PRIMO: PARETO-OPTIMAL INFORMATION-CONSTRAINED COMPRESSION FOR MIXTURE-OF-EXPERTS LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture-of-Experts (MoE) architectures scale large language models (LLMs) efficiently, but every expert must be stored although only a few are active per token, so memory becomes the deployment bottleneck. Existing pipelines merge experts into a shared base by heuristics, truncate the deltas at energy thresholds that bound neither error nor cost, and deploy one fixed configuration for every task. We propose PRIMO (Pareto-optimal Information-constrained Mixture-of-experts Compression), which casts post-training MoE compression as one constrained problem: minimize model size subject to explicit bounds on reconstruction error and computational overhead. Its key insight is that the costly stages are task-independent while the right compression level is task-dependent, and routing links the two. Once per model, MI-constrained decomposition builds the shared base by minimizing a routing-weighted KL surrogate of information loss, PGD low-rank compression with a rank search fits the deltas to both bounds, and a Pareto frontier stores the non-dominated size-error configurations. At deployment, task-adaptive selection tightens the error bound by the target task's routing shift and picks a configuration by a linear scan after one forward pass, without re-optimization. On four MoE LLMs (Mixtral-8×7B, DeepSeek-MoE-16B, Phi-3.5-MoE, Qwen2-57B-A14B) and 12 tasks, PRIMO attains the lowest perplexity and highest average accuracy in all eight settings, and its lead over D^2-MoE widens with compression, reaching 5.12 against 6.46 WikiText-2 perplexity at 60% compression of Mixtral-8×7B. Under routing shift, task-adaptive selection raises HumanEval Pass@1 from 21.5% (static configuration) to 33.5%, and its gain grows with the shift.

1 INTRODUCTION

LLMs have achieved remarkable success across diverse applications (Touvron et al., 2023), while MoE models have emerged as a promising paradigm for efficiently scaling model capacity through sparse activation (Jiang et al., 2024; Muennighoff et al., 2025). By activating only a subset of expert networks per input token, MoE architectures such as Mixtral-8×7B, DeepSeek-V3, and Qwen-MoE achieve state-of-the-art performance across benchmarks, often matching or surpassing larger dense models (Liu et al., 2024a; Yang et al., 2025). However, MoE models face critical limitations in deployment: their massive parameter counts create prohibitive storage requirements, substantial memory overhead, and significant barriers to resource-constrained environments (Cao et al., 2025).

To mitigate these limitations, various compression techniques have been developed for MoE architectures (Cai et al., 2025). Expert pruning methods remove entire experts deemed less important based on activation frequency or gradient-based importance scores (Lu et al., 2024; Liu et al., 2024b; Xie et al., 2024). Expert merging approaches consolidate multiple experts into fewer compact representations through clustering and weighted averaging (Li et al., 2024; Chen et al., 2025a; Li et al., 2026). Low-rank decomposition techniques factorize expert weight matrices to reduce parameter counts while preserving functional behavior (Gu et al., 2025; Yang et al., 2024b; Liu et al., 2025). However, existing methods suffer from three fundamental challenges. First, heuristic merging strategies construct shared base weights without principled criteria, leading to unpredictable information loss that directly translates to performance degradation (Miao et al., 2025). Second, energy-based truncation thresholds for low-rank compression provide no explicit control over reconstruction error or computational overhead, resulting in highly variable outcomes across different experts even under identical compression settings (Li et al., 2025a). Third, static compression configurations determined

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.

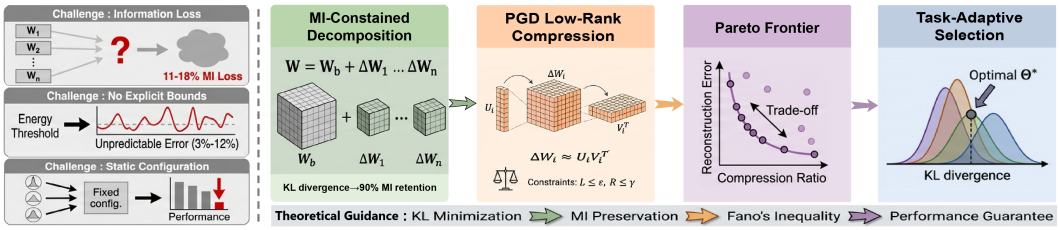


Figure 1: **Left:** Existing MoE compression loses expert information, has no explicit error or cost bounds, and deploys one static configuration. **Right:** PRIMO answers each in one constrained framework: MI-constrained decomposition preserves information, PGD compression meets both bounds, and a Pareto frontier built once offline lets task-adaptive selection fit each task at deployment.

on a single calibration dataset fail to generalize across diverse downstream tasks, causing performance gaps that strongly correlate with routing pattern divergence and grow worse under further distribution shift across deployment scenarios (Zhao et al., 2025b).

To address these challenges, we propose PRIMO, which casts MoE compression as one constrained problem (Eq. (1)): minimize model size subject to a reconstruction-error bound ϵ and a computational bound γ. Its key insight is that the costly stages are task-independent while the right compression level is task-dependent; its four components are the stages of this problem, and two quantities connect them. The error bound ϵ links the stages: (1) Mutual Information (MI)-constrained decomposition fixes the shared base weight and thereby the deltas whose error is bounded; (2) Projected Gradient Descent (PGD) low-rank compression and a rank search select the smallest rank whose factors meet ϵ within γ; (3) the Pareto frontier indexes the resulting configurations by the size and error they achieve; and (4) task-adaptive selection converts the target task's routing shift into a tightened bound ϵ_{adj}. The calibration routing distribution is the second link: it weights the KL objective that builds the base and is the reference against which deployment-time routing shift is measured. The analysis relates the KL surrogate to MI loss, shows convergence of PGD to a stationary point, and bounds error amplification under routing shift. Our main contributions:

- We formulate MoE compression as a constrained problem in which delta compression and deployment-time configuration share one error bound and one computational bound, on a shared base built from a routing-weighted KL objective, replacing the merging heuristics, energy thresholds and fixed configurations examined in Section 2.
- We introduce routing-shift-aware selection: a Pareto frontier computed once offline and a tightening of ϵ driven by the KL divergence between target and calibration routing, which adapts the compression level to a task from a small target sample with a linear scan and no re-optimization at deployment.
- On four MoE LLMs and 12 tasks, PRIMO attains the lowest WikiText-2 perplexity among compared methods at every evaluated ratio (5.12 versus 6.46 for D^2-MoE on Mixtral-8×7B at 60%), and task-adaptive selection raises HumanEval Pass@1 from 21.5% to 33.5% over a static configuration.

2 Motivation

When applying existing MoE compression methods, we identify three challenges.

Challenge 1: Unpredictable Information Loss During Merging. Existing delta decomposition methods construct shared base weights using heuristic merging strategies (e.g., simple averaging, Fisher-weighted averaging), assuming that such merging preserves most task-relevant information. To test this assumption, we measure MI retention across six merging strategies on Mixtral-8×7B. As shown in Figure 2, all strategies exhibit significant information loss. Even Fisher-weighted merging, the best-performing heuristic, loses 11 to 18% of MI depending on the layer. More critically, we find strong negative correlation ($r = -0.89$) between MI retention and perplexity increase, indicating that this "hidden" information loss translates into performance degradation. This raises a key question: *What principled criteria should guide base weight construction to minimize information loss?*

Challenge 2: Heuristic Compression Without Explicit Error Bounds. Current methods compress delta weights using truncation-based SVD with manually tuned rank thresholds, typically based on

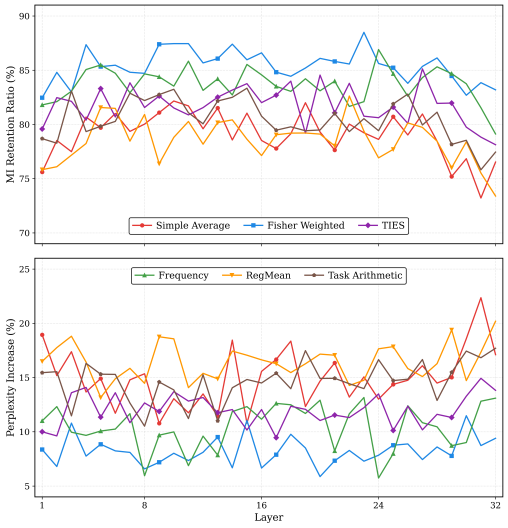


Figure 2: MI retention (**top**) and perplexity increase (**bottom**) across layers for six merging strategies on Mixtral-8×7B. Overall correlation: $r = -0.89$. Details in Appendix A.2.

Table 1: Expert-wise compression analysis on Mixtral-8×7B; tolerated error, optimal rank and speedup differ widely across experts.

Expert	Sensitivity	Speedup	Optimal Rank	Error Tol.
E1	0.85	1.40×	35%	9%
E2	1.12	1.28×	45%	5%
E3	0.71	1.52×	28%	12%
E4	0.92	1.36×	40%	7%
E5	1.38	1.21×	55%	3%
E6	0.76	1.48×	30%	11%
E7	0.89	1.38×	38%	8%
E8	0.95	1.33×	42%	6%

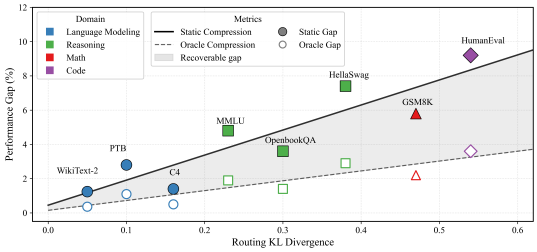


Figure 3: Performance gap vs. routing KL divergence ($r > 0.8$). Shaded area indicates recoverable gap. Details in Appendix A.4.

energy retention ratios (e.g., retaining singular values capturing 90% energy (Gu et al., 2025)). Such a threshold fixes the fraction of singular-value energy kept, while the resulting rank, and with it the computational cost, depends on each expert's spectrum; experts also differ in the error they tolerate. In Table 1, the reconstruction error an expert tolerates before a 1% accuracy loss ranges from 3% (E5) to 12% (E3), and inference speedup ranges from 1.21× to 1.52×. This raises a key question: *How can we enforce explicit bounds on both reconstruction error and computational overhead?*

Challenge 3: Static Configuration Fails Across Input Distributions. MoE models are deployed across diverse downstream tasks with varying input characteristics and expert routing patterns. However, existing compression methods use fixed configurations determined on a single calibration dataset, ignoring distribution shift between calibration and deployment. As shown in Figure 3, the performance gap strongly correlates ($r > 0.8$) with routing pattern divergence. While in-distribution performance (WikiText-2) is well-preserved with approximately 1% gap, out-of-distribution tasks suffer up to 9% degradation (HumanEval). The shaded region indicates the gap between static and oracle compression, revealing substantial room for improvement. This raises a key question: *How can we adapt compression dynamically to varying input distributions?*

3 Problem Formulation

To tackle the challenges in Section 2, we formulate MoE compression as a constrained optimization problem. Consider an MoE model with L layers, each containing N experts. Let $\mathbf{W}_i^{(l)} \in \mathbb{R}^{m\times n}$ denote the weight matrix of the i-th expert in layer l. Each expert weight is decomposed as $\mathbf{W}_i^{(l)} = \mathbf{W}_b^{(l)} + \Delta\mathbf{W}_i^{(l)}$, where the decomposition is performed independently per layer. The goal is to minimize model size subject to reconstruction error and computational complexity constraints:

$$\min_{\Theta} \ \mathcal{C}(\Theta) \quad \text{s.t.} \quad \mathcal{L}(\Theta) \le \epsilon, \ \ \mathcal{R}(\Theta) \le \gamma, \tag{1}$$

where $\Theta = \{(\mathbf{W}_b^{(l)}, \{\mathbf{U}_i^{(l)}, \mathbf{V}_i^{(l)}\}_{i=1}^N)\}_{l=1}^L$ consists of shared base weights $\mathbf{W}_b^{(l)} \in \mathbb{R}^{m\times n}$ and low-rank factors $\mathbf{U}_i^{(l)} \in \mathbb{R}^{m\times r}$, $\mathbf{V}_i^{(l)} \in \mathbb{R}^{n\times r}$ satisfying $\Delta\mathbf{W}_i^{(l)} \approx \mathbf{U}_i^{(l)}(\mathbf{V}_i^{(l)})^\top$. We assume uniform rank r across experts and layers for tractability (see Appendix A.5). $\mathcal{C}(\Theta)$ measures total parameter count; $\mathcal{L}(\Theta)$ is the global reconstruction error; and $\mathcal{R}(\Theta)$ is the computational complexity constraint.

4 Solution

To solve the constrained optimization problem in Section 3, we propose PRIMO, a build-once, select-per-task framework, as illustrated in Figure 4. PRIMO comprises four components: MI-

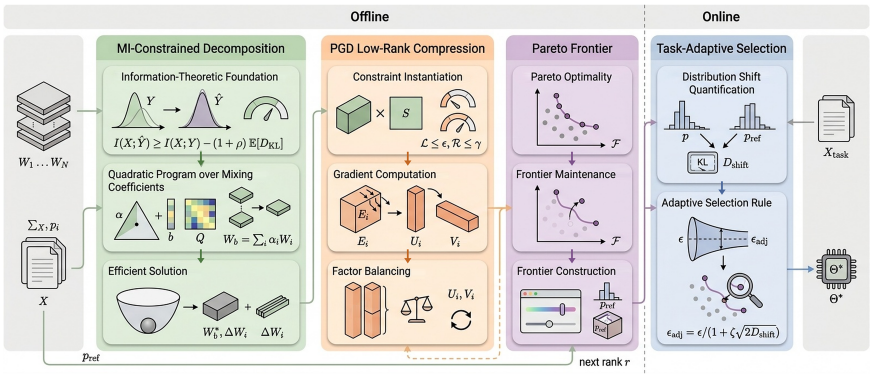


Figure 4: Overview of PRIMO. **Offline**: MI-constrained decomposition, PGD low-rank compression, and Pareto frontier construction build $\mathcal{F}$. **Online**: task-adaptive selection picks Θ^* from $\mathcal{F}$ per task.

constrained decomposition builds a shared base by minimizing a routing-weighted KL surrogate of information loss, PGD low-rank compression and a rank search fit the residual deltas under explicit error and computational bounds, the Pareto frontier stores the resulting size–error trade-offs, and task-adaptive selection picks one configuration per task. Our key insight is that the costly stages of MoE compression are task-independent while the right compression level is task-dependent, and the routing distribution connects the two: it weights the objective that builds the base offline, and at deployment its shift measures how much the error bound must tighten. PRIMO therefore compresses each model once and adapts to a new task with a single forward pass.

4.1 MI-Constrained Decomposition

To address Challenge #1, we propose MI-constrained decomposition, as illustrated in Figure 4 (green). It bounds each expert's information loss by a KL divergence, builds the shared base weight through a convex quadratic program, and solves it to the global optimum.

Information-Theoretic Foundation. The following derivation applies per layer; we omit layer superscripts for clarity. Let X denote the input activations, $Y = \mathbf{W}X + \xi$ the original expert output, and $\hat{Y} = \mathbf{W}_b X + \xi$ the reconstructed output, where $\mathbf{W}_b$ denotes the shared base weight. Under the Gaussian assumption with $\xi \sim \mathcal{N}(0, \Sigma_\xi)$, where $\Sigma_\xi \succ 0$ is estimated from inter-expert output diversity and induces a Mahalanobis metric weighting reconstruction error (see Appendix A.7.5), the conditional entropies satisfy $H(Y|X) = H(\hat{Y}|X) = H(\xi)$, since both outputs share the same conditional covariance. Therefore, the MI gap reduces to an entropy difference:

$$I(X;Y) - I(X;\hat{Y}) = H(Y) - H(\hat{Y}). \tag{2}$$

For Gaussian distributions, this entropy difference is bounded by the expected KL divergence up to a factor $1+\rho$ that depends on the signal-to-noise ratio and the reconstruction error (Theorem in Appendix A.7.3):

$$I(X;\hat{Y}) \geq I(X;Y) - (1+\rho)\,\mathbb{E}_X[D_{\mathrm{KL}}(p(Y|X)\|p(\hat{Y}|X))]. \tag{3}$$

This bound makes KL divergence minimization a surrogate objective for information preservation, so the goal becomes finding the base weight $\mathbf{W}_b$ that minimizes the expected KL divergence across all experts.

Quadratic Program over Mixing Coefficients. To make the optimization tractable, the base weight is parameterized as a convex combination of expert weights. Let $\Delta^{N-1} = \{\boldsymbol{\alpha} \in \mathbb{R}^N : \alpha_i \geq 0, \sum_i \alpha_i = 1\}$ denote the $(N-1)$-dimensional probability simplex. The parameterization is:

$$\mathbf{W}_b = \sum_{i=1}^{N} \alpha_i \mathbf{W}_i, \quad \boldsymbol{\alpha} \in \Delta^{N-1}. \tag{4}$$

The expected KL divergence over the routing distribution $\{p_i\}_{i=1}^N$ admits a closed-form expression:

$$\mathbb{E}_{i\sim p}[D_{\mathrm{KL}}] = \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{Q}\boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \mathbf{b} + c. \tag{5}$$

The matrix $\mathbf{Q} \in \mathbb{R}^{N \times N}$ captures pairwise expert interactions via trace inner products:

$$Q_{jk} = \text{tr}(\Sigma_\xi^{-1} \mathbf{W}_j \Sigma_X \mathbf{W}_k^\top). \tag{6}$$

Similarly, the vector $\mathbf{b} \in \mathbb{R}^N$ encodes the per-expert alignment with the routing-weighted average:

$$b_j = \sum_{i=1}^{N} p_i \cdot \text{tr}(\Sigma_\xi^{-1} \mathbf{W}_i \Sigma_X \mathbf{W}_j^\top). \tag{7}$$

Notably, Q_{jk} is independent of the routing distribution $\{p_i\}$, while b_j retains this dependence. Combining with the simplex constraint yields a convex quadratic program:

$$\min_{\boldsymbol{\alpha} \in \Delta^{N-1}} \ \frac{1}{2} \boldsymbol{\alpha}^\top \mathbf{Q} \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \mathbf{b} + c. \tag{8}$$

Efficient Solution. Since $\mathbf{Q}$ is a Gram matrix, it is positive semi-definite, so the QP in Eq. (8) is convex and attains its global optimum on the simplex. We solve it with a standard convex solver. Details in Appendix A.7.2.

Eq. (3) links the KL surrogate to MI loss; Appendix A.7.3 gives the retention level that a KL budget implies under the conditions of Appendix A.7.4. The decomposition returns $\mathbf{W}_b^* = \sum_i \alpha_i^* \mathbf{W}_i$ and $\Delta \mathbf{W}_i = \mathbf{W}_i - \mathbf{W}_b^*$.

4.2 PGD Low-Rank Compression

To address Challenge #2, we propose PGD low-rank compression, as illustrated in Figure 4 (orange). It instantiates the error and computational bounds for each delta, updates the low-rank factors by projected gradient descent, and rebalances them periodically for numerical stability.

Constraint Instantiation. The following optimization applies per layer; we omit layer superscripts for clarity. For each expert i with target rank r, low-rank factors $\mathbf{U}_i \in \mathbb{R}^{m \times r}$ and $\mathbf{V}_i \in \mathbb{R}^{n \times r}$ are sought such that $\Delta \mathbf{W}_i \approx \mathbf{U}_i \mathbf{V}_i^\top$. A naive approach would minimize the Frobenius norm $\|\Delta \mathbf{W}_i - \mathbf{U}_i \mathbf{V}_i^\top\|_F^2$, but this treats all weight elements equally regardless of their importance during inference. Instead, activation statistics are incorporated to weight the reconstruction error by actual usage patterns. For expert i, the relative reconstruction error is defined as:

$$\mathcal{L}_i(\mathbf{U}_i, \mathbf{V}_i) = \frac{\left\|(\Delta \mathbf{W}_i - \mathbf{U}_i \mathbf{V}_i^\top)\mathbf{S}\right\|_F^2}{\|\Delta \mathbf{W}_i \mathbf{S}\|_F^2}, \tag{9}$$

where $\mathbf{S} = (\Sigma_X + \lambda \mathbf{I})^{1/2}$ with small $\lambda > 0$ for numerical stability (see Appendix A.12.2). The input covariance Σ_X is estimated from calibration data and shared across all experts within each layer.

The reconstruction error constraint $\mathcal{L}(\Theta) \leq \epsilon$ is instantiated via two-level aggregation: first averaging across experts within each layer, then averaging across all layers:

$$\mathcal{L}^{(l)}(\Theta) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_i^{(l)}, \qquad \mathcal{L}(\Theta) = \frac{1}{L} \sum_{l=1}^{L} \mathcal{L}^{(l)}(\Theta). \tag{10}$$

With this definition, ϵ specifies the average relative reconstruction error across $L \times N$ expert weight matrices. The computational complexity constraint $\mathcal{R}(\Theta) \leq \gamma$ in Eq. (1) is instantiated as:

$$\mathcal{R}(\Theta) = \frac{r(m+n)}{mn} \leq \gamma, \tag{11}$$

ensuring that the low-rank approximation does not exceed the target computational overhead.

Gradient Computation. The gradients with respect to the low-rank factors are derived as follows. Let $\mathbf{E}_i = \Delta \mathbf{W}_i - \mathbf{U}_i \mathbf{V}_i^\top$ denote the residual matrix and $D_i = \|\Delta \mathbf{W}_i \mathbf{S}\|_F^2$ denote the normalization constant. Since $\mathbf{S}$ is symmetric, we have $\mathbf{S}\mathbf{S}^\top = \mathbf{S}^2$. Taking derivatives of Eq. (9):

$$\nabla_{\mathbf{U}_i} \mathcal{L}_i = -\frac{2}{D_i} \mathbf{E}_i \mathbf{S}^2 \mathbf{V}_i, \quad \nabla_{\mathbf{V}_i} \mathcal{L}_i = -\frac{2}{D_i} \mathbf{S}^2 \mathbf{E}_i^\top \mathbf{U}_i. \tag{12}$$

Factor Balancing. Gradient updates may cause factor imbalance where $\|\mathbf{U}_i\|_F$ and $\|\mathbf{V}_i\|_F$ diverge, degrading numerical stability. To address this, we apply periodic factor balancing via SVD re-parameterization. Given updated factors $(\mathbf{U}'_i, \mathbf{V}'_i)$, the SVD of their product is computed:

$$\mathbf{U}'_i \mathbf{V}'^{\top}_i = \sum_{k=1}^{r} \sigma_k \mathbf{u}_k \mathbf{v}_k^{\top}. \tag{13}$$

The balanced factors are obtained by distributing singular values evenly: $\mathcal{B}_r(\mathbf{U}'_i, \mathbf{V}'_i) = (\tilde{\mathbf{U}}_i, \tilde{\mathbf{V}}_i)$, where $\tilde{\mathbf{U}}_i = [\sqrt{\sigma_1}\mathbf{u}_1, \ldots, \sqrt{\sigma_r}\mathbf{u}_r]$ and $\tilde{\mathbf{V}}_i = [\sqrt{\sigma_1}\mathbf{v}_1, \ldots, \sqrt{\sigma_r}\mathbf{v}_r]$. This operation preserves the product $\mathbf{U}_i\mathbf{V}_i^{\top}$ exactly while ensuring $\|\tilde{\mathbf{U}}_i\|_F = \|\tilde{\mathbf{V}}_i\|_F$ (see Appendix A.8.2 for details).

Combining gradient descent with factor balancing, the complete PGD iteration at step t is:

$$\mathbf{U}_i^{(t+\frac{1}{2})} = \mathbf{U}_i^{(t)} - s_t \nabla_{\mathbf{U}_i} \mathcal{L}_i, \quad \mathbf{V}_i^{(t+\frac{1}{2})} = \mathbf{V}_i^{(t)} - s_t \nabla_{\mathbf{V}_i} \mathcal{L}_i, \tag{14}$$

followed by applying factor balancing $\mathcal{B}_r$ to obtain $(\mathbf{U}_i^{(t+1)}, \mathbf{V}_i^{(t+1)})$.

The factors are initialized via truncated SVD of $\Delta\mathbf{W}_i\mathbf{S}$, and the step size s_t is determined via Armijo line search. Convergence to a stationary point at rate $O(1/T)$ is established in Appendix A.8. A binary search over the rank then keeps the smallest rank whose factors $\{\mathbf{U}_i, \mathbf{V}_i\}_{i=1}^N$ satisfy the constraints in Eq. (1) (Algorithm 1).

4.3 Pareto Frontier

To supply task-adaptive selection with candidates, we propose the Pareto frontier, as illustrated in Figure 4 (purple). It defines Pareto optimality over model size and reconstruction error, maintains the non-dominated configurations, and constructs the frontier offline.

Pareto Optimality. Model size $\mathcal{C}(\Theta)$ and reconstruction error $\mathcal{L}(\Theta)$ conflict: stronger compression reduces memory footprint but increases reconstruction error. A configuration Θ is Pareto-optimal if no other configuration dominates it in model size and reconstruction error. The Pareto frontier $\mathcal{F}$ collects all such non-dominated configurations:

$$\mathcal{F} = \{\Theta^* : \Theta^* \text{ is not dominated by any } \Theta' \in \mathcal{S}\}, \tag{15}$$

where $\mathcal{S}$ denotes the set of evaluated configurations satisfying $\mathcal{R}(\Theta) \le \gamma$, and Θ' dominates Θ^* iff $\mathcal{C}(\Theta') \le \mathcal{C}(\Theta^*)$, $\mathcal{L}(\Theta') \le \mathcal{L}(\Theta^*)$ with at least one strict inequality.

