JORA: JOINT EXPERT-OUTPUT REWEIGHTING FOR TRAINING-FREE MOE ACCELERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Skipping experts lowers the cost of mixture-of-experts (MoE) inference, but can substantially degrade model quality. The resulting approximation depends on the retained experts and the coefficients used to combine their outputs. We introduce JORA, a training-free method that jointly estimates these coefficients after expert selection to approximate the original routed mixture. Offline calibration records expert-output inner products. At inference, these statistics and the current routing weights define a small regularized system, avoiding evaluation of omitted experts. Model parameters remain unchanged. JORA supports per-token prefill retention and shared-pool decoding; our analysis establishes optimality for the fixed-set calibration objective and characterizes sensitivity to calibration mismatch. Across text and vision-language benchmarks, JORA outperforms the evaluated expert-skipping and re-routing baselines in most task–budget comparisons, with improvements under prefill-only, decode-only, and joint reduction. In single-GPU vLLM serving, Qwen3-30B-A3B with native prefill and a 32-expert decode pool achieves a maximum observed end-to-end speedup of $1.31\times$ over Native for 256 input and 1,024 output tokens at concurrency 64. Across 26 matched serving configurations, JORA adds a median 4.38% to Drop's end-to-end latency while retaining at least 93.62% of its output throughput.

1 INTRODUCTION

Mixture-of-experts (MoE) models increase capacity without evaluating every expert for every token (Shazeer et al., 2017; Fedus et al., 2022). A router selects a small subset of experts and assigns weights to their outputs (Yang et al., 2025; Muennighoff et al., 2024). Further inference savings are possible by skipping some of these experts (Lu et al., 2024b; Xu et al., 2026) or coordinating their use across a batch (Wu et al., 2026a; Gupta et al., 2024). Both strategies change the mixture passed to subsequent layers. Selecting fewer experts is therefore only part of the problem: the outputs that remain must still approximate what the original route would have produced.

The original router weights do not account for the omitted contributions. Rescaling and calibrated folding can reduce this mismatch (Chen & Yao, 2026; Wu et al., 2026b), but adjustments based on isolated expert pairs need not work well together. If two retained outputs share a component, separate pairwise corrections can overcount it. Conversely, outputs that are poor substitutes individually may approximate an omitted contribution when combined. The appropriate coefficient of one expert thus depends on the other outputs available for that token. We use this dependence to fit the weighted sum of omitted contributions with all executed outputs jointly. Reweighting cannot recover components outside their span; it can reduce the error in how the available outputs are combined.

We introduce **JORA**, joint expert-output reweighting for training-free MoE acceleration (Figure 1). For each layer, offline calibration evaluates experts on common inputs and records their output inner products in a Gram table. Given the selected set and current routing weights, JORA uses this table to solve a small regularized system for all retained coefficients. Only the coefficients adapt to the current route and execution set: the tables and model parameters remain fixed, and omitted experts are not evaluated online.

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.

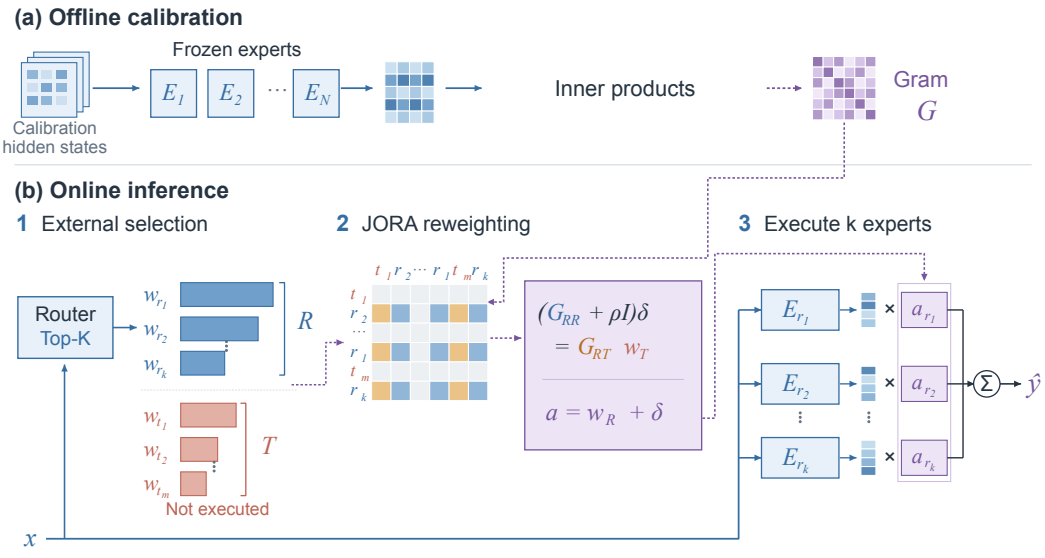


Figure 1: **Set-conditioned reconstruction with JORA.** (a) Common calibration states yield a layer-scaled Gram G. (b) For deletion, $R \subseteq A$ and $T = A \setminus R$. The system $(G_{RR} + \rho I)\delta = G_{RT} w_T$ couples the retained outputs to the weighted omitted mixture and gives $a = w_R + \delta$. Re-routed targets use the zero-extended anchor in Section 3. Only selected routed experts execute; shared experts and residual paths stay unchanged. Solid/dashed lines carry activations/statistics; colors and widths are schematic.

The selection policy is separate from this coefficient estimation. JORA takes the native route and selected expert set as inputs, so native-ranked prefill retention, shared-pool decoding, and other expert-skipping policies can use the same reconstruction objective.

Our contributions are:

- We provide a post-selection reweighting procedure that can be combined with different expert-selection policies. Given the native route and a selected set of k experts, one online $k \times k$ system jointly estimates their coefficient adjustments from calibrated output statistics.
- We decompose output error into a component outside the retained span and an aggregation error. For a fixed set, we establish optimality under the regularized calibration objective and bound sensitivity to statistical mismatch and numerical error.
- On Qwen3-30B-A3B, adding JORA to NAEE, Alloc-MoE, Ban, and ASET lowers cached-continuation perplexity on WikiText-2, PTB, and C4 (Appendix A.3.1). Cross-model task comparisons and same-input diagnostics assess quality and reconstruction; single-GPU experiments quantify serving gains and the additional runtime of reweighting relative to matched Drop.

2 Related Work

2.1 Expert Selection and Static Compression

Expert skipping reduces the number of experts evaluated for each token. NAEE studies post-training pruning and skipping (Lu et al., 2024b). ACE combines parameter-derived contribution proxies with router gates (Xu et al., 2026), whereas elbow-based routing adapts expert counts from router probabilities (Pan et al., 2026). Alloc-MoE uses sensitivity profiling and routing scores to allocate budgets across layers and tokens (Liu et al., 2026). Other policies account for token importance or modality: MC-MoE protects important tokens during dynamic pruning (Huang et al., 2025), and MoDES applies layer- and modality-aware skipping to multimodal models (Huang et al., 2026).

Static compression instead changes the deployed expert bank. REAP ranks experts by router-weighted output norms (Lasby et al., 2026), and EEP searches pruning and parameter-merging configurations without gradients (Liu et al., 2024a). MC-SMoE aligns neurons through weight matching, uses routing-guided grouping and frequency-weighted merging, and then applies low-rank and sparse compression (Li et al., 2024). MAESTRO combines cross-layer Markov-based expert ranking with pruning and attention-LoRA recovery (Goel et al., 2026). PreMoE compiles domain-specific or multidomain pools from calibration-based router-logit utility (Pei et al., 2026). These methods reduce the stored expert bank, which need not reduce the number of experts executed per token.

2.2 Batch-aware routing

Batch-aware routing reduces weight-access costs by coordinating tokens' expert choices. SERE re-routes assignments using expert similarity and a batch-level expert pool (Wu et al., 2026a). Lynx consolidates low-affinity assignments (Gupta et al., 2024), and OEA reuses experts activated elsewhere in the batch (Oncescu et al., 2025). XShare chooses shared subsets using aggregate routing utility (Vankov et al., 2026). The batch expert union measures a different resource from per-token assignments or resident parameters, so equal retention ratios need not imply matched latency.

2.3 Output reweighting and reconstruction

Reconstruction is used in post-training mask tuning (Kwon et al., 2022), expert-output merging (Miao et al., 2025), and MoE-to-dense pruning and distillation (Kim et al., 2026). GRAIL computes a Gram-based ridge map and merges it into downstream weights (Tang et al., 2026).

For reduced expert mixtures, reference-mass scaling adjusts a shared gain while preserving relative weights (Chen & Yao, 2026). ExFold formulates budgeted output approximation and implements it through offline pairwise scalar calibration, online loss-based target selection, and weight folding; a confidence-based extension distributes part of a source's weight across several targets (Wu et al., 2026b). C3PO optimizes test-time mixing weights through reference-example surrogate objectives (Li et al., 2025).

JORA applies calibrated reconstruction to route-dependent execution sets while keeping model parameters frozen. The retained-output Gram and current weighted omitted mixture define one joint coefficient system, allowing aggregation coefficients to adapt to the selected set using cached output geometry.

3 JORA: set-conditioned mixture reconstruction

3.1 Reconstructing the routed output

After an expert-selection policy chooses which experts to execute, JORA adjusts their aggregation weights to approximate the original routed output. For a token state x, let A be the native Top-K expert set and w their native aggregation weights. Write $E_i(x) \in \mathbb{R}^d$ for expert i's output and F_A for the matrix whose columns are these outputs. The original mixture is then $y = F_A w$. Shared experts and residual branches remain unchanged.

Suppose the policy selects a nonempty set R of $k \leq K$ experts, leaving $T = A \setminus R$ unexecuted. We use the native weights of experts still present in R as reference coefficients a^0. If re-routing introduces an expert outside A, its reference coefficient is zero because it had no contribution to the original mixture. In the usual deletion setting, this simply gives $a^0 = w_R$. With F_R and F_T collecting the corresponding expert outputs, a coefficient adjustment δ gives

$$\widehat{y} = F_R a, \qquad a = a^0 + \delta, \qquad y - \widehat{y} = \underbrace{F_T w_T}_{r} - F_R \delta. \tag{1}$$

Here r is the contribution lost by omitting T, while $F_R\delta$ is the change obtained by reweighting the executed outputs. JORA chooses δ to reduce the mismatch between them. Offline output statistics make this adjustment possible without evaluating the omitted experts during inference.

3.2 OFFLINE ESTIMATION OF OUTPUT GEOMETRY

Following calibrated Gram reconstruction (Tang et al., 2026), we collect input states x_u at each routed layer ℓ during frozen native-model runs. All N_ℓ experts in the layer are evaluated on the same selected input states. Taking the weighted average of their output inner products gives

$$\widetilde{G}_{ij} = \sum_u \omega_u E_i(x_u)^\top E_j(x_u), \qquad G = \widetilde{G}/s_\ell, \tag{2}$$

where $\omega_u \geq 0$ and $\sum_u \omega_u = 1$. In the main experiments, $\omega_u = 1/n_\ell$ for n_ℓ calibration states, and $s_\ell = \mathrm{tr}(\widetilde{G})/N_\ell > 0$ is their mean output energy across all routed experts. The scale is computed offline, not tuned, and remains fixed during inference.

Diagonal entries encode output energies; off-diagonal entries encode inner products. The shared scale preserves the relative magnitudes of these entries. Evaluating all experts on common inputs with nonnegative weights makes G positive semidefinite. Online coefficient estimation gathers G_{RR} and G_{RT} from the fixed table. Appendix A.2 details the main calibration; Appendix B describes the diagnostic configurations and their use of fixed layer scales.

3.3 ONLINE SET-CONDITIONED COEFFICIENT ESTIMATION

We fit $F_R\delta$ to the omitted mixture r. For the current route, let $q = G_{RT}w_T$ and $e_0 = w_T^\top G_{TT} w_T$. Holding (A, w, R) fixed across the calibration inputs in Equation (2) gives the weighted-average squared reconstruction error, scaled by $1/s_\ell$:

$$\begin{aligned} E_G(\delta) &= e_0 - 2\delta^\top q + \delta^\top G_{RR}\delta, \\ \delta^\star &= \arg\min_{\delta\in\mathbb{R}^k} J_G(\delta), \qquad J_G(\delta) = E_G(\delta) + \rho\|\delta\|_2^2, \quad \rho > 0. \end{aligned} \tag{3}$$

Each entry of q is a scaled weighted-average inner product between a retained output and the weighted omitted mixture. The block G_{RR} couples their coefficient estimates through inner products among retained outputs. The ridge penalty $\rho\|\delta\|_2^2 = \rho\|a - a^0\|_2^2$ limits departures from the reference coefficients. The resulting system is

$$H\delta^\star = q, \qquad H = G_{RR} + \rho I_k, \qquad a = a^0 + \delta^\star. \tag{4}$$

Here I_k is the $k \times k$ identity. Inference gathers G_{RR} and G_{RT}; the constant e_0 is not needed. For each layer and inference phase, G and ρ remain fixed across routes. The coefficients may be signed because the solve is unconstrained. They are applied after the expert nonlinearities without softmax, renormalization, clipping, or shrinkage.

Algorithm 1 uses the table when its cached offline validity flag is $v_G = 1$. The operation $\mathrm{CholSolve}$ performs a Cholesky factorization followed by two triangular solves and returns $\perp$ on failure. The function finite checks all coefficient entries. If the table or solve is unusable, the finite fallback coefficient vector b_R is set to a^0 for deletion or to the selector-mapped coefficients for re-routing. Mapped coefficients enter only the fallback, not the joint solve. Fallback events are counted separately.

