MoE-Spec: Expert Budgeting for MoE Speculative Decoding

Anonymous authors
Paper under double-blind review

Abstract

Speculative decoding accelerates Large Language Model (LLM) inference by verifying multiple candidate tokens in parallel. For Mixture-of-Experts (MoE) models, each candidate token routes to different experts, and the target model must load the union of all activated experts to verify them. As the number of candidates grows, unique experts per layer approach the full model, and verification becomes as expensive as running a dense model. We propose MoE-Spec, a training-free method that enforces a fixed expert budget at each layer during verification. MoE-Spec aggregates routing probabilities across candidate tokens and loads only the top-B experts. Because the remaining experts are excluded, the budget B controls a tradeoff between throughput and quality. At conservative budgets, MoE-Spec matches EAGLE quality and often improves throughput; tighter budgets trade quality for additional throughput. Experiments on OLMoE-1B-7B, Qwen3-30B-A3B, and Mixtral-8x7B across reasoning, code generation, and summarization show that at conservative budgets MoE-Spec matches EAGLE quality while improving throughput, and at aggressive budgets retaining $\geq$90% of autoregressive quality, throughput improves 15–45% depending on model architecture. Code will be released upon acceptance.

1 Introduction

Mixture-of-Experts (MoE) models are increasingly used in state-of-the-art language models, including DeepSeek-R1 (Guo et al., 2025) and Llama-4 (Meta AI, 2025). MoEs reduce inference cost through sparse activation: each token activates only k of the N total experts per layer, so memory bandwidth scales with active rather than total parameters. Speculative decoding accelerates LLM inference by verifying multiple drafted tokens in parallel (Leviathan et al., 2023; Chen et al., 2023). For dense models, the weight-loading cost is amortized across all candidates in a verification pass. In the memory-bound regime, additional candidates therefore add relatively little cost compared with loading the model weights.

For MoE models, verification cost depends on how many unique experts appear across the draft tree. Each token routes independently to its own subset of experts, and the target model must load the *union* of all selected experts to verify the tree (Figure 2a). As draft size grows, so does this union: on OLMoE-1B-7B, a 127-token tree activates 54 of 64 experts per layer, approaching the full model (Figure 2b). At that point, verification can become more expensive than autoregressive decoding, negating the speedup from speculation.

Most of the bandwidth cost comes from experts with low routing probability, whose weighted contribution to the layer output is small. Routing weight across the draft tree is concentrated (Figure 2c): on OLMoE, the top 32 of 64 experts capture 93% of aggregate routing probability. The remaining experts add bandwidth cost but change the layer output little when removed. This means verification cost can be reduced by loading fewer experts, with the excluded experts contributing small changes to the layer output. The number of experts kept controls a tradeoff between throughput and quality that does not exist in dense speculative decoding.

We propose MoE-Spec, a method that enforces a fixed expert budget during verification (Figure 1). Given a draft tree, we aggregate routing probabilities across all tokens and load only the top-B experts per layer. Tokens whose preferred experts fall outside this set are rerouted to available alternatives.

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.

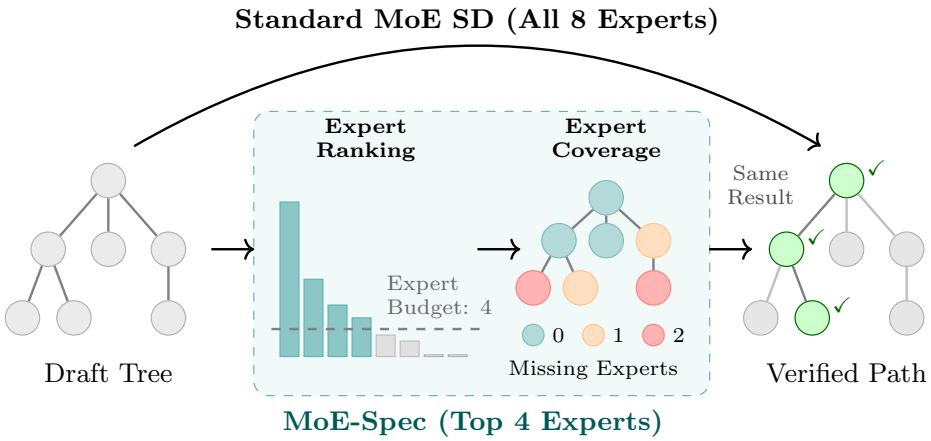


Figure 1: Overview of MoE-Spec. Standard MoE speculative decoding loads all N experts (N = 8) activated across the draft tree (top path). MoE-Spec ranks experts by aggregate routing probability and enforces a budget B (here $B = 4$), loading only top-scoring experts (bottom path). Tokens are color-coded by the number of missing experts from their natural routing. In this example, both paths accept the same three tokens despite MoE-Spec loading half as many experts.

The method is training-free and uses only information the router already computes, and requires no retraining or changes to draft construction or token acceptance.

We evaluate MoE-Spec on three MoE architectures across reasoning, code generation, and summarization. Conservative budgets match EAGLE-3 quality (Li et al., 2025) and often improve throughput; aggressive budgets retaining $\geq 90\%$ of autoregressive quality improve throughput by 15–45%. Our contributions are:

- We identify a sparsity collapse in tree-based MoE verification: a 127-token tree activates 54 of 64 OLMoE experts, although the top 32 carry 93% of aggregate routing weight.
- We introduce MoE-Spec, a training-free method whose expert budget controls the quality–throughput tradeoff without changes to model weights, draft construction, or token acceptance.
- Across three architectures and five benchmarks, MoE-Spec improves throughput by 15–45% over EAGLE at aggressive budgets retaining $\geq 90\%$ of autoregressive quality.
- An output-aware Oracle matches router-based reconstruction error with 25% fewer experts, quantifying the headroom for improved selection.

2 Background

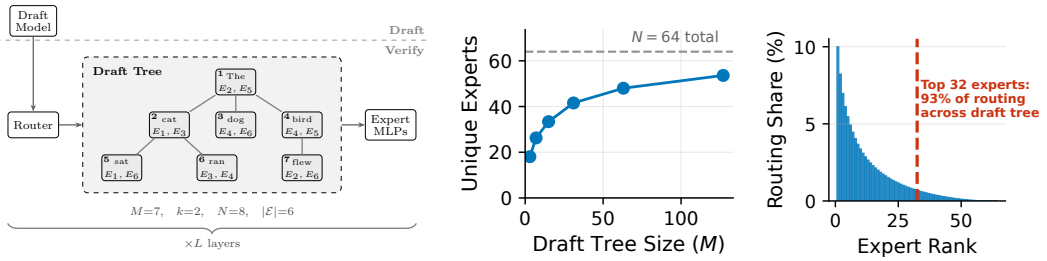
2.1 Mixture-of-Experts

Mixture-of-Experts (MoE) models achieve parameter efficiency by activating only a subset of model weights for each token. In an MoE layer, the standard feed-forward network is replaced with N parallel expert networks $\{E_1, \ldots, E_N\}$ and a learned router that selects k experts to process each token (where $k \ll N$).