Frontier Maintenance. As new configurations are evaluated, the frontier is maintained incrementally. For a candidate Θ_{new}, we update:

$$\mathcal{F} \leftarrow (\mathcal{F} \setminus \mathcal{D}) \cup \begin{cases} \{\Theta_{\text{new}}\} & \text{if } \Theta_{\text{new}} \text{ is non-dominated} \\ \emptyset & \text{otherwise} \end{cases}, \tag{16}$$

where $\mathcal{D} = \{\Theta \in \mathcal{F} : \Theta_{\text{new}} \text{ dominates } \Theta\}$. This update keeps $\mathcal{F}$ equal to the non-dominated set of the configurations evaluated so far. More details are provided in Appendix A.9.

Frontier Construction. Algorithm 1 (See Appendix A.11.2) constructs the frontier by sweeping compression levels and updating $\mathcal{F}$ incrementally. For each configuration $\Theta_k \in \mathcal{F}$, we store: (i) the model size metric $\mathcal{C}(\Theta_k)$, (ii) the reconstruction error $\mathcal{L}(\Theta_k)$, and (iii) the compressed parameters $(\mathbf{W}_b, \{\mathbf{U}_i, \mathbf{V}_i\})$. We also record the reference routing distribution on calibration data:

$$p_{\text{ref}}^{(l)}(i) = \frac{1}{k\,|\mathcal{X}|} \sum_{x \in \mathcal{X}} \mathbf{1}[i \in \text{Top-}k(G^{(l)}(x))], \tag{17}$$

where $G^{(l)}(x)$ is the router output for input x at layer l and k is the number of active experts per token.

Upon completion, we obtain: (1) the Pareto frontier $\mathcal{F}$ containing K configurations that span the compression-performance spectrum, and (2) the reference routing distributions $\{p_{\text{ref}}^{(l)}\}_{l=1}^L$ for distribution shift estimation. Both are computed once per model and cached for deployment.

4.4 TASK-ADAPTIVE SELECTION

To address Challenge #3, we propose task-adaptive selection, as illustrated in Figure 4 (blue). It measures the routing shift of the target task from the calibration reference, tightens the error bound by this shift, and selects the most compressed configuration that satisfies it.

Distribution Shift Quantification. Given a small sample $\mathcal{X}_{\text{task}}$ from the target task, we estimate the routing distribution and measure its divergence from the calibration reference:

$$D_{\text{shift}} = \frac{1}{L}\sum_{l=1}^{L} D_{\text{KL}}(p^{(l)} \| p_{\text{ref}}^{(l)}) = \frac{1}{L}\sum_{l=1}^{L}\sum_{i=1}^{N} p^{(l)}(i) \log \frac{p^{(l)}(i)}{p_{\text{ref}}^{(l)}(i)}, \tag{18}$$

where $p^{(l)}(i)$ is computed analogously to Eq. (17) on target samples. To ensure numerical stability when $p_{\text{ref}}^{(l)}(i) = 0$, we apply additive smoothing (see Appendix A.10.1).

Adaptive Selection Rule. For a fixed configuration, the routing-weighted reconstruction error on the target task satisfies $\mathcal{L}_{\text{task}} \leq \mathcal{L}_{\text{ref}}(1 + \zeta\sqrt{2D_{\text{shift}}})$, where $\mathcal{L}_{\text{ref}}$ is the routing-weighted error on calibration data and ζ bounds the largest deviation of a per-expert error from its layer's reference, relative to $\mathcal{L}_{\text{ref}}$ (Appendix A.10.2). Following this form, we adjust the constraint:

$$\epsilon_{\text{adj}} = \frac{\epsilon}{1 + \zeta\sqrt{2D_{\text{shift}}}}, \tag{19}$$

and select from $\mathcal{F}$ the configuration with maximum compression satisfying $\mathcal{L}(\Theta) \leq \epsilon_{\text{adj}}$. This requires only a linear scan over $\mathcal{F}$ with no optimization at deployment time. (When target samples are unavailable, set $D_{\text{shift}} = 0$ and select directly from $\mathcal{F}$ based on resource constraints.)

The components form one pipeline: the offline stages run once per model and cache $\mathcal{F}$ with $\{p_{\text{ref}}^{(l)}\}$, and deployment needs only one forward pass over $\mathcal{X}_{\text{task}}$. A task with a larger routing shift tightens ϵ_{adj} and receives a less compressed Θ^*; Algorithm 1 (Appendix A.11) lists the full procedure.

5 EXPERIMENT

5.1 IMPLEMENTATION DETAILS

Experimental Setup. Our framework is built upon PyTorch 2.1.0 (Paszke et al., 2019) with CUDA 12.1 and Hugging Face Transformers 4.36.0 (Wolf et al., 2020). We evaluate on four state-of-the-art MoE models spanning four architecture families with 8 to 64 routed experts per layer: Mixtral-8×7B (Jiang et al., 2024), DeepSeek-MoE-16B (Dai et al., 2024), Phi-3.5-MoE (Abdin et al., 2024), and Qwen2-57B-A14B (Yang et al., 2024a), all loaded in bfloat16. Details in Appendix A.12.1.

Software and Hardware. Experiments are conducted on four NVIDIA A100 80GB GPUs with NVLink interconnect. For calibration, we use 128 sequences of 2048 tokens from C4 (Raffel et al., 2020), following standard practice in MoE compression (Lu et al., 2024; Xie et al., 2024); each baseline keeps its original calibration setup. SVD computations use randomized algorithms for efficiency on large matrices, avoiding full-matrix materialization overhead. Details in Appendix A.12.1.

Method-Specific Settings. For MI-constrained decomposition, we use regularization $\lambda = 10^{-6}$. For PGD compression, we use Armijo backtracking line search with relative-change threshold 10^{-6} or 100 iterations. For task-adaptive selection, we use 128 target-task samples and $\zeta = 1.0$ across all four MoE backbones; all other hyperparameters retain their defaults. Details in Appendix A.12.2.

Pareto Search Space. The Pareto frontier uses the ϵ-constraint method with binary search over the rank. Complexity bounds $\gamma \in \{0.25, 0.30, 0.35, 0.40\}$ (default 0.30) and error bounds $\epsilon \in \{0.03, 0.05, 0.08, 0.10\}$ define the search grid. Base weight sparsity follows D^2-MoE (Gu et al., 2025) (10% static + 10% dynamic) and stays fixed during the search. Details in Appendix A.12.3.

Baselines, Evaluation Metrics, and Implementation Details. We benchmark against delta decomposition, expert merging, expert pruning, sparsity and low-rank baselines under identical evaluation protocols. We report WikiText-2, PTB and C4 perplexity (PPL, ↓) and average accuracy over nine commonsense, math and knowledge tasks (↑). Details in Appendix A.12.4–A.13 and A.17.

Table 2: Performance of PRIMO and baselines on Mixtral-8×7B and DeepSeekMoE-16B-Base at 20%, 40% and 60% compression, and on Phi-3.5-MoE and Qwen2-57B-A14B at 40%. We report perplexity (↓) on three language modeling datasets and accuracy (↑) on seven zero-shot tasks plus MMLU and GSM8K, with Avg over all nine. Gray: full model; pink: best compressed method.

Model	Ratio	Method	Wiki-2↓	PTB↓	C4↓	Openb.	ARC_e	WinoG.	HellaS.	ARC_c	PIQA	MathQA	MMLU	GSM8K	Avg↑
Mixtral 8×7B	0%	Full Model (Jiang et al., 2024)	3.98	12.99	6.78	0.36	0.84	0.76	0.65	0.57	0.82	0.43	0.68	0.59	0.63
	20%	NAEE (Lu et al., 2024)	4.77	16.09	8.89	0.32	0.76	0.72	0.58	0.47	0.79	0.40	0.59	0.52	0.57
		MoE-I^2 (Yang et al., 2024b)	4.86	26.50	11.07	0.32	0.79	0.74	0.55	0.48	0.78	0.37	0.58	0.51	0.57
		D^2-MoE (Gu et al., 2025)	4.65	16.32	8.59	0.33	0.80	0.75	0.61	0.51	0.81	0.39	0.60	0.54	0.59
		PRIMO (Ours)	4.18	14.12	7.28	0.35	0.82	0.75	0.63	0.51	0.81	0.42	0.65	0.54	0.61
	40%	NAEE (Lu et al., 2024)	6.44	22.15	13.86	0.25	0.63	0.64	0.46	0.36	0.72	0.35	0.49	0.43	0.48
		MoE-I^2 (Yang et al., 2024b)	6.74	60.45	22.44	0.26	0.71	0.66	0.43	0.38	0.69	0.31	0.50	0.44	0.49
		D^2-MoE (Gu et al., 2025)	5.28	20.54	10.10	0.32	0.78	0.73	0.57	0.47	0.78	0.34	0.57	0.51	0.56
		PRIMO (Ours)	4.58	15.38	8.24	0.33	0.80	0.74	0.61	0.49	0.78	0.38	0.61	0.53	0.58
	60%	NAEE (Lu et al., 2024)	11.43	47.28	31.16	0.17	0.42	0.55	0.33	0.23	0.62	0.26	0.37	0.32	0.36
		MoE-I^2 (Yang et al., 2024b)	13.52	182.99	74.62	0.18	0.44	0.55	0.32	0.22	0.58	0.23	0.37	0.32	0.36
		D^2-MoE (Gu et al., 2025)	6.46	23.63	12.76	0.28	0.72	0.71	0.51	0.38	0.73	0.31	0.52	0.46	0.51
		PRIMO (Ours)	5.12	17.86	9.58	0.31	0.75	0.72	0.57	0.45	0.76	0.35	0.50	0.45	0.54
DeepSeek MoE-16B -Base	0%	Full Model (Dai et al., 2024)	6.38	9.47	9.82	0.32	0.76	0.70	0.58	0.44	0.79	0.31	0.45	0.19	0.50
	20%	NAEE (Lu et al., 2024)	9.44	15.02	15.34	0.32	0.71	0.66	0.55	0.40	0.77	0.29	0.40	0.17	0.47
		MoE-I^2 (Yang et al., 2024b)	7.69	11.59	13.72	0.26	0.71	0.68	0.49	0.38	0.73	0.29	0.38	0.16	0.45
		D^2-MoE (Gu et al., 2025)	6.84	11.10	11.88	0.30	0.74	0.69	0.55	0.41	0.76	0.31	0.41	0.17	0.48
		PRIMO (Ours)	6.72	10.15	10.58	0.31	0.74	0.69	0.57	0.42	0.77	0.30	0.43	0.18	0.49
	40%	NAEE (Lu et al., 2024)	8.55	14.47	17.98	0.23	0.67	0.67	0.41	0.32	0.69	0.26	0.35	0.15	0.42
		MoE-I^2 (Yang et al., 2024b)	9.73	15.75	19.75	0.23	0.64	0.66	0.41	0.31	0.68	0.26	0.34	0.14	0.41
		D^2-MoE (Gu et al., 2025)	7.93	14.07	15.18	0.26	0.69	0.65	0.45	0.36	0.72	0.28	0.38	0.16	0.44
		PRIMO (Ours)	7.35	11.28	12.15	0.29	0.71	0.67	0.51	0.39	0.74	0.29	0.40	0.17	0.46
	60%	NAEE (Lu et al., 2024)	23.20	49.89	48.63	0.17	0.49	0.58	0.33	0.24	0.61	0.23	0.29	0.12	0.34
		MoE-I^2 (Yang et al., 2024b)	15.83	32.20	38.60	0.17	0.48	0.58	0.32	0.23	0.61	0.22	0.28	0.12	0.33
		D^2-MoE (Gu et al., 2025)	11.67	27.73	27.63	0.21	0.54	0.61	0.35	0.29	0.63	0.24	0.31	0.13	0.37
		PRIMO (Ours)	8.52	13.89	15.28	0.26	0.65	0.63	0.46	0.35	0.68	0.26	0.36	0.15	0.42
Phi-3.5 MoE	0%	Full Model (Abdin et al., 2024)	3.48	8.43	8.22	0.40	0.77	0.76	0.68	0.56	0.79	0.38	0.78	0.89	0.67
	40%	NAEE (Lu et al., 2024)	8.18	20.07	16.11	0.35	0.73	0.73	0.61	0.48	0.76	0.37	0.60	0.68	0.59
		MoE-I^2 (Yang et al., 2024b)	7.46	20.95	20.95	0.29	0.59	0.67	0.27	0.40	0.70	0.25	0.58	0.66	0.49
		D^2-MoE (Gu et al., 2025)	6.07	13.79	14.01	0.34	0.72	0.73	0.65	0.53	0.78	0.38	0.72	0.82	0.63
		PRIMO (Ours)	4.28	10.15	10.08	0.37	0.74	0.73	0.66	0.52	0.78	0.38	0.73	0.81	0.64
Qwen2 57B-A14B	0%	Full Model (Yang et al., 2024a)	5.12	9.18	8.86	0.33	0.75	0.74	0.63	0.46	0.81	0.39	0.77	0.81	0.63
	40%	NAEE (Lu et al., 2024)	6.81	11.34	11.57	0.31	0.73	0.73	0.55	0.46	0.76	0.36	0.60	0.63	0.57
		MoE-I^2 (Yang et al., 2024b)	24.90	77.05	22.50	0.26	0.70	0.46	0.71	0.41	0.75	0.30	0.66	0.70	0.55
		D^2-MoE (Gu et al., 2025)	8.19	11.23	12.70	0.33	0.75	0.75	0.61	0.45	0.79	0.36	0.75	0.79	0.62
		PRIMO (Ours)	5.68	9.95	9.98	0.33	0.75	0.75	0.62	0.46	0.80	0.38	0.76	0.78	0.63

Table 3: Performance of Mixtral-8×7B compressed by different methods under 20% compression.

Methods	Wiki-2↓	PTB↓	C4↓	Openb.	ARC_e	WinoG.	HellaS.	ARC_c	PIQA	MathQA	MMLU	GSM8K	Avg↑
Full Model (Jiang et al., 2024)	3.98	12.99	6.78	0.36	0.84	0.76	0.65	0.57	0.82	0.43	0.68	0.59	0.63
NAEE (Lu et al., 2024)	4.77	16.09	8.89	0.32	0.76	0.72	0.58	0.47	0.79	0.40	0.59	0.52	0.57
SparseGPT (2:4) (Frantar & Alistarh, 2023)	4.69	21.11	9.19	0.30	0.77	0.74	0.56	0.45	0.77	0.35	0.57	0.51	0.56
MoE-I^2 (Yang et al., 2024b)	4.86	26.50	11.07	0.32	0.79	0.74	0.55	0.48	0.78	0.37	0.58	0.51	0.57
ASVD (Yuan et al., 2025b)	9.44	47.29	20.30	0.25	0.71	0.66	0.48	0.40	0.73	0.35	0.52	0.46	0.51
LoSparse (Li et al., 2023)	953.51	805.16	1273.12	0.20	0.27	0.49	0.28	0.26	0.53	0.20	0.27	0.24	0.30
MC-SMoE (Li et al., 2024)	1341.36	1316.52	1478.13	0.26	0.28	0.51	0.29	0.25	0.54	0.19	0.28	0.25	0.31
MoE-Compress (He et al., 2025)	6.12	14.67	11.61	0.30	0.73	0.70	0.54	0.46	0.73	0.33	0.55	0.49	0.54
D^2-MoE (Gu et al., 2025)	4.65	16.32	8.59	0.33	0.80	0.75	0.61	0.51	0.81	0.39	0.60	0.54	0.59
PRIMO (Ours)	4.18	14.12	7.28	0.35	0.82	0.75	0.63	0.51	0.81	0.42	0.65	0.54	0.61

5.2 Main Results

Compression Performance. PRIMO attains the lowest perplexity on all three language modeling datasets and the highest average accuracy in all eight settings, across four MoE LLMs and three compression ratios. Its advantage grows with compression: on Mixtral-8×7B the WikiText-2 gap to D^2-MoE widens from 0.47 at 20% to 1.34 at 60%, and on DeepSeekMoE-16B from 0.12 to 3.15, where PRIMO reaches 8.52 against 11.67 for D^2-MoE and 23.20 for the expert-pruning method NAEE. Pruning-based methods degrade fastest because they remove whole experts, while PRIMO keeps every expert as a shared base plus a low-rank delta whose error is bounded explicitly. The fine-grained Qwen2-57B-A14B follows the same pattern: at 40% compression PRIMO lowers perplexity from 8.19 for D^2-MoE to 5.68 and matches the full model's average accuracy of 0.63 (Table 2).

Comparison with State-of-the-Art Methods. Against eight methods spanning pruning, sparsity, low-rank, merging and delta decomposition on Mixtral-8×7B at 20% compression, PRIMO attains the lowest perplexity on all three datasets and the best or tied-best accuracy on all nine benchmarks. It retains 96.8% of the full model's average accuracy (0.61 vs. 0.63), and its WikiText-2 perplexity rises by only 0.20 over the full model, against 0.67 for D^2-MoE and 0.71 for SparseGPT. MC-SMoE and LoSparse exceed 800 perplexity, and ASVD, which has no shared base across experts, trails PRIMO by 10 points of average accuracy (0.51 vs. 0.61). D^2-MoE shares PRIMO's delta decomposition, so its gap to PRIMO, largest on MMLU (0.60 vs. 0.65), reflects the MI-constrained base and PGD compression, replacing its Fisher merging and truncated SVD (Table 3).

Memory Efficiency and Inference Throughput. PRIMO keeps inference fast: on Mixtral-8×7B at 60%, 70% and 80% compression it reaches 280.78, 304.09 and 317.39 tokens/sec, the highest at every

ratio and 1.1–1.3% above D^2-MoE at equal size and memory, while LoSparse needs 2.3–4.5× more TFLOPs and runs up to 1.77× slower; selection adds no runtime cost (Table 15, Appendix A.14).

5.3 Ablation Study

Effectiveness of MI-Constrained Decomposition. The base weight determines what every delta must reconstruct. Parameter-space merges leave large residuals: averaging reaches 7.66 perplexity, TIES 12.45 and RegMean 187.19. D^2-MoE's Fisher merging (5.28) helps, and PRIMO, built on the MI-constrained base, reaches 4.58 (Table 16, Appendix A.15).

Synergy of Base and Delta Optimization. The MI-constrained base lowers perplexity by 0.35 over the Fisher-weighted base under both Vanilla SVD (6.22 to 5.87) and Truncation-Aware SVD (5.28 to 4.93), so its gain holds for either compressor. On the MI-constrained base, PGD with the rank search lowers perplexity by a further 0.35, to 4.58. The two choices add up: together they reduce perplexity from 6.22 to 4.58, a 26% reduction at the same 40% compression ratio (Table 4; full results in Appendix A.16).

Table 4: Base and delta ablation (Mixtral-8×7B, 40%, WikiText-2 PPL↓).

Base	Delta compressor	PPL
Full model		3.98
Fisher	Vanilla SVD	6.22
Fisher	Truncation-Aware SVD	5.28
MI-constrained	Vanilla SVD	5.87
MI-constrained	Truncation-Aware SVD	4.93
MI-constrained	**PGD (PRIMO)**	**4.58**

Impact of Task-Adaptive Selection. The static policy deploys the calibration-chosen configuration for every task; the adaptive policy tightens the error bound by each task's routing shift and selects a less compressed configuration when needed. The two coincide without shift (WikiText-2), and the gain grows with the shift: 11.1% on MMLU, 29.1% on GSM8K and 55.8% on HumanEval, where Pass@1 rises from 21.5% to 33.5% (Table 5; full results in Appendix A.18).

Table 5: Static vs. adaptive selection on Mixtral-8×7B under routing shift (KL).

Task	KL	Static	Adaptive
WikiText-2 (PPL↓)	0.00	4.58	4.58
C4 (PPL↓)	0.12	8.24	**8.15**
MMLU (Acc)	0.35	54.1	**60.1**
GSM8K (Acc)	0.48	41.2	**53.2**
HumanEval (Pass@1)	0.62	21.5	**33.5**

6 Related Work

MoE Models. The MoE paradigm enables efficient scaling by activating only a subset of experts per token (Shazeer et al., 2017). GShard (Lepikhin et al., 2021) and Switch Transformer (Fedus et al., 2022) advanced sharding and top-1 routing. Recent architectures go further: Mixtral-8×7B (Jiang et al., 2024) matches LLaMA-2 70B using only 13B active parameters, DeepSeekMoE (Dai et al., 2024; Deng et al., 2025) introduces fine-grained expert segmentation with shared expert pools, and Qwen2-MoE demonstrates state-of-the-art quality-efficiency trade-offs. However, a key deployment challenge remains: all expert parameters must reside in limited GPU memory despite sparse activation, motivating MoE-specific compression techniques (Cai et al., 2025; Zhang et al., 2025a).

MoE Compression. Pruning (Lu et al., 2024; Xie et al., 2024; Zhao et al., 2025a) and merging (Li et al., 2024; Chen et al., 2025a; Zhang et al., 2025c; Miao et al., 2025) remove or fuse experts by routing statistics, with subspace (Li et al., 2026) and channel-wise (Zhang et al., 2025b) variants. Delta methods keep a shared base plus residuals: D^2-MoE (Gu et al., 2025) uses Fisher merging and SVD, and MoE-I^2 (Yang et al., 2024b) adds inter-expert pruning; others use structured SVD (Li et al., 2025a), shift-aware quantization (Huang et al., 2025; Chen et al., 2025b) and bit-packed merging (Zhao et al., 2025b). PRIMO shares D^2-MoE's decomposition and ASVD's activation-aware weighting (Yuan et al., 2025b); it sets ranks and the configuration under explicit error and computational bounds, builds the base from routing-weighted KL, and adapts to routing shift.

7 Conclusion

PRIMO casts MoE compression as one constrained problem: minimize model size under explicit error and computational bounds. Its key insight is that the costly stages are task-independent while the right compression level is task-dependent, and routing links the two: it weights the KL objective that builds the shared base offline, and its shift on a target task tightens the error bound online. Across four MoE LLMs and 12 tasks, PRIMO attains the lowest perplexity at every ratio, with a lead over D^2-MoE that grows with compression, and task-adaptive selection, needing one forward pass and no retraining, raises Pass@1 on HumanEval, the most shifted task, from 21.5% to 33.5%.

REFERENCES

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, Weizhu Chen, Yen-Chun Chen, Yi-Ling Chen, Hao Cheng, Parul Chopra, Xiyang Dai, Matthew Dixon, Ronen Eldan, Victor Fragoso, Jianfeng Gao, Mei Gao, Min Gao, Amit Garg, Allie Del Giorno, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Wenxiang Hu, Jamie Huynh, Dan Iter, Sam Ade Jacobs, Mojan Javaheripi, Xin Jin, Nikos Karampatziakis, Piero Kauffmann, Mahoud Khademi, Dongwoo Kim, Young Jin Kim, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Liden, Xihui Lin, Zeqi Lin, Ce Liu, Liyuan Liu, Mengchen Liu, Weishung Liu, Xiaodong Liu, Chong Luo, Piyush Madan, Ali Mahmoudzadeh, David Majercak, Matt Mazzola, Caio César Teodoro Mendes, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Liliang Ren, Gustavo de Rosa, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Yelong Shen, Swadheen Shukla, Xia Song, Masahiro Tanaka, Andrea Tupini, Praneetha Vaddamanu, Chunyu Wang, Guanhua Wang, Lijuan Wang, Shuohang Wang, Xin Wang, Yu Wang, Rachel Ward, Wen Wen, Philipp Witte, Haiping Wu, Xiaoxia Wu, Michael Wyatt, Bin Xiao, Can Xu, Jiahang Xu, Weijian Xu, Jilong Xue, Sonali Yadav, Fan Yang, Jianwei Yang, Yifan Yang, Ziyi Yang, Donghan Yu, Lu Yuan, Chenruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone. *arXiv e-prints*, art. arXiv:2404.14219, April 2024. doi: 10.48550/arXiv.2404.14219.

Mengting Ai, Tianxin Wei, Yifan Chen, Zhichen Zeng, Ritchie Zhao, Girish Varatkar, Bita Darvish Rouhani, Xianfeng Tang, Hanghang Tong, and Jingrui He. Resmoe: Space-efficient compression of mixture of experts llms via residual restoration. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025.

Fares Al Mohamad, Leonhard Donle, Felix Dorfner, Laura Romanescu, Kristin Drechsler, Mike P Wattjes, Jawed Nawabi, Marcus R Makowski, Hartmut Häntze, Lisa Adams, et al. Open-source large language models can generate labels from radiology reports for training convolutional neural networks. *Academic Radiology*, 32(5):2402–2410, 2025.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *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. 2357–2367, 2019.

Larry Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1):1–3, 1966.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=vXxardq6db`.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Sikai Bai, Haoxi Li, Jie Zhang, Zicong Hong, and Song Guo. Diep: Adaptive mixture-of-experts compression through differentiable expert pruning. In *Advances in Neural Information Processing Systems*, 2025.

Amir Beck. *First-order methods in optimization*. SIAM, 2017.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *International conference on machine learning*, pp. 531–540. PMLR, 2018.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004. ISBN 978-0521833783.