3.4 PREFILL AND BATCH-POOL DECODING

Prefill and batched decode place different demands on expert selection. Reducing the experts per token removes token–expert assignments, but a decoding batch may still access the same distinct experts. When weight access limits runtime, a shared expert pool can constrain this batch-level cost more directly.

Our phase-specific configuration combines per-token prefill retention with batch-pool decode. Prefill retains the native-ranked set $R_t = A_t[1{:}k_P]$, including in joint phase reduction. For decode, we adopt ExFold's dynamic pool selector and source-to-target mapping (Wu et al., 2026b). Let $U = \cup_t A_t$ over the current decode tokens at one layer. With offline output-norm estimates h_e and pool cap D, the selector gives

$$S = \mathrm{Top}_{\min(D,|U|),\, e\in U}\left(h_e \sum_t w_{t,e}\right), \qquad R_t = \{\pi_S(e) : e \in A_t\}, \tag{5}$$

Algorithm 1 JORA: online reconstruction at one routed MoE layer

Require: $x, (A, w), R, \{E_e\}$; G, ρ, v_G, b_R
Ensure: $\widehat{y}$
$T \leftarrow A \setminus R$, $k \leftarrow |R|$
$a^0 \leftarrow \mathbf{0}_k$; $a^0_e \leftarrow w_e$ for $e \in A \cap R$ ▷ Zero-extend native weights
if $T = \varnothing$ **then return** $\sum_{e \in A} w_e E_e(x)$
$a \leftarrow b_R$ ▷ Fallback
if $v_G = 1$ **then**
  $H \leftarrow G_{RR} + \rho I_k$, $q \leftarrow G_{RT} w_T$
  $\widetilde{\delta} \leftarrow \mathrm{CholSolve}(H, q)$
  if $\widetilde{\delta} \neq \perp$ **then**
    if $\mathrm{finite}(a^0 + \widetilde{\delta})$ **then** $a \leftarrow a^0 + \widetilde{\delta}$
  end if
end if
return $\widehat{y} = \sum_{e \in R} a_e E_e(x)$ ▷ Execute R only

where $w_{t,e} = 0$ outside A_t. The mapping π_S keeps experts in S and maps each omitted source to a target in S using the calibrated pair loss. Merging duplicate targets gives $|\cup_t R_t| \leq \min(D, |U|)$ and $|R_t| \leq K$. The pool is recomputed at each layer and decoding step; tokens may execute different subsets of it, rather than every expert in S. JORA then estimates coefficients on each R_t using the original route and Equation (4), rather than the folded coefficients. If $D \geq |U|$, native execution is unchanged.

We denote this configuration by (k_P, D_{pool}). The $P_{k_P} D_{k_D}$ controls instead retain native-ranked experts per token in both phases; they isolate reweighting without imposing a batch-pool cap. Their measured quality and diagnostic results are reported separately from the pool evaluation specified in Appendix A.4.

In continuous batching, scheduler metadata identifies prefill and decode tokens. A decode-pool cap applies only to the latter: prompt tokens may require additional experts in the same batch. Appendix C.6 describes phase boundaries and mixed-batch execution.

For $k_t = |R_t|$ and $m_t = |A_t \setminus R_t|$, coefficient estimation costs $O(\sum_t [k_t m_t + k_t^3])$, independent of expert widths. Pure deletion gives $m_t = K - k_t$. A layer's FP32 Gram occupies $O(N_\ell^2)$ storage, or 64 KiB for 128 experts per phase. Let C_0 be unchanged execution time, M_K and M_R the native and selected expert-path times, and C_J the added selection, mapping, coefficient, and scheduling overhead not already included in M_R. If these times do not overlap,

$$\mathrm{Speedup} = \frac{C_0 + M_K}{C_0 + M_R + C_J} > 1 \quad \Longleftrightarrow \quad C_J < M_K - M_R. \tag{6}$$

Here the expert path includes dispatch, padding, communication, and weight access. When execution overlaps, the gain must instead be measured end to end; Appendix A.5 reports serving times and separate coefficient-path diagnostics.

3.5 Reconstructability and estimation guarantees

The span of the executed outputs determines which parts of the omitted mixture can be reconstructed. At a given input, let $P_R = F_R F_R^{\dagger}$ be the orthogonal projector onto this span, with $\dagger$ denoting the Moore–Penrose pseudoinverse. For a fixed nonempty R and the reference weights a^0 from Section 3.1,

$$\|r - F_R \delta\|_2^2 = \underbrace{\|(I - P_R) r\|_2^2}_{\text{outside the retained span}} + \underbrace{\|P_R r - F_R \delta\|_2^2}_{\text{aggregation mismatch}}. \tag{7}$$

The first term is unaffected by coefficient changes. Reweighting acts on the second term by fitting the component of the omitted mixture within the retained span (Figure 6). This decomposition holds for both deletion and re-routing.

For the calibrated objective, $G \succeq 0$ and $\rho > 0$ make H positive definite, so Equation (4) has a unique solution. Comparing it with any correction δ_b on the same R and reference weights gives

$$J_G(\delta_b) - J_G(\delta^\star) = \|\delta_b - \delta^\star\|_H^2, \tag{8}$$

$$E_G(0) - E_G(\delta^\star) = q^\top H^{-1} q + \rho\|\delta^\star\|_2^2 \geq 0, \tag{9}$$

where $\|u\|_H^2 = u^\top H u$. Equation (8) establishes optimality for the common regularized objective. Equation (9) also guarantees lower calibrated reconstruction error than the unadjusted reference when $q \neq 0$; for deletion, that reference is direct dropping. The ridge penalty bounds the adjustment through $\rho\|\delta^\star\|^2 \leq e_0$. Proofs are in Appendix C.

The joint solve computes the correction for the entire omitted mixture in one system. Fitting each omitted expert separately would give the same aggregate correction if every fit used all of R, the same Gram, and the same ridge penalty (Appendix C.4). Restricting those fits to fewer targets, or imposing a shared gain, can change the solution. Equation (8) measures the resulting gap under the common objective.

These guarantees concern the calibrated Gram, which may differ from the output geometry conditional on a route. Appendix C.5 combines calibration mismatch and numerical solver error in a residual, then bounds the excess conditional regularized loss by the squared residual divided by ρ.

4 Experiments

We evaluate task quality on Qwen3-30B-A3B across five task families, treating per-token prefill retention and batch-shared decode pools as different budgets. Same-input diagnostics test whether the calibrated Gram improves reconstruction on held-out states. The appendix extends these comparisons to other models, tasks, and expert selectors.

4.1 Evaluation setup

Models and calibration. Qwen3-30B-A3B ($K = 8$) is the main task-evaluation model; Qwen1.5-MoE-A2.7B ($K = 4$) and DeepSeek-V2-Lite ($K = 6$) extend the architecture coverage (Yang et al., 2025; Qwen Team, 2024a; DeepSeek-AI, 2024). Qwen2-57B-A14B Base ($K = 8$) is also included in the cross-model comparison (Qwen Team, 2024b). OLMoE-1B-7B-0924 ($K = 8$) remains in the perplexity and fidelity experiments (Muennighoff et al., 2024). The experimental environment is specified in Appendix A.1. Gram tables and regularization remain fixed during evaluation. Further calibration settings and diagnostics are described in Appendices A.2 and B; the Qwen3-VL extension and benchmark allocation are in Appendix D.

Comparison methods. Native executes the original route. Drop and JORA retain the same native-ranked experts in the prefill comparison and differ only in their aggregation coefficients. ExFold retains its original selector (Wu et al., 2026b). MC-MoE-ODP uses online dynamic pruning; REAP and MC-SMoE change the resident expert bank and occupy a separate comparison block. For batched decode, ExFold and JORA share dynamic pool selection and source-to-target mapping. Joint reduction uses equal prefill counts and decode-pool caps, but JORA retains native-ranked prefill experts while ExFold uses its original prefill selector. Baseline-specific budgets, including SERE's batch-union policy, are defined in Appendix A.3.2.

Tasks and metrics. Table 1 compares three phase settings on ten tasks spanning knowledge, commonsense, mathematics, instruction following, and code. All reported downstream task scores are arithmetic means over three evaluation runs with fixed settings. Generated-answer evaluations use temperature zero, one completion per item per run, and at most 4,096 new tokens per completion. Task-specific scorers evaluate the responses, with pass@1 for code. Serving metrics are specified in Table 6. Generation settings and scoring rules are described in Appendices A and D.

4.2 Task quality and phase-specific budgets

Task quality at half the prefill budget. With prefill activation reduced from eight to four experts per token and native decode unchanged, JORA exceeds ExFold on all ten tasks in Table 1(a), by

Table 1: **Qwen3-30B-A3B task quality: prefill-only, batched decode-only, and joint reduction.** Scores are percentages (↑); code uses pass@1. Avg. is the unweighted mean of the ten displayed task scores. Bold marks column maxima among non-Native methods within each of (a)–(c), including ties. Native and static-compression rows are unbolded references. t_k: experts/token; d_D: shared-pool cap at active $B = 32$; S_4: SERE's variable-size batch union. Native (batch) is the separate reference for (b,c). CSQA/OBQA/SIQA denote CommonsenseQA/OpenBookQA/SocialIQA. Budget definitions are in Appendix A.3.2. Scores in (a) use `eager_tokenwise_v1`.

			Knowledge		Commonsense			Math		Instr.	Code		
Method	P	D	MMLU Pro	GPQA D	CSQA	OBQA	SIQA	GSM8K	MATH 500	IFEval	Human Eval+	MBPP+	Avg.
(a) Prefill-only reduction; native decode													
Native	t_8	t_8	67.90	53.03	80.21	90.27	76.13	93.78	86.60	84.97	85.98	75.22	79.41
Drop	t_4	t_8	62.07	41.41	76.77	84.73	73.87	92.62	82.33	78.87	79.88	72.57	74.51
JORA	t_4	t_8	**67.04**	**50.51**	**79.44**	88.27	**74.31**	**93.23**	**87.33**	**83.92**	**84.76**	**75.31**	**78.41**
MC-MoE-ODP	ODP$_{.5}$	t_8	66.18	47.47	77.75	**88.73**	73.51	92.12	86.13	82.01	81.10	74.43	76.94
ExFold	t_4	t_8	66.31	48.48	77.61	86.20	73.15	92.75	87.07	82.69	82.32	74.87	77.15
(b) Batched decode-only reduction; native prefill; active $B = 32$													
Native (batch)	t_8	t_8	67.78	53.03	80.04	90.13	76.19	93.76	86.73	85.03	85.98	75.22	79.39
ExFold	t_8	d_{64}	67.83	44.95	80.21	89.27	**76.31**	92.57	87.67	84.04	84.76	73.81	78.14
JORA	t_8	d_{64}	**67.99**	**53.03**	**80.62**	**89.93**	76.13	**93.99**	**89.40**	83.55	**85.98**	**74.34**	**79.50**
ExFold	t_8	d_{32}	67.12	45.45	79.25	88.33	75.37	92.47	86.47	79.61	81.71	72.75	76.85
JORA	t_8	d_{32}	67.76	41.41	80.48	89.13	76.25	93.66	88.07	79.73	83.54	73.54	77.36
SERE	t_8	S_4	67.05	45.45	79.36	89.07	76.15	92.24	87.00	**86.20**	82.32	**74.34**	77.92
(c) Joint prefill and batched decode reduction; active $B = 32$													
ExFold	t_4	d_{64}	67.07	42.42	78.35	84.27	74.65	92.87	87.13	**82.50**	84.76	73.90	76.79
JORA	t_4	d_{64}	**67.15**	**48.48**	**79.22**	**86.00**	74.41	**93.18**	**88.13**	82.19	**85.37**	74.60	**77.87**
ExFold	t_4	d_{32}	66.96	42.42	78.41	83.93	74.67	92.70	86.07	77.20	80.49	74.25	75.71
JORA	t_4	d_{32}	66.91	47.47	79.14	85.40	**74.74**	93.15	86.20	77.88	81.71	**74.69**	76.73
Static compression													
REAP	R64		60.90	35.35	69.92	77.93	68.78	88.98	83.53	70.06	76.83	65.08	69.74
MC-SMoE	M		63.92	39.39	75.57	85.07	73.13	92.17	81.80	78.87	79.88	72.49	74.23

0.26–2.44 percentage points. ExFold uses its original selector, so this comparison evaluates the complete methods at the same prefill budget. Drop and JORA instead share native-ranked selection, isolating the aggregation rule.

Against Drop, JORA improves all ten task scores: GPQA-Diamond rises from 41.41% to 50.51% (+9.10 percentage points), CommonsenseQA from 76.77% to 79.44%, and GSM8K from 92.62% to 93.23%. The gains on MMLU-Pro and MATH500 are 4.97 and 5.00 percentage points. Dynamic-pruning and static-compression rows retain their distinct budget definitions.

Phase-specific budgets. Table 1(b) preserves native prefill and constrains the shared decode pool at active batch size $B = 32$, with $D_{\text{pool}} \in \{64, 32\}$. Panel (c) combines these pool budgets with four-expert prefill and compares the complete phase-specific policies. A pool cap counts distinct experts across the batch, not experts per token; SERE instead forms a variable-size union of primary experts.

Transfer across architectures and selectors. Table 13 reports joint-phase results on Qwen2-57B-A14B Base, Qwen1.5-MoE-A2.7B, and DeepSeek-V2-Lite. The per-token prefill budget is halved, and the decode pool is capped at 50% of each layer's routed expert bank, with active batch size $B = 32$. Drop and JORA use the same selection rules and differ in aggregation. Visual tasks are reported separately (Table 14). Beyond native-ranked retention, JORA lowers cached-continuation PPL with NAEE, Alloc-MoE, Ban, and ASET on Qwen3; Table 5 gives the results and realized-budget qualifications.

