BASE: BATCH-AWARE SELECTION OF EXPERTS USING PREDICTED REMOVAL ERROR FOR EFFICIENT MOE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models continue to grow in parameter count and capacity, making them increasingly expensive to serve. In large-scale serving systems, autoregressive decoding is often bottlenecked by transferring model weights from accelerator high-bandwidth memory into on-chip SRAM. This memory bottleneck becomes more severe as models and context lengths grow, placing increasing pressure on memory bandwidth. Mixture-of-experts (MoE) models reduce computation by activating only a small subset of experts per token, but this sparsity does not translate directly to batched decoding. Different requests select different experts; therefore, the combined active set across many concurrent requests can span a substantial fraction of the expert pool and require significantly more expert weights to be transferred. Most expert-reduction techniques make retention decisions independently for each token and therefore do not address this batch-level expansion. More recently, batch-aware methods have attempted to coordinate expert use across concurrent requests and reuse experts already fetched for the batch. Yet their selection criteria are based primarily on router rankings or expert statistics collected during calibration. Consequently, these criteria are not directly tied to the output error caused by dropping an expert, nor do they capture how an expert's contribution changes across tokens at inference time. We instead rank experts according to how much their removal would change the MoE-layer output. To apply this criterion during serving, we train a lightweight linear predictor during calibration that estimates the expert removal cost for each incoming token. We also develop custom GPU kernels to efficiently perform cost prediction, batch-level score aggregation, and expert selection. Across three MoE architectures, BASE improves the quality–efficiency tradeoff without retraining. On Qwen3-30B-A3B, it improves average accuracy by 29.5 points over the strongest baseline at comparable throughput under a tight expert budget. At a higher expert budget, it is 60% faster than dense inference while remaining within 0.4 accuracy points.

1 INTRODUCTION

Scaling model size and training compute has consistently improved language-model performance, but it has also made inference increasingly expensive (Kaplan et al., 2020; Hoffmann et al., 2022). In large-scale serving, autoregressive decoding is often constrained by memory bandwidth, as model weights and a growing key-value (KV) cache must be accessed for every generated token (Kwon et al., 2023; Rajbhandari et al., 2022). Serving systems therefore treat decoding as a distinct phase with its own cost profile (Zhong et al., 2024b; Agrawal et al., 2024). Mixture-of-experts (MoE) architectures reduce per-token computation by activating only a sparse subset of model parameters (Jacobs et al., 1991; Shazeer et al., 2017; Fedus et al., 2022; Jiang et al., 2024), yet this sparsity does not necessarily translate into efficient batched decoding.

This limitation arises because MoE sparsity is defined per token, whereas serving cost is incurred across the batch. At each decoding step, every active request contributes one token; although each token selects only a few experts, their selections can differ substantially, causing the batch-wise union to grow rapidly and cover a large fraction of the expert pool (Wu et al., 2026a; Oncescu et al., 2025).

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.

In the memory-bound decoding regime, each distinct expert in this union requires its weights to be fetched from high-bandwidth memory, so latency is governed primarily by the size of the batch-level active set rather than the sparsity of any individual token (Rajbhandari et al., 2022; Oncescu et al., 2025). Efficient batched decoding therefore requires controlling which experts are loaded for the batch as a whole.

Existing methods reduce MoE serving costs by pruning or merging experts before inference (Lasby et al., 2025; Chen et al., 2024; Li et al., 2024b), reducing expert execution per token (Huang et al., 2024; Lu et al., 2024), or coordinating expert use across a batch (Gupta et al., 2024; Wu et al., 2026a; Oncescu et al., 2025). We focus on the last setting, where the objective is to limit the number of distinct expert weights fetched at each decode step.

Existing batch methods still leave open how experts should be selected according to their contribution to the current batch. SERE and OEA construct the active set from token routing decisions, so an expert selected by only one token can add an expert to the batch active set even if its contribution to the batch is small (Wu et al., 2026a; Oncescu et al., 2025). ExFold instead imposes a fixed batch budget and combines current router weights with expert output norms measured during calibration (Wu et al., 2026b). However, a fixed calibrated norm cannot capture how an expert's output varies across tokens in the current batch. This leads to the central question we study: **Given a fixed batch budget, which experts should be fetched to minimize the change in the MoE layer output?**

To address this question, we introduce BASE, which scores experts using the squared change in the MoE layer output caused by removing an expert's weighted contribution. The exact error contains the individual removal costs of omitted experts and cross terms between experts omitted together. Our analysis shows that selection is robust to discarding the cross terms, allowing BASE to rank experts independently and retain those with the largest estimated removal costs under a fixed batch budget.

The remaining challenge is that removal cost depends on an expert's output for the current token, which is unavailable before the expert is fetched and executed. BASE resolves this with a lightweight linear predictor for each expert, trained during calibration, that estimates output energy from the current token representation. BASE combines these estimates with router weights, aggregates removal costs across the batch, and loads the experts with the largest predicted costs under the fixed budget.

Our contributions are threefold. **1)** We formulate batch expert selection as minimizing the change in the MoE layer output under a fixed expert budget, and show that discarding cross terms has little effect on the selected set. **2)** We predict each routed expert's output energy from the current token representation before execution, combine it with router weights to estimate removal cost, and use backfill when a routed expert is not loaded. **3)** We implement BASE with custom GPU kernels and evaluate it across three MoE architectures and eight benchmarks. BASE improves average accuracy by 3.0 to 29.5 points over existing methods at comparable speed under tight budgets, while relaxed settings stay within 0.4 points of dense model quality with 22 to 60% higher throughput.

2 Related Work

Prior work reduces MoE inference cost through offline expert compression (Lasby et al., 2025; Li et al., 2024b; Chen et al., 2024), adaptive expert execution for individual tokens (Huang et al., 2024; Lu et al., 2024), or coordination of expert use across a batch (Wu et al., 2026a; Oncescu et al., 2025; Gupta et al., 2024; Chen et al., 2026; Wu et al., 2026b). BASE belongs to the last category, but differs in estimating expert importance from the predicted output contribution of the current tokens rather than from router signals or fixed calibration statistics.

Offline Expert Pruning and Merging. Offline methods permanently reduce the global expert pool through pruning, merging, decomposition, or regularization (Yang et al., 2024; Liu et al., 2024; Li et al., 2024b; Chen et al., 2024; Muzio et al., 2024; Jaiswal et al., 2025). REAP is particularly relevant because it combines router weights with expert output magnitudes and relates this score to reconstruction error (Lasby et al., 2025). Its pruning decisions, however, are fixed from calibration data. BASE instead estimates removal costs for the current tokens and chooses the active expert set separately for each decode batch.

Token-Level Adaptive Expert Execution. Other methods adapt the number of experts executed for individual tokens using router confidence, sensitivity, or learned importance criteria (Huang et al., 2024; Li et al., 2023; Zhong et al., 2024a; Lu et al., 2024; Huang et al., 2025; Yang et al., 2025b). Because different tokens can retain different experts, reducing computation per token does not directly control the number of distinct experts fetched for the batch.

Batch-Aware Expert Selection. SERE and OEA construct the batch active set from the union of preferred experts across tokens (Wu et al., 2026a; Oncescu et al., 2025). OEA fills missing routes with experts already loaded for the batch, while SERE uses calibrated expert similarities to choose replacements. Because their active set is formed from a union, its size depends on routing overlap and can grow when only a few tokens request additional experts. Lynx instead uses router confidence and expert popularity within the current batch to reduce the active set (Gupta et al., 2024), but does not estimate the output change caused by removing an expert. ExFold is closest to BASE because it combines a fixed batch budget with output information (Wu et al., 2026b). However, it uses expert output norms measured during calibration, whereas BASE predicts output energy from the current token representation. ExFold also transfers the contribution of a removed expert to a retained expert using a calibrated scalar, while BASE uses backfill among the token's preferred experts.

Expert Offloading and Prefetching. Systems such as Fiddler and Pre-gated MoE reduce serving cost through CPU and GPU placement, offloading, or prefetching (Kamahori et al., 2025; Hwang et al., 2024; Balmau et al., 2025). BASE instead reduces the number of expert weights accessed from GPU high-bandwidth memory during decoding.

3 METHOD

3.1 PROBLEM FORMULATION

Consider an MoE layer with N routed experts $E_1, \ldots, E_N : \mathbb{R}^d \to \mathbb{R}^d$. For token t with input z_t, the router selects K experts $R_t \subseteq [N]$ with weights $w_{t,u}$, producing $y_t = \sum_{u \in R_t} w_{t,u} E_u(z_t)$. We assume token-choice routing, omit shared experts from the notation, and let ℓ index the layer (Zhou et al., 2022; Dai et al., 2024).

Batched decoding. At each decode step, the batch $\mathcal{B}$ contains one token from each active sequence. Although each token routes to only K experts, the batch can require $\left|\bigcup_{t \in \mathcal{B}} R_t\right|$ distinct experts, which can be much larger than K.