Given a token representation $\mathbf{h} \in \mathbb{R}^d$, the router computes a probability distribution over experts via a linear projection followed by softmax:[1]

$$g_i(\mathbf{h}) = \frac{\exp(\mathbf{w}_i^\top \mathbf{h})}{\sum_{j=1}^{N} \exp(\mathbf{w}_j^\top \mathbf{h})}, \tag{1}$$

[1]Some architectures, such as DeepSeek-V3, use sigmoid-based routing instead. Our method applies to both, though we focus on softmax-routed models in experiments.



(a) MoE verification (b) Expert explosion (c) Heavy-tailed routing

Figure 2: Motivation for expert budgeting. (a) During verification, each token routes to k experts; the target model must load all unique experts $\mathcal{E}$ across the draft tree. (b) As tree size M grows, unique experts per layer approach N, negating sparse activation benefits. (c) Routing probabilities are heavy-tailed: the top 32 of 64 experts capture 93% of routing weight for a tree size of 60.

where $\mathbf{w}_i \in \mathbb{R}^d$ is the routing weight vector for expert i. Let $\mathcal{T}_k(\mathbf{h}) \subset \{1, \ldots, N\}$ denote the indices of the top-k experts by routing probability. The MoE layer output is a weighted combination of the selected experts:

$$\text{MoE}(\mathbf{h}) = \sum_{i \in \mathcal{T}_k(\mathbf{h})} \bar{g}_i(\mathbf{h}) \cdot E_i(\mathbf{h}), \tag{2}$$

where $\bar{g}_i(\mathbf{h})$ denotes the mixing weights. These are either the raw routing probabilities $g_i(\mathbf{h})$ or renormalized values $g_i(\mathbf{h}) / \sum_{j \in \mathcal{T}_k(\mathbf{h})} g_j(\mathbf{h})$, depending on the architecture (e.g., OLMoE renormalizes; Qwen3 does not).

Inference cost thus scales with the number of *active* parameters. For a single token, only k experts are loaded; when processing multiple tokens jointly, the model must load the *union* of all selected experts, which can approach N as token count grows.

2.2 Speculative Decoding with Draft Trees

Speculative decoding accelerates autoregressive generation by drafting multiple candidate tokens and verifying them in parallel (Leviathan et al., 2023; Chen et al., 2023). A lightweight *draft model* proposes candidate continuations, and the full *target model* verifies these candidates in a single forward pass. Accepted tokens are emitted immediately; rejected tokens trigger resampling from a corrected distribution. This process is provably lossless for greedy decoding as the output distribution matches standard autoregressive decoding.[2]

Modern methods such as EAGLE (Li et al., 2024b;c) extend this to *tree-structured* drafts, where multiple candidate tokens are proposed at each position. The target model verifies all candidates simultaneously using tree attention, accepting a path of tokens consistent with the target distribution.

For dense models, verification cost is effectively constant regardless of draft tree size: all model parameters are loaded once per forward pass, and the additional compute for extra tokens is negligible in the memory-bound regime. This makes aggressive speculation attractive—larger trees increase the expected number of accepted tokens per iteration. For MoE models, however, this assumption breaks down; we analyze this problem in Section 3.1.

3 Method

3.1 Verification Overhead in Tree Decoding

As described in Section 1, MoE verification cost scales with expert diversity across the draft tree. We formalize this as follows. Let $\mathcal{D} = \{\mathbf{h}_1, \ldots, \mathbf{h}_M\}$ denote a draft tree of M token representations. To

[2]In practice, floating-point non-associativity means that different batch compositions during verification can produce different reduction orders, causing small numerical differences even at greedy decoding (Gond et al., 2026). This affects all speculative decoding methods equally.

verify the tree, the target model must load the union of all selected experts:

$$\mathcal{E}(\mathcal{D}) = \bigcup_{t=1}^{M} \mathcal{T}_k(\mathbf{h}_t). \tag{3}$$

As Figure 2b shows, $|\mathcal{E}|$ approaches N for large trees, negating sparse activation benefits. However, routing weight is concentrated on a subset of these experts (Figure 2c). By enforcing a budget $B < |\mathcal{E}|$ and loading only the top-B experts, we reduce bandwidth while preserving verification quality. Because excluded experts are not loaded, the budget B introduces a tradeoff between throughput and output quality.

The severity of verification overhead depends on model sparsity, and the trend in MoE design is toward more experts with fixed k, which makes the problem worse. On Mixtral (N=8, k=2), each token touches 25% of the expert pool, so a 60-token tree saturates to the full model and there is little to cut. On OLMoE (N=64, k=8), each token touches 12.5%, the tree union reaches about 50 experts, and the remaining 14 carry little aggregate routing weight. On Qwen3 (N=128, k=8), each token touches 6%, and budgeting yields the largest gains in our experiments (Section 4.2). As N grows with k fixed, the expert union during verification expands, but each token's share of it shrinks, so more of the union consists of experts with low routing weight that can be excluded cheaply.

3.2 Expert Importance Ranking

Given an expert budget $B < N$, we select a shortlist $\mathcal{S}$ of B experts to load during verification. We consider three ranking strategies.

Static Ranking. The simplest approach precomputes a fixed ordering from calibration data by counting selection frequency:

$$c_i = \sum_{\mathbf{h} \in \text{calib}} \mathbf{1}[i \in \mathcal{T}_k(\mathbf{h})]. \tag{4}$$

The shortlist consists of the B experts with highest c_i. This incurs zero runtime overhead and enables permanent model pruning. For example, deploying only 32 of 64 experts halves memory requirements. At moderate budgets ($B \geq N/2$), static ranking performs comparably to adaptive methods on most tasks (Section D). The gap widens at aggressive budgets, where input-dependent routing becomes more valuable.

Router-Based Ranking. Our primary method aggregates routing probabilities across the draft tree:

$$s_i = \sum_{t=1}^{M} g_i(\mathbf{h}_t). \tag{5}$$

The shortlist $\mathcal{S}$ consists of the B experts with highest s_i. This requires no computation beyond what the router already produces, and adapts to each tree's routing pattern.

Oracle Ranking. As an output-aware reference, we consider a greedy oracle with access to routing decisions of the target model and expert outputs. The oracle greedily selects experts to minimize reconstruction error against the unpruned output. Let $O_t^* = \sum_{i \in \mathcal{T}_k(\mathbf{h}_t)} \bar{g}_i(\mathbf{h}_t) E_i(\mathbf{h}_t)$ denote the unbudgeted output for token t. Starting from $\mathcal{S} = \emptyset$, the oracle iteratively selects

$$i^* = \arg\min_{i \notin \mathcal{S}} \sum_{t=1}^{M} \left\| O_t^* - \text{MoE}_{\mathcal{S} \cup \{i\}}^{\text{sub}}(\mathbf{h}_t) \right\|_2^2. \tag{6}$$

This requires running all experts and is impractical for speedup, but estimates the headroom available to selection methods that use information beyond router probabilities.