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A survey on mixture of experts in large language models. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

Shiyi Cao, Shu Liu, Tyler Griggs, Peter Schafhalter, Xiaoxuan Liu, Ying Sheng, Joseph E Gonzalez, Matei Zaharia, and Ion Stoica. Moe-lightning: High-throughput moe inference on memory-constrained gpus. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pp. 715–730, 2025.

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 *Forty-second International Conference on Machine Learning*, 2025a. URL https://openreview.net/forum?id=hslOzRxzXL.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374, 2021.

Yuanteng Chen, Yuantian Shao, Peisong Wang, and Jian Cheng. Eac-moe: Expert-selection aware compressor for mixture-of-experts large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12942–12963, 2025b.

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. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Thomas M Cover, Joy A Thomas, and John Kieffer. *Elements of information theory*, volume 2. wiley New York, 1991.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024. doi: 10.18653/v1/2024.acl-long.70.

Zehang Deng, Wanlun Ma, Qing-Long Han, Wei Zhou, Xiaogang Zhu, Sheng Wen, and Yang Xiang. Exploring deepseek: A survey on advances, applications, challenges and future directions. *IEEE/CAA Journal of Automatica Sinica*, 12(5):872–893, 2025.

Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in neural information processing systems*, 27, 2014.

Robert M Fano and David Hawkins. Transmission of information: A statistical theory of communications. *American Journal of Physics*, 29(11):793–794, 1961.

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.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pp. 10323–10337. PMLR, 2023.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. A framework for few-shot language model evaluation. *Zenodo*, 2021.

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 *International Conference on Machine Learning*, 2025.

Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2): 217–288, 2011.

Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.

Haoyu He, Haozheng Luo, and Qi R Wang. St-moe-bert: A spatial-temporal mixture-of-experts framework for long-term cross-city mobility prediction. In *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Human Mobility Prediction Challenge*, pp. 10–15, 2024.

Shwai He, Daize Dong, Liang Ding, and Ang Li. Towards efficient mixture of experts: A holistic study of compression techniques, 2025. URL https://arxiv.org/abs/2406.02500.

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.

Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shiming Zhang, Xianglong Liu, Michele Magno, and Xiaojuan Qi. BiLLM: Pushing the limit of post-training quantization for LLMs. In *International Conference on Machine Learning*, 2024.

Wei Huang, Yue Liao, Jianhui Liu, Ruifei He, Haoru Tan, Shiming Zhang, Hongsheng Li, Si Liu, and Xiaojuan Qi. Mixture compressor for mixture-of-experts LLMs gains more. In *International Conference on Learning Representations*, 2025.

Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Accurate post training quantization with small calibration sets. In *International conference on machine learning*, pp. 4466–4475. PMLR, 2021.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic, 2023. URL https://arxiv.org/abs/2212.04089.

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. ISSN 0899-7667. doi: 10.1162/neco.1991.3.1.79. URL https://doi.org/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. *arXiv preprint arXiv:2401.04088*, 2024.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. In *International Conference on Learning Representations*, 2023.

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.

Lujun Li, Qiyuan Zhu, Jiacheng Wang, Xiaoyu Qin, Wei Li, Hao Gu, Sirui Han, and Yike Guo. Submoe: 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.

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. URL https://openreview.net/forum?id=eFWG9Cy3WK.

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 *Forty-second International Conference on Machine Learning*, 2025a. URL https://openreview.net/forum?id=acJ3vdFljk.

Yan Li, Pengfei Zheng, Shuang Chen, Zewei Xu, Yuanhao Lai, Yunfei Du, and Zhengang Wang. Speculative moe: Communication efficient parallel moe inference with speculative token and expert pre-scheduling. *arXiv preprint arXiv:2503.04398*, 2025b.

Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao. Losparse: Structured compression of large language models based on low-rank and sparse approximation. In *International Conference on Machine Learning*, pp. 20336–20350. PMLR, 2023.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In P. Gibbons, G. Pekhimenko, and C. De Sa (eds.), *Proceedings of Machine Learning and Systems*, volume 6, pp. 87–100, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/file/42a452cbafa9dd64e9ba4aa95cc1ef21-Paper-Conference.pdf.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.

Enshu Liu, Junyi Zhu, Zinan Lin, Xuefei Ning, Matthew B Blaschko, Shengen Yan, Guohao Dai, Huazhong Yang, and Yu Wang. Efficient expert pruning for sparse mixture-of-experts language models: Enhancing performance and reducing inference costs. *arXiv preprint arXiv:2407.00945*, 2024b.

Zehua Liu, Han Wu, Ruifeng She, Xiaojin Fu, Xiongwei Han, Tao Zhong, and Mingxuan Yuan. Molae: Mixture of latent experts for parameter-efficient language models. *arXiv preprint arXiv:2503.23100*, 2025.

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 Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6159–6172, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.334. URL https://aclanthology.org/2024.acl-long.334/.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL https://aclanthology.org/J93-2004/.

Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

Ruijie Miao, Yilun Yao, Zihan Wang, Zhiming Wang, Bairen Yi, LingJun Liu, Yikai Zhao, and Tong Yang. Mergemoe: Efficient compression of moe models via expert output merging. *arXiv preprint arXiv:2510.14436*, 2025.

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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018. doi: 10.18653/v1/d18-1260.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Evan Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. OLMoe: Open mixture-of-experts language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xXTkbTBmqq.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

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.

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 *International Conference on Machine Learning*, pp. 18332–18346. PMLR, 2022.

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.

S Muhammad Ahmed Hassan Shah, Syed Faizan Hussain Shah, Asad Ullah, Atif Rizwan, Ghada Atteia, and Maali Alabdulhafith. Arabic sentiment analysis and sarcasm detection using probabilistic projections-based variational switch transformer. *IEEE Access*, 11:67865–67881, 2023.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

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*, 2017.

Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. Osqp: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4): 637–672, 2020.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, 2024.

Jibin B Thomas and KV Shihabudheen. Neural architecture search algorithm to optimize deep transformer model for fault detection in electrical power distribution systems. *Engineering Applications of Artificial Intelligence*, 120:105890, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

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*, 2025.

Miles Williams and Nikolaos Aletras. On the impact of calibration data in post-training quantization and pruning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10100–10118, 2024.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.

Yanyue Xie, Zhi Zhang, Ding Zhou, Cong Xie, Ziang Song, Xin Liu, Yanzhi Wang, Xue Lin, and An Xu. Moe-pruner: Pruning mixture-of-experts large language model using the hints from its router. *arXiv preprint arXiv:2410.12013*, 2024.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36:7093–7115, 2023.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report. arXiv preprint arXiv:2407.10671, 2024a.

An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, Junyang Lin, Kai Dang, Kexin Yang, Le Yu, Mei Li, Minmin Sun, Qin Zhu, Rui Men, Tao He, Weijia Xu, Wenbiao Yin, Wenyuan Yu, Xiafei Qiu, Xingzhang Ren, Xinlong Yang, Yong Li, Zhiying Xu, and Zipeng Zhang. Qwen2.5-1m technical report, 2025. URL https://arxiv.org/abs/2501.15383.

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 Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 10456–10466, Miami, Florida, USA, November 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.612. URL https://aclanthology.org/2024.findings-emnlp.612/.

Yueming Yuan, Ahan Gupta, Jianping Li, Sajal Dash, Feiyi Wang, and Minjia Zhang. X-moe: Enabling scalable training for emerging mixture-of-experts architectures on hpc platforms. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1315–1331, 2025a.

Zhihang Yuan, Yuzhang Shang, and Zhen Dong. PB-LLM: Partially binarized large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=BifeBRhikU.

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, 2025b. URL https://arxiv.org/abs/2312.05821.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019. doi: 10.18653/v1/p19-1472.

Danyang Zhang, Junhao Song, Ziqian Bi, Xinyuan Song, Yingfang Yuan, Tianyang Wang, Joe Yeong, and Junfeng Hao. Mixture of experts in large language models, 2025a. URL https://arxiv.org/abs/2507.11181.

Dengming Zhang, Xiaowen Ma, Zhenliang Ni, Zhenkai Wu, Han Shu, Xin Jiang, and Xinghao Chen. Expert merging: Model merging with unsupervised expert alignment and importance-guided layer chunking. *arXiv preprint arXiv:2509.25712*, 2025b.

Mingyang Zhang, Jing Liu, Ganggui Ding, Linlin Ou, Xinyi Yu, and Bohan Zhuang. Channel merging: Preserving specialization for merged experts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 22479–22487, 2025c.

Yiran Zhao, Guizhen Chen, Kenji Kawaguchi, Lidong Bing, and Wenxuan Zhang. Pruning general large language models into customized expert models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 23377–23391, 2025a.

Yushu Zhao, Zheng Wang, and Minjia Zhang. Puzzlemoe: Efficient compression of large mixture-of-experts models via sparse expert merging and bit-packed inference. *arXiv preprint arXiv:2511.04805*, 2025b.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

AI USE STATEMENT

We used generative AI tools to translate the text (Claude Opus 5.0), to check grammar and spelling and polish the wording (GPT-5.6), to guide the drawing of the figures (Gemini), and to check the content (Grok). We did not use generative AI tools to modify code. We have reviewed all AI-assisted work: we checked every reference for hallucinated citations. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The details needed to reproduce the results of this paper, including the hyperparameters, the calibration and evaluation data, the compression configurations, the complete algorithm and the software and hardware setup, are disclosed in this appendix.

APPENDIX

Table 6 lists the contents of each part of the appendix.

APPENDIX A.0: LIMITATIONS

Comparison with Recent Expert Pruning. This paper lacks a comparison with recent expert-pruning methods at 40% and 60% compression. A reader who must choose between removing whole experts and decomposing them needs to know how PRIMO compares with the strongest current pruning methods in the high-compression regime, where the two approaches diverge most. At these ratios the pruning-based baselines are NAEE and MoE-I^2, and we have not run newer expert-pruning methods there.

Table 6: What each part of the appendix holds.

Section	Contents
Appendix A.0	Limitations
Appendix A.1	Extended related work
Appendix A.2	Information loss of merging strategies
Appendix A.3	Expert sensitivity
Appendix A.4	Distribution shift
Appendix A.5	Constraint analysis
Appendix A.6	Background
Appendix A.7	MI-constrained decomposition
Appendix A.8	PGD convergence analysis
Appendix A.9	Pareto frontier
Appendix A.10	Theoretical analysis of constraint adjustment
Appendix A.11	End-to-end integration
Appendix A.12	Experimental setup
Appendix A.13	Algorithm complexity analysis
Appendix A.14	Memory efficiency and inference throughput
Appendix A.15	Effectiveness of MI-constrained decomposition
Appendix A.16	Synergy of base and delta optimization
Appendix A.17	PGD hyperparameter sensitivity
Appendix A.18	Impact of task-adaptive selection
Appendix A.19	Selector sensitivity
Appendix A.20	MI retention and downstream performance
Appendix A.21	Offline compression cost

Calibration Budget. This paper lacks a study of how the size of the calibration set affects PRIMO. A reader who has only a few calibration sequences, or who can afford many more, needs to know how the covariance estimates, the shared base and the selected configuration change with this budget. Every PRIMO result uses 128 calibration sequences of 2048 tokens, and we have not varied this number.

Mixed-Domain Batches. This paper lacks an evaluation of task-adaptive selection on batches that mix domains. A serving system that batches requests from different tasks, such as code and chat, needs to know how one configuration chosen per batch performs when the routing of the batch mixes several task distributions. We evaluate selection on one target task at a time and have not tested mixed batches.

APPENDIX A.1: EXTENDED RELATED WORK

Mixture-of-Experts Models. The MoE paradigm was first introduced by Jacobs et al. (1991) and later adapted to deep learning by Shazeer et al. (2017), who proposed the sparsely-gated MoE layer that activates only top-k experts per token. This design enables scaling model capacity without proportional increases in computational cost. Recent advances have successfully scaled MoE architectures to production systems. GShard (Lepikhin et al., 2021; Dai et al., 2024) introduced automatic sharding for distributed training, enabling 600B parameter multilingual models. Switch Transformer (Fedus et al., 2022; Shah et al., 2023; Thomas & Shihabudheen, 2023) simplified routing to top-1 gating, achieving 4$\times$ pre-training speedup over dense models. ST-MoE (Zoph et al., 2022; He et al., 2024) addressed training stability and transferability issues. More recent work includes Mixtral-8$\times$7B (Jiang et al., 2024; Huang et al., 2025; Al Mohamad et al., 2025), which achieves competitive performance with LLaMA-2 70B using only 13B active parameters, and DeepSeekMoE (Dai et al., 2024; Deng et al., 2025), which introduces fine-grained expert segmentation. DeepSpeed-MoE (Rajbhandari et al., 2022; Li et al., 2025b; Yuan et al., 2025a) provides system-level optimizations for efficient MoE training and inference. Despite their computational efficiency, MoE models present deployment challenges due to high memory requirements: all expert parameters must be loaded even though only a subset is activated per token. This motivates the development of compression techniques specifically designed for MoE architectures.

Quantization for LLMs. Quantization reduces memory footprint by representing weights with lower precision. GPTQ (Frantar et al., 2022) achieves accurate 4-bit quantization through layer-wise approximate second-order optimization. AWQ (Lin et al., 2024) identifies activation-aware importance for weight quantization. Recent work pushes toward extreme compression: PB-LLM (Yuan et al., 2024) and BiLLM (Huang et al., 2024) explore 1-bit quantization, though with notable accuracy degradation.

Pruning for LLMs. Pruning removes redundant parameters to reduce model size. Unstructured pruning methods like SparseGPT (Frantar & Alistarh, 2023) achieve high sparsity but require specialized hardware for acceleration. Structured pruning methods including LLM-Pruner (Ma et al., 2023) and SliceGPT (Ashkboos et al., 2024) remove entire channels or layers, enabling practical speedup but often suffering from larger accuracy drops.

Low-Rank Decomposition for LLMs. Singular Value Decomposition (SVD) approximates weight matrices with low-rank factors, reducing both storage and computation. ASVD (Yuan et al., 2025b) incorporates activation statistics for importance-aware decomposition. SVD-LLM (Wang et al., 2025) introduces truncation-aware data whitening and hierarchical updates, achieving state-of-the-art performance among SVD-based methods. However, applying these techniques directly to MoE models ignores the structural properties of expert weights.

Expert Pruning. Expert pruning reduces memory by removing entire experts or portions of expert weights. NAEE (Lu et al., 2024) identifies non-redundant experts and trims unimportant weight connections. MoE-Pruner (Xie et al., 2024) leverages router information to guide pruning decisions. However, complete expert removal inevitably causes significant information loss, particularly for experts that specialize in rare but important patterns.

Expert Merging. Expert merging consolidates multiple experts into fewer representatives. Li et al. (2024) first merges experts based on routing policy similarity, then applies standard compression. EEP (Liu et al., 2024b) introduces a two-stage pipeline combining pruning with merging. These methods reduce the number of distinct expert weights but rely on heuristic similarity metrics without theoretical guarantees on information preservation.

Delta Decomposition. Delta decomposition factorizes expert weights into shared base weights plus expert-specific residuals. D^2-MoE (Gu et al., 2025) constructs base weights using Fisher-weighted averaging and compresses delta weights via truncated SVD. MoE-I^2 (Yang et al., 2024b) combines inter-expert pruning with intra-expert low-rank decomposition.

APPENDIX A.2: INFORMATION LOSS

A.2.1 EXPERIMENTAL SETUP

Model and Dataset. We conduct experiments on Mixtral-8×7B (Jiang et al., 2024), a sparse mixture-of-experts model with 8 experts per layer and 32 MoE layers. For calibration data, we use 1,024 randomly sampled sequences from the C4 dataset (Raffel et al., 2020), each with a maximum length of 2,048 tokens.

Merging Strategies. We evaluate six representative merging strategies for constructing shared base weights $\mathbf{W}_b$ from expert weights $\{\mathbf{W}_i\}_{i=1}^{8}$:

- **Simple Average**: $\mathbf{W}_b = \frac{1}{N}\sum_{i=1}^{N}\mathbf{W}_i$
- **Fisher Weighted** (Matena & Raffel, 2022): $\mathbf{W}_b = \sum_{i=1}^{N}\frac{F_i}{\sum_j F_j}\mathbf{W}_i$, where F_i is the Fisher information matrix diagonal
- **Frequency Weighted**: $\mathbf{W}_b = \sum_{i=1}^{N} f_i\mathbf{W}_i$, where f_i is the routing frequency of expert i
- **TIES** (Yadav et al., 2023): Trim-Elect-Sign merging with conflict resolution
- **RegMean** (Jin et al., 2023): Regression-based merging

- **Task Arithmetic** (Ilharco et al., 2023): $\mathbf{W}_b = \mathbf{W}_{\text{init}} + \sum_{i=1}^{N} \lambda_i (\mathbf{W}_i - \mathbf{W}_{\text{init}})$

Mutual Information Estimation. For each of the 32 MoE layers, we measure the MI between layer inputs $\mathbf{X} \in \mathbb{R}^{B \times d}$ (where B is batch size and d is hidden dimension) and outputs $\mathbf{Y} \in \mathbb{R}^{B \times d}$ using the MINE estimator (Belghazi et al., 2018):

$$I(\mathbf{X}; \mathbf{Y}) \geq \sup_{T_\theta} \mathbb{E}_{p(\mathbf{x},\mathbf{y})}[T_\theta(\mathbf{x}, \mathbf{y})] - \log \mathbb{E}_{p(\mathbf{x})p(\mathbf{y})}[e^{T_\theta(\mathbf{x},\mathbf{y})}], \tag{20}$$

where T_θ is a neural network with two hidden layers of 256 units, trained for 1,000 iterations with Adam optimizer (lr=10^{-4}).

The MI retention ratio is defined as:

$$\text{MI Retention} = \frac{I(\mathbf{X}; \hat{\mathbf{Y}})}{I(\mathbf{X}; \mathbf{Y})} \times 100\%, \tag{21}$$

where $\hat{\mathbf{Y}}$ is the reconstructed output using merged base weights and delta reconstruction.

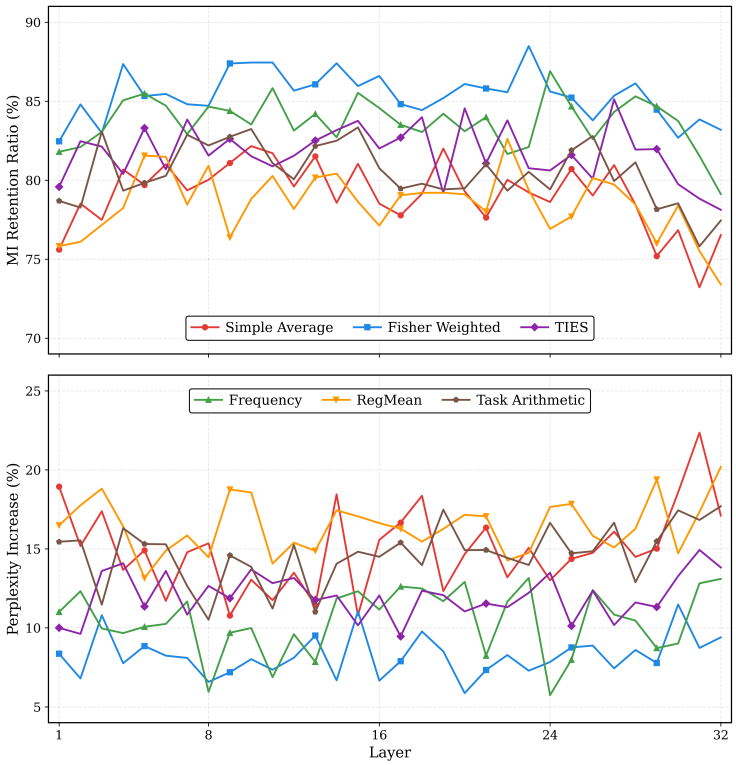


Figure 5: Layer-wise mutual information (MI) retention ratio (%, top) and perplexity increase (%, bottom) for six merging strategies on Mixtral-8×7B across 32 MoE layers. Fisher Weighted achieves the best MI retention (85.4% avg) and lowest perplexity increase (8.2% avg), while RegMean performs worst (78.5% MI retention, 16.4% PPL increase). The strong negative correlation ($r = -0.89$, $p < 0.001$) between MI retention and perplexity increase supports MI as an optimization objective for base weight construction.

Perplexity Evaluation. We evaluate perplexity on WikiText-2 (Merity et al., 2017) test set using standard sliding window approach with stride 512. Perplexity increase is computed as:

$$\text{PPL Increase} = \frac{\text{PPL}_{\text{compressed}} - \text{PPL}_{\text{original}}}{\text{PPL}_{\text{original}}} \times 100\% \tag{22}$$

A.2.2 FIGURE DESCRIPTION

Figure 5 presents a two-panel visualization of information preservation across 32 layers of Mixtral-8×7B. The top panel illustrates MI retention ratio while the bottom panel shows the corresponding perplexity increase, enabling direct comparison of the trade-off between information preservation and model performance degradation.

Top Panel: MI Retention Ratio. The top panel displays MI retention ratio (%) on the Y-axis against layer index (1–32) on the X-axis. Six colored lines represent different merging strategies:

- **Fisher Weighted** (blue squares): Best performer, averaging 85.4% retention
- **Frequency** (green triangles): 83.7% average
- **TIES** (purple diamonds): 81.8% average
- **Task Arithmetic** (brown pentagons): 80.5% average
- **Simple Average** (red circles): 79.1% average
- **RegMean** (orange inverted triangles): Worst at 78.5%

All methods exhibit layer-dependent variation, with MI retention fluctuating across the 32 layers due to varying expert specialization patterns.

Bottom Panel: Perplexity Increase. The bottom panel shows perplexity increase (%) using identical X-axis and color coding. The inverse relationship with MI retention is visually apparent: methods with lower MI retention exhibit higher perplexity increase. Fisher Weighted achieves the lowest average PPL increase (8.2%), while RegMean shows the highest (16.4%).

Correlation Analysis. Computing Pearson correlation across all 192 data points (6 methods × 32 layers), we obtain $r = -0.89$ ($p < 0.001$), confirming strong negative correlation between MI retention and perplexity increase. The Pearson correlation coefficient is computed as:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \cdot \sum_{i=1}^{n}(y_i - \bar{y})^2}}, \tag{23}$$

where x_i denotes the MI retention ratio and y_i denotes the perplexity increase for the i-th data point, with $n = 192$ total observations. Per-method correlations range from $r = -0.52$ (Fisher Weighted) to $r = -0.80$ (Simple Average).

Detailed Data. Table 7 presents complete experimental results. We report the average MI retention ratio, average perplexity increase, and per-method Pearson correlation coefficient for each merging strategy evaluated on Mixtral-8×7B.

Table 7: MI retention and perplexity increase for six merging strategies.

Method	Avg MI Ret.	Avg PPL Inc.	Corr. (r)
Fisher Weighted	85.4%	8.2%	−0.52
Frequency	83.7%	10.4%	−0.66
TIES	81.8%	12.0%	−0.56
Task Arithmetic	80.5%	14.7%	−0.66
Simple Average	79.1%	15.0%	−0.80
RegMean	78.5%	16.4%	−0.78
Overall	–	–	**−0.89**

A.2.3 KEY OBSERVATIONS

The analysis yields the following findings, which motivate MI-constrained base construction:

- **Universal Information Loss**: All merging strategies lose 14.6% to 21.5% of MI on average, with no method achieving consistent >90% retention across all layers.

- **MI-Performance Link**: The strong negative correlation ($r = -0.89$) links MI retention to perplexity, which makes MI a suitable objective for base weight construction.
- **Fisher Superiority**: Fisher-weighted merging outperforms alternatives by 2–7% in MI retention, but still lacks principled information-preservation criteria.
- **Method Ranking**: Fisher > Frequency > TIES > Task Arithmetic > Simple Average > RegMean in terms of MI retention.

APPENDIX A.3: EXPERT SENSITIVITY

A.3.1 EXPERIMENTAL SETUP

Model Configuration. We analyze Mixtral-8×7B with 8 experts per MoE layer. Each expert consists of three weight matrices: gate projection ($\mathbf{W}_{\text{gate}} \in \mathbb{R}^{14336\times4096}$), up projection ($\mathbf{W}_{\text{up}} \in \mathbb{R}^{14336\times4096}$), and down projection ($\mathbf{W}_{\text{down}} \in \mathbb{R}^{4096\times14336}$).

SVD Compression Protocol. For each expert i, we apply truncated SVD to the delta weights $\Delta\mathbf{W}_i = \mathbf{W}_i - \mathbf{W}_b$:

$$\Delta\mathbf{W}_i \approx \mathbf{U}_i\mathbf{\Sigma}_i\mathbf{V}_i^\top, \tag{24}$$

where $\mathbf{U}_i$ and $\mathbf{V}_i$ are orthogonal matrices, and $\mathbf{\Sigma}_i$ is a diagonal matrix containing the top-r singular values. We sweep rank ratios from 10% to 70% in 5% increments, yielding 13 compression levels per expert.

