ECCoMoE: Elastic Compute Conditioned Mixture of Experts

Anonymous authors
Paper under double-blind review

Abstract

Sparse mixture-of-experts (MoE) models run the same number of experts for every token, whether the token is easy or hard to predict. Learned methods that vary this number train it into the backbone or train one module per operating point; training-free rules learn no per-token decisions. We present ECCoMoE (Elastic Compute Conditioned Mixture of Experts), which adds a small allocator to each MoE layer of a frozen backbone. For each token, it decides how many of the router's top experts to run, but never which ones or with what weights. The compute budget is an input, so one checkpoint serves both a continuous budget dial and a free mode in which the allocator chooses on its own. It runs inside vLLM's routing stage as a fused Triton kernel. On three MoE backbones and four reasoning and coding benchmarks, at the same average number of experts per token, the dial gains most over interpolated fixed top-k at tight budgets, up to 20 points near two experts per token. On Qwen3.6-35B-A3B at t=2 ($\mathbb{E}[k]\approx$2.3), the dial beats fixed k at matched average expert count on all four benchmarks and shows no detected difference from top-8 on three, including MATH-500 (92.3% versus 91.3%). We also find that renormalizing the kept experts' weights, as top-k routers usually do, collapses fixed k performance on two of the three backbones, and fewer experts per token do not guarantee a cheaper response, because the allocator changes how much the model writes.

1 Introduction

Mixture-of-experts (MoE) layers let language models grow in parameters without growing in compute (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2021). A router sends each token to a few experts, and only those run. The number of experts per token k is fixed during training. Every token at every layer gets the same k, whether it is a comma or the next step of a physics problem. The often cheaper option at deployment is a smaller fixed k for every token. The performance cost depends on the model. Going from eight experts to two costs Gemma-4 (Team et al., 2026) most of its MATH-500 accuracy (Hendrycks et al., 2021; Lightman et al., 2024) but barely moves Qwen3.6 (Qwen Team, 2026) (Figure 2). Learned alternatives pretrain or fine-tune the backbone or router, or train a small module on a frozen backbone once per operating point; training-free rules can vary the count from one model but learn no per-token decisions.

ECCoMoE takes a different route (Figure 1). At each MoE layer of a frozen model, a two-layer allocator MLP reads the MoE layer's input and chooses how many of the router's top experts the token keeps. The allocator decides how many top experts to run, the router is remains responsible for deciding which ones and how much each weighs, so keeping all of them recovers the pretrained layer exactly. The budget is an allocator input, not a training setting, so one checkpoint covers a continuous range of budgets, each chosen when a server loads the checkpoint. Dropping the budget term on a quarter of training steps also teaches the same checkpoint to allocate on its own. Instead of using reinforcement learning for training, we relax the hard choice of how many experts to keep into a soft mask over the router's ranking, so gradients reach the allocator directly through the frozen backbone. At serving time the choice is a single argmax fused into vLLM's routing stage (Kwon et al., 2023). Dropped experts become zero-weight slots, so routing tensors keep their shape and CUDA graphs still apply.

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.