3.3 Expert Coverage

Given shortlist $\mathcal{S}$, we must handle tokens whose natural routing includes experts outside $\mathcal{S}$. We consider two policies.

Truncation. Each token uses its natural top-k routing, but experts outside $\mathcal{S}$ contribute zero:

$$\text{MoE}_{\mathcal{S}}^{\text{trunc}}(\mathbf{h}_t) = \sum_{i \in \mathcal{T}_k(\mathbf{h}_t) \cap \mathcal{S}} \bar{g}_i(\mathbf{h}_t) \cdot E_i(\mathbf{h}_t). \tag{7}$$

Probabilities $\bar{g}_i$ are computed over the original top-k set (not re-computed for the reduced set). Tokens whose natural routing overlaps partially with $\mathcal{S}$ receive fewer than k experts. In the extreme case where $\mathcal{T}_k(\mathbf{h}_t) \cap \mathcal{S} = \emptyset$, the MoE output is zero and only the residual connection passes through. This effectively skips the MoE computation for that token at that layer.

Substitution. Top-k selection is constrained to $\mathcal{S}$. Let $\mathcal{T}_k^{\mathcal{S}}(\mathbf{h}_t)$ denote the $\min(k, |\mathcal{S}|)$ highest-scoring experts within $\mathcal{S}$, ranked by $g_i(\mathbf{h}_t)$. This definition also covers the partial shortlists created during greedy Oracle construction:

$$\text{MoE}_{\mathcal{S}}^{\text{sub}}(\mathbf{h}_t) = \sum_{i \in \mathcal{T}_k^{\mathcal{S}}(\mathbf{h}_t)} \bar{g}_i^{\mathcal{S}}(\mathbf{h}_t) \cdot E_i(\mathbf{h}_t), \tag{8}$$

where $\bar{g}_i^{\mathcal{S}}$ are the routing probabilities over the constrained selection. If a token's natural top-k includes experts outside $\mathcal{S}$, those experts are replaced by the highest-scoring available alternatives within the shortlist. Since all evaluated budgets satisfy $B \geq k$, every token receives exactly k experts in the final budgeted model, though they may differ from natural routing. We compare both policies empirically in Section 4.4.

4 Experiments

4.1 Experimental Setup

Models. We evaluate on three MoE architectures spanning different scales and expert configurations: OLMoE-1B-7B (Muennighoff et al., 2024), with 64 experts and $k = 8$ active per token (7B total, 1B active parameters); Qwen3-30B-A3B, with 128 experts and $k = 8$ (30B total, 3B active); and Mixtral-8x7B (Jiang et al., 2024), with 8 experts and $k = 2$ (47B total, 13B active). For speculative decoding, we use EAGLE-3 (Li et al., 2025) draft models for OLMoE and Qwen3, and the original EAGLE (Li et al., 2024b) for Mixtral.

Benchmarks. We evaluate on five benchmarks spanning mathematical reasoning, code generation, and summarization. GSM8K and MATH500 test multi-step arithmetic and competition-level math problems respectively, measured by accuracy. HumanEval and MBPP evaluate code generation via pass@1 on function synthesis tasks. CNN/DailyMail tests abstractive summarization, measured by ROUGE-L F1. We evaluate on 1,319 GSM8K examples, 500 MATH500 examples, 164 HumanEval problems, 257 MBPP problems, and 500 CNN/DailyMail examples.

Baselines. We compare against two baselines: autoregressive decoding (AR), which generates tokens sequentially without speculation, and EAGLE, the current state-of-the-art for tree-structured speculative decoding. We use EAGLE-3 for OLMoE and Qwen3, and EAGLE-1 for Mixtral where EAGLE-3 draft models are unavailable. All draft model adapters are from the official EAGLE repository.

Implementation. We implement MoE-Spec by extending the EAGLE-3 codebase[3] to add expert budgeting during the verification phase (Section B). We use the open-source EAGLE-3 implementation without changing draft-tree construction or token acceptance. During verification, we cap the number of experts executed per layer across all tokens in the draft tree. All main results use router-based expert selection with the substitution policy; ablations over selection methods and coverage policies appear in Section 4.4. Following Li et al. (2025), we use draft trees of 60 tokens and evaluate at greedy decoding ($T = 0$). Full hyperparameters and model configurations are in Section A.

[3] `https://github.com/SafeAILab/EAGLE`

Table 1: Main results at greedy decoding ($T = 0$), reporting speedup, task quality, mean acceptance length (τ), and expert budget (B). MoE-Spec$_C$ is the highest-throughput iso-quality configuration relative to EAGLE(-3); MoE-Spec$_A$ maximizes throughput while retaining $\geq 90\%$ of AR quality. Speedups exceeding EAGLE(-3) are **bold**.

	GSM8K				MATH500				HumanEval				MBPP				CNN/DM			
Method	Spd	Acc	τ	B	Spd	Acc	τ	B	Spd	P@1	τ	B	Spd	P@1	τ	B	Spd	R-L	τ	B
Mixtral-8x7B																				
AR	1.0×	45	–	–	1.0×	28	–	–	1.0×	23	–	–	1.0×	64	–	–	1.0×	18	–	–
EAGLE	1.6×	44	2.9	–	1.4×	28	2.6	–	1.7×	26	3.3	–	1.7×	61	3.3	–	1.3×	18	2.2	–
MoE-Spec$_C$	**2.0**×	44	2.8	5	**1.5**×	28	2.6	7	**2.0**×	28	3.3	6	1.7×	61	3.3	8	1.3×	18	2.2	8
MoE-Spec$_A$	**2.4**×	42	2.8	3	**1.8**×	26	2.6	5	**2.1**×	22	3.2	5	**2.0**×	59	3.2	6	**2.4**×	17	2.1	2
OLMoE-1B-7B																				
AR	1.0×	60	–	–	1.0×	23	–	–	1.0×	34	–	–	1.0×	44	–	–	1.0×	16	–	–
EAGLE-3	1.7×	60	3.0	–	1.9×	21	3.1	–	2.6×	35	4.1	–	2.3×	44	4.0	–	0.9×	17	1.2	–
MoE-Spec$_C$	1.7×	61	2.9	40	**2.3**×	23	3.1	24	**2.7**×	35	4.0	40	**2.4**×	44	4.0	40	0.9×	17	1.2	64
MoE-Spec$_A$	**2.0**×	58	2.8	24	**2.3**×	21	3.0	16	**2.8**×	32	4.0	32	**2.4**×	44	4.0	40	**1.2**×	15	1.1	16
Qwen3-30B-A3B																				
AR	1.0×	84	–	–	1.0×	71	–	–	1.0×	87	–	–	1.0×	86	–	–	1.0×	16	–	–
EAGLE-3	2.4×	85	3.8	–	2.1×	71	3.5	–	2.2×	87	3.7	–	2.0×	85	3.2	–	1.8×	15	2.4	–
MoE-Spec$_C$	**2.6**×	85	3.8	56	**2.5**×	71	3.4	32	2.2×	87	3.7	128	**2.2**×	85	3.1	56	**2.2**×	16	2.4	24
MoE-Spec$_A$	**3.5**×	82	3.6	16	**2.9**×	67	3.4	16	**2.4**×	82	3.7	56	**2.5**×	80	3.1	32	**2.3**×	15	2.3	16