Metric Definitions. We define four metrics to characterize expert compression behavior:

- **Sensitivity Score**: The ratio of accuracy degradation to reconstruction error, measuring how vulnerable an expert is to compression-induced errors:

$$\text{Sensitivity}_i = \left|\frac{\Delta\text{Acc}_i}{\Delta\text{Error}_i}\right| = \left|\frac{\text{Acc}_{\text{orig}} - \text{Acc}_{\text{comp}}}{\|\Delta\mathbf{W}_i - \mathbf{U}_i\mathbf{\Sigma}_i\mathbf{V}_i^\top\|_F/\|\Delta\mathbf{W}_i\|_F}\right|. \tag{25}$$

  Higher sensitivity indicates greater vulnerability to compression.

- **Speedup**: Wall-clock inference time ratio measured on 1,000 prompts (batch size 1, sequence length 512) on NVIDIA A100 GPU:

$$\text{Speedup}_i = \frac{T_{\text{original}}}{T_{\text{compressed}}}. \tag{26}$$

- **Optimal Rank**: The minimum rank ratio required to achieve <1% accuracy degradation on 5-shot MMLU (Hendrycks et al., 2021):

$$r_i^* = \min\{r : \text{Acc}_{\text{orig}} - \text{Acc}_{\text{comp}}(r) < 1\%\}. \tag{27}$$

- **Error Tolerance**: The maximum relative reconstruction error (measured on delta weights) that maintains <1% accuracy degradation:

$$\epsilon_i^{\text{tol}} = \max\{e : \text{Acc}_{\text{orig}} - \text{Acc}_{\text{comp}}(e) < 1\%\}, \tag{28}$$

  where $e = \|\Delta\mathbf{W}_i - \mathbf{U}_i\mathbf{\Sigma}_i\mathbf{V}_i^\top\|_F/\|\Delta\mathbf{W}_i\|_F$ denotes the relative reconstruction error on delta weights. Note that ϵ_i^{tol} represents the per-expert tolerance threshold, which informs the global constraint ϵ in Eq. (1). This metric is related to optimal rank through the singular value distribution: experts with faster spectral decay (stronger low-rank structure) exhibit higher error tolerance at the same rank ratio.

Evaluation Protocol. For each expert, we compress the expert at 13 rank ratios (10%–70%), measure the reconstruction error $\|\Delta\mathbf{W} - \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top\|_F/\|\Delta\mathbf{W}\|_F$, evaluate 5-shot MMLU accuracy across 57 subjects, measure the inference speedup on an A100 GPU, and fit a sensitivity curve from which we extract the optimal rank and the error-tolerance threshold.

Table 8: Expert-wise compression analysis on Mixtral-8×7B.

Expert	Sensitivity	Speedup	Optimal Rank	Error Tolerance
E1	0.85	1.40×	35%	9%
E2	1.12	1.28×	45%	5%
E3	0.71	1.52×	28%	12%
E4	0.92	1.36×	40%	7%
E5	1.38	1.21×	55%	3%
E6	0.76	1.48×	30%	11%
E7	0.89	1.38×	38%	8%
E8	0.95	1.33×	42%	6%
Mean	0.95	1.37×	39%	7.6%
Std	0.21	0.10	8.5%	2.9%

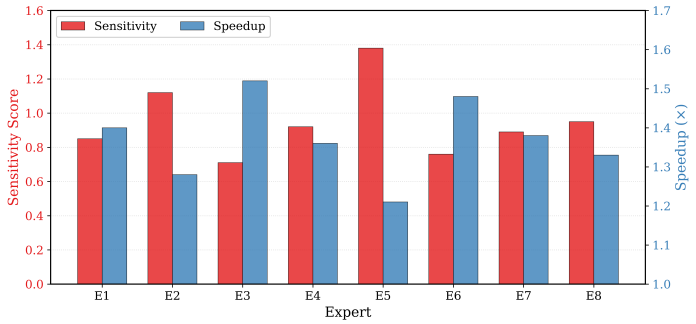


Figure 6: Expert-wise sensitivity (red bars, left Y-axis) and speedup (blue bars, right Y-axis) on Mixtral-8×7B.

A.3.2 Results

Table 1 in the main text presents the per-expert values; Table 8 reproduces them with the mean and standard deviation across all eight experts to quantify the heterogeneity in compression behavior.

Figure 6 provides a visual representation of the sensitivity-speedup relationship across all eight experts. Sensitivity and speedup move in opposite directions: the most sensitive expert (E5) gains the least speed and the least sensitive expert (E3) gains the most.

Left Y-Axis: Sensitivity Score (Red Bars). Red bars represent sensitivity scores on a scale from 0 to 1.6. E5 has the highest sensitivity (1.38) and is the most vulnerable to compression, while E3 has the lowest (0.71) and tolerates aggressive compression well; the $\sim 2\times$ variation ($1.38/0.71 \approx 1.94$) shows that experts respond very differently to the same reconstruction error.

Right Y-Axis: Speedup (Blue Bars). Blue bars show inference speedup ratios on a scale from $1.0\times$ to $1.7\times$. E3 achieves the highest speedup ($1.52\times$) with the lowest sensitivity, E5 the lowest speedup ($1.21\times$) with the highest sensitivity, and E2 combines high sensitivity (1.12) with a moderate speedup ($1.28\times$).

A.3.3 Statistical Analysis

Sensitivity-Speedup Relationship. Across experts, sensitivity and speedup move in opposite directions: E5 is the most sensitive expert and gains the least speed ($1.21\times$), while E3 is the least sensitive and gains the most ($1.52\times$). This motivates bounding reconstruction error and computational overhead explicitly and jointly.

Optimal Rank and Error Tolerance. The optimal rank varies from 28% (E3) to 55% (E5), a $\sim 2\times$ difference. Error tolerance represents the maximum reconstruction error (on delta weights) that maintains $<1\%$ accuracy degradation. E3 tolerates up to 12% reconstruction error, while E5 tolerates

only 3% error. This 4× difference (12%/3%) demonstrates why heuristic energy-based thresholds fail to provide reliable quality control: the same compression strategy produces acceptable results for some experts but unacceptable degradation for others.

This motivates explicit constraint enforcement: PRIMO bounds the average reconstruction error ($\mathcal{L} \leq \epsilon$) and the computational overhead ($\mathcal{R} \leq \gamma$) directly.

A.3.4 Key Observations

The expert-wise analysis gives three observations that motivate explicit constraint enforcement:

- **Heterogeneous Error Tolerance**: The tolerated reconstruction error varies from 3% (E5) to 12% (E3), a 4× spread.
- **Heterogeneous Speedup**: Inference speedup ranges from 1.21× (E5) to 1.52× (E3), so latency requirements cannot be met reliably without an explicit complexity bound.
- **Need for Explicit Bounds**: These findings motivate constrained optimization approaches that explicitly bound reconstruction error ($\mathcal{L} \leq \epsilon$) and computational overhead ($\mathcal{R} \leq \gamma$), rather than relying on heuristic energy retention thresholds.

APPENDIX A.4: Distribution Shift

A.4.1 Experimental Setup

Compression Configuration. We compress Mixtral-8×7B at 50% overall compression ratio using D^2-MoE (Gu et al., 2025) with Fisher-weighted base construction. The model is calibrated on 1,024 sequences from WikiText-2 (Merity et al., 2017) training set.

Evaluation Tasks. We evaluate on 8 downstream tasks spanning 4 domains:

- **Language Modeling**: WikiText-2 (Merity et al., 2017), PTB (Marcus et al., 1993), C4 (Raffel et al., 2020)
- **Reasoning**: MMLU (Hendrycks et al., 2021), OpenbookQA (Mihaylov et al., 2018), HellaSwag (Zellers et al., 2019)
- **Math**: GSM8K (Cobbe et al., 2021)
- **Code**: HumanEval (Chen et al., 2021)

Routing Distribution Analysis. For each task, we collect routing decisions from 1,000 evaluation samples. Let $p_{\text{cal}}(e|l)$ denote the routing distribution over experts e at layer l on calibration data, and $p_{\text{task}}(e|l)$ the distribution on task data. To ensure numerical stability, we apply additive smoothing to p_{cal} with $\epsilon_{\text{smooth}} = 10^{-6}$ (see Appendix A.10.1 for details). We then compute KL divergence:

$$D_{\text{KL}}^{(l)} = \sum_{e=1}^{8} p_{\text{task}}(e|l) \log \frac{p_{\text{task}}(e|l)}{\tilde{p}_{\text{cal}}(e|l)}. \tag{29}$$

The reported routing KL divergence is averaged across all 32 layers:

$$\text{Routing KL} = \frac{1}{32} \sum_{l=1}^{32} D_{\text{KL}}^{(l)}. \tag{30}$$

Performance Gap Metrics. We define two gap metrics to quantify the impact of distribution shift on compression quality. The **Static Gap** measures the performance degradation when using a fixed compression configuration determined at calibration time, which is the standard approach in prior work (Frantar & Alistarh, 2023; Sun et al., 2024). The **Oracle Gap** represents an upper bound on achievable performance, obtained by re-calibrating the compression on task-specific data (Hubara et al., 2021). Formally:

- **Static Gap**: $\text{Gap}_{\text{static}} = \text{Perf}_{\text{original}} - \text{Perf}_{\text{static-compressed}}$

- **Oracle Gap**: $\text{Gap}_{\text{oracle}} = \text{Perf}_{\text{original}} - \text{Perf}_{\text{oracle-compressed}}$

The difference between these two gaps quantifies the "recoverable gap"—the performance that could theoretically be recovered through task-adaptive compression.

A.4.2 Detailed Data

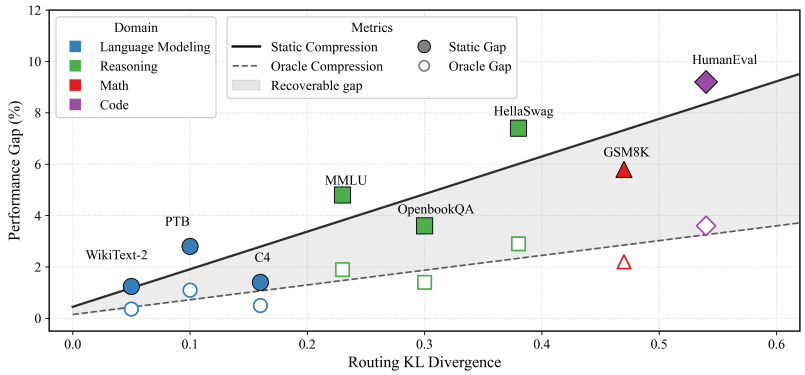


Figure 7: Relationship between routing distribution shift (KL divergence) and compression-induced performance degradation across 8 downstream tasks.

Figure 7 presents a scatter plot analyzing the relationship between routing distribution shift and compression-induced performance degradation. The visualization reveals a strong positive correlation between KL divergence and performance gap, while also highlighting the substantial room for improvement through adaptive compression strategies.

Axes and Data Points. The X-axis represents routing KL divergence ranging from 0 to 0.6, quantifying how much the expert selection distribution on each task deviates from the calibration distribution. The Y-axis shows the performance gap in percentage (0–12% range), measuring the accuracy or perplexity degradation relative to the uncompressed model. Filled markers indicate static compression gap using fixed calibration-time configurations, while empty markers of the same shape and color represent oracle compression gap achieved through task-specific re-calibration.

Domain Encoding. Tasks are grouped by domain using distinct marker shapes and colors to reveal domain-specific patterns. Language modeling tasks (WikiText-2, PTB, C4) are shown as blue circles, reasoning tasks (MMLU, OpenbookQA, HellaSwag) as green squares, math (GSM8K) as red triangles, and code (HumanEval) as purple diamonds. This encoding clearly shows that in-domain tasks (language modeling) cluster at low KL divergence, while out-of-domain specialized tasks exhibit progressively higher distribution shift.

Trend Lines and Shaded Region. The solid black line represents the linear regression fit for static compression, while the dashed gray line shows the corresponding fit for oracle compression. The gray shaded area between these two lines represents the "recoverable gap", which is the performance improvement achievable by adapting compression configurations to the target task distribution rather than relying on fixed calibration-time settings.

Table 9 presents the complete numerical results for all 8 evaluation tasks, including the routing KL divergence from calibration distribution, static compression gap, and oracle compression gap. These values correspond to the data points visualized in Figure 7 and form the basis for our statistical analysis.

A.4.3 Statistical Analysis

Correlation Coefficients. We compute Pearson correlation coefficients between routing KL divergence and both gap metrics. Both correlations are strongly positive and exceed the critical value $r_{\text{critical}} = 0.707$ for $n = 8$ samples at $\alpha = 0.05$, confirming statistical significance.

Table 9: Distribution shift analysis across 8 downstream tasks for D^2-MoE at 50% compression with WikiText-2 calibration (Appendix A.4.1).

Task	Domain	KL Div.	Static Gap	Oracle Gap
WikiText-2	LM	0.05	1.2%	0.4%
PTB	LM	0.10	2.8%	1.1%
C4	LM	0.16	1.4%	0.5%
MMLU	Reasoning	0.23	4.8%	1.9%
OpenbookQA	Reasoning	0.30	3.6%	1.4%
HellaSwag	Reasoning	0.38	7.4%	2.9%
GSM8K	Math	0.47	5.8%	2.2%
HumanEval	Code	0.54	9.2%	3.6%

Linear Regression Models. We fit linear regression models of each gap on routing KL divergence. The static-gap slope is about 2.5 times the oracle-gap slope, which indicates that static compression is 2.5× more sensitive to distribution shift than oracle compression, highlighting the potential benefits of adaptive approaches.

Non-Monotonic Patterns. Higher KL divergence does not always correspond to a higher performance gap. Specifically, C4 (KL=0.16) shows a lower gap than PTB (KL=0.10) with 1.4% vs. 2.8%, OpenbookQA (KL=0.30) shows a lower gap than MMLU (KL=0.23) with 3.6% vs. 4.8%, and GSM8K (KL=0.47) shows a lower gap than HellaSwag (KL=0.38) with 5.8% vs. 7.4%.

A.4.4 Key Observations

The analysis of distribution shift yields the following findings, which motivate adaptive compression:

1. **Strong KL-Gap Correlation**: The correlation between routing KL and the static gap indicates that routing distribution shift predicts compression-induced performance degradation. This correlation exceeds standard significance thresholds for $n = 8$ samples ($r_{\text{critical}} = 0.707$ at $\alpha = 0.05$).
2. **Domain-Dependent Shift**: Language modeling tasks cluster at low KL (0.05–0.16), while specialized tasks (Math, Code) exhibit high KL (0.47–0.54). This reflects the similarity between WikiText-2 calibration data and LM evaluation tasks.
3. **Substantial Recoverable Gap**: Oracle compression recovers 60–67% of the static gap across tasks, demonstrating significant headroom for input-adaptive compression. For example, HumanEval's gap reduces from 9.2% to 3.6% (61% recovery) when using task-specific calibration.
4. **Need for Dynamic Adaptation**: The 7.7× gap difference between WikiText-2 (1.2%) and HumanEval (9.2%) motivates compression strategies that adapt to input distributions at inference time rather than relying on fixed calibration-time configurations.

APPENDIX A.5: Constraint Analysis

This appendix clarifies the physical interpretation of the computational complexity constraint in the problem formulation (Eq. (1)).

A.5.1 Complexity Constraint Interpretation

For an expert weight matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$, the original representation requires mn parameters. Under low-rank factorization $\mathbf{W} \approx \mathbf{U}\mathbf{V}^\top$ with $\mathbf{U} \in \mathbb{R}^{m\times r}$ and $\mathbf{V} \in \mathbb{R}^{n\times r}$, the compressed representation requires $mr + nr = r(m + n)$ parameters. Therefore, the ratio:

$$\mathcal{R}(\Theta) = \frac{r(m+n)}{mn} \tag{31}$$

represents the fraction of parameters retained after low-rank approximation.

This ratio simultaneously captures the FLOPs ratio for expert weight computations (Denton et al., 2014): the original computation $\mathbf{W}\mathbf{x}$ requires $2mn$ FLOPs, while the factorized computation $\mathbf{U}(\mathbf{V}^\top\mathbf{x})$ requires $2r(m+n)$ FLOPs. Since expert FFN computations constitute the dominant cost in MoE inference, the constraint parameter γ directly specifies the target computational overhead as a fraction of the original.

A.5.2 Uniform Rank Assumption

Throughout this paper, we employ a uniform rank r across all layers and experts in the parameterization of Eq. (1). While Table 1 demonstrates that different experts exhibit varying sensitivities and optimal rank ratios (ranging from 28% to 55%), we adopt uniform rank for the following reasons:

1. **Tractability**: Uniform rank gives one interpretable bound on computational overhead and keeps the rank search one-dimensional. Per-expert ranks would expand the search from $O(R)$ to $O(R^N)$ rank assignments per layer for R candidate ranks (with $N = 8$ experts and $R = 10$, 10^8 assignments against 10).
2. **Per-expert fitting**: Within the shared rank, each expert's factors are fitted to its own activation-weighted delta, so the content of the low-rank factors adapts to each expert.

The global error constraint $\mathcal{L}(\Theta) \leq \epsilon$ in Eq. (1) ensures that the average reconstruction error across all experts remains bounded, even with varying individual expert sensitivities.

APPENDIX A.6: Background

This appendix provides essential background on MoE architectures, delta decomposition, the D^2-MoE baseline, and mutual information fundamentals.

A.6.1 Mixture-of-Experts Architecture

In Transformer-based LLMs, MoE layers replace standard FFN with N parallel expert networks and a gating router (Shazeer et al., 2017). For input $\mathbf{x} \in \mathbb{R}^d$, the output is:

$$\text{MoE}(\mathbf{x}) = \sum_{i=1}^{N} g_i(\mathbf{x}) \cdot E_i(\mathbf{x}), \tag{32}$$

where only top-k experts are activated via sparse gating: $g_i(\mathbf{x}) = \text{Softmax}(\mathbf{W}_g\mathbf{x})_i$ if $i \in$ Top-k, else 0.

Each expert E_i employs the SwiGLU activation (Shazeer, 2020):

$$E_i(\mathbf{x}) = \mathbf{W}^{(i)}_{\text{down}} \left(\sigma(\mathbf{W}^{(i)}_{\text{gate}}\mathbf{x}) \odot (\mathbf{W}^{(i)}_{\text{up}}\mathbf{x}) \right), \tag{33}$$

where $\mathbf{W}^{(i)}_{\text{gate}}, \mathbf{W}^{(i)}_{\text{up}} \in \mathbb{R}^{d_{\text{ff}} \times d}$ and $\mathbf{W}^{(i)}_{\text{down}} \in \mathbb{R}^{d \times d_{\text{ff}}}$. Expert weights dominate model size: in Mixtral-8×7B, they constitute 97% of total parameters (45.1B out of 46.7B).

A.6.2 Delta Decomposition

Delta decomposition exploits redundancy among expert weights by factorizing each $\mathbf{W}_i \in \mathbb{R}^{m \times n}$ as:

$$\mathbf{W}_i = \mathbf{W}_b + \Delta\mathbf{W}_i, \tag{34}$$

where $\mathbf{W}_b$ captures shared knowledge across experts and $\Delta\mathbf{W}_i$ encodes expert-specific behavior.

A key empirical finding is that delta weights exhibit strong low-rank structure (Gu et al., 2025), enabling compression via $\Delta\mathbf{W}_i \approx \mathbf{U}_i\mathbf{V}_i^\top$ with $r \ll \min(m, n)$. The compression ratio is:

$$\text{Ratio} = 1 - \frac{mn + N \times r(m+n)}{N \times mn}. \tag{35}$$

A.6.3 D^2-MoE Baseline

D^2-MoE (Gu et al., 2025) is a representative delta decomposition method with three components:

Fisher-Weighted Merging. The base weight is constructed as $\mathbf{W}_b = \sum_i \alpha_i \mathbf{W}_i$, where $\alpha_i \propto \text{tr}(\mathbf{F}_i)$ and $\mathbf{F}_i$ is the Fisher information matrix.

Truncated SVD. Delta weights are compressed via SVD with energy retention threshold τ:

$$r^* = \min \left\{ r : \frac{\sum_{j=1}^{r} \sigma_j^2}{\sum_j \sigma_j^2} \geq \tau \right\}. \tag{36}$$

Semi-Dynamic Pruning. Base weight columns are pruned in two phases: static pruning removes consistently inactive columns; dynamic pruning adapts to current input activations.

A.6.4 Mutual Information Preliminaries

Mutual information (MI) quantifies statistical dependence between random variables (Cover et al., 1991):

$$I(X;Y) = H(X) - H(X|Y) = D_{\text{KL}}(p(X,Y) \| p(X)p(Y)), \tag{37}$$

where $H(\cdot)$ denotes entropy. Key properties include:

- **Data Processing Inequality**: For Markov chain $X \to Y \to Z$: $I(X;Z) \leq I(X;Y)$.
- **Gaussian Case**: For $Y = \mathbf{W}X + \xi$ with $\xi \sim \mathcal{N}(0, \Sigma_\xi)$:

$$I(X;Y) = \frac{1}{2} \log \det \left(\mathbf{I} + \Sigma_\xi^{-1} \mathbf{W} \Sigma_X \mathbf{W}^\top \right). \tag{38}$$

For compression, the MI-KL connection is central: $\mathbb{E}[D_{\text{KL}}(p(Y|X) \| p(\hat{Y}|X))]$ enters the lower bound on preserved MI (Eq. (3)). In practice, MI is estimated via the MINE neural estimator (Belghazi et al., 2018) or closed-form Gaussian expressions.

APPENDIX A.7: MI-Constrained Decomposition

This appendix provides detailed derivations for the MI-constrained decomposition framework, including the quadratic program formulation, the water-filling solution, and the information-preservation bound under stated assumptions.

A.7.1 Derivation of Quadratic Program

We derive the quadratic program in Eq. (8) from the KL divergence minimization objective under the Gaussian model assumption. Note that this derivation applies independently to each layer l; we omit the layer superscript (l) for notational simplicity.

Notation. Throughout this appendix, $\text{tr}(\cdot)$ denotes the matrix trace (sum of diagonal elements), and $\| \cdot \|_F$ denotes the Frobenius norm.

Gaussian Model Setup. Under the Gaussian model $Y = \mathbf{W}X + \xi$ with $\xi \sim \mathcal{N}(0, \Sigma_\xi)$, the conditional distribution is $p(Y|X) = \mathcal{N}(\mathbf{W}X, \Sigma_\xi)$. Similarly, the reconstructed output has distribution $p(\hat{Y}|X) = \mathcal{N}(\hat{\mathbf{W}}X, \Sigma_\xi)$.

KL Divergence Between Gaussians. The KL divergence between two Gaussians with the same covariance is:

$$D_{\text{KL}}(p(Y|X) \| p(\hat{Y}|X)) = \frac{1}{2} ((\mathbf{W} - \hat{\mathbf{W}})X)^\top \Sigma_\xi^{-1} ((\mathbf{W} - \hat{\mathbf{W}})X). \tag{39}$$

Taking expectation over $X \sim \mathcal{N}(0, \Sigma_X)$:

$$\mathbb{E}_X[D_{\text{KL}}] = \frac{1}{2} \text{tr} \left(\Sigma_\xi^{-1} (\mathbf{W} - \hat{\mathbf{W}}) \Sigma_X (\mathbf{W} - \hat{\mathbf{W}})^\top \right). \tag{40}$$

Expansion with Mixing Coefficients. For expert i with weight $\mathbf{W}_i$ and reconstructed weight $\hat{\mathbf{W}}_i = \mathbf{W}_b$, substituting $\mathbf{W}_b = \sum_j \alpha_j \mathbf{W}_j$:

$$\mathbb{E}_X[D_{\text{KL}}^{(i)}] = \frac{1}{2}\text{tr}\left(\Sigma_\xi^{-1}\left(\mathbf{W}_i - \sum_j \alpha_j \mathbf{W}_j\right)\Sigma_X\left(\mathbf{W}_i - \sum_k \alpha_k \mathbf{W}_k\right)^\top\right). \tag{41}$$

Taking expectation over the routing distribution $\{p_i\}_{i=1}^N$:

$$\mathbb{E}_{i\sim p}[\mathbb{E}_X[D_{\text{KL}}]] = \sum_{i=1}^N p_i \cdot \frac{1}{2}\text{tr}\left(\Sigma_\xi^{-1}\left(\mathbf{W}_i - \sum_j \alpha_j \mathbf{W}_j\right)\Sigma_X\left(\mathbf{W}_i - \sum_k \alpha_k \mathbf{W}_k\right)^\top\right). \tag{42}$$

Term-by-Term Expansion. To clarify how each term depends on the routing distribution, we expand $\mathbf{D}_i\Sigma_X\mathbf{D}_i^\top$ where $\mathbf{D}_i = \mathbf{W}_i - \sum_j \alpha_j \mathbf{W}_j$:

$$\mathbf{D}_i\Sigma_X\mathbf{D}_i^\top = \underbrace{\mathbf{W}_i\Sigma_X\mathbf{W}_i^\top}_{\text{(I)}} \underbrace{- \sum_j \alpha_j(\mathbf{W}_i\Sigma_X\mathbf{W}_j^\top + \mathbf{W}_j\Sigma_X\mathbf{W}_i^\top)}_{\text{(II)}} + \underbrace{\sum_j\sum_k \alpha_j\alpha_k\mathbf{W}_j\Sigma_X\mathbf{W}_k^\top}_{\text{(III)}}. \tag{43}$$

Applying $\sum_{i=1}^N p_i \cdot \frac{1}{2}\text{tr}(\Sigma_\xi^{-1}\cdot)$ to each term:

- **Term (I)** contains $\mathbf{W}_i$, contributing the constant $c = \sum_{i=1}^N p_i \cdot \frac{1}{2}\text{tr}(\Sigma_\xi^{-1}\mathbf{W}_i\Sigma_X\mathbf{W}_i^\top)$.
- **Term (II)** contains $\mathbf{W}_i$ in the cross product, contributing $\boldsymbol{\alpha}^\top\mathbf{b}$ where $b_j = \sum_{i=1}^N p_i \cdot \text{tr}(\Sigma_\xi^{-1}\mathbf{W}_i\Sigma_X\mathbf{W}_j^\top)$.
- **Term (III)** involves only $\mathbf{W}_j$ and $\mathbf{W}_k$ with no dependence on i. Since the summand $\text{tr}(\Sigma_\xi^{-1}\mathbf{W}_j\Sigma_X\mathbf{W}_k^\top)$ is constant with respect to i, we can factor out $\sum_{i=1}^N p_i = 1$:

$$\sum_{i=1}^N p_i \sum_j\sum_k \alpha_j\alpha_k \frac{1}{2}\text{tr}(\Sigma_\xi^{-1}\mathbf{W}_j\Sigma_X\mathbf{W}_k^\top) = \frac{1}{2}\boldsymbol{\alpha}^\top\mathbf{Q}\boldsymbol{\alpha}, \tag{44}$$

  where $Q_{jk} = \text{tr}(\Sigma_\xi^{-1}\mathbf{W}_j\Sigma_X\mathbf{W}_k^\top)$ is independent of $\{p_i\}$.