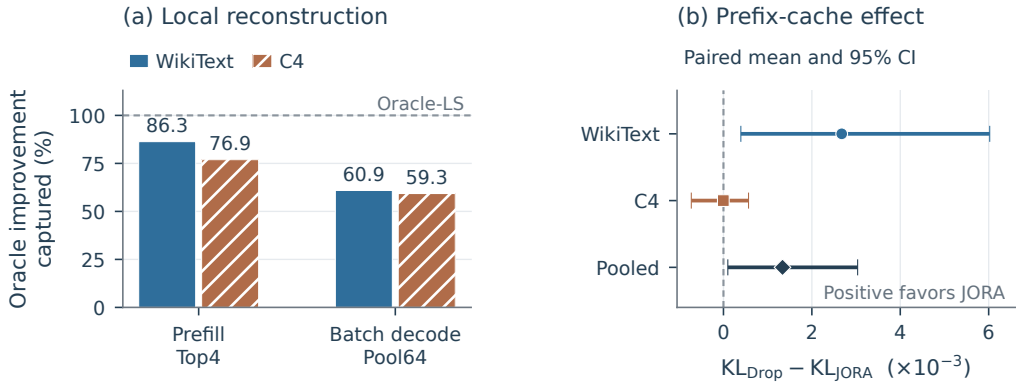


Figure 2: **Calibrated reconstruction and prefix-cache effects on Qwen3-30B-A3B.** (a) Fraction η of Oracle-LS's local improvement over Drop captured on native-input states; pool decode uses $B = 32$. (b) A separate $B = 2$ intervention applies Top4 only during prefill, then transfers its KV cache to native Top8 suffix computation. Points and 95% paired document-bootstrap intervals show $\Delta\mathrm{KL} = \mathrm{KL}_{\mathrm{Drop}} - \mathrm{KL}_{\mathrm{JORA}}$ over targets 2–128; positive values favor JORA. Both panels use 64 WikiText and 64 C4 selection-set documents with $\rho = 10^{-3}$. Protocols and a summary of the controls are in Appendix B.5.

4.3 Reconstruction and Predictive Fidelity

To test the averaged Gram, we hold inputs, native routes, and retained sets fixed across coefficient rules. Oracle-LS evaluates the actual expert outputs and gives the minimum reconstruction error within their retained span. The independently validation-selected Qwen3-30B-A3B configuration captures 82.62–83.71% of this Oracle improvement over Drop across 256 held-out documents (Table 10). The fraction measures recoverable local error, not task accuracy.

Complementary Qwen3 diagnostics examine shared-pool reconstruction and prefix-cache propagation on 128 reused selection-set documents (Figure 2). On native-input states, JORA captures 81.60% of the pooled Oracle improvement for Top4 prefill and 60.36% for Pool64 decode; total layer-scaled error falls by 13.34% and 5.43%, respectively. In the separate cache intervention, all-layer Top4 prefill followed by native decode reduces pooled KL to Native by 7.63%. Its paired difference interval excludes zero; WikiText improves while C4 changes little. The latter comparison isolates an effect transmitted through prefix KV rather than through reduced-expert suffix computation.

Local reconstruction gains need not translate into lower final hidden-state error or better predictions. We examine these differences through full-model fidelity, coefficient ablations, and longer-prefix evaluations in Tables 11 and 3 and Appendix B.

4.4 Serving Efficiency

We measure Qwen3-30B-A3B under single-GPU vLLM continuous batching (Kwon et al., 2023), including selection, reweighting, and dispatch. Figure 3 separates controlled phase timings from request-level latency. Drop uses the same execution budgets and selection rules, isolating the additional cost of JORA's reweighting.

The largest observed E2E gain is $1.31\times$ with native prefill and Pool32 decode on $256 \to 1024$ requests at concurrency 64. The same trace gives $1.32\times$ TPOT speedup and $1.31\times$ output throughput. Prefill-only serving reaches $1.15\times$ E2E speedup on $2048 \to 32$ at concurrency 32, while joint Top4/Pool32 reduction reaches $1.12\times$ on $2048 \to 256$ at concurrency 64. Controlled engine windows reach $1.30\times$ for prefill ($L = 512$, $B = 8$) and $1.34\times$ for decode ($L = 512$, $B = 128$, Pool32). Decode gains are not monotonic in batch size: both methods slow down at $B = 96$ in Figure 3(b).

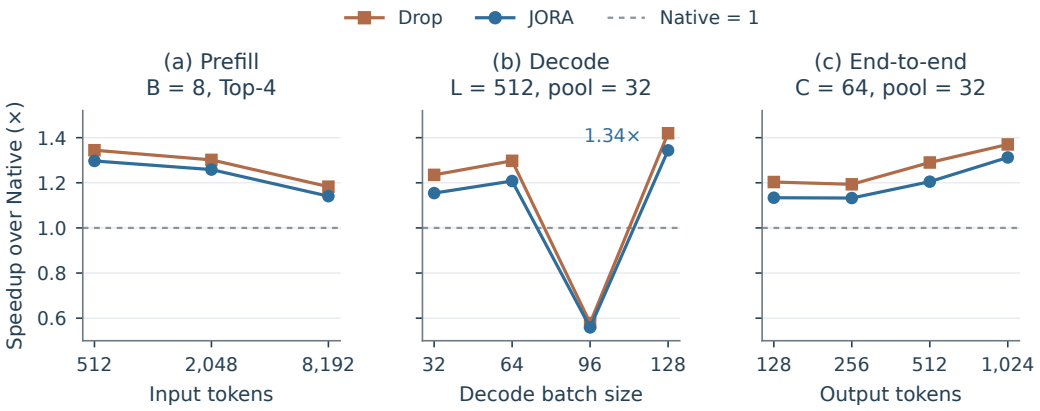


Figure 3: **Phase and workload dependence of speedup on Qwen3-30B-A3B.** (a) Top4 prefill at fixed $B = 8$. (b) Pool32 decode over 128 steps, starting at context length $L = 512$. These panels use Native/method GPU-window time, excluding scheduler preparation. (c) Mean E2E latency ratios with native prefill, Pool32, 256 input tokens, and client concurrency 64. Values above one indicate improvement. Appendix A.5 reports absolute timings, Pool64 comparisons, and the remaining workloads.

Runtime cost of reweighting. Across 26 matched serving configurations, JORA increases mean E2E latency by 3.16–7.05% relative to Drop (median 4.38%) and retains 93.62–96.93% of its output throughput. At the peak E2E configuration, the 4.40% latency increment consumes 11.88% of Drop's latency saving over Native; when that saving is small, reweighting can eliminate it. The isolated coefficient entry costs 8.98–27.32 μs per call across the sampled phase–budget configurations, averaged over three layers. Appendix A.5 defines the timing boundaries and reports the full sweep.

5 Conclusion

JORA estimates aggregation coefficients after expert selection, using calibrated output geometry and the current native mixture. It supports token-wise prefill and batch-pool decoding without changing model parameters. The fixed-set solve minimizes the regularized calibration objective, with sensitivity to statistical mismatch characterized by our analysis. At half the native prefill budget on Qwen3-30B-A3B with native decode, JORA improves all ten task scores over Drop and ExFold. The selector experiments also show lower cached-continuation PPL when JORA is applied after NAEE, Alloc-MoE, Ban, and ASET. Same-input reconstruction improves in both phases, although predictive effects remain model- and budget-dependent. Serving measurements place the cost of reweighting at a 3.16–7.05% latency increment over matched Drop. Whether reduced expert execution offsets this cost depends on sequence lengths, batch size, and pool budget.

AI USE DISCLOSURE

We used AI tools solely for language polishing to improve the clarity and rigor of the manuscript, and for assisting with the search and organization of related work. All research ideas, methodology, experiments, analysis, and conclusions were developed and conducted by the authors. The authors take full responsibility for the final content of this work.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. URL https://arxiv.org/abs/2108.07732.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025. URL https://arxiv.org/abs/2511.21631.

Mark Chen, Jerry Tworek, Heewoo Jun, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Xing Chen and Hengshuai Yao. Training-free halving of activated experts in fine-grained mixture-of-experts models. *arXiv preprint arXiv:2609.04575*, 2026. URL https://arxiv.org/abs/2609.04575.

Yuanteng Chen, Peisong Wang, Yuantian Shao, Nanxin Zeng, Chang Xu, and Jian Cheng. Ban&Pick: Enhancing performance and efficiency of MoE-LLMs via smarter routing. *arXiv preprint arXiv:2509.06346*, 2026. URL https://arxiv.org/abs/2509.06346v3. Version 3.

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. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024. URL https://arxiv.org/abs/2405.04434.

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

Palaash Goel, Ayush Maheshwari, and Tanmoy Chakraborty. It takes a MAESTRO to prune bad experts. *arXiv preprint arXiv:2607.08601*, 2026. URL https://arxiv.org/abs/2607.08601v2. Version 2.

Vima Gupta, Jae Hyung Ju, Kartik Sinha, Ada Gavrilovska, and Anand Padmanabha Iyer. Lynx: Enabling efficient MoE inference through dynamic batch-aware expert selection. *arXiv preprint arXiv:2411.08982*, 2024. URL https://arxiv.org/abs/2411.08982. Revised May 2026.

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*, 2021a. URL https://arxiv.org/abs/2009.03300.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021b. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

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. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/abc1943857a42935ceacff03c524bb44-Abstract-Conference.html.

Yushi Huang, Zining Wang, Zhihang Yuan, Yifu Ding, Ruihao Gong, Jinyang Guo, Xianglong Liu, and Jun Zhang. MoDES: Accelerating mixture-of-experts multimodal large language models via dynamic expert skipping. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026. URL https://arxiv.org/abs/2511.15690.

Junhyuck Kim, Jihun Yun, Haechan Kim, Gyeongman Kim, Joonghyun Bae, and Jaewoong Cho. Pruning and distilling mixture-of-experts into dense language models. *arXiv preprint arXiv:2605.28207*, 2026. URL https://arxiv.org/abs/2605.28207.

Woosuk Kwon, Sehoon Kim, Michael W. Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. A fast post-training pruning framework for transformers. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/987bed997ab668f91c822a09bce3ea12-Abstract-Conference.html.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023. URL https://arxiv.org/abs/2309.06180.

Mike Lasby, Ivan Lazarevich, Nish Sinnadurai, Sean Lie, Yani Ioannou, and Vithursan Thangarasa. REAP the experts: Why pruning prevails for one-shot MoE compression. *arXiv preprint arXiv:2510.13999*, 2026. URL https://arxiv.org/abs/2510.13999v3. Version 3, revised May 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. *arXiv preprint arXiv:2310.01334*, 2024. URL https://arxiv.org/abs/2310.01334v2. Version 2.

Zhongyang Li, Ziyue Li, and Tianyi Zhou. C3PO: Critical-layer, core-expert, collaborative pathway optimization for test-time expert re-mixing. In *Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2504.07964.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023. URL https://arxiv.org/abs/2305.20050.

Baihui Liu, Kaiyuan Tian, Wei Wang, Zhaoning Zhang, Linbo Qiao, and Dongsheng Li. AllocMoE: Budget-aware expert activation allocation for efficient mixture-of-experts inference. *arXiv preprint arXiv:2604.08133*, 2026. URL https://arxiv.org/abs/2604.08133.

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*, 2024a. URL https://arxiv.org/abs/2407.00945.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In

Advances in Neural Information Processing Systems, volume 36, 2023a. URL https://papers.nips.cc/paper_files/paper/2023/hash/43e9d647ccd3e4b7b5baab53f0368686-Abstract-Conference.html.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. MMBench: Is your multi-modal model an all-around player? In *European Conference on Computer Vision*, 2024b. URL https://arxiv.org/abs/2307.06281.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xucheng Yin, Cheng-lin Liu, Lianwen Jin, and Xiang Bai. OCRBench: On the hidden mystery of OCR in large multimodal models. *arXiv preprint arXiv:2305.07895*, 2023b. URL https://arxiv.org/abs/2305.07895.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. MathVista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations*, 2024a. URL https://arxiv.org/abs/2310.02255.

Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024b. URL https://aclanthology.org/2024.acl-long.334/.

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/.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 2263–2279, 2022. URL https://aclanthology.org/2022.findings-acl.177/.

Minesh Mathew, Viraj Bagal, Rubèn Pérez Tito, Dimosthenis Karatzas, Ernest Valveny, and C. V. Jawahar. InfographicVQA. *arXiv preprint arXiv:2104.12756*, 2021a. URL https://arxiv.org/abs/2104.12756.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. DocVQA: A dataset for VQA on document images. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021b. URL https://arxiv.org/abs/2007.00398.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

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. URL https://arxiv.org/abs/2510.14436.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018. URL https://arxiv.org/abs/1809.02789.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, et al. OLMoE: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*, 2024. URL https://arxiv.org/abs/2409.02060.

Costin-Andrei Oncescu, Qingyang Wu, Wai Tong Chung, Robert Wu, Bryan Gopal, Junxiong Wang, Tri Dao, and Ben Athiwaratkun. Opportunistic expert activation: Batch-aware expert routing for faster decode without retraining. *arXiv preprint arXiv:2511.02237*, 2025. URL https://arxiv.org/abs/2511.02237.

Robin Pan, Raymond Liu, Daniel Fang, Adelina Andrei, and Rosa Wu. Elbow-based MoE routing: A training-free inference time plugin for expert selection. *arXiv preprint arXiv:2608.04401*, 2026. URL https://arxiv.org/abs/2608.04401v1.

Zehua Pei, Ying Zhang, Hui-Ling Zhen, Tao Yuan, Xianzhi Yu, Zhenhua Dong, Sinno Jialin Pan, Mingxuan Yuan, and Bei Yu. PreMoE: Proactive inference for efficient mixture-of-experts. *arXiv preprint arXiv:2505.17639*, 2026. URL https://arxiv.org/abs/2505.17639v4. Version 4, revised July 2026.