MoE decoding is typically memory-bound (Rajbhandari et al., 2022; Oncescu et al., 2025). We model the layer latency as

$$T_\ell \approx \tau_{\text{fetch}} |A_\ell| + \tau_{\text{comp}} \sum_{t \in \mathcal{B}} |R'_t|, \tag{1}$$

where A_ℓ is the set of experts fetched for the batch and R'_t is the set executed for token t. Here, τ_{fetch} is the per-expert weight-transfer cost and τ_{comp} is the per-token, per-expert computation cost. In the memory-bound regime, the first term dominates, making latency primarily dependent on $|A_\ell|$ (Oncescu et al., 2025). On our inference stack, decode time is approximately linear in the expert budget over $4 \leq M \leq 32$, with $R^2 \geq 0.996$ across all three architectures (Appendix A.4). We therefore constrain the active set to $|A_\ell| \leq M$ and ask which experts should be fetched for the current batch.

Unlike standard token routing (Shazeer et al., 2017; Fedus et al., 2022) and union-based methods such as SERE and OEA (Wu et al., 2026a; Oncescu et al., 2025), we directly constrain the number of experts fetched and select the active set according to the resulting change in the MoE output.

3.2 WHAT IS THE COST OF LEAVING AN EXPERT OUT?

For token t, define the weighted contribution $a_{t,u} \triangleq w_{t,u} E_u(z_t)$, and let $U_t \triangleq R_t \setminus A_\ell$ denote its omitted experts. The resulting batch output error is

$$\mathcal{E}(A_\ell) = \sum_{t \in \mathcal{B}} \left\| \sum_{u \in U_t} a_{t,u} \right\|_2^2 = \underbrace{\sum_{t \in \mathcal{B}} \sum_{u \in U_t} \|a_{t,u}\|_2^2}_{\text{individual damage}} + \underbrace{2 \sum_{t \in \mathcal{B}} \sum_{u < v,\, u,v \in U_t} \langle a_{t,u}, a_{t,v} \rangle}_{\text{cross terms}}. \tag{2}$$

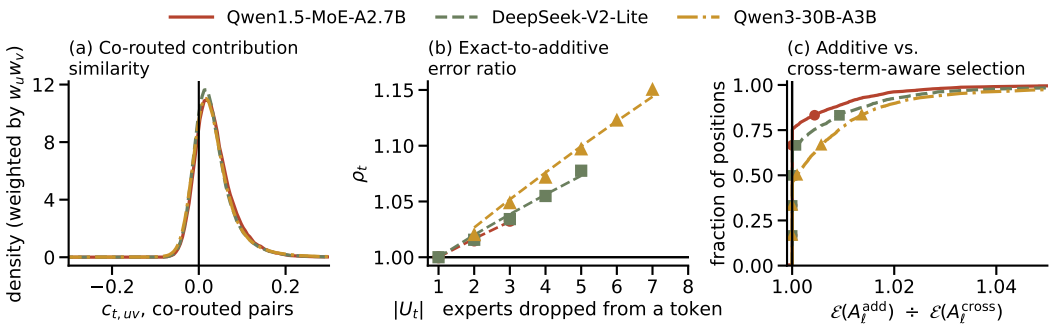


Figure 1: **Cross terms affect error magnitude but have limited impact on expert selection. (a)** $w_{t,u}w_{t,v}$-weighted distributions of cosine similarity between expert contributions routed to the same token. **(b)** Exact-to-additive error ratio as the literal number of omitted experts increases. The active set is the batch-wise union of each token's top-1 expert; dashed curves show one-parameter fits. **(c)** Exact-error ratio between additive and cross-term-aware selection under the same fixed budget $M = 12$. Panels (a) and (b) exclude the first and last MoE layers.

The individual removal cost of expert u, aggregated across the batch, is

$$D_u \triangleq \sum_{t \in \mathcal{B},\, u \in R_t} w_{t,u}^2 \|E_u(z_t)\|_2^2. \tag{3}$$

Ignoring the cross terms gives $\mathcal{E}_{\text{add}}(A_\ell) = \sum_{u \notin A_\ell} D_u$, which under $|A_\ell| = M$ is minimized by retaining the M experts with the largest D_u. Thus, additive selection reduces the problem to ranking experts by their individual removal costs.

However, the additive objective is not generally identical to the exact objective. When several experts are omitted for the same token, their contributions interact through cross terms of the form $\langle a_{t,u}, a_{t,v} \rangle = \|a_{t,u}\|_2 \|a_{t,v}\|_2 \cos\theta_{t,uv}$. Large or systematically structured cross terms could invalidate an independent per-expert score. In that case, the cost of omitting expert u would depend strongly on which other experts are omitted alongside it. This leads to the first question our method must answer: **Can we safely ignore the cross terms when selecting which experts to retain?**

A second question follows from the same geometry. SERE compensates for an unavailable expert by replacing it with a similar expert already loaded for the batch (Wu et al., 2026a). Such substitution is useful only when the corresponding expert outputs point in sufficiently similar directions. This leads to our second question: **Can the output of one active expert serve as a reliable surrogate for an unavailable expert?**

Therefore, before designing either the selection rule or the treatment of omitted experts, we measure the geometry of expert outputs produced for the same token. Figure 1 summarizes this analysis.

3.3 Expert Selection Is Robust to Discarding Cross Terms

The exact objective in Equation 2 contains cross terms between omitted expert contributions. Discarding them is useful only if it does not substantially change which experts should be retained. We examine this across all three architectures using the ground-truth outputs of experts in each token's original top-K routing set. For two experts u and v routed to token t, let $c_{t,uv}$ denote the cosine similarity between their outputs.

Co-routed expert outputs are nearly orthogonal. Figure 1(a) shows that these similarities concentrate near zero, with weighted mean absolute values of 0.045, 0.040, and 0.043 for Qwen1.5-MoE-A2.7B, DeepSeek-V2-Lite, and Qwen3-30B-A3B, respectively.

Near-orthogonality does not guarantee that several small cross terms cannot accumulate. We therefore measure their combined effect using the ratio between exact and additive error:

$$\rho_t \triangleq \frac{\left\| \sum_{u \in U_t} a_{t,u} \right\|_2^2}{\sum_{u \in U_t} \|a_{t,u}\|_2^2}, \tag{4}$$

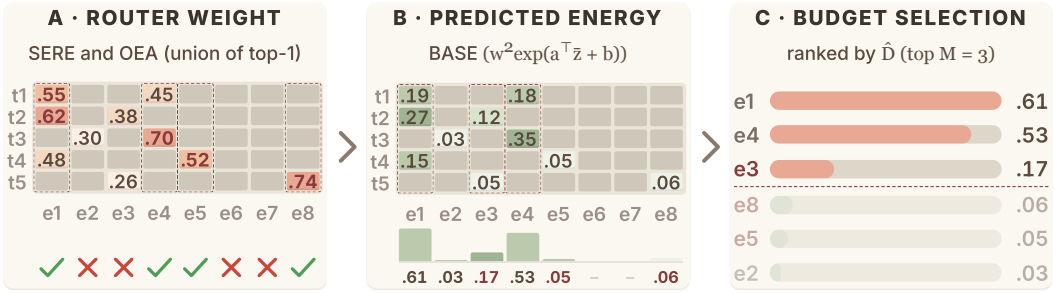


Figure 2: **Binary union and budgeted selection produce different active sets.** Constructed example with five tokens, eight experts, and two routed experts per token. **(a)** SERE and OEA fetch the union of each token's top-1 expert, shown by dashed columns, producing four active experts. **(b)** BASE computes each routed pair's predicted removal cost, $w_{t,u}^2 \widehat{e}_u(z_t)$, and sums these values across tokens to obtain the batch-level expert score $\widehat{D}_u$. **(c)** BASE fetches the $M = 3$ highest-scoring experts. Experts e_5 and e_8 enter the binary union despite low predicted removal costs, while e_3 is selected by BASE despite never being a token's top-1 expert.

where $U_t = R_t \setminus A_\ell$ is the set of omitted experts. A value of $\rho_t = 1$ corresponds to exact additivity; values above or below one indicate amplification or cancellation from the cross terms.

Cross-term bias is small and predictable. For this analysis, we construct A_ℓ as the union of each token's top-1 expert, matching the active-set construction used by SERE and OEA. Figure 1(b) groups tokens by the number of omitted experts, $|U_t|$, and reports their mean ρ_t. The ratio increases gradually with $|U_t|$, showing that cross terms introduce a modest bias in the additive error as more experts are omitted. Each point corresponds to tokens with the same integer value of $|U_t|$. The dashed curves show one-parameter fits, with fitting details provided in Appendix A.2.

Thus, the additive objective does not perfectly predict the magnitude of the exact error. The more important question for our method is whether this bias changes which experts should be retained.