This analysis reveals the fundamental asymmetry: Q_{jk} loses its dependence on routing because $\mathbf{W}_i$ does not appear in Term (III), while b_j and c retain their dependence through Terms (I) and (II).

Quadratic Form. Combining the three terms from Eq. (43) yields the quadratic form:

$$\mathbb{E}_{i\sim p}[D_{\text{KL}}] = \frac{1}{2}\boldsymbol{\alpha}^\top\mathbf{Q}\boldsymbol{\alpha} - \boldsymbol{\alpha}^\top\mathbf{b} + c, \tag{45}$$

where, as derived above:

$$Q_{jk} = \text{tr}(\Sigma_\xi^{-1}\mathbf{W}_j\Sigma_X\mathbf{W}_k^\top), \tag{46}$$

$$b_j = \sum_{i=1}^N p_i \cdot \text{tr}(\Sigma_\xi^{-1}\mathbf{W}_i\Sigma_X\mathbf{W}_j^\top), \tag{47}$$

$$c = \sum_{i=1}^N p_i \cdot \frac{1}{2}\text{tr}(\Sigma_\xi^{-1}\mathbf{W}_i\Sigma_X\mathbf{W}_i^\top). \tag{48}$$

Note that Q_{jk} no longer contains $\sum_i p_i$ because this factor equals 1 as shown in Eq. (44), while b_j and c retain their routing dependence.

The matrix $\mathbf{Q}$ is positive semi-definite as it can be expressed as a Gram matrix. Define $\mathbf{w}_j = \text{vec}(\Sigma_\xi^{-1/2}\mathbf{W}_j\Sigma_X^{1/2})$, then $Q_{jk} = \mathbf{w}_j^\top\mathbf{w}_k$, confirming positive semi-definiteness. Q_{jk} measures how much experts j and k agree in their input-output mappings, weighted by the input covariance Σ_X and the metric Σ_ξ^{-1}.

A.7.2 Quadratic Program Solution

This section derives the solution to the QP in Eq. (8) and provides a water-filling interpretation that offers geometric intuition for the optimal mixing coefficients.

KKT Conditions. The Lagrangian for the quadratic program with simplex constraint is:

$$\mathcal{L}(\boldsymbol{\alpha}, \nu, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{Q}\boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \mathbf{b} + c + \nu \left(\sum_{j=1}^{N} \alpha_j - 1 \right) - \sum_{j=1}^{N} \lambda_j \alpha_j, \tag{49}$$

where $\nu \in \mathbb{R}$ enforces the equality constraint and $\lambda_j \geq 0$ enforce non-negativity. The Karush-Kuhn-Tucker (KKT) conditions (Boyd & Vandenberghe, 2004) require stationarity ($[\mathbf{Q}\boldsymbol{\alpha}]_j - b_j + \nu - \lambda_j = 0$), primal feasibility ($\sum_j \alpha_j = 1$, $\alpha_j \geq 0$), dual feasibility ($\lambda_j \geq 0$), and complementary slackness ($\lambda_j \alpha_j = 0$).

General Solution. For nonsingular $\mathbf{Q}$ with all $\alpha_j > 0$, complementary slackness gives $\lambda_j = 0$ for all j, and stationarity gives:

$$[\mathbf{Q}\boldsymbol{\alpha}]_j = b_j - \nu \quad \Rightarrow \quad \alpha_j = [\mathbf{Q}^{-1}\mathbf{b}]_j - \nu[\mathbf{Q}^{-1}\mathbf{1}]_j. \tag{50}$$

Practical Implementation. The water-filling interpretation below provides geometric intuition for understanding the optimal solution structure. For practical computation, we solve the QP in Eq. (8) using OSQP (Stellato et al., 2020), a standard convex optimization solver that returns the optimum to solver tolerance.

Water-Filling Interpretation. The optimal solution admits a water-filling interpretation analogous to the classical power allocation problem in information theory (Cover et al., 1991). In the classical setting, given N parallel Gaussian channels with noise variances $\{\sigma_j^2\}$ and total power constraint P_{total}, the capacity-achieving allocation is $P_j^* = \max(0, \mu - \sigma_j^2)$ where μ is the water level. The analogy to our setting is summarized in Table 10:

Table 10: Analogy between classical water-filling and MI-constrained decomposition.

Concept	Classical Water-Filling	MI-Constrained Decomposition
Resource	Transmit power P_j	Mixing coefficient α_j
Channel quality	$1/\sigma_j^2$	$[\mathbf{Q}^{-1}\mathbf{b}]_j$
Water level	μ	Threshold from KKT
Constraint	$\sum_j P_j = P_{\text{total}}$	$\sum_j \alpha_j = 1$
Objective	Maximize capacity	Minimize KL divergence

Geometrically, imagine pouring water (total mass 1) into a container where expert j corresponds to a column with floor height proportional to $-[\mathbf{Q}^{-1}\mathbf{b}]_j$. The water level rises until total volume equals 1, and the water depth at each column corresponds to α_j^*.

A.7.3 Information-Preservation Bound (Conditional)

This section provides proofs for the information preservation bounds stated in the main text. We first establish the mutual information gap bound via direct Gaussian analysis with explicit error control, then use Fano's inequality to interpret its significance for downstream task performance.

Preliminaries. For random variables X and Y, the mutual information is $I(X;Y) = H(Y) - H(Y|X)$. Fano's inequality (Fano & Hawkins, 1961) relates mutual information to classification error: for any estimator $\hat{Z}$ of a discrete variable Z from observation Y, the error probability satisfies:

$$P_e = P(\hat{Z} \neq Z) \geq \frac{H(Z) - I(Z;Y) - 1}{\log |\mathcal{Z}|}. \tag{51}$$

Notation. Consistent with Section 4.1, we use $\mathbf{W}$ to denote an expert weight $\mathbf{W}_i$, and $\hat{\mathbf{W}} = \mathbf{W}_b = \sum_j \alpha_j \mathbf{W}_j$ to denote the shared base weight serving as its reconstruction. The reconstruction error is thus $\Delta\mathbf{W} = \mathbf{W} - \hat{\mathbf{W}} = \mathbf{W}_i - \mathbf{W}_b$.

Theorem 1 (Mutual Information Gap Bound)**.** *Let X denote input activations, $Y = \mathbf{W}X + \xi$ the original expert output, and $\hat{Y} = \hat{\mathbf{W}}X + \xi$ the reconstructed output, where $\xi \sim \mathcal{N}(0, \Sigma_\xi)$ is independent of $X \sim \mathcal{N}(0, \Sigma_X)$. Define $\Delta\mathbf{W} = \mathbf{W} - \hat{\mathbf{W}}$ and the effective signal-to-noise ratio* $\text{SNR} = \|\Sigma_\xi^{-1/2}\hat{\mathbf{W}}\Sigma_X^{1/2}\|_2^2$*. Then the mutual information gap satisfies:*

$$I(X;Y) - I(X;\hat{Y}) \leq (1+\rho) \cdot \mathbb{E}_X[D_{\text{KL}}(p(Y|X)\|p(\hat{Y}|X))], \tag{52}$$

where the remainder factor ρ is given by (for $\Delta\mathbf{W} \neq \mathbf{0}$):

$$\rho = \frac{2\|\Sigma_\xi^{-1/2}\hat{\mathbf{W}}\Sigma_X^{1/2}\|_F}{\|\Sigma_\xi^{-1/2}\Delta\mathbf{W}\Sigma_X^{1/2}\|_F} \leq \frac{2\sqrt{m \cdot \text{SNR}}}{\|\Sigma_\xi^{-1/2}\Delta\mathbf{W}\Sigma_X^{1/2}\|_F}, \tag{53}$$

where $\text{SNR}_{\max} = \max_i \|\Sigma_\xi^{-1/2}\hat{\mathbf{W}}_i\Sigma_X^{1/2}\|_2^2$ *is the maximum effective SNR across experts.*

Proof. The proof proceeds in three steps.

Step 1: Entropy Decomposition. Under the Gaussian model, the conditional distributions are:

$$p(Y|X=x) = \mathcal{N}(\mathbf{W}x, \Sigma_\xi), \tag{54}$$

$$p(\hat{Y}|X=x) = \mathcal{N}(\hat{\mathbf{W}}x, \Sigma_\xi). \tag{55}$$

Since both distributions share the same covariance Σ_ξ, the conditional entropies are identical:

$$H(Y|X) = H(\hat{Y}|X) = H(\xi) = \frac{1}{2}\log\det(2\pi e\Sigma_\xi). \tag{56}$$

Therefore, the mutual information gap reduces to an entropy difference:

$$I(X;Y) - I(X;\hat{Y}) = [H(Y) - H(Y|X)] - [H(\hat{Y}) - H(\hat{Y}|X)] = H(Y) - H(\hat{Y}). \tag{57}$$

Step 2: Conditional KL Divergence. For two Gaussians with the same covariance, the KL divergence is:

$$D_{\text{KL}}(p(Y|X=x)\|p(\hat{Y}|X=x)) = \frac{1}{2}(\mathbf{W}x - \hat{\mathbf{W}}x)^\top\Sigma_\xi^{-1}(\mathbf{W}x - \hat{\mathbf{W}}x) = \frac{1}{2}(\Delta\mathbf{W}x)^\top\Sigma_\xi^{-1}(\Delta\mathbf{W}x). \tag{58}$$

Taking expectation over $X \sim \mathcal{N}(0, \Sigma_X)$:

$$\mathbb{E}_X[D_{\text{KL}}] = \frac{1}{2}\text{tr}\left(\Sigma_\xi^{-1}\Delta\mathbf{W}\Sigma_X\Delta\mathbf{W}^\top\right). \tag{59}$$

Step 3: Entropy-KL Connection with Explicit Error Bound. For Gaussian $Y = \mathbf{W}X + \xi$, the marginal distribution is $Y \sim \mathcal{N}(0, \Sigma_Y)$ where $\Sigma_Y = \mathbf{W}\Sigma_X\mathbf{W}^\top + \Sigma_\xi$. Similarly, $\hat{Y} \sim \mathcal{N}(0, \Sigma_{\hat{Y}})$ with $\Sigma_{\hat{Y}} = \hat{\mathbf{W}}\Sigma_X\hat{\mathbf{W}}^\top + \Sigma_\xi$. The entropy difference is:

$$H(Y) - H(\hat{Y}) = \frac{1}{2}\log\det(\Sigma_Y) - \frac{1}{2}\log\det(\Sigma_{\hat{Y}}) = \frac{1}{2}\log\det(\Sigma_Y\Sigma_{\hat{Y}}^{-1}). \tag{60}$$

Define $\mathbf{A} = \Sigma_Y\Sigma_{\hat{Y}}^{-1} = \mathbf{I} + (\Sigma_Y - \Sigma_{\hat{Y}})\Sigma_{\hat{Y}}^{-1}$. Using the matrix inequality $\log\det(\mathbf{I}+\mathbf{B}) \leq \text{tr}(\mathbf{B})$ for $\mathbf{B}$ with real eigenvalues $\lambda_i(\mathbf{B}) > -1$ (which follows from $\log(1+\lambda) \leq \lambda$ for eigenvalues $\lambda > -1$):

$$\frac{1}{2}\log\det(\mathbf{A}) \leq \frac{1}{2}\text{tr}\left((\Sigma_Y - \Sigma_{\hat{Y}})\Sigma_{\hat{Y}}^{-1}\right). \tag{61}$$

The condition $\lambda_i(\mathbf{B}) > -1$ is equivalent to $\mathbf{A} = \mathbf{I} + \mathbf{B}$ having positive eigenvalues. Since $\Sigma_Y, \Sigma_{\hat{Y}} \succ 0$, the matrix $\mathbf{A} = \Sigma_Y\Sigma_{\hat{Y}}^{-1}$ is similar to $\Sigma_{\hat{Y}}^{-1/2}\Sigma_Y\Sigma_{\hat{Y}}^{-1/2} \succ 0$, confirming $\lambda_i(\mathbf{A}) > 0$ and hence the required condition.

Now we compute $\Sigma_Y - \Sigma_{\hat{Y}}$:

$$\begin{aligned}\Sigma_Y - \Sigma_{\hat{Y}} &= \mathbf{W}\Sigma_X\mathbf{W}^\top - \hat{\mathbf{W}}\Sigma_X\hat{\mathbf{W}}^\top \\ &= (\hat{\mathbf{W}} + \Delta\mathbf{W})\Sigma_X(\hat{\mathbf{W}} + \Delta\mathbf{W})^\top - \hat{\mathbf{W}}\Sigma_X\hat{\mathbf{W}}^\top \\ &= \Delta\mathbf{W}\Sigma_X\hat{\mathbf{W}}^\top + \hat{\mathbf{W}}\Sigma_X\Delta\mathbf{W}^\top + \Delta\mathbf{W}\Sigma_X\Delta\mathbf{W}^\top. \end{aligned} \tag{62}$$

Substituting into Eq. (61):

$$\begin{aligned}H(Y) - H(\hat{Y}) &\le \frac{1}{2}\text{tr}\left((\Delta\mathbf{W}\Sigma_X\hat{\mathbf{W}}^\top + \hat{\mathbf{W}}\Sigma_X\Delta\mathbf{W}^\top + \Delta\mathbf{W}\Sigma_X\Delta\mathbf{W}^\top)\Sigma_{\hat{Y}}^{-1}\right) \\ &= \text{tr}\left(\Delta\mathbf{W}\Sigma_X\hat{\mathbf{W}}^\top\Sigma_{\hat{Y}}^{-1}\right) + \frac{1}{2}\text{tr}\left(\Delta\mathbf{W}\Sigma_X\Delta\mathbf{W}^\top\Sigma_{\hat{Y}}^{-1}\right), \end{aligned} \tag{63}$$

where we used the cyclic property of trace and symmetry.

The key observation is that:

$$\Sigma_{\hat{Y}}^{-1} = (\hat{\mathbf{W}}\Sigma_X\hat{\mathbf{W}}^\top + \Sigma_\xi)^{-1} \preceq \Sigma_\xi^{-1}, \tag{64}$$

since $\hat{\mathbf{W}}\Sigma_X\hat{\mathbf{W}}^\top \succeq \mathbf{0}$ implies $\Sigma_{\hat{Y}} \succeq \Sigma_\xi$, and matrix inversion reverses the order.

For the second term in Eq. (63), applying Eq. (64):

$$\frac{1}{2}\text{tr}\left(\Delta\mathbf{W}\Sigma_X\Delta\mathbf{W}^\top\Sigma_{\hat{Y}}^{-1}\right) \le \frac{1}{2}\text{tr}\left(\Delta\mathbf{W}\Sigma_X\Delta\mathbf{W}^\top\Sigma_\xi^{-1}\right) = \mathbb{E}_X[D_{\text{KL}}], \tag{65}$$

where the equality follows from Eq. (59).

For the first term in Eq. (63), we analyze the cross term:

$$R_{\text{cross}} = \text{tr}\left(\Delta\mathbf{W}\Sigma_X\hat{\mathbf{W}}^\top\Sigma_{\hat{Y}}^{-1}\right). \tag{66}$$

Remainder Factor Derivation. Let $\mathbf{G} = \Sigma_\xi^{-1/2}\hat{\mathbf{W}}\Sigma_X^{1/2}$ and $\mathbf{E} = \Sigma_\xi^{-1/2}\Delta\mathbf{W}\Sigma_X^{1/2}$. Then $\Sigma_{\hat{Y}} = \Sigma_\xi^{1/2}(\mathbf{I} + \mathbf{G}\mathbf{G}^\top)\Sigma_\xi^{1/2}$, so $R_{\text{cross}} = \langle\mathbf{E}, (\mathbf{I} + \mathbf{G}\mathbf{G}^\top)^{-1}\mathbf{G}\rangle_F$ and $|R_{\text{cross}}| \le \|\mathbf{E}\|_F\|\mathbf{G}\|_F$, while $\mathbb{E}_X[D_{\text{KL}}] = \|\mathbf{E}\|_F^2/2$ by Eq. (59). Combining with Eq. (65) gives $I(X;Y) - I(X;\hat{Y}) \le (1 + 2\|\mathbf{G}\|_F/\|\mathbf{E}\|_F)\,\mathbb{E}_X[D_{\text{KL}}]$, and $\|\mathbf{G}\|_F \le \sqrt{m}\,\|\mathbf{G}\|_2 = \sqrt{m \cdot \text{SNR}}$, which proves Eq. (53). If $\mathbf{E} = \mathbf{0}$, then $\Sigma_Y = \Sigma_{\hat{Y}}$ and both sides vanish.

Information Retention Ratio. As a direct consequence of Theorem 1, we establish the following bound on information retention.

Corollary 2. *Under the conditions of Theorem 1, suppose the effective SNR satisfies* $\text{SNR}_{\max} \le S_0$ *for some constant* S_0*. If* $\mathbb{E}[D_{\text{KL}}] \le \delta$*, then the remainder factor* ρ *is bounded by Eq.* (53)*, and:*

$$\frac{I(X;\hat{Y})}{I(X;Y)} \ge 1 - \frac{(1+\rho)\delta}{I(X;Y)}. \tag{67}$$

Proof. By Theorem 1, $I(X;\hat{Y}) \ge I(X;Y) - (1+\rho)\mathbb{E}[D_{\text{KL}}] \ge I(X;Y) - (1+\rho)\delta$. Dividing both sides by $I(X;Y)$ yields the result.

Downstream Task Interpretation via Fano's Inequality. Fano's inequality relates task-label mutual information to a lower bound on classification error.

For any downstream classification task estimating a discrete label $Z \in \mathcal{Z}$ from model outputs, Fano's inequality states:

$$P_e \ge \frac{H(Z) - I(Z;Y) - 1}{\log|\mathcal{Z}|}. \tag{68}$$

The attainable classification error is thus limited by $I(Z;Y)$: losing information raises the lower bound on error.

A.7.4 Assumptions for the Information Retention Bound

Modeling Assumptions. The above analysis relies on the following regularity conditions: (i) the Gaussian approximation for activations, which is commonly adopted in neural network compression literature (Frantar et al., 2022; Hassibi & Stork, 1992); (ii) moderate compression where reconstruction errors remain small relative to the original weights, specifically $\|\Delta\mathbf{W}\|_F \lesssim 0.3\|\mathbf{W}\|_F$; (iii) bounded effective signal-to-noise ratio $\text{SNR}_{\max} \lesssim 10$; and (iv) sufficient calibration data for accurate covariance estimation.

Empirical Validation of Assumptions. We test condition (i) on Mixtral-8×7B via per-layer Kolmogorov–Smirnov tests on expert input activations ($\alpha = 0.05$): 27 of 32 layers (84.4%) pass; the 5 failing layers all lie at network edges (layers 0, 1, 2, 30, 31), reflecting the well-known non-Gaussianity of input embeddings and final-layer outputs. Replacing the Gaussian QP with a Fisher-information weighted base on these 5 edge layers changes WikiText-2 PPL from 4.58 to 4.67. Condition (ii) is verified by direct measurement: $\|\Delta\mathbf{W}\|_F/\|\mathbf{W}\|_F \in [0.18, 0.27]$ at 40–70% compression ratios. Condition (iv) is satisfied at 128 calibration samples (Section 5). Appendix A.20 relates retention to downstream quality.

Interpretation. The corollary connects the optimization objective (minimizing KL divergence) to information preservation. By controlling the KL divergence δ to be a small fraction of $I(X;Y)$, the analysis bounds the loss of input-output information; the downstream effect is measured directly in Appendix A.20.

A.7.5 Estimation of Covariance Matrices

This section specifies how the covariance matrices Σ_X and Σ_ξ are estimated in practice.

Input Covariance Estimation. The input covariance $\Sigma_X^{(l)}$ is estimated from calibration data as the sample covariance of hidden states before expert routing:

$$\Sigma_X^{(l)} = \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}} \mathbf{h}^{(l)}(x)(\mathbf{h}^{(l)}(x))^\top, \tag{69}$$

where $\mathbf{h}^{(l)}(x) \in \mathbb{R}^n$ denotes the hidden state at layer l before expert routing (i.e., the input to the MoE block). This layer-wise estimation is shared across all N experts, consistent with the goal of constructing a unified base weight $\mathbf{W}_b^{(l)}$.

Role of Σ_ξ. In Eq. (5), Σ_ξ^{-1} appears as a weighting matrix in the quadratic form $Q_{jk} = \text{tr}(\Sigma_\xi^{-1}\mathbf{W}_j\Sigma_X\mathbf{W}_k^\top)$. Mathematically, Σ_ξ^{-1} defines a Mahalanobis metric in the output space: the KL divergence $D_{\text{KL}} = \frac{1}{2}(\Delta\mathbf{W}x)^\top\Sigma_\xi^{-1}(\Delta\mathbf{W}x)$ measures reconstruction error under this metric. The choice of Σ_ξ determines which output dimensions the KL surrogate weights most.

Estimation Procedure. We estimate Σ_ξ from the empirical output covariance across experts. For each layer l, let $\bar{\mathbf{W}}^{(l)} = \frac{1}{N}\sum_{i=1}^N \mathbf{W}_i^{(l)}$ denote the simple average of expert weights. The output space metric is estimated as:

$$\hat{\Sigma}_\xi^{(l)} = \frac{1}{N}\sum_{i=1}^N (\mathbf{W}_i^{(l)} - \bar{\mathbf{W}}^{(l)})\Sigma_X^{(l)}(\mathbf{W}_i^{(l)} - \bar{\mathbf{W}}^{(l)})^\top + \lambda\mathbf{I}, \tag{70}$$

where $\Sigma_X^{(l)}$ is the input covariance defined in Eq. (69), and $\lambda = 10^{-6}$ is the regularization constant, ensuring $\hat{\Sigma}_\xi^{(l)} \succ 0$.

Rationale. This estimation adapts the metric to the inherent diversity among experts. Output dimensions with high inter-expert variance receive smaller weights in Σ_ξ^{-1}, allowing larger reconstruction errors in directions where experts naturally differ. Output dimensions with low inter-expert variance receive larger weights, enforcing precise reconstruction of the shared structure. This adaptive weighting makes the KL surrogate focus on preserving the common knowledge across experts rather than expert-specific variations.

Validity of Theoretical Results: Σ_ξ as a Modeling Choice. Theorem 1 and Eq. (8) hold for any fixed $\Sigma_\xi \succ 0$. The form in Eq. (70) is selected so that the resulting Mahalanobis metric adapts to expert diversity (the rationale above); the regularization $\lambda \mathbf{I}$ ($\lambda = 10^{-6}$) ensures $\hat{\Sigma}_\xi^{(l)} \succ 0$, hence the QP is always well-posed and the SNR bounds in Eq. (53) are well-defined. The estimation is performed once before optimization and held fixed throughout.

APPENDIX A.8: PGD CONVERGENCE ANALYSIS

This appendix provides the convergence analysis for the PGD algorithm for low-rank compression of delta weights.

A.8.1 PROBLEM SETUP

Note that this analysis applies independently to each layer l; we omit the layer superscript (l) throughout this appendix for notational simplicity.

Recall the optimization objective for each expert i:

$$\min_{\mathbf{U}_i, \mathbf{V}_i} \mathcal{L}_i(\mathbf{U}_i, \mathbf{V}_i) = \frac{\left\|(\Delta \mathbf{W}_i - \mathbf{U}_i \mathbf{V}_i^\top)\mathbf{S}\right\|_F^2}{\|\Delta \mathbf{W}_i \mathbf{S}\|_F^2} \quad \text{s.t.} \quad \text{rank}(\mathbf{U}_i \mathbf{V}_i^\top) \le r, \tag{71}$$

where $\mathbf{S} = (\Sigma_X + \lambda \mathbf{I})^{1/2} \in \mathbb{R}^{n \times n}$ with regularization constant $\lambda = 10^{-4}$ (Appendix A.12.2) is the regularized input covariance square root for the current layer, shared across all experts, and the target rank r is determined by the complexity constraint $r(m+n)/(mn) \le \gamma$. The normalization constant $D_i = \|\Delta \mathbf{W}_i \mathbf{S}\|_F^2$ is fixed throughout the PGD optimization, since $\Delta \mathbf{W}_i = \mathbf{W}_i - \mathbf{W}_b$ is determined by MI-constrained decomposition (executed once before PGD low-rank compression). This ensures that $\mathcal{L}_i$ represents a dimensionless relative error, and that the optimization is a standard weighted least-squares problem rather than fractional programming.