Qwen Team. Qwen1.5-MoE: Matching 7b model performance with 1/3 activated parameters. Technical report, 2024a. URL https://qwenlm.github.io/blog/qwen-moe/.

Qwen Team. Qwen2-57B-A14B: Base model card. Hugging Face model card, 2024b. URL https://huggingface.co/Qwen/Qwen2-57B-A14B. Performance table, accessed September 25, 2026.

Qwen Team. Qwen3-VL-30B-A3B-Instruct: Model card and configuration, 2025. URL https://huggingface.co/Qwen/Qwen3-VL-30B-A3B-Instruct. Configuration consulted September 20, 2026.

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

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof Q&A benchmark. *arXiv preprint arXiv:2311.12022*, 2023. URL https://arxiv.org/abs/2311.12022.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019. URL https://arxiv.org/abs/1907.10641.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. SocialIQA: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019. URL https://arxiv.org/abs/1904.09728.

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. *arXiv preprint arXiv:1701.06538*, 2017. URL https://arxiv.org/abs/1701.06538.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards VQA models that can read. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. URL https://arxiv.org/abs/1904.08920.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2018. URL https://arxiv.org/abs/1811.00937.

Wenwu Tang, Dong Wang, Lothar Thiele, and Olga Saukh. GRAIL: Post-hoc compensation by linear reconstruction for compressed networks. In *Conference on Parsimony and Learning*, volume 328 of *Proceedings of Machine Learning Research*, pp. 881–895, 2026. URL https://proceedings.mlr.press/v328/tang26a.html.

Daniil Vankov, Nikita Ivkin, Kyle Ulrich, Xiang Song, Ashish Khetan, and George Karypis. XShare: Collaborative in-batch expert sharing for faster MoE inference. *arXiv preprint arXiv:2602.07265*, 2026. URL https://arxiv.org/abs/2602.07265.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ad236edc564f3e3156e1b2feafb99a24-Abstract.html.

Juntong Wu, Jialiang Cheng, Fuyu Lv, Dan Ou, and Li Yuan. SERE: Similarity-based expert re-routing for efficient batch decoding in MoE models. In *International Conference on Learning Representations*, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/5d07eb2335d88b1dfcefcc3d2da2a230-Abstract-Conference.html.

Juntong Wu, Yifei Liu, Junyi Chen, Siqi Fan, Chaoran Feng, Minghao Li, Liujie Zhang, Weihang Chen, and Li Yuan. ExFold: Unified expert folding for training-free MoE prefill-decode acceleration. *arXiv preprint arXiv:2608.24938*, 2026b. URL https://arxiv.org/abs/2608.24938.

Zukang Xu, Zhixiong Zhao, Xing Hu, Jiangyong Yu, Houji Wen, Jun Li, Zhe Jiang, and Dawei Yang. ACE: Adaptive calibration-free expert skipping for MoE-based LLMs. *arXiv preprint arXiv:2609.05228*, 2026. URL https://arxiv.org/abs/2609.05228.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Haoqi Yang, Luohe Shi, Qiwei Li, Zuchao Li, Ping Wang, Hao Huang, and Hai Zhao. Faster MoE LLM inference for extremely large models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 43133–43151. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.2140. URL https://aclanthology.org/2026.findings-acl.2140/.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. URL https://arxiv.org/abs/2311.16502.

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. URL https://aclanthology.org/P19-1472/.

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

A EXPERIMENTAL DETAILS AND EXTENDED COMPARISONS

A.1 EVALUATION PROTOCOL

Correctness tests and warmup are excluded from the reported results. Compared methods use the same evaluation protocol within each task. Appendix A.4 specifies shared-pool decoding; Appendix D gives task-specific scoring. Serving and online-path measurements are reported in Appendix A.5.

Experimental environment. All experiments use NVIDIA H20-3e 141GB GPUs, BF16 model weights, and non-thinking inference. Eight GPUs are available for independent single-GPU jobs; the exploratory likelihood and cache-intervention experiments each use four. The NVIDIA driver is 580.105.08 (CUDA 13.0 support). Serving uses vLLM continuous batching, with Qwen3-30B-A3B on one GPU per process (tensor and pipeline parallelism both one); independent measurement groups run on separate cards. The validation-selected diagnostics (Appendix B) use vLLM 0.28 V1, eager and batch-invariant execution, Triton MoE, and FlashAttention-2, with FP64 statistical accumulation and Oracle calculations and FP32 logits. The cache intervention (Appendix B.5) uses Transformers 4.57.6 and PyTorch 2.10.0+cu128, eager attention, FP32 coefficient solves, and FP64 metric accumulation.

Standard-Prefill PPL. The standard protocol uses the complete WikiText-2 and PTB test texts and 128 valid 2,048-token C4 windows (Merity et al., 2017; Marcus et al., 1993; Raffel et al., 2020). Compared methods share tokenization, concatenation boundaries, BOS/EOS policy, loss masks, and window stride. Perplexity over scored positions $\mathcal{I}$ is

$$\mathrm{PPL}_P = \exp\left(\frac{\sum_{t\in\mathcal{I}} -\log p(x_t \mid x_{<t})}{|\mathcal{I}|}\right). \tag{10}$$

Full-block forwards use the prefill budget, so changing only the decode budget leaves this measurement unchanged. Table 2 gives the scored-token counts.

Cached-Suffix PPL. The cached-suffix protocol uses up to 128 non-overlapping windows with a 1,024-token prefix and a 256-token suffix. The first suffix prediction occurs at the prefill boundary; the remaining 255 use cached decode. Each method constructs its own KV cache under its prefill policy and teacher-forces the suffix under its decode policy. Tables report all-suffix PPL unless specified otherwise, with the scored-token counts in Table 2.

Table 2: **Scored-token counts for standard-prefill and cached-suffix PPL evaluation.** The two protocols are evaluated separately. Within each protocol/model/corpus, compared methods share the same evaluation samples and scored-token count. Cached totals include the prefill boundary prediction. Qwen3-VL uses text-only inputs here.

	Standard-prefill			Cached-suffix		
Model	WT2	PTB	C4	WT2	PTB	C4
OLMoE	288,729	98,056	262,016	32,768	19,456	32,768
Qwen1.5-MoE	299,077	95,194	262,016	32,768	18,944	32,768
DeepSeek-V2-Lite	311,124	96,861	262,016	32,768	19,200	32,768
Qwen3	299,077	95,194	262,016	32,768	18,944	32,768
Qwen3-VL	299,077	95,194	262,016	32,768	18,944	32,768

Task scoring. For our measured cross-model panels, MMLU reports pooled accuracy over 14,042 questions; ARC-Challenge and HellaSwag report length-normalized accuracy over 1,172 and 10,042 questions, respectively (Hendrycks et al., 2021a; Clark et al., 2018; Zellers et al., 2019). Standard likelihood scoring evaluates complete candidates through prefill. Cached-continuation scoring evaluates the first candidate token at the prefix boundary and subsequent candidate tokens through decode.

Generation settings and response scoring. Generated-answer evaluations use non-thinking, greedy decoding (temperature 0), one completion per example per run, and at most 4,096 new tokens with normal EOS termination. Top-$p = 0.95$ is configured but inactive under vLLM's temperature-zero decoding. Likelihood/PPL/KL evaluations and fixed-length serving workloads retain their own protocols.

Methods share prompts, parsers, and scorers within each comparison. Code tests run in a network-isolated, resource-limited sandbox; incomplete subsets are not reported as full-dataset results. Truncations, parse failures, invalid programs, timeouts, and empty outputs follow the task scorer.

A.2 Offline geometry and matched coefficient controls

For the main calibration, we collect frozen native trajectories on examples separate from the scored test sets. OLMoE-1B-7B-0924, Qwen1.5-MoE-A2.7B, DeepSeek-V2-Lite, Qwen2-57B-A14B Base, and Qwen3-30B-A3B each use 128 WikiText-2 training articles (`wikitext-2-raw-v1`). Qwen3-VL-30B-A3B-Instruct uses 67 GSM8K training requests and 61 TextVQA training requests with images. Each example supplies 16 states per layer, giving $n_\ell = 2{,}048$; all six models use $\rho = 10^{-5}$. The independent diagnostics use the settings in Appendix B.

On Qwen3-VL (128 routed experts, native Top8), JORA changes only language-backbone MoE computations at visual and textual positions (Bai et al., 2025; Qwen Team, 2025). Image resolution, resizing/tiling, encoder, projector, and image-token counts remain fixed across methods.

All routed experts receive the same states with $\omega_u = 1/n_\ell$. The scale $s_\ell = \mathrm{tr}(\widetilde{G})/N_\ell$ averages their squared output norms, without router weighting or division by hidden width. Thus $\mathrm{tr}(G)/N_\ell = 1$ up to rounding, and ridge ρI on G equals $\rho s_\ell I$ on $\widetilde{G}$. Tables and scales remain fixed across inference batches.

Gram construction and storage. The matrix in Equation (2) is positive semidefinite: for any $v \in \mathbb{R}^{N_\ell}$,

$$v^\top G v = \frac{1}{s_\ell} \sum_u \omega_u \left\| \sum_{i=1}^{N_\ell} v_i E_i(x_u) \right\|_2^2 \geq 0.$$

We accumulate inner products in high precision over token blocks and store G in FP32. Offline checks of positive scale, finite entries, symmetry, and matrix construction determine the validity flag $v_G \in \{0, 1\}$ in Algorithm 1; inference does not rescan the table.

Matched coefficient definitions. The coefficient controls fix $R \subseteq A$, $T = A \setminus R$, $q = G_{RT} w_T$, $H = G_{RR} + \rho I$, and the layer scale. Direct Drop uses $\delta = 0$; weight-mass rescaling uses $a = \gamma w_R$ with $\gamma = (\sum_{i \in A} w_i)/(\sum_{i \in R} w_i)$ for a nonzero denominator. Calibrated common scaling minimizes the ridge objective on $\delta = \beta w_R$:

$$\beta^\star = \frac{w_R^\top q}{w_R^\top H w_R}, \qquad \delta_{\mathrm{common}} = \beta^\star w_R \tag{11}$$

for nonzero w_R, and zero correction otherwise.

For each omitted source t, the one-/two-target controls minimize its sourcewise ridge objective over sets $U_t \subseteq R$ of size at most one/two. Candidate coefficients are $c_U = (G_{UU} + \rho I)^{-1} G_{Ut}$ and aggregate as $\delta = \sum_{t \in T} w_t \, \mathrm{embed}_R(c_{U_t})$, without shrinkage. The diagonal control replaces G_{RR} by $\mathrm{diag}(G_{RR})$, leaving q unchanged. Full-source ridge uses $U_t = R$ for all sources and is algebraically identical to JORA (Appendix C.4); this control checks numerical equivalence. Two-target fitting also has full support when $k \leq 2$, so this comparison uses $k = 4$.

Local error measurement. The Direct Drop/JORA local NMSE in Table 3 uses 256 held-out documents, equally divided between WikiText training and C4 validation holdouts, with eight Native-trajectory prefill states per document at each of 48 layers. On these 2,048 states per layer, output NMSE is

$$\mathrm{NMSE}(M) = \frac{\sum_t \|y_t - \widehat{y}_t^M\|_2^2}{\sum_t \|y_t\|_2^2}. \tag{12}$$

Table 3: **Coefficient-rule comparison on Qwen3-30B-A3B at Top4 prefill.** The local NMSE diagnostic and the PTB/MMLU-Pro benchmarks use separate evaluation configurations. Expert reduction is applied only during prefill; MMLU-Pro generation keeps decode at the native expert budget. Full-source ridge checks numerical equivalence to JORA.

Coefficient rule	Local NMSE ↓	PTB PPL ↓	MMLU-Pro (%) ↑
Direct drop	0.009241	18.78	62.13
Weight-mass rescaling	0.008012	18.22	63.33
Calibrated common scaling	0.007124	17.57	64.08
Sourcewise, one target	0.007412	17.78	63.91
Sourcewise, two targets	0.006889	17.45	64.36
Diagonal retained Gram	0.006143	16.89	65.07
Full-source ridge	0.004117	16.46	67.00
JORA	0.004112	16.46	67.00

Direct Drop and JORA share inputs, routes, retained sets, and denominator. The validation-selected diagnostic configuration differs from the PTB and MMLU-Pro benchmark configurations. Appendix B gives the Oracle comparisons and calibration-size sensitivity.

A.3 Extended quality and baseline comparisons

Supporting perplexity comparisons. Table 4 reports the full-block PPL sweep at half-budget prefill retention on five checkpoints, including OLMoE but not Qwen2. JORA was tuned on the scored corpora; these exploratory same-set results are separate from task-accuracy runs and do not exercise decode.

Table 4: **Standard-prefill perplexity at native-ranked half-budget retention (lower is better).** Native retains K experts and reduced methods retain $\lceil K/2 \rceil$; decode is not exercised. JORA is selected on the scored corpus, so these are exploratory same-set results rather than held-out tests. Qwen3-VL uses text-only inputs.