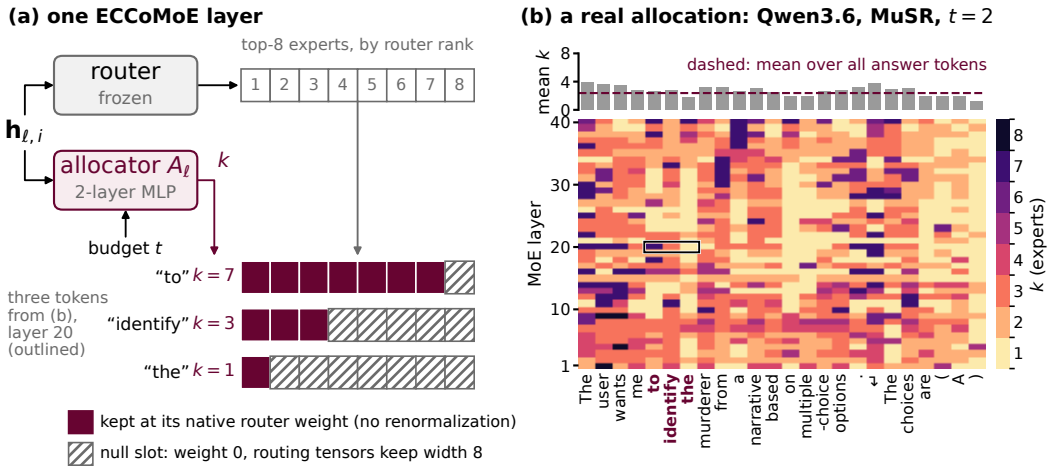

Figure 1: (a) ECCoMoE adds a small allocator to each MoE layer of a frozen model. It reads the MoE layer's input and a budget t, and picks how many of the router's top-8 experts each token uses; the rest are skipped at zero weight. (b) Its choices on Qwen3.6 at t=2 for the first 24 answer tokens of a MuSR response (Sprague et al., 2024), chosen by rule; this window spends more experts than most. Outlined: the three tokens drawn in (a). Table 4 tests whether these per-token choices matter.

We evaluate ECCoMoE on three backbones, covering softmax and sigmoid routers, and on four reasoning and coding benchmarks. We compare each operating point with an interpolated fixed-k reference at the same average number of experts per token, which we call matched average expert count, and measure energy per response. Our main findings are:

- At matched average expert count the dial gains up to 20 points at tight budgets, where fixed k degrades fastest, and at generous ones mostly no difference is detected under our uncertainty estimate (Table 1). Free mode, which needs no budget, is weaker: it falls below fixed k on three of the twelve pairs. Our frozen-backbone adaptations of BEAM and Ada-K, trained on the same traces, do not beat the dial overall (Section 4.4).
- The allocator's per-token decisions matter where experts are scarce. On MuSR at a budget of two, replacing them with random counts or another token's profile costs accuracy on every backbone (Table 4).
- Top-k routers usually renormalize the kept experts' weights. At two experts per token this collapses two of the three backbones, so the choice of baseline alone can decide a comparison (Section 4.2).
- Fewer experts per token do not guarantee a cheaper response. The allocator changes how much the model writes, and at a budget of two it costs more energy per response than the unmodified model on six of the twelve pairs, three of them beyond noise, all on Gemma-4 (Section 4.5).

2 Related work

Routing in sparse MoE. Token-choice top-k routing with a k fixed in training is the default in MoE language models (Jiang et al., 2024; DeepSeek-AI, 2024), including all three backbones we study. Expert-choice routing (Zhou et al., 2022) instead lets each expert pick its tokens, so the number of experts a token gets varies. But the routing is fixed at training time, and a token's compute depends on the rest of the batch.

Adaptive expert counts. One line of work builds a variable k into the model: top-p or threshold routing (Huang et al., 2024; Yang et al., 2024), null or zero-computation experts (Zeng et al., 2024; Jin et al., 2025), auto-tuned expert counts (Guo et al., 2025), ReLU routing (Wang et al., 2025), and per-token counts learned with the model (Aghdam et al., 2024). These methods pretrain or

fine-tune the backbone. BEAM also fine-tunes the model while it learns binary masks over the router's top-k (Wu et al., 2026). A second line keeps the pretrained model frozen. Ada-K trains per-layer allocators with reinforcement learning (Zhao et al., 2025); as published, it and BEAM fix one operating point per training run. NAEE skips experts by a gate-weight rule (Lu et al., 2024), and Alloc-MoE splits a global budget, given as an input, across layers and tokens without training (Liu et al., 2026). ECCoMoE also keeps the backbone frozen and takes the budget as an input, but learns its per-token allocator, so one single checkpoint serves a continuous dial along with a free mode without any retraining.

Adaptive computation. Other methods vary depth or width: adaptive halting (Graves, 2016; Banino et al., 2021; Dehghani et al., 2018), early exit (Schuster et al., 2022), Mixture-of-Depths (Raposo et al., 2024) and elastic sub-networks (Cai et al., 2024). ECCoMoE varies only how many routed experts run.