4.2 Main Results

Table 1 compares autoregressive decoding (AR), EAGLE, and MoE-Spec across three MoE architectures and five benchmarks at greedy decoding ($T = 0$). We report two MoE-Spec operating points: *iso-quality* (MoE-Spec$_C$) selects the tightest budget that does not degrade the task metric relative to EAGLE, and *throughput-maximizing* (MoE-Spec$_A$) selects the most aggressive budget retaining $\geq$90% of AR quality. At iso-quality, MoE-Spec improves speedup over EAGLE on 10 of 15 configurations. At the aggressive setting, gains are larger: Qwen3 reaches 3.5× on GSM8K (vs. EAGLE's 2.4×), and Mixtral reaches 2.4× on GSM8K and CNN/DM (vs. EAGLE's 1.6× and 1.3×).

Sparser models benefit more from expert budgeting. Qwen3-30B-A3B (128 experts, k=8) maintains near-AR quality across a wide budget range. OLMoE-1B-7B (64 experts, k=8) shows steeper quality degradation at aggressive budgets. Mixtral-8x7B (8 experts, k=2) has less room to budget but still sees large iso-quality gains on GSM8K (2.0× vs. EAGLE's 1.6×) and HumanEval (2.0× vs. 1.7×, with higher P@1). Since acceptance lengths remain similar to EAGLE across configurations, the speedup gains primarily come from reduced verification cost per step.

Figure 4 shows the full quality-speedup tradeoff across all model-benchmark combinations. Unlike EAGLE, which provides a single operating point, MoE-Spec traces a Pareto curve from conservative to aggressive budgets. The tradeoff varies by task: code generation (HumanEval, MBPP) tolerates tighter budgets than reasoning (GSM8K, MATH500), consistent with routing concentration patterns (Section D). As MoE architectures scale toward more experts with sparser routing (e.g., DeepSeek-V3 with 256 experts, k=8), the expert union during verification will grow faster with tree size, but routing concentration will also increase, making budgeting both more necessary and more effective.

4.3 Speedup Analysis

The speedup gains in Table 1 arise because MoE-Spec loads at most B experts per layer while EAGLE-3 must load all experts activated across the draft tree, even those used by only a minority of tokens (Figure 2c). Figure 3 illustrates this effect on OLMoE-1B-7B at a fixed expert budget $B = 32$ as a function of tree size. For this experiment, we aggregate across all 5 datasets outlined in Section 4.1. Figure 3a shows that the average active number of experts per draft tree is significantly smaller for MoE-Spec compared to EAGLE-3 even at moderate tree sizes. For example, at tree size of 31, MoE-Spec uses on average 29 experts compared to 39 for EAGLE-3. At larger tree sizes, this gap widens: MoE-Spec saturates at 32 experts while EAGLE-3 approaches using all 64 experts per tree.

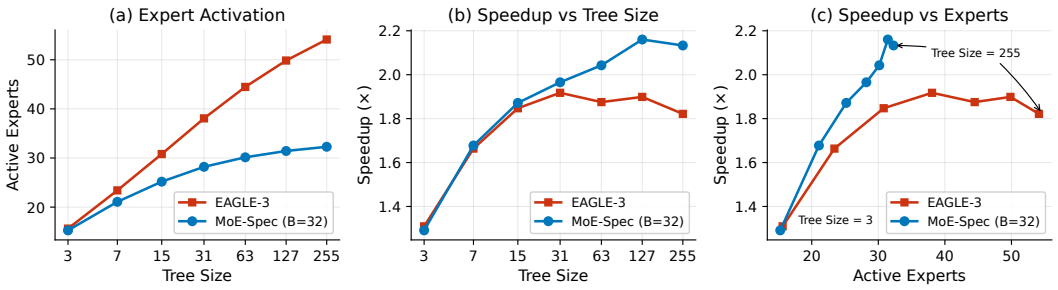


Figure 3: Expert activation and speedup on OLMoE-1B-7B. (a) Unique experts activated during verification as tree size increases. EAGLE-3 loads over 50 of 64 experts at large trees; MoE-Spec with $B = 32$ saturates at the budget. (b) Speedup relative to AR. EAGLE-3 peaks at tree size 31 then declines as expert loading dominates; MoE-Spec continues improving at larger trees. (c) Speedup vs. active experts. At tree size 255, EAGLE-3 loads 54 experts for 1.85× speedup; MoE-Spec loads 32 experts for 2.1× speedup.

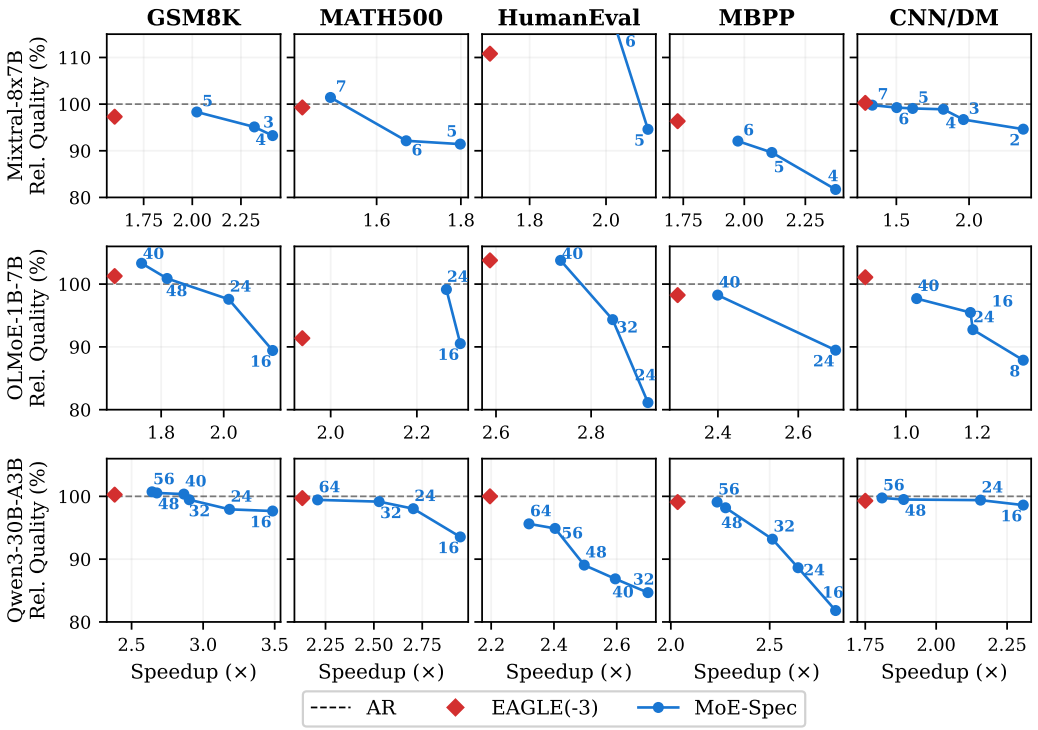


Figure 4: Quality-speedup tradeoff at $T = 0$ across three models (rows) and five benchmarks (columns). Each point is labeled with the expert budget B. Dashed line marks AR quality; diamonds mark EAGLE(-3).