Additive scoring preserves selection quality. Figure 1(c) evaluates the fixed-budget selection problem. At each decode position, we compare the additive selection A_ℓ^{add} with a selection A_ℓ^{cross} that accounts for the cross terms, using the same budget $M = 12$. We construct A_ℓ^{cross} greedily by repeatedly adding the expert that produces the largest reduction in the exact objective. This accounts for all cross terms, although it does not exhaustively search all subsets of size M.

Both selections are evaluated using the exact error from Equation 2. Panel (c) reports $\mathcal{E}(A_\ell^{\text{add}})/\mathcal{E}(A_\ell^{\text{cross}})$, which measures the additional exact error incurred by additive selection relative to the cross-term selection. The 95th-percentile ratios are 1.018, 1.025, and 1.032 for Qwen1.5-MoE-A2.7B, DeepSeek-V2-Lite, and Qwen3-30B-A3B, respectively. Thus, in 95% of measured layer and decode position pairs, discarding the cross terms increases the exact error by no more than 1.8%, 2.5%, and 3.2%.

These results show an important distinction: **cross terms affect the magnitude of the error, but have little effect on which experts should be retained.**

The same geometry also informs how unavailable experts should be handled. SERE redirects an unavailable route to a similar expert already in the batch active set (Wu et al., 2026a), which assumes that one expert's output can approximate another's.

We test this assumption directly. Suppose that an omitted expert u is replaced by an active expert v while retaining u's routing weight. The substitution error is

$$\|w_{t,u}E_u(z_t) - w_{t,u}E_v(z_t)\|_2^2 = w_{t,u}^2 \|E_u(z_t) - E_v(z_t)\|_2^2 . \tag{5}$$

By comparison, dropping expert u produces error $w_{t,u}^2\|E_u(z_t)\|_2^2$. Substitution is better only if $2\langle E_u(z_t), E_v(z_t)\rangle > \|E_v(z_t)\|_2^2$. When the two output norms are comparable, this approximately requires $c_{t,uv} > 1/2$, far above the similarities observed in Figure 1(a). This supports using the additive score D_u for selection without treating one expert's output as a surrogate for another. Table 3 further shows that substitution performs worse than simply dropping the unavailable contribution.

3.4 Predicting Expert Output Energy

Computing D_u requires $\|E_u(z_t)\|_2^2$, which is unavailable before expert execution. We therefore predict this quantity from the current token representation. For each expert u, calibration examples use the target $s_u(z) = \log(\|E_u(z)\|_2^2 + \epsilon)$. After normalizing the input as $\bar{z} = z/\|z\|_2$, we fit the linear model

$$\widehat{s}_u(z) = a_u^\top \bar{z} + b_u, \qquad \widehat{e}_u(z) = \exp(\widehat{s}_u(z)), \qquad \widehat{D}_u = \sum_{t \in \mathcal{B},\, u \in R_t} w_{t,u}^2 \widehat{e}_u(z_t). \tag{6}$$

Each predictor is fit independently by ridge regression on routed token-expert pairs collected during calibration. At inference, BASE selects the M experts with the largest $\widehat{D}_u$.

We also evaluate a fixed mean expert energy and a predictor with a shared nonlinear residual in Section 4.3; the linear predictor gives the best accuracy–efficiency tradeoff.

3.5 Inference Procedure

At each MoE layer, BASE computes $\widehat{D}_u$ for the routed experts and loads the M experts with the largest scores (Figure 2). Each token then executes its K highest-ranked experts within this active set. If one of its original routes is unavailable, it backfills the slot with its next-preferred expert already in the active set, similar to OEA (Oncescu et al., 2025). Backfill therefore requires additional expert computation but no additional expert weight transfer.

Backfilled experts use their original router scores, with the normalization denominator computed from the token's original top-K experts rather than renormalized over the executed experts as in OEA. Selection- and fill-rule ablations are given in Sections 4.3 and 4.4; kernel details are provided in Appendix A.5.

4 Experiments

4.1 Experimental Setup

Models. We evaluate BASE on three instruction-tuned MoE models with different routing configurations. Qwen3-30B-A3B-Instruct-2507 (Yang et al., 2025a) contains 48 MoE layers, 128 routed experts per layer, and activates 8 routed experts per token. Qwen1.5-MoE-A2.7B-Chat (Bai et al., 2023) contains 24 MoE layers, 60 routed experts, and activates 4 routed experts per token. DeepSeek-V2-Lite-Chat (DeepSeek-AI et al., 2024) contains 26 MoE layers, 64 routed experts, and activates 6 routed experts per token.

Benchmarks and metrics. We evaluate all methods on the full, unsubsampled sets from eight generative benchmarks: CMMLU (Li et al., 2024a), BBH (Suzgun et al., 2022), MATH (Hendrycks et al., 2021), BoolQ (Clark et al., 2019), GSM8K (Cobbe et al., 2021), MBPP (Austin et al., 2021), MATH-401 (Yuan et al., 2023), and HumanEval (Chen et al., 2021). We report accuracy for CMMLU, BBH, MATH, BoolQ, GSM8K, and MATH-401, pass rate for MBPP, and pass@1 for HumanEval. The overall score is the average of the eight benchmark scores.

Baselines and implementation. We compare BASE with four batch-aware expert-selection methods, SERE (Wu et al., 2026a), OEA (Oncescu et al., 2025), Lynx (Gupta et al., 2024), and ExFold (Wu et al., 2026b). All methods are served with vLLM 0.9.2 (V0 engine) in bf16 with batch size 16 and identical prompting and sampling settings: temperature 0.7, top-p 0.8, top-k 20, and seed 0. Following common practice (Oncescu et al., 2025; Gupta et al., 2024), all methods run prefill with the dense model and apply expert selection only during decoding, where inference is memory-bound and cost grows with the number of experts fetched. We report variation across seeds in Appendix A.3. Each run uses a single NVIDIA H100 80GB GPU. During offline calibration, BASE fits its linear expert-energy predictors on a subset of the FineWeb-Edu sample-10BT training split (Penedo et al., 2024). The selected text is packed into 2,048-token sequences, totaling approximately 2.5 million calibration tokens per model. The ridge regressions are solved in closed form. On one NVIDIA RTX PRO 6000 Blackwell Max-Q GPU, calibration takes 38.9 minutes for Qwen3-30B-A3B, 17.4 minutes for DeepSeek-V2-Lite, and 9.7 minutes for Qwen1.5-MoE-A2.7B. The