3 METHODS

3.1 PROBLEM FORMULATION

Consider a pretrained MoE layer ℓ with E routed experts $e_1, \dots, e_E$. For token i with MoE input $\mathbf{h}_{\ell,i}$, the router gives every expert a ranking score, $\mathbf{s}_{\ell,i} \in \mathbb{R}^E$. For softmax routers these are its logits. For sigmoid routers they are the sigmoid of the logits plus a learned per-expert bias. Let $\mathrm{TopK}(\mathbf{s}, k)$ denote the indices of the k highest entries of $\mathbf{s}$, in descending order. The pretrained layer runs the experts in $\mathrm{TopK}(\mathbf{s}_{\ell,i}, k_{\max})$, where $k_{\max}$ is its native top-k. It weights each expert j by the gate $g_{\ell,i,j}$ from its own rule: a softmax over the logits, or the sigmoid scores. The selection bias affects only the ranking.

Conventional top-k routing uses all $k_{\max}$ experts for every token. ECCoMoE replaces this fixed count with a decision learned per token and per layer, $k_{\ell,i} = A_\ell(\mathbf{h}_{\ell,i}, t) \in \{1, \dots, k_{\max}\}$. Here A_ℓ is the ECCoMoE allocator and $t \in [1, k_{\max}]$ is an optional compute budget, which free mode omits. The layer keeps the first $k_{\ell,i}$ experts of the native ranking, drops the rest, and does not renormalize the surviving weights:

$$\mathbf{y}_{\ell,i} = \sum_{j \,\in\, \mathrm{TopK}(\mathbf{s}_{\ell,i},\, k_{\ell,i})} g_{\ell,i,j}\, e_j(\mathbf{h}_{\ell,i}). \tag{1}$$

The allocator changes only *how many* experts run, not *which* ones or their weights. Setting $k_{\ell,i} = k_{\max}$ therefore recovers the pretrained layer exactly.

3.2 ECCOMOE ALLOCATOR ARCHITECTURE

Each MoE layer gets its own allocator A_ℓ, a 2-layer MLP. It reads the MoE input $\mathbf{h}_{\ell,i}$ and scores every possible expert count $k = 1, \dots, k_{\max}$:

$$\mathbf{z}_{\ell,i} = \mathrm{SiLU}\big(W_\ell^{(1)} \mathbf{h}_{\ell,i} + \mathbf{b}_\ell^{(1)}\big), \qquad W_\ell^{(1)} \in \mathbb{R}^{E \times d}, \tag{2}$$

$$a_{\ell,i,k} = \big(W_\ell^{(2)} \mathbf{z}_{\ell,i} + \mathbf{b}_\ell^{(2)}\big)_k - \gamma_\ell\, |k - t|, \qquad W_\ell^{(2)} \in \mathbb{R}^{k_{\max} \times E}. \tag{3}$$

Here γ_ℓ is a learned scalar per layer. The last term of Eq. 3 penalizes counts far from the requested budget t. Free mode drops this term, so the allocator scores counts from the MoE input alone. Training needs a differentiable way to turn these logits into a count, and serving needs a cheap one (Section 3.3).

3.3 TRAINING THE ALLOCATOR

We train each backbone's allocators on its own chain-of-thought traces, about 20M tokens per backbone. The prompts are a 70/30 mix of WildChat (Zhao et al., 2024) and math and science problems (Li et al., 2023; Ling et al., 2017), and we keep only complete, untruncated answers (Appendix A). The backbone, experts and router stay frozen; only the allocators learn.