The speedup for EAGLE peaks at tree size 31 then declines as straggler experts dominate verification cost, while MoE-Spec continues improving throughout (Figure 3b). Both methods achieve similar gains at small tree sizes where expert counts remain low. Beyond tree size 15, EAGLE peaks at 1.95× then declines to 1.85× at tree size 255. The memory bandwidth cost of loading straggler experts, those activated by only a few tokens in the tree, outpaces the benefit from longer accepted sequences. MoE-Spec avoids this overhead and continues improving throughout, reaching 2.1× at tree size 255.

Plotting speedup against active experts directly (Figure 3c) reveals a saturation point around 32 experts, beyond which additional loading yields diminishing returns. EAGLE operates past this point at large tree sizes (54 experts at tree size 255), while MoE-Spec stays below it by construction.

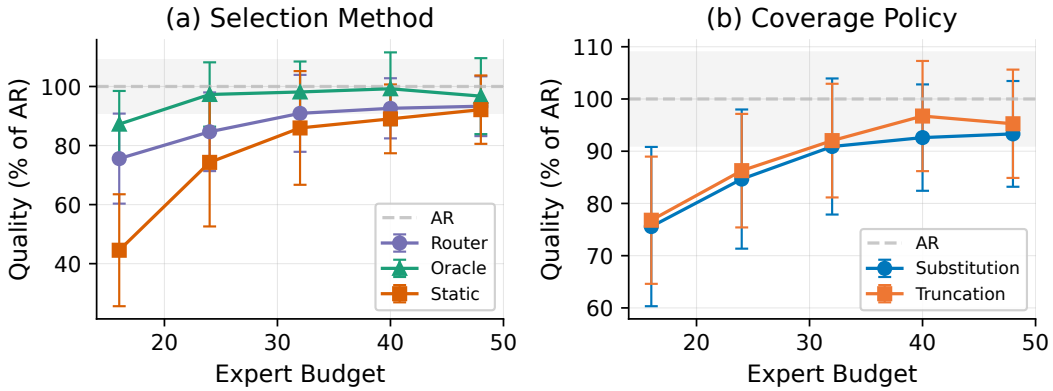


Figure 5: Design ablations on OLMoE-1B-7B at $T = 1$ (averaged over 5 seeds), normalized to autoregressive (AR) baseline. (a) Selection methods: Static ranking fails at low budgets while Router-based ranking tracks the output-aware Oracle. (b) Coverage policies: Truncation and Substitution perform comparably across budgets.

CUDA-event profiling confirms that the MoE block is the largest component of an uncapped EAGLE iteration at batch size one. It accounts for 48–52% of iteration time on OLMoE and 54–58% on Qwen3. Draft-tree construction accounts for 25–27% and 10–12%, respectively, while target-model attention accounts for 18–20% and 24–27%. The remaining time is spent in normalization, residual operations, the output projection, and framework overhead. Budgeting reduces only expert-weight traffic within the MoE block and adds 2–3% selection overhead, so end-to-end speedup is necessarily smaller than the ratio between uncapped and budgeted expert counts.

4.4 Ablation Studies

We evaluate the impact of the expert selection strategy on generation quality across varying budgets. Figure 5a compares three ranking methods using the Substitution coverage policy: Router-based (aggregating routing probabilities across the draft tree), Static (fixed ordering from calibration data), and Oracle (greedy selection using target model outputs). Static ranking fails at low budgets ($B \leq 16$), achieving near-zero accuracy on GSM8K and MATH500 because a fixed expert set cannot adapt to input variability. Router-based ranking remains robust across the budget range, tracking the output-aware Oracle even at tight budgets.

The Oracle method achieves near-AR quality even at $B = 24$ (37.5% of experts), suggesting room for improved selection strategies. While the Oracle is impractical for acceleration (it requires executing the full model to obtain routing decisions), better selection methods could close the remaining gap between Router and Oracle. Router-based ranking offers the best practical tradeoff: it incurs no additional model execution cost while outperforming Static ranking by a wide margin at low budgets, and converging to within 5–10% of AR at moderate budgets ($B \geq 32$). We use Router-based ranking as the default.

Figure 5b examines how to handle tokens whose preferred experts fall outside the selected budget, using Router-based selection. Truncation zeros out missing experts while preserving natural routing weights; Substitution replaces missing experts with the highest-scoring available alternatives within the shortlist. Both policies perform comparably across all benchmarks and expert budgets. We select Substitution as the default to maintain a constant number of active experts (k) per token.

4.5 Reconstruction Error Analysis

Router-based ranking tracks but does not match Oracle across the budget range. To quantify this gap, we measure reconstruction error: the normalized squared distance between the budgeted MoE output and the full output. Oracle achieves the same reconstruction error as Router while using 25% fewer experts (Section C). The source of this gap is that Oracle has access to the actual expert outputs during selection, while Router relies only on routing probabilities. The 25% gap indicates that independent

router scores leave room for improved selection. Methods that incorporate joint routing patterns or information about expert outputs could achieve the same reconstruction error at tighter budgets. The analysis in Section E shows that routing decisions have strong pairwise structure: some expert pairs activate together 10–33$\times$ more often than expected under uniform routing.

5 Related Work

Speculative Decoding. Speculative decoding accelerates autoregressive generation by drafting candidate tokens with a lightweight model and verifying them in parallel with the target model (Leviathan et al., 2023; Chen et al., 2023). Subsequent work improved draft quality through learned drafters: EAGLE (Li et al., 2024b) predicts future hidden states autoregressively, EAGLE-2 (Li et al., 2024c) introduces dynamic draft trees based on confidence, and EAGLE-3 (Li et al., 2025) scales performance through training-time test. Medusa (Cai et al., 2024) adds multiple decoding heads for parallel token prediction. These methods assume verification cost is constant, which holds for dense models but breaks down for MoEs.