Since the factors $\mathbf{U}_i \in \mathbb{R}^{m \times r}$ and $\mathbf{V}_i \in \mathbb{R}^{n \times r}$ have fixed column dimension r, the rank constraint $\text{rank}(\mathbf{U}_i \mathbf{V}_i^\top) \le r$ is implicitly satisfied by the parameterization. The PGD algorithm alternates between gradient descent and factor balancing:

$$\mathbf{U}_i^{(t+\frac{1}{2})} = \mathbf{U}_i^{(t)} - \eta_t \nabla_{\mathbf{U}_i} \mathcal{L}_i, \tag{72}$$

$$\mathbf{V}_i^{(t+\frac{1}{2})} = \mathbf{V}_i^{(t)} - \eta_t \nabla_{\mathbf{V}_i} \mathcal{L}_i, \tag{73}$$

$$(\mathbf{U}_i^{(t+1)}, \mathbf{V}_i^{(t+1)}) = \mathcal{B}_r(\mathbf{U}_i^{(t+\frac{1}{2})}, \mathbf{V}_i^{(t+\frac{1}{2})}), \tag{74}$$

where $\mathcal{B}_r$ denotes the factor balancing operator, which re-parameterizes the factors via SVD to ensure numerical stability.

Properties of S. Since Σ_X is symmetric positive semi-definite, $\mathbf{S} = (\Sigma_X + \lambda \mathbf{I})^{1/2}$ is also symmetric, satisfying $\mathbf{S}\mathbf{S}^\top = \mathbf{S}^2 = \Sigma_X + \lambda \mathbf{I}$.

A.8.2 SMOOTHNESS AND FACTOR BALANCING

A.8.2.1 LIPSCHITZ CONTINUITY OF GRADIENT

The objective $\mathcal{L}_i(\mathbf{U}_i, \mathbf{V}_i)$ is a polynomial in $(\mathbf{U}_i, \mathbf{V}_i)$, so its gradient is Lipschitz continuous on every bounded set; $D_i = \|\Delta \mathbf{W}_i \mathbf{S}\|_F^2$ is the normalization constant. The PGD iterates with factor balancing remain uniformly bounded, yielding a global Lipschitz constant $L_{\max}$ over all iterations.

Proof. Let $D_i = \|\Delta \mathbf{W}_i \mathbf{S}\|_F^2$ be the normalization constant. The gradients are $\nabla_{\mathbf{U}_i} \mathcal{L}_i = -\frac{2}{D_i} \mathbf{E}_i \mathbf{S}\mathbf{S}^\top \mathbf{V}_i$ and $\nabla_{\mathbf{V}_i} \mathcal{L}_i = -\frac{2}{D_i} \mathbf{S}\mathbf{S}^\top \mathbf{E}_i^\top \mathbf{U}_i$, where $\mathbf{E}_i = \Delta \mathbf{W}_i - \mathbf{U}_i \mathbf{V}_i^\top$. The Hessian with respect to $\mathbf{U}_i$ is:

$$\nabla^2_{\mathbf{U}_i} \mathcal{L}_i = \frac{2}{D_i} \mathbf{V}_i^\top \mathbf{S}\mathbf{S}^\top \mathbf{V}_i \otimes \mathbf{I}_m. \tag{75}$$

The spectral norm is bounded by $\frac{2}{D_i} \|\mathbf{V}_i\|_2^2 \|\mathbf{S}\|_2^2$. A similar bound holds for the Hessian with respect to $\mathbf{V}_i$. The mixed blocks consist of a term linear in $\mathbf{E}_i$ and a term bilinear in $(\mathbf{U}_i, \mathbf{V}_i)$, so the full Hessian is bounded on every bounded set.

A.8.2.2 FACTOR BALANCING OPERATOR

While the rank constraint is implicitly satisfied by the factor dimensions, gradient updates may cause factor imbalance: the norms $\|\mathbf{U}_i\|_F$ and $\|\mathbf{V}_i\|_F$ can diverge significantly while their product $\mathbf{U}_i\mathbf{V}_i^\top$ remains unchanged. This imbalance degrades numerical conditioning and can slow convergence.

To address this, we introduce the factor balancing operator $\mathcal{B}_r : \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r} \to \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r}$. Given factors $(\mathbf{U}', \mathbf{V}')$, we compute the thin SVD of their product:

$$\mathbf{U}'\mathbf{V}'^\top = \sum_{k=1}^{r} \sigma_k \mathbf{u}_k \mathbf{v}_k^\top, \tag{76}$$

and define the balanced factors as:

$$\mathcal{B}_r(\mathbf{U}', \mathbf{V}') = (\tilde{\mathbf{U}}, \tilde{\mathbf{V}}), \quad \text{where } \tilde{\mathbf{U}} = [\sqrt{\sigma_k}\mathbf{u}_k]_{k=1}^r,\ \tilde{\mathbf{V}} = [\sqrt{\sigma_k}\mathbf{v}_k]_{k=1}^r. \tag{77}$$

The operator $\mathcal{B}_r$ satisfies the following key properties:

Lemma 3 (Properties of Factor Balancing)**.** *The factor balancing operator $\mathcal{B}_r$ satisfies:*

- ***Product Preservation:*** $\tilde{\mathbf{U}}\tilde{\mathbf{V}}^\top = \mathbf{U}'\mathbf{V}'^\top$.
- ***Norm Equalization:*** $\|\tilde{\mathbf{U}}\|_F = \|\tilde{\mathbf{V}}\|_F = \sqrt{\sum_{k=1}^r \sigma_k}$.
- ***Objective Invariance:*** $\mathcal{L}_i(\tilde{\mathbf{U}}, \tilde{\mathbf{V}}) = \mathcal{L}_i(\mathbf{U}', \mathbf{V}')$.

Proof. Property 1 follows directly from the construction: $\tilde{\mathbf{U}}\tilde{\mathbf{V}}^\top = \sum_{k=1}^r \sqrt{\sigma_k}\mathbf{u}_k(\sqrt{\sigma_k}\mathbf{v}_k)^\top = \sum_{k=1}^r \sigma_k \mathbf{u}_k\mathbf{v}_k^\top = \mathbf{U}'\mathbf{V}'^\top$. Property 2 follows from $\|\tilde{\mathbf{U}}\|_F^2 = \sum_{k=1}^r \sigma_k = \|\tilde{\mathbf{V}}\|_F^2$. Property 3 follows from Property 1: since $\mathcal{L}_i(\mathbf{U}, \mathbf{V})$ depends on $(\mathbf{U}, \mathbf{V})$ only through the product $\mathbf{U}\mathbf{V}^\top$, and this product is preserved by $\mathcal{B}_r$, the objective value is unchanged. Property 3 is crucial for convergence analysis: factor balancing does not affect the objective value, so it can be applied after each gradient step without interfering with the descent property.

A.8.2.3 BOUNDEDNESS OF ITERATES

The following lemma establishes that PGD iterates with factor balancing remain uniformly bounded, which is essential for ensuring a global Lipschitz constant.

Lemma 4 (Bounded Factors)**.** *Let $\{(\mathbf{U}^{(t)}, \mathbf{V}^{(t)})\}_{t\geq 0}$ be the PGD iterates with factor balancing initialized from truncated SVD. Then for all $t \geq 0$:*

$$\max(\|\mathbf{U}^{(t)}\|_2^2, \|\mathbf{V}^{(t)}\|_2^2) \leq M^2 := \|\mathbf{S}^{-1}\|_2 \left(\|\Delta\mathbf{W}\mathbf{S}\|_2 + \|\Delta\mathbf{W}\mathbf{S}\|_F \cdot \sqrt{\mathcal{L}^{(0)}}\right), \tag{78}$$

where $\mathcal{L}^{(0)}$ is the initial objective value.

Proof. After factor balancing (Eq. (77)), the balanced factors satisfy $\|\tilde{\mathbf{U}}\|_2^2 = \|\tilde{\mathbf{V}}\|_2^2 = \sigma_1(\mathbf{U}\mathbf{V}^\top) = \|\mathbf{U}\mathbf{V}^\top\|_2$. By the Armijo condition of Section A.8.4, $\mathcal{L}^{(t+1)} \leq \mathcal{L}^{(t)}$, so $\mathcal{L}^{(t)} \leq \mathcal{L}^{(0)}$ for all $t \geq 0$. Let $D = \|\Delta\mathbf{W}\mathbf{S}\|_F^2$. The objective definition gives:

$$\mathcal{L}^{(t)} = \frac{\|(\Delta\mathbf{W} - \mathbf{U}^{(t)}(\mathbf{V}^{(t)})^\top)\mathbf{S}\|_F^2}{D} \geq \frac{(\|\mathbf{U}^{(t)}(\mathbf{V}^{(t)})^\top\mathbf{S}\|_2 - \|\Delta\mathbf{W}\mathbf{S}\|_2)^2}{D}, \tag{79}$$

where the inequality follows from the reverse triangle inequality applied to spectral norms. Rearranging and using $\mathcal{L}^{(t)} \leq \mathcal{L}^{(0)}$:

$$\|\mathbf{U}^{(t)}(\mathbf{V}^{(t)})^\top\|_2 \leq \|\mathbf{S}^{-1}\|_2\|\mathbf{U}^{(t)}(\mathbf{V}^{(t)})^\top\mathbf{S}\|_2 \leq \|\mathbf{S}^{-1}\|_2 \left(\|\Delta\mathbf{W}\mathbf{S}\|_2 + \sqrt{D \cdot \mathcal{L}^{(0)}}\right). \tag{80}$$

Substituting $\sqrt{D} = \|\Delta\mathbf{W}\mathbf{S}\|_F$ completes the proof. □

Implication. This lemma ensures a uniform smoothness constant:

$$L_{\max} := \sup_{\mathbf{Z}\in\mathcal{K}} \|\nabla^2\mathcal{L}(\mathbf{Z})\|_2 < \infty, \tag{81}$$

where $\mathcal{K}$ is a bounded set containing the balanced iterates and the gradient-step segments. This enables the standard convergence analysis with a fixed smoothness constant.

A.8.3 CONVERGENCE THEOREM

Convergence Rate. By Lemma 4, the PGD iterates with factor balancing satisfy $\max(\|\mathbf{U}^{(t)}\|_2^2, \|\mathbf{V}^{(t)}\|_2^2) \leq M^2$ for all $t \geq 0$. This ensures a uniform Lipschitz constant $L_{\max}$ (Section A.8.2.3) over all iterations. Throughout this analysis, $\nabla\mathcal{L}^{(t)} := \nabla\mathcal{L}(\mathbf{U}^{(t)}, \mathbf{V}^{(t)})$ denotes the gradient evaluated at the balanced factors at the beginning of iteration t. The PGD sequence $\{(\mathbf{U}^{(t)}, \mathbf{V}^{(t)})\}$ with step size $\eta \leq 1/L_{\max}$ satisfies:

$$\min_{0\leq t\leq T-1} \|\nabla\mathcal{L}^{(t)}\|_F^2 \leq \frac{2(\mathcal{L}^{(0)} - \mathcal{L}^*)}{\eta T} = O(1/T), \tag{82}$$

where $\mathcal{L}^*$ is the optimal value in the factor space.

Proof. The proof follows the standard analysis for projected gradient descent on smooth functions (Beck, 2017).

Step 1: Descent Lemma. By L-smoothness of $\mathcal{L}$:

$$\mathcal{L}^{(t+1)} \leq \mathcal{L}^{(t)} + \langle\nabla\mathcal{L}^{(t)}, \mathbf{Z}^{(t+1)} - \mathbf{Z}^{(t)}\rangle + \frac{L}{2}\|\mathbf{Z}^{(t+1)} - \mathbf{Z}^{(t)}\|_F^2, \tag{83}$$

where $\mathbf{Z}^{(t)} = (\mathbf{U}^{(t)}, \mathbf{V}^{(t)})$ denotes the concatenated variables.

Step 2: Factor Balancing Invariance. By Lemma 3 (Property 3), factor balancing preserves the objective value:

$$\mathcal{L}(\mathbf{U}^{(t+1)}, \mathbf{V}^{(t+1)}) = \mathcal{L}(\mathcal{B}_r(\mathbf{U}^{(t+\frac{1}{2})}, \mathbf{V}^{(t+\frac{1}{2})})) = \mathcal{L}(\mathbf{U}^{(t+\frac{1}{2})}, \mathbf{V}^{(t+\frac{1}{2})}). \tag{84}$$

This invariance is the key property enabling our convergence analysis: the factor balancing step does not affect the descent achieved by the gradient step.

Step 3: Sufficient Decrease. For step size $\eta \leq 1/L$, we establish sufficient decrease in the factor space. Let $(\mathbf{U}^{(t+\frac{1}{2})}, \mathbf{V}^{(t+\frac{1}{2})})$ denote the factors after gradient update. By L-smoothness of $\mathcal{L}$ in the factor space:

$$\mathcal{L}(\mathbf{U}^{(t+\frac{1}{2})}, \mathbf{V}^{(t+\frac{1}{2})}) \leq \mathcal{L}(\mathbf{U}^{(t)}, \mathbf{V}^{(t)}) - \frac{\eta}{2}\|\nabla\mathcal{L}^{(t)}\|_F^2. \tag{85}$$

Combining with the factor balancing invariance (Eq. (84)):

$$\mathcal{L}^{(t+1)} = \mathcal{L}(\mathbf{U}^{(t+\frac{1}{2})}, \mathbf{V}^{(t+\frac{1}{2})}) \leq \mathcal{L}^{(t)} - \frac{\eta}{2}\|\nabla\mathcal{L}^{(t)}\|_F^2. \tag{86}$$

This is the standard descent inequality for gradient descent on smooth functions, with factor balancing having no effect on the convergence rate.

Step 4: Telescoping Sum. Summing over $t = 0, \ldots, T-1$:

$$\sum_{t=0}^{T-1} \frac{\eta}{2}\|\nabla\mathcal{L}^{(t)}\|_F^2 \leq \mathcal{L}^{(0)} - \mathcal{L}^{(T)} \leq \mathcal{L}^{(0)} - \mathcal{L}^*. \tag{87}$$

Step 5: Final Bound. Taking the minimum over all iterations:

$$\min_{0\leq t\leq T-1} \|\nabla\mathcal{L}^{(t)}\|_F^2 \leq \frac{1}{T}\sum_{t=0}^{T-1} \|\nabla\mathcal{L}^{(t)}\|_F^2 \leq \frac{2(\mathcal{L}^{(0)} - \mathcal{L}^*)}{\eta T}. \tag{88}$$

This establishes $O(1/T)$ convergence to a stationary point in the factor space.

A.8.4 PRACTICAL CONSIDERATIONS

Step Size Selection. The Lipschitz constant L depends on the current iterates and may be difficult to estimate. We employ Armijo line search (Armijo, 1966) to adaptively select the step size η_t at each iteration. Starting from an initial step size $\eta_0 = 1$, the algorithm repeatedly halves the step size until the sufficient decrease condition is satisfied:

$$\mathcal{L}^{(t+1)} \leq \mathcal{L}^{(t)} - c \cdot \eta_t \|\nabla\mathcal{L}^{(t)}\|_F^2. \tag{89}$$

Initialization. The factors are initialized via truncated SVD of the activation-weighted delta weights $\Delta\mathbf{W}_i\mathbf{S}$:

$$\Delta\mathbf{W}_i\mathbf{S} = \sum_{k=1}^{\min(m,n)} \sigma_k \mathbf{u}_k \mathbf{v}_k^\top \approx \mathbf{U}_i^{(0)} (\mathbf{S}\mathbf{V}_i^{(0)})^\top, \tag{90}$$

where $\mathbf{U}_i^{(0)} = [\sqrt{\sigma_1}\mathbf{u}_1, \ldots, \sqrt{\sigma_r}\mathbf{u}_r]$ and $\mathbf{V}_i^{(0)} = \mathbf{S}^{-1}[\sqrt{\sigma_1}\mathbf{v}_1, \ldots, \sqrt{\sigma_r}\mathbf{v}_r]$.

Convergence Criterion. The algorithm terminates when the relative change in objective falls below a threshold:

$$\frac{|\mathcal{L}^{(t+1)} - \mathcal{L}^{(t)}|}{|\mathcal{L}^{(t)}| + \delta} < \tau. \tag{91}$$

A.8.5 Monotonicity of Reconstruction Error

A key property enabling efficient optimization is the monotonicity of the reconstruction error with respect to rank.

Lemma 5 (Monotonicity)**.** *For fixed delta weights $\{\Delta\mathbf{W}_i^{(l)}\}$ and activation statistics $\{\mathbf{S}^{(l)}\}$, the optimal reconstruction error $\mathcal{L}^*(\Theta; r)$ is monotonically non-increasing in rank r:*

$$r_1 < r_2 \quad \Rightarrow \quad \mathcal{L}^*(\Theta; r_1) \geq \mathcal{L}^*(\Theta; r_2). \tag{92}$$

Proof. Let $(\mathbf{U}_i^{(l)*}, \mathbf{V}_i^{(l)*})$ denote the optimal rank-r_1 factors for expert i in layer l. For rank $r_2 > r_1$, we can construct a feasible rank-r_2 solution by padding with zeros:

$$\tilde{\mathbf{U}}_i^{(l)} = [\mathbf{U}_i^{(l)*}, \mathbf{0}_{m\times(r_2-r_1)}], \quad \tilde{\mathbf{V}}_i^{(l)} = [\mathbf{V}_i^{(l)*}, \mathbf{0}_{n\times(r_2-r_1)}]. \tag{93}$$

This achieves the same reconstruction error as the rank-r_1 solution. Since the optimal rank-r_2 solution can only improve upon this, we have $\mathcal{L}^*(\Theta; r_2) \leq \mathcal{L}^*(\Theta; r_1)$.

Remark (Solution Quality). Since $\mathbf{S} \succ 0$ is symmetric, the objective in Eq. (9) equals $\|\Delta\mathbf{W}_i\mathbf{S} - \mathbf{U}_i(\mathbf{S}\mathbf{V}_i)^\top\|_F^2/D_i$. Under the reparameterization $\tilde{\mathbf{V}}_i = \mathbf{S}\mathbf{V}_i$, this reduces to standard low-rank approximation of $\Delta\mathbf{W}_i\mathbf{S}$, for which truncated SVD provides the global optimum (Eckart-Young-Mirsky theorem). This motivates our truncated SVD initialization; the formal guarantee for the PGD iterations is convergence to a stationary point at rate $O(1/T)$.

APPENDIX A.9: Pareto Frontier

This appendix provides conceptual background on multi-objective optimization and Pareto frontier construction.

A.9.1 Pareto Frontier Construction

Stage 2 of Algorithm 1 (Appendix A.11.2) constructs the Pareto frontier: each (γ, ϵ_k) grid point yields at most one configuration, which is submitted to the non-dominated update of Eq. (16). Stage 1 computes the reference routing distributions once per model.

A.9.2 Competing Objectives in MoE Compression

In MoE compression, two objectives conflict: model size $\mathcal{C}(\Theta)$, where smaller is better for deployment efficiency, and reconstruction error $\mathcal{L}(\Theta)$, where lower is better for model quality.

These objectives compete because aggressive compression (lower $\mathcal{C}$) inevitably increases reconstruction error (higher $\mathcal{L}$). Multi-objective problems admit a set of optimal solutions representing different trade-offs. This motivates the Pareto frontier framework.

A.9.3 Pareto Frontier Fundamentals

Dominance Relation. Configuration Θ' *dominates* Θ (written $\Theta' \prec \Theta$) if Θ' is at least as good as Θ in both objectives and strictly better in at least one:

$$\Theta' \prec \Theta \quad \Leftrightarrow \quad \mathcal{C}(\Theta') \leq \mathcal{C}(\Theta),\ \mathcal{L}(\Theta') \leq \mathcal{L}(\Theta),\ \text{with at least one strict inequality.} \tag{94}$$

Pareto Optimality. A configuration Θ^* is *Pareto-optimal* if no other configuration dominates it. Intuitively, Θ^* cannot be improved in one objective without sacrificing the other.

Pareto Frontier. The Pareto frontier $\mathcal{F}$ is the set of all Pareto-optimal configurations. Geometrically, $\mathcal{F}$ forms the lower-left boundary in the $(\mathcal{C}, \mathcal{L})$ space, meaning no evaluated configuration lies below and to the left of this boundary.

A.9.4 Frontier Maintenance

As stated in Eq. (16), the frontier is maintained incrementally. For each new candidate Θ_{new}:

1. **Dominance check**: If any $\Theta \in \mathcal{F}$ dominates Θ_{new}, discard Θ_{new}.
2. **Removal**: Remove from $\mathcal{F}$ all configurations dominated by Θ_{new}.
3. **Insertion**: Add Θ_{new} to $\mathcal{F}$.

A.9.5 Pareto-Optimality Property

Theorem 6 (Non-Dominance of Returned Configurations)**.** *Each configuration retained in $\mathcal{F}$ by Stage 2 of Algorithm 1 is non-dominated among the configurations evaluated so far in the bi-objective problem* $\min_\Theta(\mathcal{C}(\Theta), \mathcal{L}(\Theta))$.

Proof. Each candidate Θ_k is submitted to the update in Eq. (16). The update discards a candidate dominated by a member of $\mathcal{F}$ and removes every member dominated by an accepted candidate. Starting from $\mathcal{F} = \emptyset$, induction over the candidates, using transitivity of dominance, shows that these two operations keep every member of $\mathcal{F}$ non-dominated among all configurations evaluated so far.

Remark on Discretization. Algorithm 1 enumerates configurations on a discrete grid (4 values of γ and 4 of ϵ in our experiments; see Section 5). Theorem 6 concerns non-dominance among the evaluated configurations.

APPENDIX A.10: Theoretical Analysis of Constraint Adjustment

This appendix provides the theoretical motivation for the adaptive selection rule in Section 4.4, including the numerical treatment of edge cases in KL divergence computation and the derivation of error bounds under distribution shift.

A.10.1 Additive Smoothing for KL Divergence

The KL divergence $D_{\text{KL}}(p \| p_{\text{ref}})$ in Eq. (18) is undefined when $p_{\text{ref}}(i) = 0$ but $p(i) > 0$, which occurs when an expert is never activated during calibration but becomes active on the target task. To handle this edge case, we apply additive smoothing to the reference distribution:

$$\tilde{p}_{\text{ref}}(i) = \frac{p_{\text{ref}}(i) + \epsilon_{\text{smooth}}}{1 + N \cdot \epsilon_{\text{smooth}}}, \tag{95}$$

where $\epsilon_{\text{smooth}} > 0$ is a small smoothing constant and N is the number of experts. This ensures $\tilde{p}_{\text{ref}}(i) > 0$ for all i, making the KL divergence well-defined. The denominator normalizes the smoothed distribution to sum to 1.

Bounded KL Divergence. With smoothing, the KL divergence is upper bounded:

$$D_{\mathrm{KL}}(p\|\tilde{p}_{\text{ref}}) \leq \log \frac{1 + N \cdot \epsilon_{\text{smooth}}}{\epsilon_{\text{smooth}}}. \tag{96}$$

This bound is achieved when p is a point mass on an expert with $p_{\text{ref}}(i) = 0$. In practice, observed D_{shift} values are much smaller, so the smoothing has negligible impact on the theoretical analysis.

Consistency with Theoretical Results. The error bound in the following section relies on Pinsker's inequality, which requires $D_{\mathrm{KL}}(p\|p_{\text{ref}}) < \infty$. With additive smoothing, this condition is always satisfied.

A.10.2 Error Bound under Distribution Shift

This section analyzes how reconstruction error changes under distribution shift for a fixed compression configuration. The analysis motivates the functional form of Eq. (19).

Scope. The analysis below considers a fixed compressed model Θ with low-rank factors $\{(\mathbf{U}_i^{(l)}, \mathbf{V}_i^{(l)})\}_{l,i}$ obtained from PGD optimization on reference data, and derives how the reconstruction error changes when evaluated under a different routing distribution.

Assumption 7 (Fixed Compression Configuration)**.** The reconstruction error $\mathcal{L}_i^{(l)}$ for each expert i in layer l is treated as a fixed constant determined by the compressed representation Θ. This holds when: (i) the low-rank factors $(\mathbf{U}_i^{(l)}, \mathbf{V}_i^{(l)})$ are fixed, and (ii) the activation weighting matrix $\mathbf{S}^{(l)}$ used in Eq. (9) is held constant.