A hard count has no gradient. During training we therefore turn the logits into a distribution over counts, $\mathbf{p}_{\ell,i} = \mathrm{softmax}(\mathbf{a}_{\ell,i}/\tau)$, and keep every candidate expert softly, weighted by its chance of surviving:

$$\tilde{\mathbf{y}}_{\ell,i} = \sum_{m=1}^{k_{\max}} \pi_{\ell,i,m}\, g_{\ell,i,j_m}\, e_{j_m}(\mathbf{h}_{\ell,i}), \qquad \pi_{\ell,i,m} = \sum_{k \geq m} p_{\ell,i,k}, \tag{4}$$

where j_m is the expert at rank m of $\mathrm{TopK}(\mathbf{s}_{\ell,i}, k_{\max})$. Expert j_m runs whenever the count is at least m, so $\pi_{\ell,i,m}$ is its expected mask and $\tilde{\mathbf{y}}_{\ell,i}$ is the expected output of Eq. 1. Gradients reach the allocator through these masks, so training always runs all $k_{\max}$ candidates. We anneal τ from 1.0 to 0.3 so that $\mathbf{p}_{\ell,i}$ sharpens toward the hard choice used at serving time.

The loss is the language-model loss on response tokens, plus a budget penalty and a per-layer mutual-information bonus:

$$\mathcal{L} = \mathcal{L}_{\mathrm{LM}} + \lambda_s\,(\bar{k} - t)^2 - \frac{\beta}{L} \sum_{\ell=1}^{L} \Big[H(\bar{\mathbf{p}}_\ell) - \frac{1}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} H(\mathbf{p}_{\ell,i}) \Big]. \tag{5}$$

Here $\bar{k}$ is the expected count, averaged over layers and the token positions $\mathcal{T}$ of a microbatch; $\bar{\mathbf{p}}_\ell$ is layer ℓ's mean count distribution and H is entropy. λ_s ramps linearly from 0 to 0.5 over the first 100 steps, and β=0.2. The bonus favours counts that vary across tokens while each token's choice stays confident; it acts on counts, not on expert load. Each step samples one budget $t \sim \mathcal{U}(2, 7)$, after a short warm-up on the endpoints. With probability 0.25 a step drops both the budget term and the penalty, which teaches the same checkpoint free mode. Training runs for 1,220 steps of eight 2,048-token sequences (20M tokens) with AdamW (Loshchilov & Hutter, 2017) (Appendix A).

Inference. At inference the allocator picks $k_{\ell,i} = \arg\max_k a_{\ell,i,k}$, and the layer runs the first $k_{\ell,i}$ experts of its ranking (Eq. 1). For a served budget t, the budget term of Eq. 3 is constant for each count, so we fold it into $\mathbf{b}_\ell^{(2)}$ when the checkpoint loads; each server therefore runs one budget. The allocator is fused into vLLM's routing stage and costs about as much as the router: 0.8–2.1% of routed-expert FLOPs at k=8 and 3–8% at k=2. Dropped experts become zero-weight slots, so routing tensors keep their width of $k_{\max}$ and CUDA graphs still work. The served kernels match unfused references on captured tokens (one layer for softmax routing, all layers for Laguna-XS), up to tie-breaking between experts with bit-identical scores (Appendix B).

4 Results

4.1 Experimental setup

We test three MoE backbones: Qwen3.6-35B-A3B, Gemma-4-26B-A4B and Laguna-XS-2.1 (Abadji et al., 2026). Each routes every token to 8 experts. Together they cover two common router families: softmax (Qwen3.6 with 256 experts, Gemma-4 with 128) and sigmoid with a learned selection bias (Laguna-XS with 256). Easy benchmarks saturate at almost every k on these models, so we use four harder ones: MATH-500 (500 problems), GPQA-Diamond (GPQA-D, 198) (Rein et al., 2023), MBPP+ (Austin et al., 2021; Liu et al., 2023) (378, graded against the EvalPlus extended tests) and MuSR (756).

All accuracy runs use vLLM at batch 64, a 32,768-token output cap and three seeds. The backbone is frozen in training, so we use each model card's recommended sampling. Fixed k keeps the first k of the router's top-8 experts at their native weights, the same mask the allocator applies. Fixed k=8, which we call top-8, keeps all eight experts, as the unmodified model does. We sweep fixed k and the dial t from 2 to 8, and run free mode. We leave out k=t=1: fixed top-1 collapses to 0–9% accuracy on nine of the twelve model–benchmark pairs, with responses up to 12× longer than at top-8. Energy per response comes from the GPU's board-level energy counter (NVML), read over the request window only. Every run uses one H200 GPU, with CUDA 13.0, PyTorch 2.13 and vLLM 0.28.0. The dialled settings and free mode route through our fused Triton kernel (Tillet et al., 2019), and fixed k through our PyTorch routing. Every accuracy run records per-token expert counts (mean: $\mathbb{E}[k]$), so its speed and energy include that overhead; we therefore time decoding separately against stock vLLM with the recording off (Section 4.5).