MoE Architecture and Efficiency. The sparse MoE architecture routes each token to a subset of expert networks, reducing compute while maintaining model capacity (Shazeer et al., 2017; Fedus et al., 2022; Jiang et al., 2024; Muennighoff et al., 2024). On the systems side, MegaBlocks (Gale et al., 2023) provides efficient GPU kernels through block-sparse operations, while Deja Vu (Liu et al., 2023) and PowerInfer (Song et al., 2024) train predictors to skip inactive neurons in dense models on a per-token basis. MoE-Spec operates in MoE verification, where the router already provides per-expert importance scores and the selection is across the full draft tree rather than per token. Orthogonally, compression methods such as expert pruning (Xie et al., 2024), merging (Li et al., 2024a), and quantization (Kim et al., 2023; Frantar & Alistarh, 2023) permanently reduce model size, while offloading systems like MoE-Infinity (Xue et al., 2024) cache experts between CPU and GPU. MoE-Spec operates at inference time and composes with these approaches.

Speculative Decoding for MoE. Existing work on speculative decoding for MoE models does not reduce the cost of a single verification pass. Huang et al. (2026) study MoE speculative decoding under batched serving, where cross-request expert reuse changes verification cost. We study the complementary single-request regime, where expert-weight traffic remains a primary bottleneck. Saxena et al. (2025) adapt speculation length dynamically, controlling *when* to speculate, but evaluate only chain-structured speculation with n-gram drafting, which activates far fewer unique experts per step than tree-structured methods with trained drafters. With tree-structured decoding, the expert union grows rapidly with tree size, and per-step verification cost becomes the dominant overhead. To our knowledge, MoE-Spec is the first method to target this bottleneck directly by reducing verification cost by loading fewer experts.

In memory-constrained settings where experts are offloaded to CPU, SP-MoE (Chen et al., 2025) prefetches experts via cross-model prediction during drafting, SpecMoEOff (Wang et al., 2025b) uses speculative decoding to amortize CPU-GPU transfer costs, and MoE-SpeQ (Wang et al., 2025a) combines a quantized draft model with lookahead-aware expert caching. These methods target PCIe bottlenecks; MoE-Spec targets HBM bandwidth when experts reside on GPU.

6 Conclusion

MoE-Spec shows that verification cost in MoE speculative decoding can be reduced by loading fewer experts, at a controlled cost to quality. On three architectures, this gives 15–45% throughput gains over EAGLE at aggressive budgets retaining $\geq$90% of autoregressive quality, with the budget B as the only tuning parameter. Our Oracle analysis indicates that router-based scoring is a reasonable but imperfect proxy for expert importance: an oracle achieves the same reconstruction error using 25% fewer experts. Better scoring functions could close this gap without adding significant overhead. Our experiments focus on single-request inference, where expert saturation does not occur naturally. The problem MoE-Spec addresses is likely to grow. As MoE architectures trend toward more experts with sparser routing (e.g., 256 experts with $k = 8$ in DeepSeek-V3), the expert union during verification will grow faster with tree size, but routing concentration will also increase, so budgeting should yield even larger gains.

AI USE STATEMENT

We used generative AI tools to assist with implementing, debugging, and running experiments. Additionally, AI was used to edit the manuscript for clarity and presentation. The authors reviewed and verified all AI-assisted code, experimental outputs, and manuscript edits. Generative AI was not used to generate datasets or interpret experimental results. The authors take full responsibility for the content of this work.

ETHICS STATEMENT

This work uses publicly available pretrained models and standard benchmarks and does not involve human participants or private data. MoE-Spec changes inference efficiency rather than the capabilities or training data of the underlying models, and therefore inherits their existing risks and limitations.

REPRODUCIBILITY STATEMENT

Section 3 describes the method, and Section 4.1 specifies the models, benchmarks, baselines, and evaluation protocol. The appendix provides the algorithm, hyperparameters, hardware configuration, and additional analyses. All evaluated models and benchmarks are publicly available. We will release the implementation and instructions for reproducing the experiments upon acceptance.

REFERENCES

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. arXiv preprint arXiv:2401.10774, 2024.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. arXiv preprint arXiv:2302.01318, 2023.

Liangkun Chen, Zijian Wen, Tian Wu, Xiaoxi Zhang, and Chuan Wu. Sp-moe: Speculative decoding and prefetching for accelerating moe-based model inference, 2025. URL https://arxiv.org/abs/2510.10302.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Elias Frantar and Dan Alistarh. Qmoe: Practical sub-1-bit compression of trillion-parameter models. *arXiv preprint arXiv:2310.16795*, 2023.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. Megablocks: Efficient sparse training with mixture-of-experts. *Proceedings of Machine Learning and Systems*, 5:288–304, 2023.

Raja Gond, Aditya K Kamath, Ramachandran Ramjee, and Ashish Panwar. Llm-42: Enabling determinism in llm inference with verified speculation. *arXiv preprint arXiv:2601.17768*, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Zongle Huang, Lei Zhu, Zongyuan Zhan, Ting Hu, Weikai Mao, Xianzhi Yu, Yongpan Liu, and Tianyu Zhang. Moesd: Unveil speculative decoding's potential for accelerating sparse moe. *Advances in Neural Information Processing Systems*, 38:125276–125311, 2026.

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.

Young Jin Kim, Raffy Fahim, and Hany Hassan Awadalla. Mixture of quantized experts (moqe): Complementary effect of low-bit quantization and robustness. *arXiv preprint arXiv:2310.02410*, 2023.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 23–29 Jul 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 *International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=eFWG9Cy3WK.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. arXiv preprint arXiv:2401.15077, 2024b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 7421–7432, 2024c.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pp. 22137–22176. PMLR, 2023.

Meta AI. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation, 2025. URL https://ai.meta.com/blog/llama-4-multimodal-intelligence/.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Evan Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. Olmoe: Open mixture-of-experts language models. In *The Thirteenth International Conference on Learning Representations*, 2024.

Anish Saxena, Po-An Tsai, Hritvik Taneja, Aamer Jaleel, and Moinuddin Qureshi. Utility-driven speculative decoding for mixture-of-experts, 2025. URL https://arxiv.org/abs/2506.20675.

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.

Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. Powerinfer: Fast large language model serving with a consumer-grade gpu. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, pp. 590–606, 2024.

Wenfeng Wang, Jiacheng Liu, Xiaofeng Hou, Xinfeng Xia, Peng Tang, Mingxuan Zhang, Chao Li, and Minyi Guo. Moe-speq: Speculative quantized decoding with proactive expert prefetching and offloading for mixture-of-experts. *arXiv preprint arXiv:2511.14102*, 2025a.

Zhibin Wang, Zhonghui Zhang, Yuhang Zhou, Zibo Wang, Mo Zhou, Peng Jiang, Weilin Cai, Chengying Huan, Rong Gu, Sheng Zhong, and Chen Tian. Accelerating mixture-of-experts inference by hiding offloading latency with speculative decoding, 2025b. URL https://arxiv.org/abs/2508.21706.

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

Leyang Xue, Yao Fu, Zhan Lu, Luo Mai, and Mahesh Marina. Moe-infinity: Offloading-efficient moe model serving. *arXiv preprint arXiv:2401.14361*, 2024.

Table 2: General experimental settings.