		Methods \ Tasks	Exam			Math			Code		Avg. (Acc. ↑)	tok/s (↑)
			cmmlu	boolq	bbh	math	gsm8k	math_{401}	heval	mbpp		
Qwen3-30B-A3B		Dense	84.79	89.82	76.93	72.34	88.40	80.55	88.41	78.80	82.50	845.3
	Restricted $\approx S=1$	SERE $_{S=1}$	51.04	74.07	40.20	**29.26**	38.97	40.90	16.46	9.40	37.54	**1452.9**
		OEA $_{k_0=1}$	64.64	**86.51**	43.69	27.78	33.28	34.66	**43.90**	9.60	43.01	1372.5
		Lynx $_{\alpha=0.47,\ \beta=1}$	**68.46**	84.01	51.76	19.80	38.29	65.34	29.88	17.40	**46.87**	1304.9
		ExFold $_{D=8,\ P=8}$	40.92	67.52	**51.93**	15.42	**68.16**	**70.82**	15.24	**25.20**	44.40	1395.4
		BASE (ours) $_{M=10}$	**79.46**	**89.76**	**73.55**	**63.94**	**89.01**	**80.55**	**75.00**	**59.80**	**76.38**	**1463.5**
	Relaxed $\approx S=2$	SERE $_{S=2}$	78.64	89.20	64.88	64.68	79.68	76.31	78.66	29.80	70.23	1223.5
		OEA $_{k_0=2}$	80.65	**90.61**	65.56	63.96	77.10	71.82	82.93	28.60	70.15	1190.7
		Lynx $_{\alpha=3,\ \beta=2}$	**84.62**	90.31	**75.47**	**71.78**	87.72	**83.04**	**87.20**	**73.60**	**81.72**	1047.9
		ExFold $_{D=18,\ P=8}$	83.67	**90.34**	74.91	70.68	**88.86**	**82.04**	77.44	70.60	79.82	**1225.8**
		BASE (ours) $_{M=16}$	**83.94**	90.34	**76.85**	**72.76**	**90.14**	81.55	**87.20**	**77.00**	**82.47**	**1350.2**
Qwen1.5-MoE-A2.7B		Dense	69.35	80.73	36.68	14.82	53.83	60.85	48.17	34.40	49.85	1315.6
	Restricted $\approx S=1$	SERE $_{S=1}$	46.33	78.35	32.62	5.46	24.94	46.88	6.10	13.80	31.81	1720.1
		OEA $_{k_0=1}$	67.21	79.85	**36.99**	**9.44**	**41.77**	58.10	**32.93**	29.40	**44.46**	1711.8
		Lynx $_{\alpha=0.7,\ \beta=1}$	67.74	80.34	34.17	6.12	21.76	54.11	22.56	24.20	38.88	1677.6
		ExFold $_{D=11,\ P=4}$	**69.39**	**80.58**	36.62	8.34	25.63	**61.35**	25.00	**30.40**	42.16	**1744.8**
		BASE (ours) $_{M=11}$	**69.28**	**80.43**	**37.60**	**13.04**	**47.23**	**59.35**	**39.63**	**32.80**	**47.42**	**1799.6**
	Relaxed $\approx S=2$	SERE $_{S=2}$	65.42	80.40	35.77	12.80	46.78	57.36	39.02	28.60	45.77	1476.7
		OEA $_{k_0=2}$	69.16	80.00	36.23	**14.98**	**51.78**	**60.10**	48.78	30.80	**48.98**	1460.9
		Lynx $_{\alpha=3,\ \beta=2}$	**69.44**	**80.73**	36.35	14.20	51.40	55.36	41.46	31.80	47.59	**1479.6**
		ExFold $_{D=21,\ P=4}$	**69.35**	**80.73**	**37.22**	14.26	45.56	**60.35**	**51.83**	**32.20**	48.94	1468.6
		BASE (ours) $_{M=18}$	69.26	80.64	**37.07**	**15.94**	**52.69**	59.85	**50.00**	**32.60**	**49.76**	**1609.8**
DeepSeek-V2-Lite		Dense	52.93	80.76	49.14	22.52	58.53	69.08	50.00	45.80	53.59	1056.4
	Restricted $\approx S=1$	SERE $_{S=1}$	24.04	67.80	33.18	5.12	26.08	44.39	6.10	3.80	26.31	**1633.2**
		OEA $_{k_0=1}$	36.41	**79.02**	43.76	14.52	**48.45**	63.84	**35.37**	**33.00**	**44.30**	1589.2
		Lynx $_{\alpha=0.5,\ \beta=1}$	35.66	75.44	35.66	5.42	11.30	52.62	3.66	18.00	29.72	1514.8
		ExFold $_{D=9,\ P=6}$	**42.17**	78.26	**44.82**	**15.20**	46.78	**69.08**	19.51	30.20	43.25	1621.7
		BASE (ours) $_{M=10}$	**45.86**	**79.11**	**46.64**	**19.46**	**56.63**	**68.83**	**46.34**	**44.40**	**50.91**	**1679.4**
	Relaxed $\approx S=2$	SERE $_{S=2}$	43.73	79.85	44.30	20.54	53.07	64.59	44.51	34.80	48.17	1365.2
		OEA $_{k_0=2}$	46.81	80.43	**48.04**	21.52	56.79	68.08	**53.05**	41.80	52.06	1362.1
		Lynx $_{\alpha=0.7,\ \beta=1}$	47.05	**80.86**	43.71	21.36	51.10	67.83	45.12	40.40	49.68	1302.3
		ExFold $_{D=19,\ P=6}$	**49.50**	**81.31**	47.85	**22.68**	**60.58**	**68.83**	50.00	**46.80**	**53.44**	**1381.3**
		BASE (ours) $_{M=19}$	**51.01**	80.28	**48.48**	**24.44**	**59.82**	**70.32**	**52.44**	**45.40**	**54.02**	**1435.7**

Table 1: Main quality and throughput comparison across three MoE architectures under restricted and relaxed operating regimes. Avg. is the average across eight benchmarks, and tok/s reports end-to-end throughput at batch size 16 on a single H100 GPU. Within each model and regime, **bold** marks the two highest values and shading marks the best. Dense denotes the original model with no experts skipped.

predictors can be calibrated independently and in parallel across layers and experts, and our implementation processes them in groups of eight layers. Further details are provided in Appendix A.1. BASE uses the backfill procedure described in Section 3.5. We ablate this routing rule and the expert-selection rule in Sections 4.4 and 4.3, respectively.

Comparison protocol. Because the methods use different expert-selection strategies, we compare them at matched decoding speed and measure model quality. SERE's S and OEA's k_0 both specify how many top-ranked experts are retained per token before their union forms the initial batch active set; the methods differ in how they fill the remaining routing slots. SERE also has a similarity threshold (ρ in Wu et al., 2026a) that fetches additional experts when no active expert is similar enough, improving quality but lowering decoding speed; to keep budgets matched, we set $\rho = 0$ for all SERE runs. We define two operating regimes. The *restricted* regime uses SERE $S = 1$ and OEA $k_0 = 1$ as reference settings, while the *relaxed* regime uses $S = 2$ and $k_0 = 2$. For each model, we select the BASE budget and the Lynx and ExFold settings that place them in the corresponding decoding-speed range. Lynx's α, β and ExFold's D, P are method-specific hyperparameters that control the accuracy–speed trade-off; we set them to match the decoding speed of the union-based SERE and OEA settings in each regime. These settings are fixed before quality is evaluated.

Throughput is measured end to end, including prefill, on the same 16 prompts and the same machine using a single NVIDIA H100 80GB GPU. Every sequence is forced to generate exactly 384

Selection score	Retained energy (%)	Mean accuracy
Top-1 union	85.91	58.68
Router weight sum	82.61	42.87
Square weight sum	90.98	55.50
Static expert energy	92.65	64.46
BASE linear predictor	96.47	73.50
Nonlinear predictor	98.20	72.07
Oracle energy	100.00	73.58

Table 2: Selection-rule ablation on Qwen3-30B-A3B with active-set size matched to the top-1 union. Retained energy measures the oracle energy preserved by the selected set.

Fill rule	HEval	GSM8K	MATH	BBH	Avg.	tok/s
Drop	68.90	86.00	72.00	69.91	74.20	1473.4
Backfill	**72.56**	**88.00**	**73.00**	**72.22**	**76.45**	**1479.3**
Substitution	7.93	36.00	22.50	43.52	27.49	1477.0

Table 3: Fill-rule ablation on Qwen3-30B-A3B using the BASE linear selector with $M = 10$. Drop removes unavailable contributions, backfill uses each token's next-preferred experts in the active set, and substitution redirects each unavailable expert to its most similar active expert. Avg. is the average across the four benchmarks.

tokens with end-of-sequence termination disabled, preventing any method from gaining throughput by producing shorter outputs. We report the best of three runs after warm-up.

4.2 Main Results

Table 1 reports the main quality and throughput results. Under the restricted regime, BASE achieves the highest average quality and throughput on all three architectures. On Qwen3-30B-A3B, BASE reaches an average score of 76.38, compared with 46.87 for the strongest baseline. The corresponding gains are 2.96 points on Qwen1.5-MoE-A2.7B and 6.61 points on DeepSeek-V2-Lite, showing that predicted removal error is especially useful when only a small expert set can be fetched. As more experts become available, the quality differences narrow: BASE exceeds the strongest baseline by 0.75, 0.78, and 0.58 points on Qwen3-30B-A3B, Qwen1.5-MoE-A2.7B, and DeepSeek-V2-Lite, respectively. BASE also maintains the highest throughput in the relaxed regime. Overall, BASE provides the best average quality and throughput across both operating regimes and all three evaluated architectures.

4.3 Ablation: Selection Rule

We evaluate selection criteria on Qwen3-30B-A3B with $B = 16$, dense prefill, and the same backfill procedure and evaluation examples for every method. We report mean accuracy across seven benchmarks using our Hugging Face implementation on fixed subsets: all 164 HumanEval problems, the first 200 problems of MATH, GSM8K, BoolQ, MBPP, and MATH-401, and the first 8 problems of each of the 27 BBH subtasks. We match expert-loading cost at every decode position by setting the budget to the size of the tokens' top-1 union and requiring every method to select the same number of experts. Thus, the methods differ only in how experts are ranked.

We compare the top-1 union, the sum of router weights, and the sum of squared router weights against scores that also use expert output energy. Static expert energy replaces the token-specific energy in our removal score with each expert's mean calibration energy, while BASE uses the prediction from Section 3.4. We also include a nonlinear predictor as a higher-capacity variant. We additionally use the true expert output energies to form an oracle ranking under the same budget. This oracle is not deployable because it requires executing all routed experts before selection. Retained energy measures the fraction of oracle contribution energy contained in the selected set, normalized by the energy retained by the oracle set at the same decode position. Appendix A.6 gives the full experimental setup and the formal definitions of every score, the oracle, and retained energy.

Table 2 shows that router scores alone do not reliably identify the experts whose removal most affects quality. BASE increases retained energy from 90.98% for squared router weights to 96.47%, while mean accuracy rises from 55.50% to 73.50%, close to the oracle value of 73.58%. The nonlinear predictor retains slightly more energy but does not improve accuracy, so we use the linear predictor.