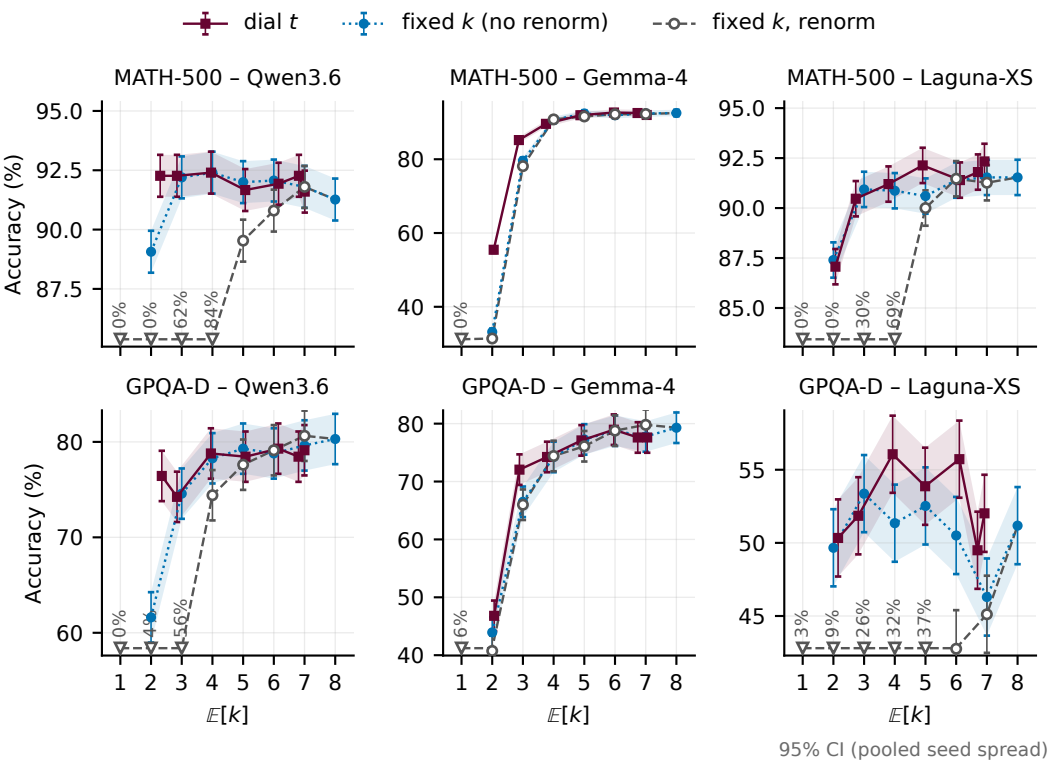

Figure 2: Accuracy against measured $\mathbb{E}[k]$ on MATH-500 and GPQA-D for the dial t=2, . . . , 8 (red), fixed k with native weights (blue) and fixed k with the kept weights renormalized (grey). Renormalized points below a panel's range sit on its floor, with their value printed. Fixed-k lines interpolate linearly between integer k. Bars and bands: 95% CI from the benchmark's pooled seed spread, assuming comparable seed variance across configurations and fixed $\mathbb{E}[k]$. Three seeds per point.