Model	Method	WikiText-2	PTB	C4
OLMoE-1B-7B-0924 ($K = 8$)	Native	6.7233	12.5348	10.2077
	Drop	8.5786	15.2605	12.0793
	JORA	8.3416	13.7619	11.6949
Qwen1.5-MoE-A2.7B ($K = 4$)	Native	6.5685	11.6884	8.3078
	Drop	6.8756	12.2381	8.6886
	JORA	6.8013	12.0320	8.5853
DeepSeek-V2-Lite ($K = 6$)	Native	5.7007	10.8784	8.0853
	Drop	5.9799	11.5171	8.4539
	JORA	5.8740	11.4309	8.2468
Qwen3-30B-A3B ($K = 8$)	Native	7.8376	15.2784	12.4270
	Drop	8.7931	18.3570	14.4129
	JORA	8.3800	16.5739	13.4175
Qwen3-VL-30B-A3B-Instruct ($K = 8$)	Native	6.2574	12.3718	10.6342
	Drop	7.3248	15.3629	12.8209
	JORA	6.8100	13.7440	11.2836

A.3.1 Reweighting after expert-skipping selectors

We test whether JORA improves aggregation beyond fixed native-ranked retention on Qwen3-30B-A3B. The four selectors are NAEE's general Top-k skipping rule with budget-matched parameters (Lu et al., 2024b), Alloc-MoE (Liu et al., 2026), the Ban component of Ban&Pick without Pick (Chen et al., 2026), and a Qwen3 adaptation of ASET (Yang et al., 2026). For each selector, we compare its baseline aggregation (Original) with JORA at target average routed-expert counts of four and six. These targets are not fixed per-token counts. Paired runs use the same selection

rule and frozen selection parameters, but propagate their own hidden states and KV caches, so later retained sets and realized budgets can differ.

Table 5 reports fixed-window, cached-continuation PPL on WikiText-2, PTB, and C4. Each method builds its own cache from the prefix and teacher-forces the continuation incrementally; PPL is computed from aggregate token NLL. These token-wise comparisons are separate from the full-block measurements in Table 4 and the shared-pool evaluation in Appendix A.4.

Table 5: **JORA with expert-skipping selectors on Qwen3-30B-A3B.** Cached-continuation PPL (↓), shown as Original → +JORA after selection. Target $\bar{k}$ is the requested average expert count, not a fixed per-token count.

Selector	Target $\bar{k}$	WikiText-2	PTB	C4
Native	8	8.0813	14.5021	11.9336
NAEE-skip	4	9.6084 → 8.8950	17.4339 → 15.0849	14.0759 → 13.0429
NAEE-skip	6	8.2274 → 8.1945	14.8961 → 14.8702	12.2273 → 12.0979
Alloc-MoE	4	14.8623 → 8.8685	30.9342 → 16.8362	20.2061 → 13.2577
Alloc-MoE	6	9.2424 → 8.3435	17.4420 → 15.2617	13.6174 → 12.2500
Ban	4	9.2835 → 8.7372	17.1950 → 15.9776	13.7411 → 12.8902
Ban	6*	8.5039 → 8.3192	15.5551 → 15.1082	12.6342 → 12.2813
ASET	4†	11.5558 → 9.7204	18.8158 → 16.4453	17.9742 → 14.5412
ASET	6	8.2743 → 8.1550	14.9165 → 14.7538	12.3868 → 12.1481

*Ban does not reach target six; the reported realized averages are approximately 5.06–5.09. †At target four, ASET's Original and +JORA paths differ in realized average expert count by more than 0.2; this pair is not a strictly matched-budget comparison.

JORA lowers PPL in all 24 reported selector–budget–corpus pairs. For Alloc-MoE at target budget four, PTB PPL decreases from 30.9342 to 16.8362; relative PPL reductions across the three corpora range from 34.39% to 45.57%. At target six, the corresponding reductions are 9.73–12.50%. NAEE and ASET also show smaller improvements at target six than at target four. Ban does not reach the requested six-expert average, while ASET's target-four pair differs in realized average count by more than 0.2; the latter is therefore not evidence of a strictly matched-budget gain. The panel supports transfer of the coefficient estimator across the tested selection rules, but does not measure task accuracy or serving speed.

A.3.2 Native baseline budgets and phase adapters

Prefill uses native-ranked Top-k_P for Drop and JORA. ExFold retains its original prefill selector, so joint comparisons match budgets rather than prefill supports. Its decode-only comparison with JORA shares the pool selector and mapping (Appendix A.4).

SERE's S_4 forms the batch union of each token's Top4 primary experts at similarity threshold $\tau_S = 0$, independently of JORA's ρ (Wu et al., 2026a). The union is formed before execution-only phase grouping. Its realized cost depends on the batch expert union and fallback routes.

MC-MoE-ODP evaluates the Online Dynamic Pruning component of MC-MoE (Huang et al., 2025) at the common weight precision, with its mixed-precision component disabled. ODP$_{.5}$ is a nominal half-native-budget operating point; protected tokens may retain more experts, so average and quantile executed counts define its realized budget.

REAP's R64 retains 64 resident routed experts per layer on the 128-expert Qwen3 backbones; generic R denotes a validation-selected, model-specific resident count (Lasby et al., 2026). M denotes the MC-SMoE static-compression configuration (Li et al., 2024). These budgets affect both phases and differ from per-token t_k.

A.4 SHARED-POOL DECODING

ExFold and JORA share the norm estimates, pair-loss table, tie-breaking rule, and pool budget in Equation (5) (Wu et al., 2026b). Duplicate targets are merged, giving $|R_t| \leq K$. ExFold uses mapped coefficients; JORA solves from the native route (A_t, w_t) and zero-extended anchor a_t^0. Numerical fallback uses finite mapped coefficients on the same R_t.

Comparison conditions. Decode-only runs retain native prefill with common duplicate handling, dispatch, and backend. Layer replay fixes inputs, routes, pools, and execution sets; full-model runs propagate each method's own states and cache, so later routes may differ. Quality comparisons fix request groups and refill order and use normal task termination. Serving comparisons match request inputs, order, and client-concurrency caps (Appendix A.5).

Batch sizes and pool budgets. The main Qwen3 pool comparisons use active $B = 32$ and $D \in \{64, 32\}$, with native or Top4 prefill. The cap applies to the batch union, not each token's expert count. For $U = \bigcup_t A_t$, execution uses at most $\min(D, |U|)$ distinct experts. When $D \geq |U|$, routing is unchanged; $B = 1, D \geq 8$ and $B = 8, D \geq 64$ are therefore no-pruning controls for native Top8 routing.

Prefill-only reduction leaves decode unrestricted; decode-only and joint reduction respectively retain native and reduced prefill. Serving workloads and timing boundaries are specified in Appendix A.5.

Cross-model joint setting. Table 13 uses $k_P = K/2$, $D/N = 50\%$, and active $B = 32$, excluding shared experts. Qwen2-57B-A14B Base, Qwen1.5-MoE-A2.7B, and DeepSeek-V2-Lite have $(K, N) = (8, 64), (4, 60), (6, 64)$ and thus $(k_P, D) = (4, 32), (2, 30), (3, 32)$. Native's 100% means unrestricted routing; dense and shared experts are unchanged. A 50% pool cap does not imply halving the executed union or latency.

Drop and JORA share native-ranked prefill selection and the decode-pool selector/mapping. Drop uses a^0 without redistribution or renormalization; JORA adds the joint correction. Native retains unrestricted routing. Methods share requests, scoring, and batch schedules.

Cached choice scoring predicts the first candidate token at the prefill boundary and later tokens through decode. Single-token candidates therefore provide only prefill evidence. Generated GSM8K answers provide a sustained-decode check.

A.5 SERVING PERFORMANCE AND ONLINE OVERHEAD

We compare Native, Drop, and JORA on Qwen3-30B-A3B. JORA uses the frozen Gram with $\rho = 10^{-5}$ and the full correction, without shrinkage. Drop follows the same budget and selection rules without joint reweighting. Paired comparisons share the physical GPU, model-residency session, engine configuration, and request inputs. Serving configurations contribute one complete post-warmup trace each. Controlled prefill windows average three passes; decode windows use one 128-step pass. Independent groups run on separate GPUs.

Request-level performance. Table 6 varies prefill input length and client concurrency with 32 output tokens. Table 7 varies the decode-pool cap, concurrency, and output length, and includes joint and mixed-length workloads. Each trace has $N = 2C$ requests, where C limits outstanding client requests rather than fixing the active decode batch. Output lengths are fixed for timing; the mixed trace contains 32 requests each of $256 \to 512$ and $8192 \to 64$. With send time t_s, first-token time t_f, last-token time t_l, and $n > 1$ output tokens, $\mathrm{TTFT} = t_f - t_s$, $\mathrm{E2E} = t_l - t_s$, and $\mathrm{TPOT} = (t_l - t_f)/(n-1)$. Output throughput divides generated tokens by trace wall time. Latency ratios divide the Native mean by the method mean; throughput ratios reverse this order.

Table 6: **Prefill-only serving on Qwen3-30B-A3B.** Drop and JORA use four experts per prefill token and native decode. L is input length, $O = 32$ is output length, C is client concurrency, and $N = 2C$ is the request count. Latencies are request means; throughput counts output tokens. E2E speedup uses the Native run within each group.

Method	TTFT (s)	TPOT (ms/token)	E2E (s)	Output (token/s)	E2E speedup
Input $L = 512$, concurrency $C = 8$, requests $N = 16$					
Native	0.242	11.21	0.590	433.00	1.000×
Drop	0.182	12.28	0.562	452.03	1.048×
JORA	0.187	12.67	0.580	436.36	1.016×
Input $L = 2048$, concurrency $C = 8$, requests $N = 16$					
Native	0.666	21.35	1.328	191.79	1.000×
Drop	0.555	20.60	1.193	212.93	1.113×
JORA	0.579	21.39	1.242	205.29	1.070×
Input $L = 8192$, concurrency $C = 8$, requests $N = 16$					
Native	2.750	110.30	6.169	41.35	1.000×
Drop	2.367	99.94	5.465	46.43	1.129×
JORA	2.461	103.05	5.655	44.72	1.091×
Input $L = 512$, concurrency $C = 32$, requests $N = 64$					
Native	0.618	22.58	1.318	772.48	1.000×
Drop	0.476	22.30	1.167	871.17	1.129×
JORA	0.494	23.10	1.210	839.91	1.089×
Input $L = 2048$, concurrency $C = 32$, requests $N = 64$					
Native	1.655	93.37	4.550	223.42	1.000×
Drop	1.322	81.07	3.835	261.26	1.186×
JORA	1.367	83.84	3.966	252.51	1.147×
Input $L = 8192$, concurrency $C = 32$, requests $N = 64$					
Native	7.790	605.65	26.566	38.28	1.000×
Drop	6.737	536.12	23.357	43.30	1.137×
JORA	6.969	554.48	24.158	41.97	1.100×

Table 7: **Shared-pool serving on Qwen3-30B-A3B.** D is the decode-pool cap; Native uses unrestricted routing. Decode-only groups keep eight prefill experts; joint and mixed groups use four for Drop/JORA. C is client concurrency and $N = 2C$. Latencies are request means and output throughput is in token/s. E2E ratios use each group's Native run. The mixed trace contains 32 requests each of $256 \to 512$ and $8192 \to 64$.

Method	D	TTFT (s)	TPOT (ms/token)	E2E (s)	Output (token/s)	E2E speedup
Decode-only: $256 \to 256$, $C = 32$, $N = 64$						
Native	∞	0.426	15.79	4.451	1837.85	1.000×
Drop	32	0.426	12.54	3.624	2256.50	1.228×
JORA	32	0.441	13.37	3.851	2112.63	1.156×
Drop	64	0.425	16.20	4.557	1795.14	0.977×
JORA	64	0.441	17.05	4.787	1699.92	0.930×
Decode-only: $256 \to 256$, $C = 64$, $N = 128$						
Native	∞	0.657	21.35	6.102	2890.23	1.000×
Drop	32	0.638	17.55	5.113	3197.38	1.193×
JORA	32	0.661	18.54	5.388	3027.25	1.132×
Drop	64	0.640	23.00	6.504	2514.58	0.938×
JORA	64	0.661	24.18	6.827	2405.50	0.894×
Decode-only: $256 \to 256$, $C = 96$, $N = 192$						
Native	∞	0.817	29.30	8.288	2950.69	1.000×
Drop	32	0.797	52.10	14.083	1741.60	0.589×
JORA	32	0.829	54.05	14.612	1674.71	0.567×
Drop	64	0.797	51.98	14.053	1745.22	0.590×
JORA	64	0.823	53.80	14.541	1687.20	0.570×

Continued on next page

Table 7: Shared-pool serving (continued).