4.4 Ablation: Fill Rule

Table 3 isolates how unavailable experts should be handled after selection. Using Qwen3-30B-A3B with $M = 10$, we hold the BASE linear selector and active set fixed and compare three fill rules, using the same implementation on all 164 HumanEval problems, the first 200 GSM8K and MATH

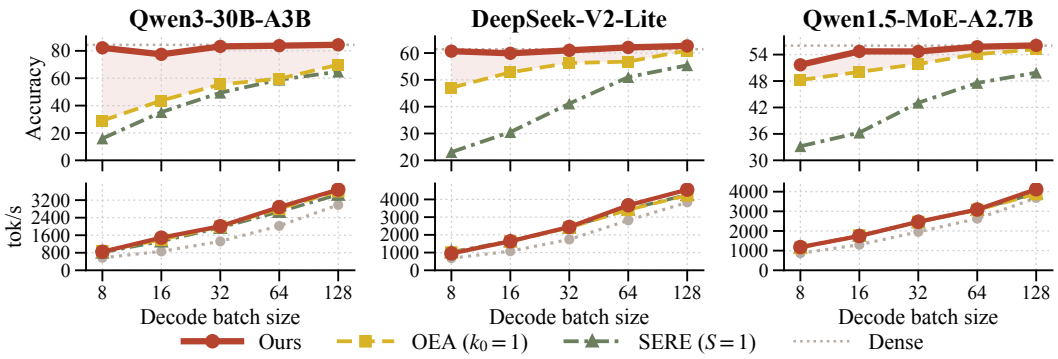


Figure 3: Effect of decode batch size on quality and throughput across three MoE architectures. The top row reports the average across HumanEval, MBPP, MATH-401, and BoolQ; the bottom row reports token throughput. The dotted gray line denotes the dense model. We vary B from 8 to 128, and BASE's expert budget is speed-matched to the faster baseline at each batch size. Appendix A.7 gives the budgets and further details.

problems, and the first 8 problems of each of the 27 BBH subtasks. Dropping simply removes unavailable contributions. Following SERE (Wu et al., 2026a), substitution redirects each unavailable expert to the most similar expert in the active set according to SERE's calibrated expert-similarity metric, while preserving the original routing weight. Substitution reduces average accuracy to 27.49, consistent with Figure 1(a), where co-routed expert outputs are nearly orthogonal and contain little shared directional information that would allow one expert's output to replace another. Backfill instead uses each token's next-preferred experts within the active set and improves average accuracy from 74.20 to 76.45. Although backfill performs additional computation, these experts have already been loaded and thus introduce no additional expert-weight transfers. The added FLOPs therefore have negligible effect on throughput in the memory-bound decoding regime.

4.5 Ablation: Effect of Decode Batch Size

Figure 3 evaluates whether BASE's advantage depends on the decode batch size. We sweep $B \in \{8, 16, 32, 64, 128\}$ across all three architectures and compare against the restricted SERE $S = 1$ and OEA $k_0 = 1$ settings. At each batch size, we first measure both baselines on the same H100 and use the faster token throughput as the speed target. We then choose the BASE expert budget to match this target, fix that budget, and evaluate quality. Thus, the comparison controls for serving speed rather than expert count. At matched throughput, BASE maintains relatively stable quality across the full range of batch sizes, while SERE and OEA degrade more sharply as the batch becomes smaller. BASE's quality advantage is therefore largest at smaller batch sizes and remains positive as the batch grows. These results show that BASE's quality advantage at comparable serving speed is consistent across batch sizes and is not specific to the $B = 16$ setting used in the main evaluation.

5 Conclusion

MoE sparsity at the token level does not guarantee efficient batched decoding, since concurrent requests can collectively activate many experts. We introduced BASE, which selects experts under a fixed batch-level budget according to their predicted effect on the MoE-layer output. Expert selection remains robust after discarding cross-expert terms, allowing BASE to rank experts independently using removal costs estimated by a lightweight linear predictor. At tight expert budgets, BASE improves average accuracy by 3.0–29.5 points over existing baselines at comparable speed. At relaxed budgets, it achieves comparable quality while maintaining higher throughput. Relative to dense inference, relaxed BASE settings are 22–60% faster while remaining within 0.4 accuracy points of dense quality; the most constrained setting is 73% faster with a 6.1-point loss. One potential limitation is that BASE requires more offline calibration time and compute than simpler baselines.

AI USE STATEMENT

In this work, we used generative AI tools to draft and revise portions of the manuscript, improve clarity and organization, assist with code debugging and LaTeX formatting, generate scripts for data visualization, organize experimental observations, and refine figure and table descriptions. We did not use generative AI tools to generate experimental data, modify reported numerical results, conduct the primary literature review, or draw scientific conclusions. Human-subject interaction, participant communication, and qualitative interview analysis are not applicable to this work. We reviewed all AI-assisted work, and AI-generated suggestions were treated as drafts. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To support reproducibility, the detailed methodology and evaluation procedure are provided in the experimental section and appendix. These sections document the model checkpoints, datasets, baseline configurations, calibration procedure, inference settings, hardware, throughput measurements, ablations, and custom GPU kernels. All evaluated models and benchmarks are publicly available, and our implementation code and evaluation scripts will be publicly released upon acceptance.

REFERENCES

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming throughput-latency tradeoff in llm inference with sarathi-serve. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 117–134, 2024.

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, 2021. URL https://arxiv.org/abs/2108.07732.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Oana Balmau, Anne-Marie Kermarrec, Rafael Pires, André Loureiro Espírito Santo, Martijn de Vos, and Milos Vujasinovic. Accelerating moe model inference with expert sharding. In *Proceedings of the 5th Workshop on Machine Learning and Systems*, pp. 192–199, 2025.

Hao Mark Chen, Zhiwen Mo, Royson Lee, Qianzhou Wang, Da Li, Shell Xu Hu, Wayne Luk, Timothy Hospedales, and Hongxiang Fan. Dynamic expert sharing: Decoupling memory from parallelism in mixture-of-experts diffusion llms, 2026. URL https://arxiv.org/abs/2602.00879.

I-Chun Chen, Hsu-Shen Liu, Wei-Fang Sun, Chen-Hao Chao, Yen-Chang Hsu, and Chun-Yi Lee. Retraining-free merging of sparse mixture-of-experts via hierarchical clustering. 2024.

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

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions, 2019. URL https://arxiv.org/abs/1905.10044.

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, 2021. URL https://arxiv.org/abs/2110.14168.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models, 2024. URL https://arxiv.org/abs/2401.06066.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, T. Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Liu, Xin Xie, Xingkai Yu, Xinnan Song, Xinyi Zhou, Xinyu Yang, Xuan Lu, Xuecheng Su, Y. Wu, Y. K. Li, Y. X. Wei, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Zheng, Yichao Zhang, Yiliang Xiong, Yilong Zhao, Ying He, Ying Tang, Yishi Piao, Yixin Dong, Yixuan Tan, Yiyuan Liu, Yongji Wang, Yongqiang Guo, Yuchen Zhu, Yuduan Wang, Yuheng Zou, Yukun Zha, Yunxian Ma, Yuting Yan, Yuxiang You, Yuxuan Liu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhewen Hao, Zhihong Shao, Zhiniu Wen, Zhipeng Xu, Zhongyu Zhang, Zhuoshu Li, Zihan Wang, Zihui Gu, Zilin Li, and Ziwei Xie. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 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*, 2022.

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.

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, 2021. URL https://arxiv.org/abs/2103.03874.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training Compute-Optimal Large Language Models, March 2022. URL http://arxiv.org/abs/2203.15556. arXiv:2203.15556 [cs].

Quzhe Huang, Zhenwei An, Nan Zhuang, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. Harder Tasks Need More Experts: Dynamic Routing in MoE Models. *arXiv preprint arXiv:2403.07652*, 2024.

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. *arXiv preprint arXiv:2511.15690*, 2025.

Ranggi Hwang, Jianyu Wei, Shijie Cao, Changho Hwang, Xiaohu Tang, Ting Cao, and Mao Yang. Pre-gated moe: An algorithm-system co-design for fast and scalable mixture-of-expert inference, 2024. URL https://arxiv.org/abs/2308.12066.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):79–87, March 1991. ISSN 0899-7667. doi: 10.1162/neco.1991.3.1.79. URL https://doi.org/10.1162/neco.1991.3.1.79.

Ajay Jaiswal, Jianyu Wang, Yixiao Li, Pingzhi Li, Tianlong Chen, Zhangyang Wang, Chong Wang, Ruoming Pang, and Xianzhi Du. Finding Fantastic Experts in MoEs: A Unified Study for Expert Dropping Strategies and Observations, April 2025. URL http://arxiv.org/abs/2504.05586. arXiv:2504.05586 [cs].

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Keisuke Kamahori, Tian Tang, Yile Gu, Kan Zhu, and Baris Kasikci. Fiddler: Cpu-gpu orchestration for fast inference of mixture-of-experts models, 2025. URL https://arxiv.org/abs/2402.07033.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models, January 2020. URL http://arxiv.org/abs/2001.08361. arXiv:2001.08361 [cs, stat].

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 ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

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*, 2025.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. Cmmlu: Measuring massive multitask language understanding in chinese, 2024a. URL https://arxiv.org/abs/2306.09212.