4.2 Renormalized versus native-weight fixed k

When fewer than $k_{\max}$ experts run, how the survivors are weighted matters on two of the three backbones. Standard top-k routers normalize the selected experts' weights to sum to one. Laguna-XS, like DeepSeek-V3, then multiplies them by a fixed scale. So running a model at a smaller k with its own router rescales the survivors to the full top-$k_{\max}$ mass. ECCoMoE instead keeps their native weights, so the routed output shrinks to the survivors' share of that mass. Chen & Yao (2026) compare the two on Qwen3.6-35B-A3B. With six experts, normalizing by the top-8 mass costs 1.1 MMLU points, against 2.6 for standard renormalization. With native weights, i.e. no renormalization, fixed k degrades far more gently (Figure 2, Table 15). On Qwen3.6 and Laguna-XS, renormalizing collapses the model at k=2: it scores at most 9% on every benchmark, against 50–89% with native weights. At k=4 it still costs Laguna-XS 9–36 points and Qwen3.6 2–8. On Gemma-4 there is no difference detected under our uncertainty estimate in 22 of the 24 comparisons at k=2–7, three seeds each. From here on we use native-weight fixed k, the stronger baseline on Qwen3.6 and Laguna-XS and not detectably weaker on Gemma-4.

4.3 Budget-conditioned dial and free mode

One checkpoint follows the budget it is given. From t=2 to t=7, the measured $\mathbb{E}[k]$ stays within 0.4 of t on every model–benchmark pair. At t=8, above the training range of 2 to 7, it stays near 7, and accuracy shows no detected difference from top-8 on any pair (Table 3). We compare with fixed k at matched average expert count: the same average number of routed experts per token over all MoE layers, prompt and generated tokens included. It does not match FLOPs, latency, energy, response length or the distinct experts a batch loads. Prompt tokens are up to 44% of the count

Table 1: Accuracy at matched average expert count ($\Delta_{\mathbb{E}}$, points): each dialled or free-mode setting minus the interpolated fixed-k reference, native-weight fixed k interpolated linearly between the bracketing integer k at the setting's measured $\mathbb{E}[k]$. $\pm$: 95% confidence interval from the benchmark's pooled seed spread, accounting for the interpolation; it assumes comparable seed variance across configurations and treats $\mathbb{E}[k]$ as fixed. **Bold**: the interval excludes zero. Three seeds each.

Model	Benchmark	t=2	t=3	t=4	t=5	t=6	t=7	t=8	free
Qwen3.6	MATH-500	**+2.3±1.1**	+0.6±1.2	0.0±1.2	−0.3±1.2	−0.1±1.2	+0.4±1.2	−0.2±1.2	0.0±1.1
	GPQA-D	**+10.3±3.3**	+1.7±3.5	+0.7±3.6	−0.8±3.6	+0.4±3.5	−1.0±3.4	−0.5±3.7	−1.0±3.3
	MuSR	**+5.3±1.5**	**+3.9±1.6**	**+1.7±1.6**	+1.0±1.6	+0.8±1.6	−0.2±1.6	+0.5±1.7	+1.3±1.5
	MBPP+	**+3.6±1.4**	+1.5±1.5	+0.1±1.6	−0.4±1.6	+0.7±1.5	−0.2±1.5	−0.6±1.6	**+1.7±1.4**
Gemma-4	MATH-500	**+20.1±1.2**	**+11.7±1.2**	**+1.7±1.1**	−0.2±1.2	+0.7±1.2	+0.3±1.1	−0.4±1.2	−0.9±1.2
	GPQA-D	+1.5±3.6	**+8.1±3.5**	+1.7±3.4	+0.1±3.6	+0.2±3.7	−0.4±3.4	−0.2±3.6	−0.3±3.6
	MuSR	**+3.1±1.7**	+1.1±1.6	**+2.8±1.6**	+0.9±1.6	+1.1±1.7	+0.3±1.5	+0.6±1.7	**−8.0±1.6**
	MBPP+	**+6.1±1.6**	−1.1±1.5	**−1.8±1.4**	−0.4±1.5	−0.4±1.5	−0.2±1.4	−0.4±1.6	**−5.3±1.6**
Laguna-XS	MATH-500	−0.6±1.2	+0.5±1.1	+0.3±1.1	**+1.5±1.2**	0.0±1.2	+0.3±1.1	+0.8±1.2	+0.1±1.1
	GPQA-D	+0.1±3.5	−0.8±3.4	**+4.6±3.6**	+1.4±3.7	**+5.7±3.6**	+1.9±3.3	**+5.4±3.6**	+1.5±3.2
	MuSR	**+2.8±1.7**	+1.1±1.5	+0.5±1.5	−1.4±1.6	−0.5±1.6	+1.4±1.5	+0.5±1.6	−0.1±1.5
	MBPP+	+1.2±1.5	−1.3±1.5	−1.0±1.5	−0.1±1.6	+1.0±1.5	+0.7±1.4	0.0±1.6	**−1.4±1.4**