Parameter	Value
Draft tree size (main)	60
Draft tree size (sweep)	3–511
Decoding temperature	0.0
Max generation length	1024
Shortlist method	Router
Token policy	Substitution
GPU type	NVIDIA A100
GPU memory	80GB

Table 3: Model-specific configurations.

Model	Experts	Budget (B)	Draft / GPUs
OLMoE-1B-7B	64 (k=8)	8–56; 64 (full)	EAGLE-3 / 1
Qwen3-30B-A3B	128 (k=8)	16–64; 128 (full)	EAGLE-3 / 1
Mixtral-8x7B	8 (k=2)	2–7; 8 (full)	EAGLE / 2

A Hyperparameters

Table 2 summarizes the general experimental settings. Table 3 provides model-specific configurations.

Draft Tree Size. We use a default draft tree size of 60 tokens, following the EAGLE-3 configuration. The tree size sweep (3–511) in Figure 3 shows that MoE-Spec's speedup grows with tree size, since larger trees activate more unique experts under standard verification. The choice of 60 balances acceptance rate against verification overhead for the baseline.

Expert Budget Selection. Budget ranges are model-specific based on the expert configuration. For OLMoE, we sweep $B \in \{8, 16, 24, 32, 40, 48, 56\}$ and include the full-budget setting $B = 64$. For Qwen3, we sweep $B \in \{16, 24, 32, 48, 56, 64\}$ and include $B = 128$. For Mixtral, we sweep $B \in \{2, 3, 4, 5, 6, 7\}$ and include $B = 8$. Main results (Table 1) report two operating points per configuration: iso-quality and throughput-maximizing.

Reproducibility. Ablation experiments use T=1 with mean and standard deviation over 5 random seeds. All models use FP16 precision. We use PyTorch 2.1 with CUDA 12.1. Model weights are loaded from HuggingFace Hub; draft model adapters are from the official EAGLE repository.

B MoE-Spec Algorithm

Algorithm 1 provides pseudocode for verification-time expert budgeting. The algorithm has three stages: (1) compute routing probabilities for all tokens in the draft tree, (2) aggregate importance scores per expert by summing routing probabilities, and (3) apply the token policy (truncation or substitution) using only the top-B experts. The aggregation step requires no additional model execution beyond what the router already computes.

C Reconstruction Error Analysis

Router-based ranking tracks but does not match Oracle across the budget range. To understand this gap, we measure reconstruction error: the normalized squared distance between the budgeted MoE output O_S and the full output O^* from Equation (2). We compute this at each MoE layer during Oracle selection, where all expert outputs are available, and average across layers and datasets.

Oracle achieves the same reconstruction error as Router while using 25% fewer experts (Figure 6). At $B = 24$, Oracle matches Router at $B = 32$ (both 3.2% error); at $B = 16$, Oracle nearly matches

Algorithm 1 MoE-Spec Expert Budgeting

Input: Hidden states $\{\mathbf{h}_1, \ldots, \mathbf{h}_M\}$, budget B, policy
Output: MoE output for each token

// Compute routing probabilities
for $t = 1, \ldots, M$ **do**
  $g_i(\mathbf{h}_t) \leftarrow \text{softmax}(\mathbf{W}_{\text{gate}}\mathbf{h}_t)$ for all i
// Aggregate importance scores
for $i = 1, \ldots, N$ **do**
  $s_i \leftarrow \sum_{t=1}^{M} g_i(\mathbf{h}_t)$
// Select shortlist
$\mathcal{S} \leftarrow$ top-B experts by s_i
// Apply token policy
for $t = 1, \ldots, M$ **do**
  if policy = truncation **then**
    $\mathcal{T}_k(\mathbf{h}_t) \leftarrow$ top-k experts by $g_i(\mathbf{h}_t)$
    $\text{out}_t \leftarrow \sum_{i \in \mathcal{T}_k(\mathbf{h}_t) \cap \mathcal{S}} \bar{g}_i(\mathbf{h}_t) \cdot E_i(\mathbf{h}_t)$
  else *// substitution*
    $\mathcal{T}_k^{\mathcal{S}}(\mathbf{h}_t) \leftarrow$ top-k in $\mathcal{S}$ by $g_i(\mathbf{h}_t)$
    $\text{out}_t \leftarrow \sum_{i \in \mathcal{T}_k^{\mathcal{S}}(\mathbf{h}_t)} \bar{g}_i^{\mathcal{S}}(\mathbf{h}_t) \cdot E_i(\mathbf{h}_t)$
return $\{\text{out}_1, \ldots, \text{out}_M\}$

Router at $B = 24$ (8.0% vs. 7.7%). Static selection incurs 20–35% error across the evaluated budget range.

Router scores each expert independently; Oracle selects greedily to minimize reconstruction error and can account for interactions between experts. Co-activation patterns across layers (Section E) show that expert routing is far from uniform, which independent scoring does not exploit.

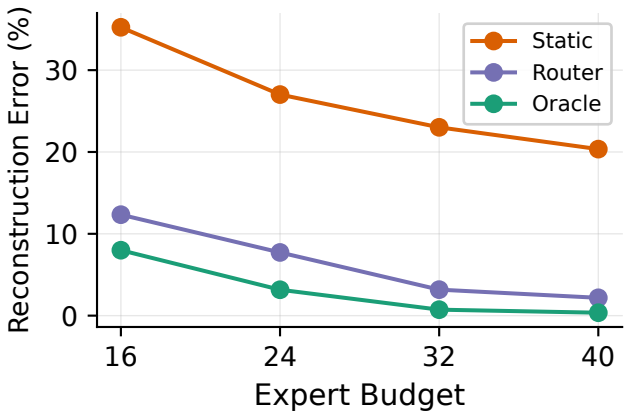


Figure 6: Reconstruction error by selection method on OLMoE-1B-7B, averaged across all MoE layers and 5 benchmarks. Oracle matches Router error using 25% fewer experts.

D Expert Coverage Across Datasets

We profile expert routing on each dataset by running the target model on calibration samples and recording routing probabilities for draft trees of size 60. Figure 7 shows cumulative routing probability captured by the top-B experts.

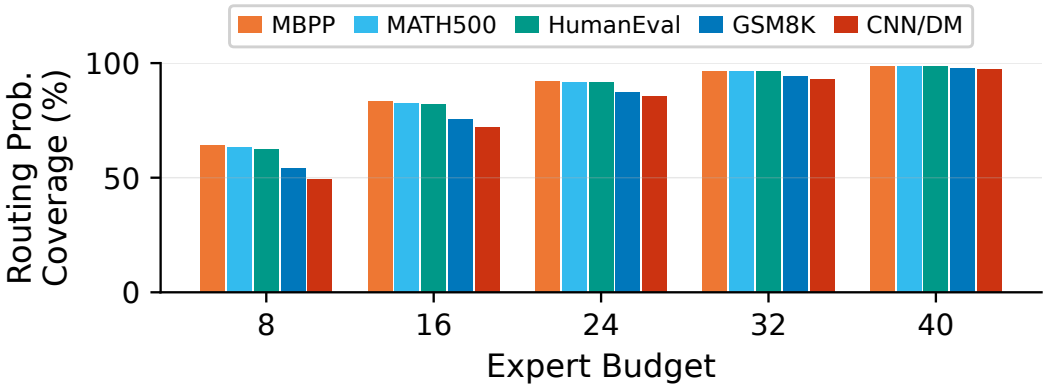


Figure 7: Routing probability coverage by expert budget across datasets on OLMoE-1B-7B with tree size 60. Datasets are ordered by coverage (best to worst): MBPP reaches 95% coverage with 8 experts, while CNN/DM requires 24 experts to reach 90%.