Jiamin Li, Qiang Su, Yitao Yang, Yimin Jiang, Cong Wang, and Hong Xu. Adaptive gating in mixture-of-experts based language models. *arXiv preprint arXiv:2310.07188*, 2023.

Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. Merge, Then Compress: Demystify Efficient SMoE with Hints from Its Routing Policy. In *The Twelfth International Conference on Learning Representations*, 2024b.

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*, 2024.

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*, 2024.

Alexandre Muzio, Alex Sun, and Churan He. Seer-moe: Sparse expert efficiency through regularization for mixture-of-experts. *arXiv preprint arXiv:2404.05089*, 2024.

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.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale, 2024. URL https://arxiv.org/abs/2406.17557.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *International conference on machine learning*, pp. 18332–18346. PMLR, 2022.

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

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them, 2022. URL https://arxiv.org/abs/2210.09261.

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 *The Fourteenth International Conference on Learning Representations*, 2026a.

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.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

Cheng Yang, Yang Sui, Jinqi Xiao, Lingyi Huang, Yu Gong, Yuanlin Duan, Wenqi Jia, Miao Yin, Yu Cheng, and Bo Yuan. MoE-i^2: Compressing mixture of experts models through inter-expert pruning and intra-expert low-rank decomposition. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 10456–10466, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.612. URL https://aclanthology.org/2024.findings-emnlp.612/.

Haoqi Yang, Luohe Shi, Qiwei Li, Zuchao Li, Ping Wang, Bo Du, Mengjia Shen, and Hai Zhao. Faster moe llm inference for extremely large models, 2025b. URL https://arxiv.org/abs/2505.03531.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. How well do large language models perform in arithmetic tasks?, 2023. URL https://arxiv.org/abs/2304.02015.

Shuzhang Zhong, Ling Liang, Yuan Wang, Runsheng Wang, Ru Huang, and Meng Li. Adapmoe: Adaptive sensitivity-based expert gating and management for efficient moe inference. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–9, 2024a.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. DistServe: Disaggregating Prefill and Decoding for Goodput-Optimized Large Language Model Serving. In *USENIX Symposium on Operating Systems Design and Implementation*, 2024b.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

A Appendix

A.1 Calibration Details

Data. We calibrate BASE using educational web text from `HuggingFaceFW/fineweb-edu`, configuration `sample-10BT`, training split. We read the data in deterministic stream order and retain documents containing more than 128 tokens under the Qwen1.5 tokenizer. This produces 2,708 documents. We reserve the final fifth of the documents for validation before packing, so the training and validation splits are disjoint by document. Using the Qwen tokenizer, the training split contains 1,199 sequences and 2,455,552 tokens, with 239 additional validation sequences. Using the DeepSeek tokenizer, the same text produces 1,230 training sequences and 2,519,040 tokens, with 246 validation sequences. All sequences contain 2,048 tokens. The validation sequences are used only for the retained-energy analysis and never for fitting the predictors.

Streaming sufficient statistics. For every MoE layer and routed expert, the calibration target is $y = \log \|E_u(z)\|_2^2$, and the input is the normalized hidden state $\bar{z} = z/\|z\|_2$. We process batches of eight sequences using teacher-forced forward passes. For each routed token-expert pair, we run the expert and compute its true output energy. We then update the sufficient statistics used for ridge regression: the sample count, first moments, Gram matrix, and input-target cross-moment. Token-level hidden states and expert outputs are discarded after every batch, so calibration does not create an activation cache. Gram matrices are accumulated in fp32, while vector and scalar statistics use fp64.

The predictors can be calibrated independently and in parallel across layers and experts. Each predictor depends only on the sufficient statistics collected for its own layer-expert pair, not on coefficients fitted for any other layer or expert. Our implementation processes groups of eight layers.

Closed-form ridge fitting. For each expert, we center the accumulated statistics and solve

$$a_u(\lambda) = (G_{c,u} + \lambda I)^{-1} q_{c,u}, \qquad b_u(\lambda) = \bar{y}_u - a_u(\lambda)^\top \bar{z}_u, \tag{7}$$

where $G_{c,u}$ and $q_{c,u}$ are the centered Gram matrix and cross-moment. One eigendecomposition per expert evaluates 25 values of λ, logarithmically spaced from 10^{-4} to 10^2. We select λ separately for each expert by marginal likelihood. Exponentiating the predicted log-energy can underestimate the mean output energy, so we apply an expert-specific correction for this bias. The correction is folded into the deployed intercept, so the inference-time predictor retains the form in Equation 6.

Model	Training tokens	Data passes	Time
Qwen3-30B-A3B	2,455,552	6	38.9 min
DeepSeek-V2-Lite	2,519,040	4	17.4 min
Qwen1.5-MoE-A2.7B	2,455,552	3	9.7 min

Table 4: Offline calibration cost on one NVIDIA RTX PRO 6000 Blackwell Max-Q GPU with 96 GB of memory.

The reported times begin with the first statistics-collection pass and end when the fitted coefficients are written. They include all forward passes, expert recomputation, and ridge solves, but exclude model loading, tokenization, and final artifact packaging. Calibration is performed once per model before deployment; no recalibration occurs during serving.

A.2 Cross-Term Bias: Formulation and Fitting

This section describes the one-parameter fits shown by the dashed curves in Figure 1(b). We compute the exact-to-additive error ratio ρ_t using the ground-truth outputs of all experts routed to each token. Therefore, this analysis contains no error from the expert-energy predictor.

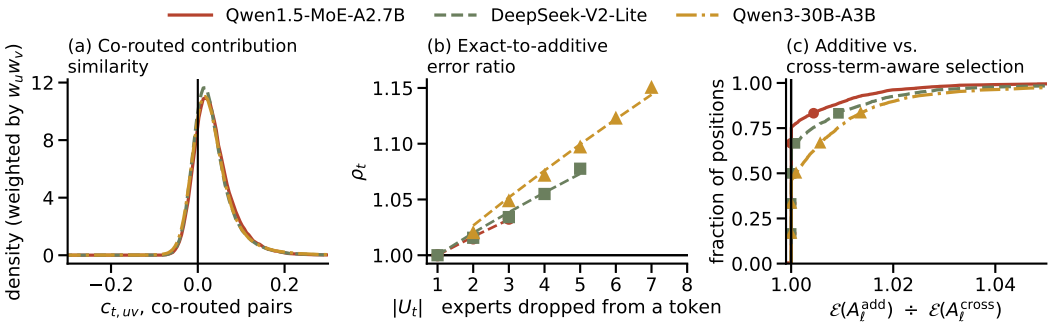


Figure 1 reproduced here for convenience.

We use the same three model checkpoints as in the main experiments and evaluate them on 16 held-out FineWeb-Edu sequences, each containing 2,048 tokens. At each sequence position, the corresponding 16 tokens form one decode batch. The active set is constructed as the union of the top-1 expert selected by each token in the batch. Shared experts are always active and are excluded from this analysis.

Recall that the exact-to-additive error ratio is

$$\rho_t = \frac{\left\|\sum_{u\in U_t} a_{t,u}\right\|_2^2}{\sum_{u\in U_t} \|a_{t,u}\|_2^2}, \tag{8}$$

where $U_t = R_t \setminus A_\ell$ is the set of experts originally routed to token t but omitted from the active set.

For an omitted expert u, let $m_{t,u} = w_{t,u}\|E_u(z_t)\|_2$ denote the magnitude of its weighted contribution. Expanding the exact reconstruction error gives

$$\left\|\sum_{u\in U_t} a_{t,u}\right\|_2^2 = \sum_{u\in U_t} m_{t,u}^2 + \sum_{\substack{u,v\in U_t \\ u\neq v}} m_{t,u} m_{t,v} c_{t,uv}, \tag{9}$$

where $c_{t,uv}$ is the cosine similarity between two omitted expert contributions. We average these similarities across omitted pairs, weighting each pair by the product of its contribution magnitudes:

$$\bar{c}_t = \frac{\sum_{u\neq v} m_{t,u} m_{t,v} c_{t,uv}}{\sum_{u\neq v} m_{t,u} m_{t,v}}, \tag{10}$$

and define their effective count as

$$|U_t|_{\text{eff}} = \frac{\left(\sum_{u\in U_t} m_{t,u}\right)^2}{\sum_{u\in U_t} m_{t,u}^2}. \tag{11}$$

These definitions give the exact identity

$$\rho_t - 1 = \bar{c}_t \left(|U_t|_{\text{eff}} - 1\right). \tag{12}$$

Thus, the deviation from additivity depends on both the alignment of the omitted contributions and the number of omitted contributions with meaningful magnitude. The effective count satisfies $1 \le |U_t|_{\text{eff}} \le |U_t|$. It equals $|U_t|$ when all omitted contributions have the same magnitude and approaches one when a single contribution dominates. When only one expert is omitted, $|U_t|_{\text{eff}} = 1$, so $\rho_t = 1$ exactly.