on MuSR, but on teacher-forced traces the dial's answer-token $\mathbb{E}[k]$ at t=2 is within about 0.1 of the served value on MATH-500 and MuSR. At a fractional $\mathbb{E}[k]$ the reference interpolates linearly between the bracketing integer k and is not a measured policy (Figure 2, Table 1). The dial's gains concentrate at small budgets. Of 84 comparisons (nominal, not corrected for multiple comparisons) it wins 18, 15 of them at $t\leq4$, loses one and shows no detected difference in 65; at the 95% level, chance alone would give about two nominal wins and two losses. The gains are largest where fixed k falls fastest (Spearman ρ=0.75 at t=2), up to 20 points on Gemma-4 MATH-500, where fixed k collapses. Gains also appear where accuracy stays usable. On Qwen3.6 at t=2 the dial beats fixed k on all four benchmarks and shows no detected difference from top-8 on three (Table 3). At tight budgets they are smallest on Laguna-XS, whose fixed-k curve barely drops; Laguna-XS also holds the only three wins above t=4 (GPQA-D at t=6, 8, MATH-500 at t=5), although its fixed-k curve on GPQA-D is not monotone (53.4% at k=3, 46.3% at k=7), so the GPQA-D wins rest on a noisy reference. Free mode settles at 4.3–5.1 experts per token. It shows no detected difference from fixed k on eight of the twelve pairs, wins on one and falls below it on three, furthest on Gemma-4.

4.4 Comparison with learned-k baselines

We also train frozen-backbone adaptations of two learned-k methods on the same traces for about as many steps, and test them the same way. BEAM masks experts out of the router's top-k and keeps native weights; Ada-K learns a per-layer policy over k with PPO (Schulman et al., 2017) and renormalizes. As published, neither takes a budget at serving time, so each setting is its own training run. Neither beats the dial overall: of the 44 nominal comparisons inside the dial's range, a baseline wins twice, both on Gemma-4 MBPP+, the one pair where the dial itself falls below fixed k. Ada-K falls short almost everywhere, and BEAM at β=0.1 shows no detected difference from the dial on 11 of the 12 pairs. Ada-K renormalizes as published, which Section 4.2 shows is harmful at small k on Qwen3.6 and Laguna-XS, though it also falls short on most Gemma-4 comparisons. BEAM at β=0.1 serves 3.7–6.5 experts per token, where the dial gains little over fixed k, so the tight-budget gains are established against fixed k. We ran no training-free budgeted rule such as Alloc-MoE. Recipes and per-benchmark results are in Appendix E (Table 17).

4.5 Realized systems savings

With expert-count recording off, the dial at $t\leq4$ decodes faster than stock vLLM once requests are batched, though by less than its expert savings (Table 2). We time 1, 8, 32 and 64 concurrent requests, each generating 256 tokens from a MATH-500 prompt. At 64 requests, t=2 runs 71–75% fewer experts and serves 1.37, 1.20 and 1.36 times stock's rate on Qwen3.6, Gemma-4 and Laguna-XS. Free mode with the fused routing serves 1.03–1.14 times. A single request gains at most 14% (Gemma-4, t=2), and for a single request Laguna-XS's dial and free mode run 5–9% slower per token than stock. Our PyTorch routing at k=8 is 7–10% slower than stock with 64 requests, and up to 27% slower per token with one.

Table 2: Serving throughput and latency of ECCoMoE against the unmodified model on stock vLLM, count recording off. Throughput: output tokens per second at 64 concurrent requests, relative to stock. Latency: ms per token for one request. One run per setting.