Method	D	TTFT (s)	TPOT (ms/token)	E2E (s)	Output (token/s)	E2E speedup
Decode-only: $256 \to 256$, $C = 128$, $N = 256$						
Native	∞	1.021	35.80	10.148	3220.23	$1.000\times$
Drop	32	0.940	25.78	7.514	4334.14	$1.351\times$
JORA	32	0.985	27.12	7.901	4119.35	$1.284\times$
Drop	64	0.940	31.69	9.021	3613.59	$1.125\times$
JORA	64	0.975	33.32	9.471	3444.01	$1.072\times$
Joint: $2048 \to 256$, $C = 64$, $N = 128$						
Native	∞	2.682	50.74	15.621	1043.10	$1.000\times$
Drop	32	2.158	43.56	13.267	1216.25	$1.177\times$
JORA	32	2.236	45.97	13.957	1156.11	$1.119\times$
Drop	64	2.157	47.12	14.173	1140.35	$1.102\times$
JORA	64	2.225	49.28	14.792	1095.56	$1.056\times$
Joint: $8192 \to 128$, $C = 32$, $N = 64$						
Native	∞	7.640	175.28	29.901	136.17	$1.000\times$
Drop	32	6.653	158.12	26.734	151.32	$1.118\times$
JORA	32	6.913	163.39	27.664	145.32	$1.081\times$
Drop	64	6.651	160.43	27.026	149.70	$1.106\times$
JORA	64	6.857	166.07	27.948	144.86	$1.070\times$
Decode-only: $256 \to 128$, $C = 64$, $N = 128$						
Native	∞	0.618	21.15	3.304	2472.54	$1.000\times$
Drop	32	0.624	16.71	2.746	2974.47	$1.203\times$
JORA	32	0.647	17.85	2.914	2809.31	$1.134\times$
Drop	64	0.621	20.56	3.232	2526.81	$1.022\times$
JORA	64	0.644	21.80	3.412	2410.61	$0.968\times$
Decode-only: $256 \to 512$, $C = 64$, $N = 128$						
Native	∞	0.610	21.30	11.496	2846.30	$1.000\times$
Drop	32	0.616	16.23	8.911	3669.83	$1.290\times$
JORA	32	0.636	17.42	9.540	3438.97	$1.205\times$
Drop	64	0.615	20.07	10.872	3009.36	$1.057\times$
JORA	64	0.638	21.10	11.421	2852.79	$1.007\times$
Decode-only: $256 \to 1024$, $C = 64$, $N = 128$						
Native	∞	0.617	26.71	27.942	2342.01	$1.000\times$
Drop	32	0.630	19.32	20.395	3208.11	$1.370\times$
JORA	32	0.650	20.18	21.291	3062.28	$1.312\times$
Drop	64	0.640	24.71	25.914	2525.22	$1.078\times$
JORA	64	0.660	25.88	27.135	2406.20	$1.030\times$
Mixed lengths, $C = 32$, $N = 64$						
Native	∞	3.523	81.20	19.686	437.54	$1.000\times$
Drop	32	3.083	74.40	18.147	453.64	$1.085\times$
JORA	32	3.195	77.18	18.823	436.63	$1.046\times$
Drop	64	3.077	77.03	18.894	439.05	$1.042\times$
JORA	64	3.175	79.70	19.541	423.48	$1.007\times$

With native prefill and Pool32 decode, the $256 \to 1024$, $C = 64$ trace reduces mean E2E latency from 27.942 to 21.291 s ($1.312\times$) and TPOT from 26.711 to 20.177 ms/token ($1.324\times$). Output throughput increases from 2,342.01 to 3,062.28 token/s ($1.308\times$); E2E latency remains 4.40% above Drop. The largest TTFT ratio is $1.291\times$ for Top4 prefill on $512 \to 32$ requests at $C = 8$. Prefill-only and joint E2E peaks are $1.147\times$ and $1.119\times$, respectively, under the conditions given in the tables.

Increment over matched Drop. For each configuration, the E2E latency increment is $100(L_{\mathrm{JORA}}/L_{\mathrm{Drop}} - 1)$, where L is mean request latency. Across all 26 matched Drop/JORA configurations, this increment ranges from 3.16% to 7.05%, with median 4.38%. Median increments are 3.46% for prefill-only, 5.06% for decode-only, and 3.60% for joint/mixed workloads. JORA's output throughput is 93.62–96.93% of Drop's (median 95.85%). These are equally weighted summaries across configurations, not repeated-run confidence intervals.

The fraction of Drop's latency saving consumed by reweighting is $(L_{\mathrm{JORA}} - L_{\mathrm{Drop}})/(L_{\mathrm{Native}} - L_{\mathrm{Drop}})$ when Drop is faster than Native. It is 11.88% at the peak E2E configuration and 65.46% for prefill-only $512 \to 32$ at $C = 8$, despite a 3.16% latency increment in the latter case. Throughput retention and retention of the latency saving are therefore distinct measures.



Figure 4: **Serving sweeps on Qwen3-30B-A3B.** (a) Top4 prefill and native decode, with 32 output tokens and concurrency $C \in \{8, 32\}$. (b) Native prefill and Pool32/64 decode on $256 \to 256$ requests, varying C. (c,d) The same decode policies with 256 input tokens and $C = 64$, varying output length; (c) reports TPOT and (d) output throughput. Latency ratios are Native/method; throughput ratios are method/Native. Each trace has $2C$ requests. Lines connect the measured points, and the horizontal reference denotes Native.

Figure 4 also shows slowdowns, including both pool policies at concurrency 96 on $256 \to 256$ requests. For the mixed workload, joint Top4/Pool32 gives $1.046\times$ mean E2E speedup but $0.998\times$ throughput. Mean request latency and whole-trace throughput can therefore move differently. Each serving point represents one complete trace, not a repeated-run confidence estimate.

Fixed engine windows. Table 8 isolates prefill and decode at controlled batch sizes. GPU timing covers model forward, the LM head, and token emission, excluding scheduler preparation and bookkeeping. Four-expert prefill reaches $1.297\times$ at $L = 512$, $B = 8$. Pool32 decode reaches $1.344\times$ at initial $L = 512$, $B = 128$ over 128 steps (3,396.899 ms total, 26.538 ms per step). At $B = 96$, both pool policies are slower than Native for $L = 512$ and $L = 2048$. These fixed-batch windows and the client-concurrency serving sweeps measure different operating conditions.

Coefficient and routing paths. Table 9 measures 11 phase–budget–batch configurations on captured inputs from layers 0, 24, and 47. Each sampled layer has three timing blocks. The isolated coefficient entry uses the same mathematics and frozen support/fallback decisions as the deployed path; the full routing entry also includes selection and mapping. CUDA events measure GPU duration. These overlapping diagnostics are not additive components of serving time.

Table 8: **Controlled engine-window timings on Qwen3-30B-A3B.** L is prompt length for prefill and initial context length for decode; B is fixed batch size and D the decode-pool cap; the dash in prefill rows means not applicable. GPU time includes model forward, LM head, and token emission, excluding scheduler preparation. Prefill uses Top4 and averages three passes; decode totals cover 128 steps in one pass. Each ratio uses the corresponding Native run.

L	B	D	Native (ms)	Drop (ms)	JORA (ms)	Drop speedup	JORA speedup
Prefill							
512	1	—	43.23	43.21	44.63	1.000×	0.969×
2048	1	—	129.70	102.85	106.99	1.261×	1.212×
8192	1	—	681.38	574.97	595.12	1.185×	1.145×
512	8	—	206.89	153.89	159.54	1.344×	1.297×
2048	8	—	937.98	720.63	745.27	1.302×	1.259×
8192	8	—	5593.83	4728.88	4903.67	1.183×	1.141×
Decode							
512	32	32	2123.80	1719.42	1839.76	1.235×	1.154×
512	32	64	2123.80	2177.05	2304.76	0.976×	0.921×
512	64	32	2735.53	2108.25	2264.69	1.298×	1.208×
512	64	64	2735.53	2631.63	2770.53	1.039×	0.987×
512	96	32	3742.67	6470.66	6695.29	0.578×	0.559×
512	96	64	3742.67	6433.55	6677.96	0.582×	0.560×
512	128	32	4565.17	3215.73	3396.90	1.420×	1.344×
512	128	64	4565.17	4046.66	4254.81	1.128×	1.073×
2048	32	32	2759.35	2351.49	2496.00	1.173×	1.106×
2048	32	64	2759.35	2810.66	2963.06	0.982×	0.931×
2048	64	32	4445.81	3516.28	3691.76	1.264×	1.204×
2048	64	64	4445.81	4230.39	4426.78	1.051×	1.004×
2048	96	32	5191.73	6695.71	6916.39	0.775×	0.751×
2048	96	64	5191.73	6631.24	6850.00	0.783×	0.758×
2048	128	32	5883.15	5268.00	5564.16	1.117×	1.057×
2048	128	64	5883.15	5678.62	5947.91	1.036×	0.989×
4096	32	32	3566.93	3205.49	3367.84	1.113×	1.059×
4096	32	64	3566.93	3656.69	3799.34	0.975×	0.939×
4096	64	32	6038.38	5105.52	5335.61	1.183×	1.132×
4096	64	64	6038.38	5763.88	6050.22	1.048×	0.998×

Table 9: **Local online-path diagnostics on Qwen3-30B-A3B.** Values are μs per layer call, averaged over sampled layers 0, 24, and 47. Each layer entry averages three timing blocks. The coefficient entry uses frozen inputs, support, and fallback decisions; the full route also includes selection and mapping. Replay Δ is online minus cached-coefficient MoE replay time on the same inputs. These overlapping diagnostics are not additive and do not measure an exclusive serving-time fraction.

Phase	Tokens	Budget	Coefficient entry	Full route	Replay Δ
Prefill	512	$k_P = 4$	26.12	28.83	28.31
Prefill	2048	$k_P = 4$	26.29	29.26	29.56
Prefill	8192	$k_P = 4$	27.32	37.30	26.64
Decode	32	$D = 32$	14.06	34.32	11.76
Decode	32	$D = 64$	8.98	27.33	12.34
Decode	64	$D = 32$	18.06	40.14	11.98
Decode	64	$D = 64$	10.88	30.59	12.14
Decode	96	$D = 32$	18.06	40.48	11.67
Decode	96	$D = 64$	11.14	30.99	12.70
Decode	128	$D = 32$	18.10	40.49	12.10
Decode	128	$D = 64$	11.20	31.25	12.76

Averaged over the three sampled layers, the coefficient entry costs 26.12–27.32 μs per call for Top4 prefill, 14.06–18.10 μs for Pool32 decode, and 8.98–11.20 μs for Pool64 decode. Decode measurements include no-solve cases when the selected pool preserves a token's route. All 33 fixed-input online/cached replay pairs pass the recorded output-equivalence checks. Their local differences do not identify the exclusive fraction of deployed serving time attributable to coefficient computation.

B GRAM APPROXIMATION AND FINAL-PREDICTION DIAGNOSTICS

The validation-selected diagnostics use native-ranked per-token budgets $P_{k_P} D_{k_D}$. Section B.5 studies shared-pool reconstruction and prefix-cache interventions on documents also used for configuration selection. Appendices A.4 and A.5 specify the shared-pool and serving protocols.

B.1 INDEPENDENT DATA, CALIBRATION, AND SELECTION

We partition WikiText training and C4 validation documents to assess held-out reconstruction and final-prediction fidelity, separately from the standard benchmarks in Appendix A. For each model, calibration C, screening V_1, selection V_2, and test T contain 256, 64, 128, and 256 documents, respectively, equally divided between sources. Hash-based assignment and cross-partition deduplication fix membership; identifiable historical matches are excluded from T. This filtering is not semantic decontamination. Each document supplies a 512-token prefix and 128-token continuation.

Qwen3-30B-A3B and OLMoE-1B-7B-0924 retain native weights and $K = 8$. These instrumented forwards evaluate fidelity, not serving latency.

We evaluate at most 12 complete candidates on V_2 for Qwen3 and six for OLMoE. Selection minimizes mean excess continuation NLL relative to Native, weighting the two sources and P_4D_8, P_8D_4, and P_4D_4 equally. Ties within 10^{-5} are resolved by full-vocabulary KL, then configuration simplicity and calibration size. The selected JORA configurations use phase-specific tables with $(\rho, n) = (0.01, 8{,}192)$ for Qwen3 and $(10, 2{,}048)$ for OLMoE, where $\rho_P = \rho_D = \rho$ and n counts calibration states per layer and phase. These settings differ from the standard benchmarks.

Calibration samples 32 states per document and phase at nested per-layer sizes 512, 2,048, and 8,192; pooled tables weight phases equally. JORA uses no additional shrinkage, survivor renormalization, or clipping. All candidates and Oracles reuse the same fixed positive layer scales s_ℓ, without candidate- or token-specific rescaling. Parameters remain frozen after V_2 selection, including budget and context sweeps.

B.2 SAME-INPUT RECONSTRUCTION AND ORACLE COMPARISONS

Local measurements sample eight states per document and phase from Native and frozen JORA P_4D_4 trajectories, reported separately. Each comparison holds the input, native route, routing weights, and retained set fixed across coefficient rules. At sampled layer–token index $i = (\ell, t)$, define

$$e_i^M = \frac{\|y_i - \widehat{y}_i^M\|_2^2}{s_\ell}, \qquad R_E = \frac{\sum_i e_i^{\mathrm{JORA}}}{\sum_i e_i^{\mathrm{Drop}}}, \qquad \eta = \frac{\sum_i (e_i^{\mathrm{Drop}} - e_i^{\mathrm{JORA}})}{\sum_i (e_i^{\mathrm{Drop}} - e_i^{\mathrm{LS}})}. \tag{13}$$

Here y_i is the actual native routed mixture. Oracle-LS projects it onto the retained output span; Oracle-Ridge minimizes J_i using the actual token Gram and deployed ρ. Its excess-loss diagnostic is

$$J_i^M = e_i^M + \rho \|\delta_i^M\|_2^2, \qquad \Gamma = \frac{\sum_i (J_i^{\mathrm{JORA}} - J_i^{\text{Oracle-Ridge}})}{\sum_i J_i^{\mathrm{Drop}}}. \tag{14}$$

Ratios aggregate numerators and denominators; mean denominator contributions at most 10^{-8} are marked undefined. Oracle-LS bounds local unregularized reconstruction, not final PPL or task accuracy. Table 10 reports the Qwen3-30B-A3B comparison at $k = 4$.

Across phases and trajectories, Qwen3-30B-A3B recovers 82.62–83.71% of the Oracle-LS improvement over Drop ($R_E = 0.8635$–0.9072). The local reconstruction error is therefore 9.28–13.65% lower than Drop's. This reduction and η describe the unregularized reconstruction error, whereas Γ measures the gap in the regularized objective.

Table 10: **Same-input local reconstruction on Qwen3-30B-A3B at $k = 4$.** Results use 256 diagnostic test documents; Native and JORA trajectories are separate. $\bar{e}$ is the mean squared routed-output error divided by the fixed layer scale, not final hidden-state NMSE. R_E, η, and Γ aggregate their numerators and denominators across sampled layer–token states. P/D denotes prefill/decode.