To obtain the dashed curves in Figure 1(b), we replace the token-dependent cosine $\bar{c}_t$ with one fitted constant $\bar{c}$ for each model:

$$\rho_t \approx 1 + \bar{c}\,(|U_t|_{\text{eff}} - 1)\,. \tag{13}$$

We group samples according to the number of omitted experts, $k = |U_t|$, and retain bins with at least 100 samples. For each bin, we compute the mean exact ratio $\bar{\rho}_k$ and mean effective count $\bar{e}_k$. We then fit $\bar{c}$ using equally weighted least squares with no intercept:

$$\bar{c} = \frac{\sum_k (\bar{\rho}_k - 1)(\bar{e}_k - 1)}{\sum_k (\bar{e}_k - 1)^2}. \tag{14}$$

The horizontal axis of Figure 1(b) remains the literal count $k = |U_t|$. The effective count enters only the dashed curve: for the tokens in bin k, we average $|U_t|_{\text{eff}}$ to obtain $\bar{e}_k$ and plot $1 + \bar{c}(\bar{e}_k - 1)$ at $x = k$. The two counts differ because omitted contributions can have unequal magnitudes. For example, Qwen3-30B-A3B tokens with $|U_t| = 7$ have a mean effective count of only 5.48, causing the fitted curve to bend below a line based directly on $|U_t|$.

Model	Fitted $\bar{c}$
Qwen1.5-MoE-A2.7B	0.021
DeepSeek-V2-Lite	0.026
Qwen3-30B-A3B	0.032

Table 5: Fitted constants for the one-parameter cross-term model in Figure 1(b).

The fitted curves remain within 0.65% of the observed mean ρ_t for every plotted bin. This confirms that cross terms introduce a small and predictable count-dependent bias in the magnitude of the additive error.

A.3 Robustness Across Sampling Seeds

We repeat the headline Qwen3-30B-A3B comparisons using seeds 0, 1, and 2. We evaluate the restricted BASE configuration with $M = 10$ against OEA with $k_0 = 1$ and SERE with $S = 1$. We also evaluate the relaxed BASE configuration with $M = 16$ against OEA with $k_0 = 2$.

Each seed controls both the vLLM engine and generation sampling, while all other inference settings remain unchanged. We use the same model, calibration data, prompts, sampling parameters, and H100 hardware as in the main evaluation, with a decode batch size of 16. This ablation covers five benchmarks: HumanEval, MBPP, MATH-401, GSM8K, and MATH-200. HumanEval, MBPP, MATH-401, and GSM8K use their full evaluation sets. To limit computational cost, MATH-200 uses the first 200 examples from the MATH test set. For every benchmark with an existing main-table reference, the seed-0 run reproduces the corresponding outputs item for item, confirming that this study uses the same inference implementation.

	Methods \ Tasks	HumanEval	MATH-200	MBPP	MATH-401	GSM8K
Restricted $\approx S=1$	BASE (ours) $_{M=10}$	73.58 ± 1.54 (75.00, 71.95, 73.78)	68.50 ± 3.28 (71.50, 69.00, 65.00)	60.27 ± 0.81 (59.80, 61.20, 59.80)	81.21 ± 1.15 (80.55, 82.54, 80.55)	88.78 ± 0.40 (89.01, 89.01, 88.32)
	OEA $_{k_0=1}$	43.49 ± 1.27 (43.90, 44.51, 42.07)	24.67 ± 2.31 (26.00, 22.00, 26.00)	10.13 ± 1.10 (9.60, 9.40, 11.40)	36.24 ± 1.51 (34.66, 36.41, 37.66)	32.80 ± 0.43 (33.28, 32.45, 32.68)
	SERE $_{S=1}$	13.01 ± 3.35 (16.46, 12.80, 9.76)	19.33 ± 1.76 (21.00, 17.50, 19.50)	8.73 ± 0.70 (9.40, 8.80, 8.00)	39.40 ± 1.50 (40.90, 37.91, 39.40)	40.46 ± 1.48 (38.97, 41.93, 40.49)
Relaxed $\approx S=2$	BASE (ours) $_{M=16}$	87.81 ± 1.61 (87.20, 86.59, 89.63)	79.17 ± 1.61 (81.00, 78.00, 78.50)	77.07 ± 0.31 (77.00, 77.40, 76.80)	81.80 ± 0.66 (81.55, 81.30, 82.54)	89.46 ± 0.59 (90.14, 89.08, 89.16)
	OEA $_{k_0=2}$	83.34 ± 0.70 (82.93, 82.93, 84.15)	68.00 ± 3.12 (69.00, 70.50, 64.50)	27.67 ± 1.45 (28.60, 28.40, 26.00)	73.07 ± 1.09 (71.82, 73.57, 73.82)	76.70 ± 0.36 (77.10, 76.57, 76.42)

Table 6: Robustness to sampling seeds on Qwen3-30B-A3B. Each entry reports the mean ± sample standard deviation over three sampling seeds, followed in parentheses by the individual results for seeds 0, 1, and 2. BASE uses the ridge-only linear predictor at the restricted budget $M = 10$ and relaxed budget $M = 16$. Bold and shading indicate the best mean within each operating regime.

Table 6 reports the mean and sample standard deviation across the three seeds, followed by the individual seed results in parentheses. BASE remains consistently stronger than its matched-speed baselines. Its sample standard deviation ranges from 0.31 to 1.61 on the four full benchmarks and from 1.61 to 3.28 on MATH-200. The closest comparison occurs against OEA on HumanEval in the relaxed regime, where BASE retains a 4.47-point advantage in mean accuracy.

A.4 Decode Latency Versus Expert Budget

We measure how the fixed expert budget affects decode latency on all three architectures. Measurements use vLLM 0.9.2 with the V0 engine, bf16 precision, decode batch size 16, and one NVIDIA RTX PRO 6000 Blackwell Max-Q GPU. We vary $M \in \{4, 8, 12, 16, 20, 24, 32\}$. For each setting, we run three trials and report the lowest decode time per output token (TPOT), the quantity directly modeled by our linear latency analysis.

We measure end-to-end decode time without adding profiling operations to the decode loop. Active-set size is measured separately to avoid affecting latency. In this experiment, the selector admits exactly M experts at each layer and decode step. The realized active-set size counts how many of those experts receive at least one token and are therefore dispatched. It can be smaller than M because the tokens in a batch may collectively request fewer than M distinct experts, leaving some admitted experts unused.

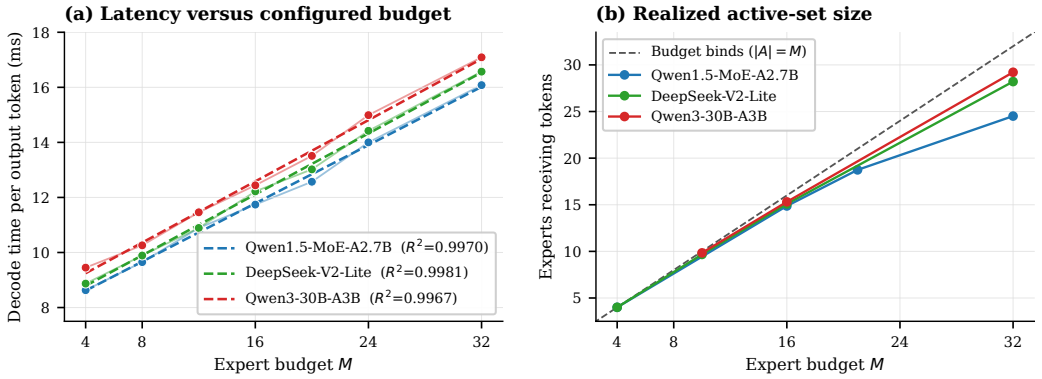


Figure 4: Decode latency as a function of the fixed expert budget. **(a)** Decode time per output token as M increases. Dashed lines show least-squares fits over $4 \le M \le 32$. **(b)** Number of experts that receive tokens under each configured budget. The dashed reference denotes $|A| = M$. At larger budgets, the batch may request fewer than M distinct experts, particularly for Qwen1.5-MoE-A2.7B.

Model	Experts	MoE layers	R^2	Intercept (ms)	Slope (μs/M)	Slope/layer (μs)
Qwen3-30B-A3B	128	48	0.9967	8.13	278.1	5.79
Qwen1.5-MoE-A2.7B	60	24	0.9970	7.55	264.5	11.02
DeepSeek-V2-Lite	64	26	0.9981	7.68	276.8	10.65

Table 7: Linear fits between decode latency and expert budget. We fit $t(M) = a + bM$ over $M \in \{4, 8, 12, 16, 20, 24, 32\}$. The slope reports the increase in decode time per output token from adding one unit to the expert budget.

Figure 4(a) shows a near-linear relationship between decode latency and M, with R^2 between 0.9967 and 0.9981 across the three architectures. This agrees with the finding of Oncescu et al.