Under Assumption 7, the analysis applies to each layer independently. Throughout, $p_{\text{ref}}^{(l)}$ denotes the smoothed reference distribution $\tilde{p}_{\text{ref}}^{(l)}$ of Appendix A.10.1, so that all divergences are finite, and both $p^{(l)}$ and $p_{\text{ref}}^{(l)}$ sum to one (Eq. (17)). For layer l, let $\mathcal{L}_{\text{ref}}^{(l)} = \sum_{i=1}^{N} p_{\text{ref}}^{(l)}(i) \cdot \mathcal{L}_i^{(l)} > 0$ denote the expected reconstruction error under the reference distribution, and $\mathcal{L}^{(l)} = \sum_{i=1}^{N} p^{(l)}(i) \cdot \mathcal{L}_i^{(l)}$ the error under the current distribution. Then for each layer:

$$\mathcal{L}^{(l)} \leq \mathcal{L}_{\text{ref}}^{(l)} \cdot \left(1 + \zeta^{(l)} \sqrt{2 D_{\mathrm{KL}}(p^{(l)}\|p_{\text{ref}}^{(l)})}\right), \tag{97}$$

where $\zeta^{(l)} = \max_i |\mathcal{L}_i^{(l)} - \mathcal{L}_{\text{ref}}^{(l)}| / \mathcal{L}_{\text{ref}}^{(l)}$ captures expert heterogeneity within layer l.

Derivation. Since both distributions sum to one, $\sum_i (p^{(l)}(i) - p_{\text{ref}}^{(l)}(i)) = 0$, so the difference in reconstruction error for layer l can be centered:

$$\mathcal{L}^{(l)} - \mathcal{L}_{\text{ref}}^{(l)} = \sum_{i=1}^{N} (p^{(l)}(i) - p_{\text{ref}}^{(l)}(i)) \cdot \mathcal{L}_i^{(l)} = \sum_{i=1}^{N} (p^{(l)}(i) - p_{\text{ref}}^{(l)}(i)) \cdot (\mathcal{L}_i^{(l)} - \mathcal{L}_{\text{ref}}^{(l)}). \tag{98}$$

By Hölder's inequality:

$$|\mathcal{L}^{(l)} - \mathcal{L}_{\text{ref}}^{(l)}| \leq \|p^{(l)} - p_{\text{ref}}^{(l)}\|_1 \cdot \max_i |\mathcal{L}_i^{(l)} - \mathcal{L}_{\text{ref}}^{(l)}|. \tag{99}$$

By Pinsker's inequality, $\|p - q\|_1 \leq \sqrt{2 D_{\mathrm{KL}}(p\|q)}$, we have:

$$|\mathcal{L}^{(l)} - \mathcal{L}_{\text{ref}}^{(l)}| \leq \sqrt{2 D_{\mathrm{KL}}(p^{(l)}\|p_{\text{ref}}^{(l)})} \cdot \max_i |\mathcal{L}_i^{(l)} - \mathcal{L}_{\text{ref}}^{(l)}|. \tag{100}$$

Dividing by $\mathcal{L}_{\text{ref}}^{(l)}$ and setting $\zeta^{(l)} = \max_i |\mathcal{L}_i^{(l)} - \mathcal{L}_{\text{ref}}^{(l)}| / \mathcal{L}_{\text{ref}}^{(l)}$ yields the per-layer result.

Global Error Bound. The routing-weighted error averaged across all L layers is:

$$\mathcal{L} = \frac{1}{L} \sum_{l=1}^{L} \mathcal{L}^{(l)}, \qquad \mathcal{L}_{\text{ref}} = \frac{1}{L} \sum_{l=1}^{L} \mathcal{L}_{\text{ref}}^{(l)}. \tag{101}$$

Similarly, the global distribution shift is defined as the average KL divergence:

$$D_{\text{shift}} = \frac{1}{L}\sum_{l=1}^{L} D_{\text{KL}}(p^{(l)} \| p_{\text{ref}}^{(l)}). \tag{102}$$

Averaging the per-layer bounds gives $\mathcal{L} \leq \mathcal{L}_{\text{ref}} + \frac{1}{L}\sum_{l=1}^{L} \mathcal{L}_{\text{ref}}^{(l)} \zeta^{(l)} \sqrt{2 D_{\text{KL}}(p^{(l)} \| p_{\text{ref}}^{(l)})}$. Bounding each $\mathcal{L}_{\text{ref}}^{(l)}\zeta^{(l)}$ by $\bar{\zeta}\,\mathcal{L}_{\text{ref}}$ and applying Jensen's inequality to the concave square root gives

$$\mathcal{L} \leq \mathcal{L}_{\text{ref}} \cdot \left(1 + \bar{\zeta}\sqrt{2 D_{\text{shift}}}\right), \qquad \bar{\zeta} = \max_{l} \zeta^{(l)} \mathcal{L}_{\text{ref}}^{(l)} / \mathcal{L}_{\text{ref}}. \tag{103}$$

Implication for Configuration Selection. To ensure $\mathcal{L}_{\text{task}} \leq \epsilon$ under distribution shift, it suffices that

$$\mathcal{L}_{\text{ref}} \leq \frac{\epsilon}{1 + \bar{\zeta}\sqrt{2 D_{\text{shift}}}} = \epsilon_{\text{adj}}. \tag{104}$$

This motivates Eq. (19): for $\zeta \geq \bar{\zeta}$, a configuration whose routing-weighted reference error satisfies the adjusted bound has routing-weighted target error at most ϵ.

A.10.3 Sample Efficiency

The routing distribution $p^{(l)}(i)$ in Eq. (17) converges rapidly with sample size. Since each input token activates exactly k experts (where k is the top-k parameter), the effective sample size for estimating N-way categorical distributions is $k \cdot |\mathcal{X}|$. By standard concentration bounds for multinomial estimation, the ℓ_1 estimation error satisfies $\|p^{(l)} - \hat{p}^{(l)}\|_1 \leq O(\sqrt{N/(k \cdot |\mathcal{X}|)})$ with high probability.

APPENDIX A.11: End-to-End Integration

This appendix provides the complete algorithmic description of the PRIMO pipeline, detailing how the four components are integrated into a unified compression framework.

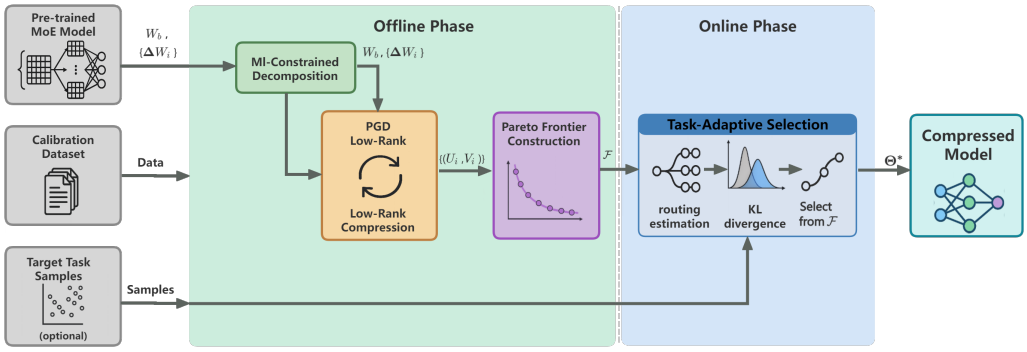


Figure 8: End-to-end Integration Pipeline

As illustrated in Figure 8, the pipeline consists of two distinct phases: an offline compression phase executed once per model, and a lightweight online phase for task-adaptive deployment.

A.11.1 Pipeline Overview

PRIMO takes three inputs: (1) a pre-trained MoE model with expert weights $\{\mathbf{W}_i^{(l)}\}_{l,i}$, (2) a calibration dataset $\mathcal{X}_{\text{cal}}$ for statistics estimation, and (3) optionally, target task samples $\mathcal{X}_{\text{task}}$ for adaptive selection. The pipeline produces a compressed model Θ^* optimized for the deployment scenario.

The key design principle is to separate expensive optimization (offline) from lightweight adaptation (online):

- **Offline Phase**: Executes once per model. Constructs the shared base weight $\mathbf{W}_b$, compresses delta weights across multiple compression levels, and builds the Pareto frontier $\mathcal{F}$.
- **Online Phase**: Executes per deployment task. Estimates distribution shift from a small number of target samples and selects the optimal configuration from $\mathcal{F}$.

A.11.2 Complete PRIMO Procedure

Algorithm 1 summarizes the complete PRIMO procedure. The algorithm proceeds in three stages. Stage 1 runs once per model: for each layer it estimates the input covariance, the output metric and the reference routing distribution on calibration data, then solves the convex quadratic program over the simplex to obtain the shared base weight $\mathbf{W}_b^{(l)}$ that minimizes the KL surrogate to information loss, together with the delta weights $\Delta\mathbf{W}_i^{(l)}$. Stage 2 sweeps the (γ, ϵ) grid: each complexity bound caps the admissible rank, and binary search retains the smallest trial rank whose PGD factors, initialized by truncated SVD and rebalanced after every Armijo step, meet the error bound; each retained configuration is submitted to the non-dominated update that maintains the Pareto frontier $\mathcal{F}$. Stage 3 runs once per task: it measures the smoothed routing KL shift from a few target samples, sets the shift to zero when none are given, tightens the tolerance to ϵ_{adj}, and returns the most compressed configuration in $\mathcal{F}$ that satisfies it, so selection is a single linear scan.

A.11.3 Online Phase

Stage 3 of Algorithm 1 performs task-adaptive selection: it estimates the target routing distribution, computes the smoothed routing shift D_{shift}, tightens the tolerance by Eq. (19), and selects the most compressed configuration in $\mathcal{F}$ by linear scan. Appendix A.10.2 gives the theoretical justification for the adjustment.

A.11.4 Input/Output Specification

Table 11 summarizes the input/output interface for each component.

Table 11: Input/output specification for PRIMO components.

Component	Input	Output	Phase
1. MI-Constrained	Expert weights $\{\mathbf{W}_i^{(l)}\}$ Covariances $\Sigma_X^{(l)}, \Sigma_\xi^{(l)}$	Base weight $\mathbf{W}_b^{(l)}$ Delta weights $\{\Delta\mathbf{W}_i^{(l)}\}$	Offline
2. PGD Compression	Delta weights $\{\Delta\mathbf{W}_i^{(l)}\}$ Bounds ϵ, γ	Low-rank factors $\{(\mathbf{U}_i^{(l)}, \mathbf{V}_i^{(l)})\}$	Offline
3. Pareto Frontier	Multiple configurations from Component 2	Frontier $\mathcal{F}$ Reference $\{p_{\text{ref}}^{(l)}\}$	Offline
4. Task-Adaptive	Frontier $\mathcal{F}$, $\{p_{\text{ref}}^{(l)}\}$ Target samples $\mathcal{X}_{\text{task}}$	Optimal Θ^*	Online

A.11.5 Computational Cost

Offline Phase. The offline phase is dominated by PGD optimization in Component 2. For a model with L layers and N experts per layer, with expert weight dimension $m \times n$ and K compression levels:

- Component 1: $O(L \cdot (|\mathcal{X}_{\text{cal}}| \cdot n^2 + N^2 \cdot mn))$ for covariance computation and QP solving
- Component 2: $O(K \cdot T \cdot L \cdot N \cdot mnr)$ for PGD iterations across all configurations
- Component 3: $O(K^2)$ for Pareto frontier maintenance (negligible)

Appendix A.21 reports the measured cost: 91.3 minutes for Mixtral-8×7B at 40% compression with $K = 10$.

Algorithm 1 PRIMO: Offline Compression and Online Task-Adaptive Selection

Require: Pre-trained MoE model with expert weights $\{\mathbf{W}_i^{(l)}\}_{l=1,i=1}^{L,N}$; calibration data $\mathcal{X}_{\text{cal}}$; grids Γ (complexity bounds) and $\mathcal{E}$ (error bounds); minimum rank $r_{\min}$; regularization λ; PGD tolerance τ and iteration cap T; tolerance ϵ; heterogeneity factor ζ; target samples $\mathcal{X}_{\text{task}}$ (optional)

Ensure: Pareto frontier $\mathcal{F}$, reference routing distributions $\{p_{\text{ref}}^{(l)}\}_{l=1}^{L}$, selected configuration Θ^*

Stage 1: MI-Constrained Decomposition (offline, once per model)

for $l = 1$ to L **do**
  Estimate $\Sigma_X^{(l)}$, $\Sigma_\xi^{(l)}$, $p_{\text{ref}}^{(l)}$ on $\mathcal{X}_{\text{cal}}$ ▷ Eqs. (69), (70), (17)
  Build $\mathbf{Q}^{(l)}$ and $\mathbf{b}^{(l)}$ with $p_i = p_{\text{ref}}^{(l)}(i)$ ▷ Eqs. (6), (7)
  $\boldsymbol{\alpha}^{(l)*} \leftarrow \arg\min_{\boldsymbol{\alpha}\in\Delta^{N-1}} \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{Q}^{(l)}\boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \mathbf{b}^{(l)}$ ▷ Eq. (8)
  $\mathbf{W}_b^{(l)} \leftarrow \sum_i \alpha_i^{(l)*}\mathbf{W}_i^{(l)}$; $\Delta\mathbf{W}_i^{(l)} \leftarrow \mathbf{W}_i^{(l)} - \mathbf{W}_b^{(l)}$ for all i ▷ Eq. (4)
  $\mathbf{S}^{(l)} \leftarrow (\Sigma_X^{(l)} + \lambda\mathbf{I})^{1/2}$ ▷ Eq. (9)
end for

Stage 2: PGD Low-Rank Compression and Pareto Frontier (offline)

$\mathcal{F} \leftarrow \emptyset$
for each $(\gamma, \epsilon_k) \in \Gamma \times \mathcal{E}$ **do**
  $r_{\text{lo}} \leftarrow r_{\min}$; $r_{\text{hi}} \leftarrow \lfloor \gamma mn/(m+n) \rfloor$; $\Theta_k \leftarrow \emptyset$ ▷ Eq. (11)
  while $r_{\text{lo}} \le r_{\text{hi}}$ **do** ▷ Binary search on rank
    $r \leftarrow \lfloor (r_{\text{lo}} + r_{\text{hi}})/2 \rfloor$
    for each layer l and expert i **do**
      Initialize $(\mathbf{U}_i^{(l)}, \mathbf{V}_i^{(l)})$ by rank-r truncated SVD of $\Delta\mathbf{W}_i^{(l)}\mathbf{S}^{(l)}$
      repeat
        Gradient step with Armijo step size s_t ▷ Eqs. (12), (14)
        $(\mathbf{U}_i^{(l)}, \mathbf{V}_i^{(l)}) \leftarrow \mathcal{B}_r(\mathbf{U}_i^{(l)}, \mathbf{V}_i^{(l)})$ ▷ Eq. (77)
      until relative change of $\mathcal{L}_i^{(l)}$ is below τ, or T iterations
    end for
    $\Theta_r \leftarrow \{(\mathbf{W}_b^{(l)}, \{\mathbf{U}_i^{(l)}, \mathbf{V}_i^{(l)}\}_{i=1}^{N})\}_{l=1}^{L}$; compute $\mathcal{L}(\Theta_r)$ ▷ Eq. (10)
    if $\mathcal{L}(\Theta_r) \le \epsilon_k$ **then**
      $\Theta_k \leftarrow \Theta_r$; $r_{\text{hi}} \leftarrow r - 1$
    else
      $r_{\text{lo}} \leftarrow r + 1$
    end if
  end while
  if $\Theta_k \neq \emptyset$ **then**
    Update $\mathcal{F}$ with $(\Theta_k, \mathcal{C}(\Theta_k), \mathcal{L}(\Theta_k))$ ▷ Eq. (16)
  end if
end for

Stage 3: Task-Adaptive Selection (online, once per task)

$D_{\text{shift}} \leftarrow 0$
if $\mathcal{X}_{\text{task}} \neq \emptyset$ **then**
  Estimate $p^{(l)}$ on $\mathcal{X}_{\text{task}}$; smooth $p_{\text{ref}}^{(l)}$ into $\tilde{p}_{\text{ref}}^{(l)}$, for all l ▷ Eqs. (17), (95)
  $D_{\text{shift}} \leftarrow \frac{1}{L}\sum_{l=1}^{L} D_{\text{KL}}(p^{(l)} \,\|\, \tilde{p}_{\text{ref}}^{(l)})$ ▷ Eq. (18)
end if
$\epsilon_{\text{adj}} \leftarrow \epsilon/(1 + \zeta\sqrt{2D_{\text{shift}}})$ ▷ Eq. (19)
$\Theta^* \leftarrow \arg\min_{\Theta\in\mathcal{F}} \mathcal{C}(\Theta)$ s.t. $\mathcal{L}(\Theta) \le \epsilon_{\text{adj}}$ ▷ Linear scan over $\mathcal{F}$
return $\mathcal{F}$, $\{p_{\text{ref}}^{(l)}\}_{l=1}^{L}$, Θ^*

Online Phase. The online phase is lightweight by design:

- Routing estimation: $O(|\mathcal{X}_{\text{task}}| \cdot L \cdot N \cdot d)$ for a single forward pass
- KL computation: $O(L \cdot N)$
- Configuration selection: $O(K)$ for linear scan over $\mathcal{F}$

The online cost is dominated by one forward pass over $\mathcal{X}_{\text{task}}$.

A.11.6 Deployment Without Target Samples

When target task samples are unavailable (e.g., zero-shot deployment), practitioners can bypass Component 4 and directly select a configuration from the Pareto frontier $\mathcal{F}$ based on resource constraints:

$$\Theta^* = \arg\min_{\Theta \in \mathcal{F}} \mathcal{C}(\Theta) \quad \text{s.t.} \quad \mathcal{L}(\Theta) \leq \epsilon. \tag{105}$$

This corresponds to setting $D_{\text{shift}} = 0$ in the adaptive selection rule.

APPENDIX A.12: Experimental Setup

A.12.1 Implementation Details

Software Environment. All experiments are implemented in PyTorch 2.1.0 (Paszke et al., 2019) with CUDA 12.1. The Hugging Face Transformers library (v4.36.0) (Wolf et al., 2020) is used for model loading and tokenization. Additional dependencies include NumPy 1.24.0 for numerical computations, SciPy 1.11.0 for optimization utilities, and the lm-evaluation-harness (v0.4.0) (Gao et al., 2021) for downstream task evaluation. All matrix operations leverage cuBLAS for GPU acceleration, and SVD computations use the randomized algorithm from scikit-learn for efficiency on large matrices (Halko et al., 2011).

Hardware Configuration. Experiments are conducted on a server equipped with four NVIDIA A100 80GB GPUs interconnected via NVLink. The system has 512GB of CPU RAM and runs Ubuntu 22.04 LTS. For Mixtral-8×7B, model loading requires approximately 90GB GPU memory in half-precision (bfloat16), distributed across two GPUs using the `device_map="auto"` strategy. For larger models (Qwen2-57B-A14B), all four GPUs are utilized. Peak GPU memory during compression is comparable to that of D^2-MoE (Gu et al., 2025), which requires 131GB for Mixtral-8×7B, 35GB for DeepSeek-MoE-16B-Base, 118GB for Phi-3.5-MoE, and 127GB for Qwen2-57B-A14B.

Models. PRIMO is evaluated on four representative open-source MoE models, consistent with the D^2-MoE evaluation setup:

- **Mixtral-8×7B** (Jiang et al., 2024): 46.7B total parameters with 8 experts per layer, activating top-2 experts per token (13B active parameters). Architecture: 32 layers, hidden dimension 4096, FFN dimension 14336, 32k vocabulary, 32k context length.
- **DeepSeek-MoE-16B-Base** (Dai et al., 2024): 16B total parameters with 64 routed experts plus 2 shared experts per layer, activating top-6 routed experts. Architecture: 28 layers, hidden dimension 2048, fine-grained expert architecture.
- **Phi-3.5-MoE** (Abdin et al., 2024): 41.9B total parameters with 16 experts per layer, activating top-2 experts (6.6B active parameters). Architecture: 32 layers, hidden dimension 4096, 128k context length.
- **Qwen2-57B-A14B** (Yang et al., 2024a): 57B total parameters with 64 experts per layer, activating top-8 experts (14B active parameters). Architecture: 28 layers, hidden dimension 3584, fine-grained expert architecture.

All models are loaded in bfloat16 precision following the original model releases. For models with shared experts (DeepSeek-MoE), the shared experts are excluded from compression to preserve cross-expert information flow, consistent with D^2-MoE (Gu et al., 2025).

Datasets. *Calibration Data.* Two calibration configurations are used depending on the method:

- **PRIMO**: 128 sequences of 2048 tokens from C4 (Raffel et al., 2020), following the standard setup in NAEE (Lu et al., 2024) and MoE-Pruner (Xie et al., 2024).

- **D^2-MoE**: 512 random samples from WikiText-2 (Merity et al., 2017), following the original implementation.

Results use each method's default calibration setup. The C4 dataset is chosen for PRIMO due to its comprehensive coverage of web text, avoiding domain-specific bias (Williams & Aletras, 2024).

Perplexity Evaluation. Language modeling capability is assessed via perplexity on: (1) WikiText-2 validation set (Merity et al., 2017), (2) Penn Treebank (PTB) (Marcus et al., 1993), and (3) C4 (Raffel et al., 2020). Sequence length is set to 2048 tokens. Lower perplexity indicates better language modeling quality.

Zero-Shot Downstream Tasks. Model performance is evaluated on seven benchmarks using the EleutherAI LM Evaluation Harness (Gao et al., 2021), following the D^2-MoE evaluation protocol:

- **Commonsense Reasoning**: HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), PIQA (Bisk et al., 2020), ARC-Easy, ARC-Challenge (Clark et al., 2018), OpenBookQA (Mihaylov et al., 2018)
- **Mathematical Reasoning**: MathQA (Amini et al., 2019)

Extended Evaluation. We additionally evaluate on:

- **World Knowledge**: MMLU (5-shot) (Hendrycks et al., 2021)
- **Mathematical Reasoning**: GSM8K (8-shot) (Cobbe et al., 2021)
- **Code Generation**: HumanEval (pass@1) (Chen et al., 2021), with MBPP (Austin et al., 2021) used as selector samples in Appendix A.19

A.12.2 Method-Specific Settings

MI-Constrained Decomposition. The quadratic program in Eq. (8) is solved with OSQP (Appendix A.7.2). Key hyperparameters:

- **KL budget**: $\delta \leq 0.1 \cdot I(X;Y)$.
- **Covariance regularization**: $\lambda = 10^{-6}$ for numerical stability in Σ_ξ estimation.
- **Routing distribution**: Empirical frequencies $\{p_i\}_{i=1}^N$ estimated from calibration data via forward passes.

The base weight $\mathbf{W}_b$ and delta weights $\{\Delta\mathbf{W}_i\}$ are computed once and reused across all Pareto configurations.

PGD Low-Rank Compression. The projected gradient descent algorithm is configured as follows:

- **Initialization**: Truncated SVD of $\Delta\mathbf{W}_i\mathbf{S}$, retaining the top-$r$ singular components.
- **Step size**: Armijo line search with initial step size $\eta_{\max} = 1.0$, contraction factor $\beta = 0.5$, and sufficient decrease parameter $c = 10^{-4}$.
- **Convergence criterion**: Relative objective change $< 10^{-6}$ or maximum 100 iterations.
- **Input covariance**: $\mathbf{S} = (\hat{\Sigma}_X + \lambda\mathbf{I})^{1/2}$ estimated from calibration activations, with $\lambda = 10^{-4}$.

Task-Adaptive Selection. The task-adaptive mechanism operates at deployment time with the following settings:

- **Reference distribution**: Per-layer reference distributions $\{p_{\text{ref}}^{(l)}\}_{l=1}^L$ are computed from calibration data during frontier construction by aggregating expert selection frequencies across all calibration samples at each layer.
- **Additive smoothing**: To ensure numerical stability when computing KL divergence, the reference distributions are smoothed via Eq. (95) with $\epsilon_{\text{smooth}} = 10^{-6}$. This handles edge cases where an expert is inactive during calibration but active on the target task.

- **Target task samples**: 128 samples from the target task are used to estimate the per-layer routing distributions $\{p^{(l)}\}_{l=1}^{L}$ via Eq. (17).
- **Distribution shift**: The KL divergence D_{shift} (Eq. (18)) is computed as the average across all L MoE layers.
- **Scaling factor**: $\zeta = 1.0$ by default in Eq. (19).
- **Selection mechanism**: linear scan over the K pre-computed Pareto configurations in $\mathcal{F}$ using the adjusted threshold ϵ_{adj} (Section 4.4); the configuration with maximum compression satisfying $\mathcal{L}(\Theta) \leq \epsilon_{\text{adj}}$ is selected. No optimization is performed at deployment time; the offline-constructed frontier $\mathcal{F}$ and the per-layer base/delta weights remain valid for the target task.

Base Weight Sparsification. Following D^2-MoE (Gu et al., 2025), structured sparsification is applied to the base weight $\mathbf{W}_b$ with the following settings:

- **Importance score**: Let $\mathbf{X} \in \mathbb{R}^{B \times n}$ denote the calibration activation matrix with B samples. Column importance is computed as $\text{Imp}_j = \|\mathbf{W}_b[:, j]\|_2 \cdot \|\mathbf{X}[:, j]\|_2$, combining weight magnitude and activation magnitude.
- **Static pre-pruning**: Columns with the lowest importance scores are permanently removed at the static sparsity level below.
- **Dynamic threshold**: At inference time, the threshold is computed via percentile-based selection on GPU.
- **Sparsity levels**: Fixed at 10% static + 10% dynamic for performance-oriented compression, following D^2-MoE (Gu et al., 2025).