Trajectory	Phase	$\bar{e}_{\mathrm{Drop}}$	$\bar{e}_{\mathrm{JORA}}$	R_E	η (%)	Γ
Native	P	0.021757	0.018787	0.8635	82.62	0.0285
Native	D	0.020842	0.018819	0.9029	82.90	0.0198
JORA	P	0.023005	0.019910	0.8655	83.71	0.0260
JORA	D	0.022073	0.020025	0.9072	83.20	0.0185

B.3 Full-model fidelity

Native and JORA propagate their own hidden states, routes, and KV caches under the same teacher-forced continuation. Final states are measured after final normalization and before the LM head, without additional unit normalization. NMSE divides summed squared state deviations by Native state energy; full-vocabulary KL is $D_{\mathrm{KL}}(p^{\mathrm{Native}} \| p^{M})$ with temperature-one softmax and no sampling filters. NMSE and KL use 127 incremental-decode predictions; NLL and PPL include all 128 suffix predictions, including the prefill boundary. Each configuration has one formal evaluation, reported across budgets in Table 11.

Table 11: **JORA full-model diagnostic fidelity with 512-token prefixes and 128-token continuations.** Each model has 256 test documents, equally divided between WikiText training and C4 validation holdouts. NMSE and full-vocabulary forward KL use the 127 decode predictions; continuation PPL uses all 128 suffix predictions. JORA uses frozen, validation-selected configurations. Lower is better.

Model	Reduced phase	Policy	Final NMSE	KL (nats)	Suffix PPL
Qwen3	Prefill	P_2D_8	0.02171	0.05712	9.206
		P_4D_8	0.00608	0.01588	8.973
		P_6D_8	0.00316	0.00620	8.843
	Decode	P_8D_2	0.11011	0.46793	13.693
		P_8D_4	0.03493	0.10630	9.428
		P_8D_6	0.00980	0.02504	9.421
	Both	P_2D_2	0.11062	0.49827	13.981
		P_4D_4	0.03825	0.11608	10.196
		P_6D_6	0.01115	0.02885	9.442
OLMoE	Prefill	P_2D_8	0.07421	0.08602	9.006
		P_4D_8	0.01868	0.01953	8.433
		P_6D_8	0.00600	0.00573	8.388
	Decode	P_8D_2	0.45431	0.68365	16.311
		P_8D_4	0.14280	0.15130	9.687
		P_8D_6	0.03512	0.03097	8.595
	Both	P_2D_2	0.51173	0.79911	18.218
		P_4D_4	0.16251	0.17247	9.696
		P_6D_6	0.03975	0.03617	8.636

At half-budget retention, both models have larger final-state and predictive errors under decode-only and joint reduction than under prefill-only reduction. Qwen3 KL is 0.01588 at P_4D_8, 0.10630 at P_8D_4, and 0.11608 at P_4D_4. These measurements include changes in states, routes, and caches.

B.4 Calibration stability and longer-prefix transfer

Increasing calibration from 512 to 8,192 states per layer improves R_E in 44/48 Qwen3 prefill layers and 23/48 decode layers, versus 5/16 and 3/16 for OLMoE; held-out reconstruction does not improve uniformly. At $k = 4$ on Native trajectories, predicted and observed improvement signs disagree on 14.69%/13.38% of Qwen3 prefill/decode layer–token states and 2.53%/0.89% for OLMoE.

The longer-prefix evaluation uses the same frozen parameters, calibration tables, and scales with 2,048-token prefixes and 128-token suffixes. After length filtering and exclusions, 30 test documents per source remain (60 per model); documents used for preliminary checks are excluded from scoring. Table 12 reports the three half-budget phase policies.

Table 12: **JORA diagnostic fidelity with longer prefixes.** Each model has 60 test documents, equally divided between WikiText training and C4 validation holdouts, with 2,048-token prefixes and 128-token continuations. JORA uses frozen, validation-selected configurations. NMSE and full-vocabulary forward KL use the 127 decode predictions; PPL uses all 128 suffix predictions. Lower is better.

Model	Policy	Final NMSE	KL (nats)	Suffix PPL
Qwen3	P_4D_8	0.00657	0.01837	8.889
	P_8D_4	0.03220	0.09686	9.751
	P_4D_4	0.03511	0.10695	9.581
OLMoE	P_4D_8	0.02390	0.02649	8.150
	P_8D_4	0.16157	0.18520	9.494
	P_4D_4	0.18422	0.21301	9.792

Both models again have lower NMSE and KL under prefill-only reduction. Because the documents and prediction positions differ from the shorter-prefix evaluation, these results assess transfer of the frozen configuration rather than the causal effect of context length.

B.5 Selection-set reconstruction and prefix-cache interventions

The Qwen3-30B-A3B diagnostics in Figure 2 use 128 documents, equally divided between WikiText training and C4 validation, with 512-token prefixes and 128-token continuations. We used these documents to select $\rho_P = \rho_D = 10^{-3}$; they are separate from the independent test partition above, and their overlap with calibration documents is not established. Gram tables and layer scales remain frozen, with full correction $a = a^0 + \delta$ and no additional shrinkage. Each configuration is evaluated once; all results below describe this selection set.

Same-input reconstruction. The local study uses $B = 32$ and eight sampled states per document, layer, and phase, with Native and JORA trajectories reported separately. Drop, JORA, and Oracle-LS share each input, original routing weights, and execution set. Equation (13) sums layer-scaled errors before taking ratios, with Oracle-LS computed from actual expert outputs.

On native-input states, Top4 prefill captures 86.34%/76.95% of the Oracle improvement on WikiText/C4, or 81.60% from pooled error sums, reducing layer-scaled error by 13.34% relative to Drop. Pool64 decode captures 60.91%/59.31% by source and 60.36% pooled, reducing error by 5.43%. Token-wise Top4 decode captures 78.56% pooled but does not constrain the batch expert union.

The first and last MoE layers show pronounced local gains. With Top4 prefill, JORA reduces squared reconstruction error relative to Drop by 46.27%/33.49% at the first layer and 67.00%/66.40% at the last layer on WikiText/C4, respectively. Under Pool64 decode, the corresponding reductions range from 19.23% to 19.67% at the first layer and from 18.74% to 47.40% at the last layer across the two sources. These gains are not uniform across layers: most middle-layer changes are small, and some layers worsen.

Budget and coefficient diagnostics. Figure 5 reports all measured token-wise budgets $k \in \{2, 4, 5, 6\}$ for both trajectories and sources; pool size is a separate resource axis. Coefficient quantiles and maxima describe the sampled states, without confidence intervals or bounds for other states.

Full-model pool comparisons. We also compare both sources under three $B = 32$ policies: Top4 prefill with native decode, native prefill with Pool64 decode, and their joint reduction. During pool decode, Drop keeps the native-route/pool intersection, whereas JORA uses the mapped execution set. Each method propagates its own states and cache. The comparison therefore measures complete policies rather than coefficient rules on identical sets.

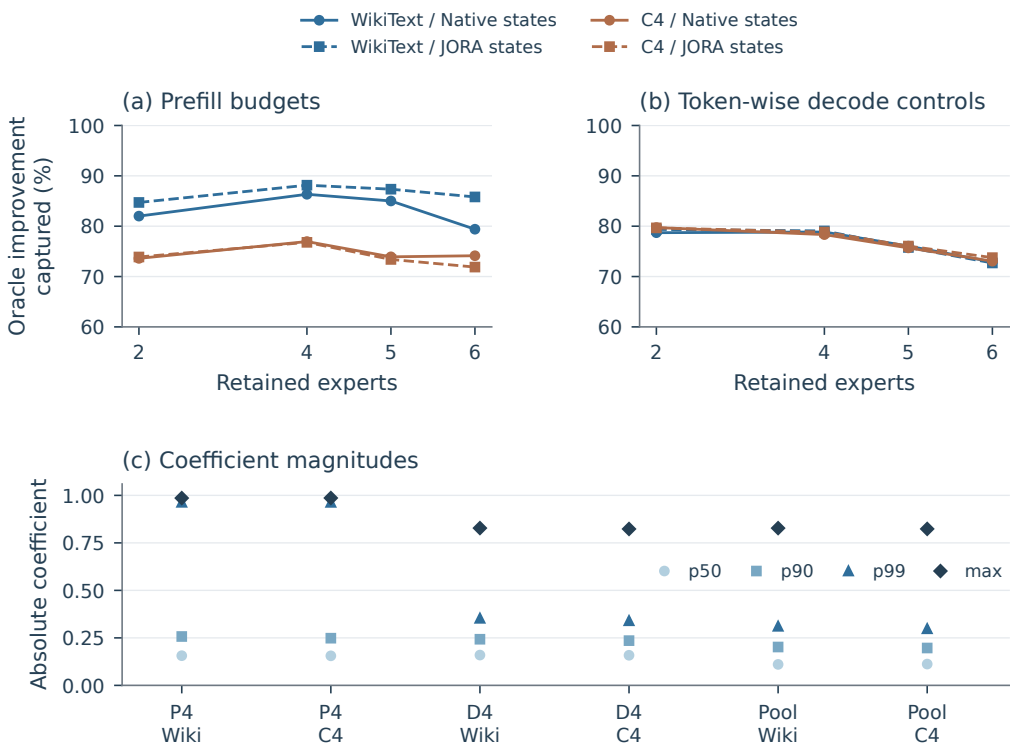


Figure 5: **Budget and coefficient diagnostics.** (a–b) Oracle-improvement recovery at every measured per-token budget, with Native/JORA input trajectories distinguished by line style. (c) Absolute-coefficient quantiles and maxima, separately by source and local setting. $D4$ is a token-wise control; Pool denotes batch-shared Pool64 decoding.

Across targets 2–128, full-vocabulary forward KL to Native decreases by 8.07%, 2.35%, and 4.37% for WikiText's prefill-only, pool-only, and joint policies, and by 4.34%, 0.60%, and 0.88% for C4. These averages exclude the prefill-boundary prediction and do not imply improvement at every continuation position. Document-level records for paired intervals are unavailable.

Cache-isolated intervention. A separate experiment tests what prefill changes transmit through the cache. The evaluation uses batch size two and processes one suffix token per step.

Each prefix pass applies Drop or JORA Top4 at designated MoE layers. The suffix receives only prefix KV and position metadata, with Native Top8 at every MoE layer and identical teacher-forced ground-truth tokens in every arm. Feeding suffix token 1 predicts target 2; scoring continues through target 128, excluding the prefill-boundary prediction and yielding 127 targets per document (16,256 per arm). Each arm has an independent cache whose newly written suffix KV evolves under native computation.

Seven intervention scopes cover all 48 layers, internal layers 2–47, and individual layers 1, 12, 24, 36, and 48; two methods per scope plus Native give 15 configurations. Single-layer arms measure common-input local error at eight fixed prefix positions. K and V errors use actual stored cache tensors, separately at every layer. Checks show exact Native/no-op agreement, unchanged cache through the intervened layer, and zero suffix intervention hits. Last-layer intervention reduces local MoE error by 66.76% relative to Drop but leaves all prefix KV and subsequent predictions exactly equal to Native: the final MoE follows its layer's cache write and cannot affect a later cache.

With Top4 prefill at all layers, JORA lowers pooled suffix KL to Native from 0.017509 to 0.016172 (7.63%) and NLL by 0.003214 nats per target relative to Drop. The 95% paired intervals for Drop-minus-JORA are $[0.099,\ 3.036] \times 10^{-3}$ for KL and $[0.457,\ 6.403] \times 10^{-3}$ for NLL; both exclude

zero. Because suffix computation is native in both arms, these gains are transmitted through the prefix KV cache. First-layer JORA also reduces K and V error in all 47 downstream cache layers.

Figure 2(b) separates the all-layer KL result by source: the pooled gain is driven by WikiText, while C4 changes little. Internal-layer-only and nonterminal single-layer KL intervals include zero. Intervals use 2,000 paired document-bootstrap samples (seed 20260924), stratified by source with 64 documents each; they condition on the selected configuration and reused samples, without multiplicity adjustment.

C Proofs and execution details

C.1 A common objective for deletion and re-routing

For any nonempty R, the zero-extended reference coefficients in Equation (1) satisfy $y - F_R a^0 = F_T w_T$. The final-coefficient stationarity equation is $(G_{RR} + \rho I_k)a = G_{RA}w + \rho a^0$. Substituting $a = a^0 + \delta$ and $G_{RA}w - G_{RR}a^0 = G_{RT}w_T$ gives Equation (4) for both deletion and re-routing. The guarantees below concern successful solves on a fixed supplied set R.

C.2 Projection geometry and the constructed example

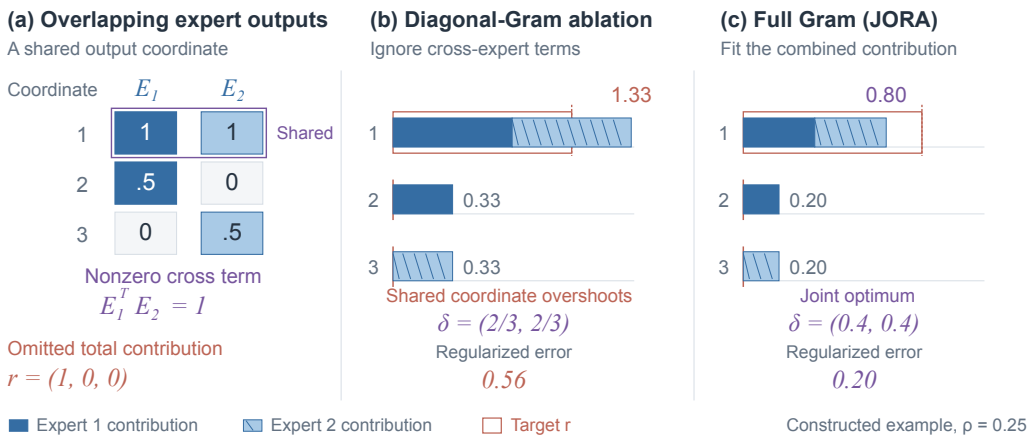


Figure 6: **Correlations couple coefficient updates.** In this constructed example, diagonal-Gram fitting overshoots a shared direction; full-Gram fitting minimizes the same regularized objective.