(2025) that decode latency scales approximately linearly with the number of active experts in the memory-bound regime. The fitted slopes are also similar, ranging from 264.5 to $278.1\,\mu$s per unit of budget. Panel (b) shows that the realized active-set size remains close to M at the budgets used in our main experiments. Across the operating budgets of $M = 10$ to 21, realized expert activation is between 89% and 98% of M. Thus, within our evaluated operating range, the configured budget closely approximates the number of experts that are actually dispatched.

A.5 Custom GPU Kernels

BASE runs after the model's standard top-K router and before expert execution. For each token t, the routing stage provides its hidden state z_t, routed expert IDs R_t, and router weights $w_{t,u}$. We normalize z_t once and reuse it across the token's K routed experts. For each routed expert u, the scoring kernel combines this token-specific input with the expert-specific linear parameters a_u and b_u to predict its output energy, $\widehat{e}_u(z_t) = \exp(a_u^\top \bar{z}_t + b_u)$. It then multiplies this prediction by $w_{t,u}^2$ and aggregates the resulting removal costs across all tokens routed to the same expert, producing one batch-level score $\widehat{D}_u$ per expert. A second kernel selects the M highest-scoring experts to form an active set shared by the batch. Finally, the backfill kernel processes each token's router ranking and selects its K highest-ranked experts within this active set. We implement this data flow using four fused Triton launches and two lightweight elementwise operations, instead of roughly 20 separate PyTorch launches per layer.

To isolate the cost introduced by BASE, we begin with vLLM's fused top-K router and cumulatively add scoring, active-set selection, and backfill. The measurements use hidden states and router logits captured from real decode steps and the final linear-predictor implementation. Table 8 reports the incremental latency on a single H100 GPU at batch size 16. The total BASE overhead is measured directly from the complete routing function; small differences from the sum of the individual stages result from rounding and interactions between consecutive kernels.

Model	M	Standard router	BASE scoring	Expert selection	**Backfill**	Total BASE overhead	Per decode step	Share of decode step
Qwen3-30B-A3B	10	10.8	+3.6	+4.4	+6.7	+14.9	0.72 ms	7.1%
DeepSeek-V2-Lite	12	6.5	+9.8	+2.3	+5.8	+17.9	0.47 ms	5.1%
Qwen1.5-MoE-A2.7B	14	9.1	+3.7	+2.2	+5.3	+11.3	0.27 ms	3.1%

Table 8: Incremental latency of BASE-specific routing operations in μs per MoE layer at batch size 16. The standard-router column contains only vLLM's fused top-K routing cost. The next three columns report the cost added by each BASE stage. Per-step latency includes all MoE layers in the corresponding model.

At batch size 16, BASE adds 11.3–$17.9\,\mu$s per MoE layer, corresponding to 0.27–0.72 ms per decode step and 3.1–7.1% of the measured decode time. Across batch sizes from 8 to 128, the overhead remains between 11 and 21 μs per layer, or 1–8% of the decode step. Its relative share decreases as the batch grows because expert computation increases much more quickly than the routing overhead.

We verify the fused implementation against the complete deployed routing function for every profiled model and batch size. The kernels admit exactly M experts, and the stage-by-stage path produces identical expert IDs and routing weights to the complete function. Thus, the measurements decompose the deployed BASE routing path rather than a separate profiling implementation.

A.6 Selection-Rule Ablation: Setup and Score Definitions

This section gives the full setup and the formal definition of each score for the selection-rule ablation in Section 4.3.

We evaluate the selection criteria on Qwen3-30B-A3B with decode batch size $B = 16$, dense prefill, and the same backfill procedure and evaluation examples for every method. This ablation was run on a single NVIDIA RTX PRO 6000 Blackwell Max-Q GPU. During offline calibration, we fit the expert-energy predictors using educational web text from the FineWeb-Edu sample-10BT training

split (Penedo et al., 2024). Documents are selected deterministically, filtered to contain more than 128 tokens, and packed into 2,048-token sequences. We report mean accuracy across HumanEval, MATH, GSM8K, BIG-Bench Hard, BoolQ, MBPP, and MATH-401.

We match expert-loading cost separately at every decode position. Specifically, the budget is the number of distinct experts in the tokens' top-1 union, and every method selects exactly that many experts. For $B = 16$, this produces a mean active-set size of 13.34, with values ranging from 1 to 16. Thus, only the criterion used to rank experts changes across methods. We compare two families: scores derived only from router information, and scores that also incorporate information about expert outputs.

Router-only selection. The first family uses only information produced by the router. The *top-1 union* retains every expert ranked first by at least one token, assigning binary importance according to union membership. The *router-weight sum* ranks experts by the sum of their routing weights across the batch, $S_u^{\text{weight}} = \sum_{t:u\in R_t} w_{t,u}$. The *squared router-weight sum* instead uses $S_u^{\text{sq}} = \sum_{t:u\in R_t} w_{t,u}^2$, matching the quadratic dependence on routing weights in our individual-damage objective. None of these criteria uses information about the outputs produced by the experts.

Router and expert-output selection. The second family augments the squared router-weight sum with information about expert output magnitude. The *static-energy* score uses $S_u^{\text{static}} = \sum_{t:u\in R_t} w_{t,u}^2 \mu_u$, where $\mu_u = \mathbb{E}_z[\|E_u(z)\|_2^2]$ is estimated during calibration and remains fixed during inference. BASE replaces this constant with the token-dependent estimate produced by our linear predictor, $S_u^{\text{BASE}} = \sum_{t:u\in R_t} w_{t,u}^2 \exp(a_u^\top \bar{z}_t + b_u)$. The *nonlinear predictor* is a more expensive variant that adds a shared nonlinear residual to this predictor.

Oracle-energy selection and evaluation. To separate prediction error from the underlying selection criterion, we compute an oracle score from the experts' realized outputs. For each token t, the true contribution energy of expert u is $w_{t,u}^2\|E_u(z_t)\|_2^2$. Aggregating this quantity across the batch gives the oracle score

$$D_u^\star = \sum_{t:u\in R_t} w_{t,u}^2 \|E_u(z_t)\|_2^2, \tag{15}$$

where R_t is the set of experts selected by the router for token t. This score cannot be computed before selection without executing every routed expert, which would defeat the purpose of reducing expert-weight transfers. The resulting *oracle-energy* selection is therefore not a deployable baseline. Instead, it provides a diagnostic upper bound under the same expert budget. Each method first uses its own importance criterion to rank the experts and select $A^{(m)}$. After the selection is fixed, we discard the method-specific scores and evaluate the selected experts using $D_u^\star$. Oracle selection sorts the experts directly by $D_u^\star$ under the same budget, producing $A^\star$. We define the energy retained by method m as

$$\text{ER}(m) = \frac{\sum_{u\in A^{(m)}} D_u^\star}{\sum_{u\in A^\star} D_u^\star}. \tag{16}$$

Because $A^\star$ contains the experts with the largest oracle scores, $0 \le \text{ER}(m) \le 1$. Values closer to 1 indicate that the method selects an expert set more similar in retained energy to the oracle selection. We compute this ratio independently at each decode position and report its mean across all positions.

A.7 Batch-Size Sweep: Experimental Details

This section gives the experimental details of the batch-size sweep in Section 4.5 and Figure 3. We sweep $B \in \{8, 16, 32, 64, 128\}$ on all three architectures and compare BASE with the restricted SERE ($S = 1$) and OEA ($k_0 = 1$) settings. All runs use vLLM 0.9.2 (V0 engine) on a single H100 GPU with the prompts, sampling settings, and seed of the main evaluation, and BASE uses the FineWeb-Edu calibration of Appendix A.1. Quality is the mean accuracy over the full HumanEval (164 problems), MBPP (500), MATH-401 (401), and BoolQ (3,270) sets, with generation limits of 512 tokens for the first three and 2,048 tokens for BoolQ.

We set the BASE expert budget at each B to match the token throughput of the faster baseline at that batch size. For $B = 8$, 16, 32, 64, and 128, this gives $M = 10$, 10, 24, 32, and 48 on Qwen3-30B-A3B, $M = 10$, 12, 19, 24, and 32 on DeepSeek-V2-Lite, and $M = 6$, 14, 21, 34, and 34 on Qwen1.5-MoE-A2.7B.

Figure 3 reports end-to-end throughput from the timing harness of the main results on a single H100: the batch is fixed to B, end-of-sequence tokens are ignored so that every sequence runs to its full length, CUDA graphs are enabled, and we report the best of three runs. The match is approximate: the faster baseline exceeds BASE at four points: OEA by 2.8% for Qwen3-30B-A3B at $B = 8$, SERE by 9.0% and 0.6% for DeepSeek-V2-Lite at $B = 8$ and $B = 16$, and OEA by 1.5% for Qwen1.5-MoE-A2.7B at $B = 16$. At these points, the accuracy of BASE is higher by 53.3, 37.6, 29.4, and 4.7 points, respectively.