A.12.3 Pareto Search Space

The Pareto frontier construction uses the ϵ-constraint method with iterative optimization, while base weight sparsity follows a fixed configuration from D^2-MoE (Gu et al., 2025) for fair comparison:

- **Rank search space**: $r \in \{r_{\min}, r_{\min} + \Delta r, \ldots, r_{\max}\}$ where $r_{\max} = \lfloor \gamma mn/(m+n) \rfloor$ is determined by the complexity bound. Binary search is used to find the optimal rank satisfying the error constraint, rather than exhaustive enumeration.
- **Base weight sparsity** (fixed): 10% static + 10% dynamic for performance-oriented compression, following D^2-MoE (Gu et al., 2025).
- **Complexity bound**: $\gamma \in \{0.25, 0.30, 0.35, 0.40\}$ controls the maximum computational overhead from low-rank factors. Default: $\gamma = 0.30$.
- **Error bound**: $\epsilon \in \{0.03, 0.05, 0.08, 0.10\}$ represents the maximum allowed global average relative reconstruction error. As defined in Eq. (10), this is computed via two-level aggregation:

$$\mathcal{L}(\Theta) = \frac{1}{L} \sum_{l=1}^{L} \left(\frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_i^{(l)} \right) \leq \epsilon, \tag{106}$$

  where L is the number of layers and N is the number of experts per layer. Thus, $\epsilon = 0.05$ means 5% average relative reconstruction error across all $L \times N$ expert weight matrices in the model. Empirically, we observe that $\epsilon = 0.05$ corresponds to approximately 10% perplexity increase on WikiText-2.

The binary search converges in $O(\log R)$ iterations where $R = (r_{\max} - r_{\min})/\Delta r$.

A.12.4 Baseline Methods

PRIMO is compared against the following baseline methods. For fair comparison, all methods use identical evaluation protocols. Calibration data follows each method's original setup (see below) to ensure faithful reproduction of published results.

D^2-MoE (Gu et al., 2025). Delta decomposition with Fisher-weighted merging and truncation-aware SVD. Hyperparameters follow the official implementation (Table 12 of the D^2-MoE paper): calibration uses 512 samples from WikiText-2 with batch size 128, and the base weight is built by Fisher-weighted averaging. The pruning ratio of the base weights is 10% in the performance setting and 60% in the throughput setting, and the static and dynamic pruning sparsities are each 5% and 30% in the two settings. The SVD truncation thresholds, given as the singular-value energy preserved, are 68.05%, 47.34%, 26.62%, 16.26% and 5.93% at 20%, 40%, 60%, 70% and 80% compression in the performance setting, and 74.30%, 53.58%, 32.86%, 22.54% and 12.18% in the throughput setting.

MC-SMoE (Li et al., 2024). Expert merging via routing-guided clustering followed by low-rank decomposition. It merges by frequency-weighted averaging based on routing statistics, uses a low-rank factor of 32 and a final threshold of 0.10 on the S matrix, and is fine-tuned for 50 epochs with learning rate 3×10^{-5}, batch size 16 and 50 warmup steps, with knowledge distillation at temperature 2 and $\lambda_{\text{kd}} = 0.2$.

NAEE (Lu et al., 2024). Enumeration-based expert pruning that minimizes the token reconstruction loss of each MoE layer. It calibrates on 128 sequences of 2048 tokens from C4, enumerates all expert combinations for $N_E \leq 8$ experts, keeps $r \in \{4, 6, 7\}$ of the 8 experts, and, following the original paper, calibrates on the MATH dataset for math benchmarks.

A.12.5 Evaluation Metrics

Model quality and efficiency are assessed via the following metrics:

Language Modeling Quality.

- **Perplexity (PPL)**: Measures language modeling quality on WikiText-2, C4, and PTB. Computed with sequence length 2048. Lower is better.
- **Average Accuracy**: Arithmetic mean across the nine accuracy benchmarks (seven zero-shot tasks, MMLU 5-shot, GSM8K 8-shot). Higher is better.
- **Few-shot Accuracy**: MMLU (5-shot) and GSM8K (8-shot) following standard evaluation protocols.

Compression Efficiency.

- **Compression Ratio**: Defined as $1 - \text{Params}_{\text{comp}}/\text{Params}_{\text{orig}}$. For example, 60% compression means the compressed model has 40% of the original parameters. Compression ratios: 20%, 40% and 60% for quality (Table 2) and 60%, 70% and 80% for efficiency (Table 15).
- **Parameter Count**: Total number of parameters in the compressed model, including base weights and low-rank factors $(\mathbf{U}_i, \mathbf{V}_i)$.

Inference Efficiency.

- **Throughput (tokens/sec)**: Measured on batch size 64, sequence length 2048, averaged over 100 forward passes after 10 warmup iterations. Reported for single A100 GPU where possible.
- **Speedup**: Ratio of compressed model throughput to original model throughput: Speedup $=$ $\text{Throughput}_{\text{comp}}/\text{Throughput}_{\text{orig}}$.
- **Latency**: End-to-end generation time for 128 output tokens with batch size 1.
- **Memory Usage**: Peak GPU memory during inference, measured via `torch.cuda.max_memory_allocated()`.
- **Memory Reduction**: $1 - \text{Memory}_{\text{comp}}/\text{Memory}_{\text{orig}}$.

Information-Theoretic Quality.

- **MI Retention**: $I(X;\hat{Y})/I(X;Y)$, estimated via MINE (Belghazi et al., 2018) on calibration data. Measures how much information is preserved after compression.
- **Reconstruction Error**: $\|\mathbf{W}_{\text{orig}} - \mathbf{W}_{\text{comp}}\|_F / \|\mathbf{W}_{\text{orig}}\|_F$, averaged across all expert weight matrices.

A.12.6 Compression Ratio Configurations

Table 12 summarizes the hyperparameter configurations for different compression ratios on Mixtral-8×7B.

Table 12: Compression configurations for Mixtral-8×7B at different compression ratios. Rank ratio is determined via Pareto optimization; sparsity and SVD energy thresholds follow D^2-MoE (Gu et al., 2025) settings for each target ratio.

Compression	Rank Ratio	Sparsity	SVD Energy	Params	Active Params
20%	0.50	10%	68.05%	37.4B	10.4B
40%	0.35	20%	47.34%	28.0B	7.8B
60%	0.25	30%	26.62%	18.7B	5.2B
70%	0.20	40%	16.26%	14.0B	3.9B
80%	0.15	50%	5.93%	9.3B	2.6B

A.12.7 Reproducibility

To ensure reproducibility, the following measures are implemented:

Random Seeds. All random seeds are fixed: `seed=42` for NumPy, PyTorch, and CUDA. For calibration data sampling, the same random state is used across all experiments.

Deterministic Operations. `torch.backends.cudnn.deterministic=True` and `torch.backends.cudnn.benchmark=False` are set to ensure reproducible results across runs.

Computational Cost. The compression pipeline consists of three main stages: calibration data forward pass, MI-constrained decomposition with PGD optimization, and Pareto frontier construction. Peak GPU memory usage is comparable to D^2-MoE (Gu et al., 2025) on the same hardware.

A.12.8 Summary of Experimental Protocol

For clarity, the complete experimental protocol is summarized as follows:

1. **Data Preparation**: Sample 128 sequences (2048 tokens each) from C4 for calibration.
2. **Forward Pass**: Run calibration data through the original model to collect activations and routing statistics.
3. **MI-Constrained Decomposition**: Compute base weights $\mathbf{W}_b$ and delta weights $\{\Delta\mathbf{W}_i\}$ by solving Eq. (8) with OSQP.
4. **PGD Compression**: Compress delta weights into low-rank factors $\{\mathbf{U}_i, \mathbf{V}_i\}$ via activation weighting.
5. **Sparsification**: Apply semi-dynamical structured pruning to base weights following D^2-MoE (Gu et al., 2025).
6. **Deployment**: For each target task, compute the distribution shift D_{shift}, tighten the tolerance to ϵ_{adj}, and select Θ^* from $\mathcal{F}$ by a linear scan (Stage 3 of Algorithm 1); no optimization is run at deployment.

7. **Evaluation**: Measure perplexity on WikiText-2/C4/PTB and accuracy on zero-shot benchmarks.
8. **Efficiency Measurement**: Record throughput, latency, and memory usage.

This protocol is applied identically across all models and compression ratios, ensuring fair and consistent evaluation.

A.12.9 Model-Specific Configurations

Due to architectural differences across MoE models, certain hyperparameters are adjusted accordingly:

Mixtral-8×7B. Standard configuration with 8 experts per layer, top-2 routing. All 8 experts participate in compression. Router weights are used for frequency estimation. This is the primary benchmark model for comparison with D^2-MoE.

DeepSeek-MoE-16B-Base. Contains 64 routed experts plus 2 shared experts per layer. Shared experts are excluded from compression to preserve cross-expert information flow, consistent with D^2-MoE. Expert pruning targets routed experts only, with top-6 routing preserved. Due to the large number of experts ($N_E = 64$), enumeration-based methods (NAEE) are intractable; genetic search is used following Yang et al. (2024b).

Phi-3.5-MoE. 16 experts per layer with top-2 routing. Similar handling to Mixtral models. The 128k context length is preserved after compression.

Qwen2-57B-A14B. 64 experts per layer with top-8 routing. Fine-grained expert architecture similar to DeepSeek-MoE. Due to memory constraints, compression is performed layer-by-layer with gradient checkpointing.

A.12.10 Expert Frequency Estimation

Expert routing frequencies $\{p_i\}_{i=1}^{N}$ are estimated from the calibration data as follows:

$$p_i = \frac{1}{k \cdot |\mathcal{D}| \cdot T_{\text{seq}}} \sum_{x \in \mathcal{D}} \sum_{t=1}^{T_{\text{seq}}} \mathbf{1}[i \in \text{Top-}k(G(x_t))], \tag{107}$$

where $\mathcal{D}$ is the calibration dataset, T_{seq} is the sequence length, $G(\cdot)$ is the router function, and k is the number of activated experts. This frequency is used for:

- Routing weights p_i in the quadratic program of Eq. (8) (PRIMO) and in Fisher information computation (D^2-MoE)
- Expert importance ranking for pruning decisions (NAEE)
- Cluster assignment weights in merging methods (MC-SMoE)
- Reference distribution p_{ref} for task-adaptive selection in PRIMO

Following Xie et al. (2024), we verify that expert frequencies are relatively stable across different calibration subsets (coefficient of variation < 0.1), supporting the use of C4 as a general-purpose calibration source.

APPENDIX A.13: Algorithm Complexity Analysis

This appendix analyzes the computational complexity of PRIMO. Table 13 summarizes the notation. For Mixtral-8×7B: $L = 32$, $N = 8$, $m = 14336$, $n = 4096$ for the gate and up projections (transposed for the down projection), $d = 4096$, $K = 10$, $T = 100$.

Table 13: Notation for complexity analysis.

Symbol	Description	Symbol	Description
L	Number of MoE layers	K	Pareto frontier size
N	Experts per layer	T	PGD iterations
m, n	Expert weight dimensions	r	Target rank
d	Hidden dimension	$\lvert\mathcal{X}\rvert$	Sample count

A.13.1 Time Complexity

MI-Constrained Decomposition. The dominant costs are covariance estimation $O(|\mathcal{X}_{\text{cal}}| \cdot n^2)$ and QP matrix construction $O(N^2 \cdot mn)$ per layer. Solving the QP over the N-simplex via OSQP requires only $O(N^3)$, which is negligible since $N \ll \min(m, n)$. Total:

$$T_{\text{MI}} = O\left(L \cdot \left(|\mathcal{X}_{\text{cal}}| \cdot n^2 + N^2 \cdot mn\right)\right) \tag{108}$$

PGD Low-Rank Compression. Each PGD iteration involves residual computation, gradient updates, and factor balancing, totaling $O(mnr)$ per expert. For K compression levels with T iterations:

$$T_{\text{PGD}} = O(K \cdot T \cdot L \cdot N \cdot mnr) \tag{109}$$

This dominates the offline phase.

Task-Adaptive Selection. A forward pass estimates routing distributions, followed by KL computation and linear scan over $\mathcal{F}$:

$$T_{\text{Adapt}} = O(|\mathcal{X}_{\text{task}}| \cdot L \cdot N \cdot d + L \cdot N + K) \tag{110}$$

A.13.2 Space Complexity

The compressed model stores one shared base weight plus N low-rank factor pairs per layer:

$$S_{\text{compressed}} = O(L \cdot (mn + N \cdot (m + n) \cdot r)) \tag{111}$$

yielding compression ratio $1 - \frac{1}{N} - \frac{(m+n)\cdot r}{mn}$. Peak working memory is $O(n^2 + N \cdot mn)$ per layer.

A.13.3 Summary and Comparison

Table 14: Complexity comparison. PRIMO trades higher offline cost for task-adaptive deployment capability.

Method	Offline Time	Online Time	Adaptive?
NAEE	$O(L \cdot N \cdot mn)$	–	No
D^2-MoE	$O(L \cdot N \cdot mnr)$	–	No
PRIMO	$O(K \cdot T \cdot L \cdot N \cdot mnr)$	$O(\lvert\mathcal{X}_{\text{task}}\rvert \cdot LNd)$	**Yes**

PRIMO's offline cost exceeds baselines by factor $O(K \cdot T)$ due to Pareto exploration.

Parallelization. PRIMO admits natural parallelization: (1) layer level: MI decomposition and PGD process layers independently; (2) expert level: PGD optimizations within each layer are independent; (3) configuration level: different compression levels can be evaluated in parallel. With L GPUs, offline complexity reduces to $O(K \cdot T \cdot N \cdot mnr)$.

APPENDIX A.14: Memory Efficiency and Inference Throughput

Table 15 reports system-level efficiency on Mixtral-8×7B from 60% to 80% compression. Model size and memory are set by the compression ratio (18.68B/34.8GiB, 14.01B/26.1GiB and 9.34B/17.4GiB at 60%, 70% and 80%). PRIMO's throughput is within 2% of D^2-MoE at every ratio, reaching 317.39 tokens/sec at 80%.

Table 15: Throughput and memory efficiency of Mixtral-8×7B at high compression ratios. Throughput measured on A100 GPU with batch size 64, sequence length 2048, averaged over 100 iterations. TFLOPs estimated from model architecture. Model size and memory are the same for all methods at a given ratio: 18.68B and 34.8GiB at 60%, 14.01B and 26.1GiB at 70%, and 9.34B and 17.4GiB at 80%.

	Ratio 60%		Ratio 70%		Ratio 80%	
Method	TFLOPs	Tokens/sec	TFLOPs	Tokens/sec	TFLOPs	Tokens/sec
NAEE	481	271.89	386	272.66	290	278.53
MoE-I^2	838	227.60	743	252.55	647	294.04
LoSparse	1150	158.90	1240	191.45	1330	198.04
D^2-MoE	481	277.72	386	300.33	290	313.29
PRIMO (Ours)	**493**	**280.78**	**396**	**304.09**	**297**	**317.39**

APPENDIX A.15: EFFECTIVENESS OF MI-CONSTRAINED DECOMPOSITION

Table 16: Effectiveness of base weight construction strategies on Mixtral-8×7B. The Fisher row is D^2-MoE and the MI-based row is PRIMO (Table 2). PRIMO attains the lowest perplexity.

Method	**WikiText-2↓**	**PTB↓**	**C4↓**
Full Model (Jiang et al., 2024)	3.98	12.99	6.78
Mean average	7.66	46.85	24.39
TIES (Yadav et al., 2023)	12.45	87.31	29.10
RegMean (Jin et al., 2023)	187.19	1206.05	612.70
Frequency (Li et al., 2024)	6.42	35.12	13.79
Fisher (Gu et al., 2025)	5.28	20.54	10.10
MI-based (Ours)	**4.58**	**15.38**	**8.24**

Table 16 compares base weight construction strategies. PRIMO reaches a WikiText-2 perplexity of 4.58, against 5.28 for D^2-MoE with its Fisher-weighted base. RegMean degrades to 187.19 perplexity. With the delta compressor held fixed, the MI-constrained base lowers perplexity by 0.35 over the Fisher-weighted base (Table 17).

APPENDIX A.16: SYNERGY OF BASE AND DELTA OPTIMIZATION

Table 17: Ablation on delta weight compression on Mixtral-8×7B at 40% compression, varying the base construction (Fisher-weighted vs. MI-constrained) and the delta compressor (Vanilla SVD, Truncation-Aware SVD, PGD).

Compression Configuration		**WikiText-2↓**	**PTB↓**	**C4↓**
Full Model (Jiang et al., 2024)		3.98	12.99	6.78
Fisher-Weighted Base (D^2-MoE Baseline)	Vanilla SVD (90% energy)	6.22	22.54	10.72
	Truncation-Aware SVD	5.28	20.54	10.10
MI-Constrained Base (PRIMO)	Vanilla SVD (90% energy)	5.87	19.96	9.79
	Truncation-Aware SVD	4.93	17.96	9.17
	PGD Low-Rank	**4.58**	**15.38**	**8.24**

Table 17 dissects the contribution of PGD low-rank compression by fixing the base weight strategy to MI-constrained and varying the delta compression method. With the base fixed to MI-constrained, Vanilla SVD reaches 5.87 perplexity (6.22 with the Fisher-weighted base of D^2-MoE), Truncation-Aware SVD reaches 4.93, and PGD reaches 4.58, 0.35 below the truncation-aware baseline at the

same 40% compression ratio. Vanilla SVD keeps 90% of the singular-value energy of each delta, whereas PGD uses the rank selected by the search under ϵ and γ (Section 4.2).

APPENDIX A.17: PGD HYPERPARAMETER SENSITIVITY

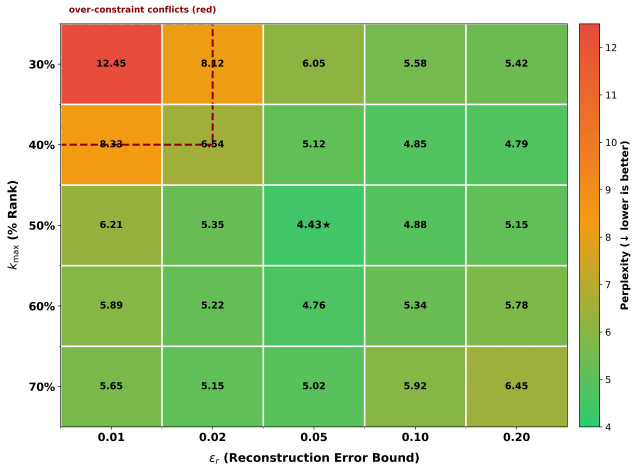


Figure 9: PGD hyperparameter sensitivity. WikiText-2 perplexity on Mixtral-8×7B (40% compression) across varying reconstruction error bounds (ϵ_r) and rank constraints ($k_{\max}$). The optimal region ($\star$) lies between over-constrained (red) and under-constrained (yellow) settings.

Figure 9 visualizes the interaction between the reconstruction error bound ϵ_r and the rank constraint $k_{\max}$. Performance forms a valley centered at an error bound of 0.05 and a rank constraint of 50%, which yields 4.43 perplexity. Over-constrained settings with strict error bounds degrade to 12.45 perplexity, and under-constrained settings degrade to 6.45.

APPENDIX A.18: IMPACT OF TASK-ADAPTIVE SELECTION

Table 18: Impact of task-adaptive selection under distribution shift. Comparison of Static Policy (fixed calibration config) versus PRIMO's Adaptive Policy across tasks with varying routing divergence (KL). Under routing shift, the adaptive policy tightens the error bound, and its gains grow with the shift (MMLU, GSM8K, HumanEval).

Task	**Routing Shift (KL Div.)**	**Static Policy (Fixed ϵ, k)**	**Adaptive Policy (PRIMO)**	**Improvement**
General (Low Shift)				
WikiText-2	0.00	4.58 (PPL)	4.58 (PPL)	+0.0%
C4	0.12	8.24 (PPL)	8.15 (PPL)	+1.1%
Specialized (High Shift)				
MMLU (5-shot)	0.35	54.1% (Acc)	60.1% (Acc)	**+11.1%**
GSM8K (8-shot)	0.48	41.2% (Acc)	53.2% (Acc)	**+29.1%**
HumanEval	0.62	21.5% (Pass@1)	33.5% (Pass@1)	**+55.8%**

Table 18 evaluates PRIMO under distribution shift. On WikiText-2, the static and adaptive policies coincide. On tasks with high routing divergence, the static policy degrades most on HumanEval (21.5% Pass@1); task-adaptive selection measures the shift, tightens the error bound, deploys a less compressed configuration from $\mathcal{F}$, and raises Pass@1 to 33.5%, a 55.8% relative improvement.

APPENDIX A.19: SELECTOR SENSITIVITY

Sample-Size Sensitivity. The task-adaptive selector requires a small set of target-task samples to estimate the routing distribution shift D_{shift}. Table 19 reports the number of samples required to recover progressively larger fractions of the static-vs-adaptive performance gap on HumanEval (the most distribution-shifted task in Table 18). Even 5 samples recover $\sim$28% of the gap; 50 samples recover $\sim$81%; performance saturates around 100 samples.

Table 19: Selector sample-size sensitivity on Mixtral-8×7B at 40% compression. HumanEval Pass@1 (%).

Target Samples	Pass@1 (%)	Recovery vs Static
0 (static)	21.5	0.0%
5	24.8	27.5%
10	26.8	44.2%
50	31.2	80.8%
100	33.1	96.7%
500 (saturated)	33.5	100.0%

Sample-Quality Sensitivity. We further test whether the target samples must be drawn from the deployment distribution itself. Table 20 shows that even mismatched samples (using WikiText-2 text to calibrate the selector for a HumanEval code-generation task) still improve over the static baseline. Matched samples recover most of the gap (96.7% at 100 samples); partially matched samples (MBPP, also code) recover 81.7% at 100 samples; severely mismatched WikiText-2 still gives a $+4.6$pp improvement at 50 samples.

Table 20: Selector sample-quality sensitivity on Mixtral-8×7B at 40% compression. HumanEval Pass@1 (%).

Target Samples	HumanEval (matched)	MBPP (partial)	WikiText-2 (mismatched)
0 (static)	21.5	21.5	21.5
50	31.2	28.3	26.1
100	33.1	31.3	27.6

APPENDIX A.20: MI RETENTION VS. DOWNSTREAM PERFORMANCE

To relate MI retention to downstream quality, we sweep the MI-retention level from 70% to 99% on Mixtral-8×7B and record both WikiText-2 perplexity and average zero-shot accuracy across the 7 commonsense reasoning benchmarks. The full uncompressed model has PPL $= 3.98$ and accuracy $= 0.63$.

Table 21: MI retention vs. downstream performance on Mixtral-8×7B (WikiText-2 PPL $\downarrow$, accuracy $\uparrow$). Below 80% retention each further 5% drop costs > 1.0 PPL; above 95% gains are marginal (< 0.3 PPL). The knee at 90% (highlighted) balances compression aggressiveness and quality preservation.

MI Retention (%)	WikiText-2 PPL	Avg Accuracy	Δ PPL vs Full
70	9.14	0.43	$+5.16$
75	7.31	0.45	$+3.33$
80	6.02	0.51	$+2.04$
85	5.15	0.53	$+1.17$
90 (default)	**4.58**	**0.58**	$\mathbf{+0.60}$
95	4.25	0.59	$+0.27$
99	4.08	0.60	$+0.10$

The 90% retention level serves as a compression-domain proxy for downstream task quality: it sits at the knee of the trade-off curve, beyond which additional MI retention yields diminishing returns.

This empirical finding complements the Corollary in Appendix A.7.3, which gives the retention level implied by a KL budget.

APPENDIX A.21: OFFLINE COMPRESSION COST

PRIMO's multi-stage pipeline (MI-constrained QP $\rightarrow$ PGD compression $\rightarrow$ Pareto enumeration) runs once, before deployment.

Table 22: Offline compression cost on Mixtral-8$\times$7B at 40% compression ($K = 10$ Pareto configurations on a 4×4 grid). Wall-clock time is one-time; once compressed, PRIMO's inference throughput is within 2% of D^2-MoE (Appendix A.14).

Method	Type	Offline (min)	Peak Memory (GB)
DiEP (Bai et al., 2025)	Pruning	13.8	139.0
MC-MoE (Huang et al., 2025)	Quantization	18.6	348.4
D^2-MoE (Gu et al., 2025)	Low-rank delta	55.0	131.0
PRIMO (Ours)	**Low-rank delta**	**91.3**	**135.2**
SVD-LLM V2 (Wang et al., 2025)	Low-rank	112.8	132.6
ResMoE (Ai et al., 2025)	Residual	191.5	124.5

PRIMO (91.3 min) sits between D^2-MoE (55 min) and SVD-LLM V2 (112.8 min), and is roughly 2$\times$ faster than ResMoE (191.5 min). The $\sim$36-minute overhead over D^2-MoE is the cost of (i) the convex QP solve for MI-constrained base-weight construction, and (ii) the per-configuration PGD refinement of $\Delta\mathbf{W}_i$ factors at $K = 10$ frontier points. DiEP and MC-MoE are faster; MC-MoE reaches a peak memory of 348.4GB. For scaling to 100B+ models, each stage operates layer-by-layer and the K Pareto configurations are mutually independent, enabling parallelization across GPUs. Once compressed, PRIMO's inference throughput is within 2% of D^2-MoE (Appendix A.14), so offline cost is amortized over the model's full serving lifetime.