For the same-input residual $r = F_T w_T$, the projector $P_R = F_R F_R^\dagger$ satisfies $P_R F_R = F_R$. Thus $r - F_R\delta = (I - P_R)r + (P_R r - F_R\delta)$. The first summand is orthogonal to the columns of F_R, whereas the second lies in their span. Pythagoras yields Equation (7). In particular, if $F_R^\top r = 0$, every nonzero represented correction adds squared error; coefficient changes cannot recover the omitted direction.

Using the outputs and $\rho = 0.25$ displayed in Figure 6, without layer-scale normalization, gives a retained Gram with diagonal entries 1.25 and off-diagonal entries 1. The displayed diagonal- and full-Gram corrections have objective $\|r - F_R\delta\|^2 + \rho\|\delta\|^2$ equal to $5/9$ and $1/5$, respectively. This constructed geometry isolates retained-output correlation.

C.3 Fixed-set optimality and improvement over the native anchor

Since $G \succeq 0$ and $\rho > 0$, $H = G_{RR} + \rho I_k \succ 0$. Completing the square gives

$$J_G(\delta) = e_0 - q^\top H^{-1} q + (\delta - H^{-1}q)^\top H (\delta - H^{-1}q). \tag{15}$$

Consequently, $\delta^\star = H^{-1}q$ is unique, and subtracting its objective value proves Equation (8). Substituting $G_{RR} = H - \rho I_k$ and $H\delta^\star = q$ into the unregularized difference yields

$$\begin{aligned} E_G(0) - E_G(\delta^\star) &= 2(\delta^\star)^\top q - (\delta^\star)^\top G_{RR}\delta^\star \\ &= q^\top H^{-1} q + \rho\|\delta^\star\|_2^2. \end{aligned} \tag{16}$$

This is positive exactly when $q \neq 0$. Furthermore, $E_G \geq 0$ by the positive semidefiniteness of G, and $J_G(\delta^\star) \leq J_G(0) = e_0$. Hence $\rho\|\delta^\star\|^2 \leq e_0$. These statements compare coefficients at a fixed execution set; they do not establish optimal subset selection or downstream accuracy.

C.4 Sourcewise equivalence and support restrictions

With a common execution support, Gram, ridge, and native anchor, full-support sourcewise ridge fitting gives $Hc_e = G_{Re}$ for each omitted expert e. Summing with native weights and applying the uniqueness of the joint solution yields

$$\sum_{e\in T} w_e c_e = H^{-1}\sum_{e\in T} G_{Re} w_e = H^{-1} G_{RT} w_T = \delta^\star. \tag{17}$$

The identity assumes the stated weighted sum without additional transformations; sparse support selection or empirical shrinkage defines a different estimator. For $T = \varnothing$, the correction is zero, but $k = K$ alone does not imply this condition after re-routing.

Restricted-support sourcewise rankings can differ from aggregate rankings because residual cross terms enter the mixture error. For example, take retained features $(1, 0)$ and $(0, 1)$, omitted features $(1, \frac{1}{2})$ and $(-1, \frac{1}{2})$, weights $\frac{1}{2}, \frac{1}{2}$, and $\rho = \frac{1}{4}$. The individually optimal one-target fits use the first feature with coefficients $\pm\frac{4}{5}$ and aggregate to zero, with $J_G = 1/4$. Choosing the second feature for both sources with coefficient $2/5$ raises each sourcewise objective from $9/20$ to $21/20$, yet gives the joint optimum $J_G = 1/20$.

C.5 Conditional calibration mismatch and numerical error

Conditional calibration regret. Let Φ contain expert outputs scaled by $1/\sqrt{s_\ell}$. Condition on information Z fixing R, T, w and the computed correction $\widetilde{\delta}$, and let $C = \mathbb{E}[\Phi^\top\Phi \mid Z]$ be finite. Replace G by C to define J_C and its minimizer $\delta_C^\star$. For numerical residual $r_{\text{solve}} = q - H\widetilde{\delta}$, define

$$\xi = r_{\text{solve}} + (C_{RT} - G_{RT})w_T - (C_{RR} - G_{RR})\widetilde{\delta}. \tag{18}$$

Then

$$J_C(\widetilde{\delta}) - J_C(\delta_C^\star) = \xi^\top (C_{RR} + \rho I_k)^{-1}\xi \leq \frac{\|\xi\|_2^2}{\rho}. \tag{19}$$

This separates solver error from routing-conditioned calibration mismatch: accurate marginal statistics need not match C. It controls local regularized reconstruction; downstream quality and runtime are measured separately.

Proof. Condition on Z, treating the calibrated table as fixed. Since $C \succeq 0$, $H_C = C_{RR} + \rho I_k \succeq \rho I_k$, and stationarity gives $H_C(\delta_C^\star - \widetilde{\delta}) = C_{RT}w_T - H_C\widetilde{\delta} = \xi$. Applying Equation (15) with C proves Equation (19). The triangle inequality then gives, for spectral-norm discrepancies $\epsilon_{RT} = \|C_{RT} - G_{RT}\|_2$ and $\epsilon_{RR} = \|C_{RR} - G_{RR}\|_2$,

$$J_C(\widetilde{\delta}) - J_C(\delta_C^\star) \leq \frac{\left(\|r_{\text{solve}}\|_2 + \epsilon_{RT}\|w_T\|_2 + \epsilon_{RR}\|\widetilde{\delta}\|_2\right)^2}{\rho}. \tag{20}$$

The constant C_{TT} term cancels in the regret. Taking $C = \Phi^\top\Phi$ gives a same-input oracle comparison. These bounds characterize reconstruction sensitivity, not a task-accuracy tolerance.

C.6 Assignments, expert unions, and phase semantics

Fewer token–expert assignments can leave the batch expert union, and hence weight access, unchanged. Scheduler metadata identifies phases per request: every prompt chunk uses prefill, including a one-token chunk; the final prompt forward predicts the first generated token, whose subsequent forward is decode.

For unequal configured budgets, routed-FFN inputs are partitioned by budget, dispatched to homogeneous expert execution, and restored to token order. This preserves the phase-specific rule but can introduce extra launches, partition/scatter work, and repeated expert-weight accesses. An external token-adaptive selector would likewise require grouping by its selected k; JORA preserves the supplied subsets, while the combined policy's quality and runtime require separate evaluation.

For batch-pool decode, S is formed on the complete scheduled decode subgroup before grouping by $|R_t|$. One target per native source and duplicate coalescing give $|R_t| \le K$, independently of D. For $U = \cup_t A_t$, $D \ge |U|$ leaves all native routes unchanged. In particular, a single $K = 8$ route is unchanged at $D \ge 8$, and $B = 8$ cannot activate more than 64 distinct experts. These settings serve as no-pruning controls. With $k_t = |R_t|$, the coefficient cost is $\sum_t O(k_t|A_t \setminus R_t| + k_t^3)$, in addition to pool construction and mapping. Zero-weight padding does not by itself skip expert computation; deployment profiling includes actual dispatch, duplicate handling, padding, and repeated weight access.

D Additional task evaluations

Table 1 compares ten Qwen3-30B-A3B tasks across three phase policies. The serving comparison is specified in Table 6. The panels below extend evaluation to other architectures and visual tasks. PPL protocols are specified in Appendix A.1.

D.1 Expanded task-quality comparisons

Table 13 compares Qwen2-57B-A14B Base, Qwen1.5-MoE-A2.7B, and DeepSeek-V2-Lite under joint reduction (Qwen Team, 2024b;a; DeepSeek-AI, 2024). Each model includes locally evaluated Native, Drop, and JORA.

Separate Qwen3 pooled-MMLU accuracy is 79.60% for Native Top8 and 77.29%/74.26%/77.72% for Drop/Drop-Renorm/JORA at native-ranked Top4 prefill with native decode. Table 14 reports Qwen3-VL-30B-A3B-Instruct visual-task comparisons.

Table 13: **Cross-model quality under joint reduction** (accuracy, %). $P = K/2$ retains half the per-token prefill experts; a 50% decode pool caps half the routed bank at active $B = 32$, excluding shared experts. Native's 100% means unrestricted routing. Budgets and comparison rules are in Appendix A.4. Scored-item counts are 1,172 for ARC-C, 1,267 for WinoGrande, and 1,319 for GSM8K.

Method	P	Pool (%)	MMLU	ARC-C	Hella Swag	Wino Grande	GSM8K
Qwen2-57B-A14B Base	($K = 8$, $N = 64$)						
Native	8	100	76.50	64.11	85.25	79.37	80.59
Drop	4	50	69.68	58.73	80.77	77.37	72.73
JORA	4	50	74.16	63.40	84.59	78.32	76.78
Qwen1.5-MoE-A2.7B	($K = 4$, $N = 60$)						
Native	4	100	62.60	53.30	73.69	67.85	61.49
Drop	2	50	56.77	46.30	68.52	62.83	57.47
JORA	2	50	60.21	50.71	71.52	66.43	59.77
DeepSeek-V2-Lite	($K = 6$, $N = 64$)						
Native	6	100	58.23	51.28	58.81	60.83	41.09
Drop	3	50	52.72	47.75	53.62	52.51	33.94
JORA	3	50	55.48	50.20	57.76	56.20	38.79

D.2 Task definitions and scoring

A one-token completion tests only the prefill boundary; preceding rationale tokens exercise decode. Matched latency and throughput measurements assess serving efficiency separately.

Table 14: **Vision–language task quality on Qwen3-VL-30B-A3B-Instruct.** The methods modify only the language-backbone MoE; image processing is unchanged. Column labels identify the benchmark variants. Native executes eight routed experts per token in both phases. DocVQA and InfographicVQA use ANLS $\times 100$, OCRBench uses its original raw score out of 1,000, and the remaining columns use percentage metrics (Appendix D).

Method	P	D	MMMU	Text VQA	DocVQA test	InfoVQA test	ChartQA test	OCR Bench	MMBench EN	MathVista testmini
Native	t_8	t_8	72.10	79.52	93.74	82.48	86.32	876	82.83	77.48
Prefill only										
Drop	t_4	t_8	69.74	75.36	91.16	79.23	82.99	825	79.97	72.90
ExFold	t_4	t_8	71.74	77.19	91.98	81.81	84.86	847	82.12	74.52
JORA	t_4	t_8	72.15	78.51	93.73	81.42	86.26	857	82.84	76.69
MC-MoE-ODP	$\text{ODP}_{.5}$	t_8	70.41	76.86	91.61	81.32	85.25	832	79.63	75.76
Decode only; dynamic batch-level policies										
ExFold	t_8	d_{32}	71.28	77.96	91.37	80.52	84.32	840	81.70	73.76
JORA	t_8	d_{32}	71.86	77.76	94.14	82.45	86.33	854	82.79	76.94
SERE	t_8	S_4	71.49	76.82	92.40	81.68	84.41	844	80.63	74.79
Prefill and decode										
ExFold	t_4	d_{32}	70.32	77.15	91.16	80.49	83.66	830	81.73	73.10
JORA	t_4	d_{32}	71.54	77.55	93.45	82.04	85.65	850	82.57	75.93
Static compression; resident-bank budget, both phases										
REAP	R64		67.19	74.81	87.95	77.59	84.34	823	78.39	74.09
MC-SMoE	M		68.77	76.57	89.76	79.59	84.18	828	80.29	73.66

Budget and selector labels follow Table 1. S_4 pools each token's Top4 primary experts, not four experts total. R64 retains 64 resident experts; M identifies static MC-SMoE compression. Pool comparisons use $B = 32$.

Cross-model choice scoring uses pooled MMLU and length-normalized ARC-Challenge/HellaSwag accuracy (Hendrycks et al., 2021a; Clark et al., 2018; Zellers et al., 2019); WinoGrande reports accuracy (Sakaguchi et al., 2019). MMLU-Pro, GPQA-Diamond, CommonsenseQA, OpenBookQA, and SocialIQA in Table 1 use method-independent final-choice extraction with fixed prompts and output caps (Wang et al., 2024; Rein et al., 2023; Talmor et al., 2018; Mihaylov et al., 2018; Sap et al., 2019). Generated-answer and likelihood scores remain separate; GPQA uses the preselected Diamond item IDs.

GSM8K compares extracted numerical answers; the fixed 500-item MATH500 release uses mathematical answer normalization/equivalence (Cobbe et al., 2021; Hendrycks et al., 2021b; Lightman et al., 2023). Extraction rules are shared across methods. IFEval strict prompt-level accuracy requires every instruction to pass the released verifier on the complete original response (Zhou et al., 2023). HumanEval+/MBPP+ use EvalPlus's original and additional executable tests; with one candidate per problem, pass@1 is the fraction passing all required tests (Chen et al., 2021; Austin et al., 2021; Liu et al., 2023a).

Visual tasks use their released evaluators (Yue et al., 2024; Singh et al., 2019; Mathew et al., 2021b;a; Masry et al., 2022; Liu et al., 2023b; 2024b; Lu et al., 2024a). We evaluate the test splits of DocVQA, InfographicVQA, and ChartQA, the original OCRBench, and MathVista testmini, alongside MMMU, TextVQA, and MMBench-EN. DocVQA and InfographicVQA report ANLS $\times 100$, TextVQA reports consensus accuracy, ChartQA reports relaxed accuracy, and OCRBench reports its raw score out of 1,000. MMMU, MMBench-EN, and MathVista use their benchmark-specific answer scoring.