Routing concentration varies across tasks. Code generation (MBPP, HumanEval) concentrates on few experts: the top 8 capture over 90% of routing probability. Summarization (CNN/DM) spreads routing more broadly, requiring 24 experts to reach 90%.

Static ranking uses a fixed expert ordering from calibration data. When routing is concentrated, the same experts dominate across inputs, so a fixed ordering works well. When routing is dispersed, different inputs select different experts, and a fixed ordering misses task-specific patterns. Router-based selection sidesteps this by aggregating probabilities per draft tree.

E Expert Co-activation Analysis

To characterize routing structure that is not captured by independent expert scores, we analyze expert co-activation patterns during inference. Co-activation measures how often pairs of experts are selected together for the same token. Nonuniform co-activation indicates statistical dependence between routing decisions, but does not by itself imply that the corresponding expert outputs are redundant.

Methodology. We run inference on 50 samples from each benchmark (GSM8K, HumanEval, MATH500, MBPP, CNN/DM) using OLMoE-1B-7B with EAGLE-3 speculative decoding. For each token processed during verification, we record which experts appear in its top-k selection. For each MoE layer, we build a 64×64 co-activation matrix where entry (i, j) counts how many times experts i and j were both selected for the same token.

Concentration Metric. Let R denote the number of token-routing observations in a layer, and let C_{ij} denote the number of observations in which experts i and j are both selected. Under uniform selection of $k = 8$ experts from $N = 64$, a specific pair is selected with probability $\frac{k(k-1)}{N(N-1)} \approx 0.014$. Its expected co-activation count is therefore $R\frac{k(k-1)}{N(N-1)}$. We define the concentration ratio as:

$$\text{Concentration} = \frac{\max_{i<j} C_{ij}}{R\frac{k(k-1)}{N(N-1)}}. \tag{9}$$

Results. Figure 8 shows co-activation heatmaps for all 16 MoE layers. Experts are reordered using hierarchical clustering. Across layers, the most frequently co-activated expert pair occurs 10× to 33× more often than expected under uniform routing (mean 23×). Router scoring uses only marginal probability sums and therefore does not represent these pairwise routing patterns. The output-aware Oracle instead selects experts using their contributions to reconstruction error. The gap between the two methods suggests that selection signals beyond independent router scores may support tighter expert budgets.

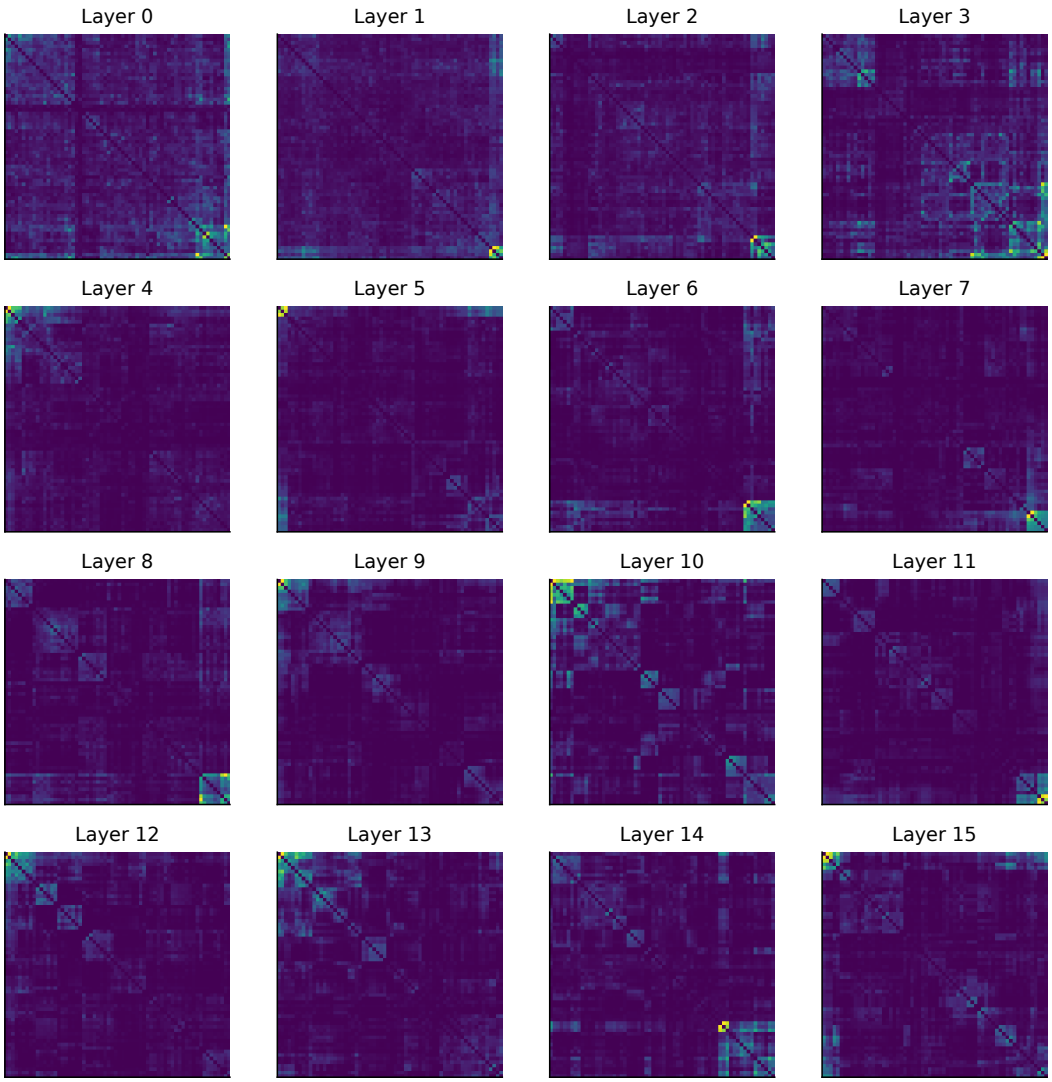


Figure 8: Expert co-activation heatmaps for all 16 MoE layers in OLMoE-1B-7B. Each cell shows normalized pairwise co-activation frequency. Experts are reordered by hierarchical clustering. Each panel exhibits nonuniform pairwise structure; the maximum off-diagonal concentration ratio ranges from 10× to 33× across layers.