	Throughput, 64 requests					Latency, 1 request (ms/token)				
Model	stock (tok/s)	t=8	t=4	t=2	free	stock	t=8	t=4	t=2	free
Qwen3.6	3743	1.04×	1.21×	1.37×	1.14×	4.4	4.4	4.3	4.3	4.4
Gemma-4	5084	0.99×	1.10×	1.20×	1.03×	4.7	4.6	4.4	4.0	4.5
Laguna-XS	3990	0.95×	1.15×	1.36×	1.08×	4.3	4.7	4.6	4.5	4.7

Table 3: Each budget and free mode against top-8 (fixed k=8): change in accuracy (points), change in energy per response (%) and throughput as the ratio of responses per second. Expert counts are not matched. Energy and throughput come from the accuracy runs, in which top-8 uses our PyTorch routing and the dial and free mode the fused kernel. Qwen3.6 at t=2 shows no detected accuracy difference from top-8 on three of the four benchmarks, and Laguna-XS at t=4 no loss on any (one gain), with up to 1.5× the throughput; Gemma-4 is the counterexample, losing accuracy on every benchmark at t=2 and t=4. **Bold**: beyond the benchmark's noise band for two three-seed means. Three seeds each.

			t=2			t=4			t=8			free		
Model	Benchmark	Top-8 Acc	ΔAcc	ΔJ/resp	Items/s	ΔAcc	ΔJ/resp	Items/s	ΔAcc	ΔJ/resp	Items/s	ΔAcc	ΔJ/resp	Items/s
Qwen3.6	MATH-500	91.3	+1.0	**−47%**	**1.50×**	+1.1	**−28%**	**1.22×**	+0.3	−4%	1.01×	+0.9	**−20%**	**1.14×**
	GPQA-D	80.3	**−3.9**	−12%	1.00×	−1.5	−7%	1.05×	−1.2	+15%	0.91×	−2.4	−5%	1.00×
	MuSR	65.2	−0.3	**−47%**	**1.51×**	+1.4	**−40%**	**1.41×**	+0.5	−10%	1.09×	+1.4	**−31%**	**1.34×**
	MBPP+	81.4	−1.1	+13%	**0.73×**	+0.3	**+46%**	**0.64×**	+1.1	+2%	0.96×	**+1.8**	**+48%**	**0.62×**
Gemma-4	MATH-500	92.5	**−37.0**	**+233%**	**0.31×**	**−2.9**	+5%	0.96×	−0.5	−2%	1.02×	−1.0	+3%	1.01×
	GPQA-D	79.3	**−32.5**	+1%	0.91×	**−5.1**	−7%	1.06×	−1.7	−2%	1.04×	−2.2	−5%	1.04×
	MuSR	69.4	**−8.6**	**+27%**	**0.74×**	**−3.2**	−1%	1.00×	+0.3	−5%	1.06×	**−9.8**	−5%	1.03×
	MBPP+	82.9	**−32.9**	**+131%**	**0.45×**	**−3.3**	−11%	1.08×	−0.2	−11%	1.10×	**−5.8**	−10%	1.08×
Laguna-XS	MATH-500	91.5	**−4.4**	−3%	0.94×	−0.3	**−12%**	**1.09×**	+0.8	−5%	1.04×	−0.7	**−10%**	1.06×
	GPQA-D	51.2	−0.9	**−25%**	**1.50×**	**+4.9**	**−21%**	**1.47×**	+0.8	−6%	**1.28×**	+2.3	−12%	1.08×
	MuSR	64.8	**−2.4**	**−12%**	1.08×	−1.0	**−26%**	**1.35×**	+0.4	−10%	**1.18×**	−1.3	**−16%**	**1.13×**
	MBPP+	81.7	**−4.3**	+6%	0.92×	−0.4	−12%	1.10×	−0.2	+4%	1.03×	−1.3	−6%	1.08×

Fewer experts per token do not always mean less energy per response. Energy comes from the accuracy runs: the dial and free mode use the fused kernel, fixed k (top-8 included) our PyTorch routing, and per-token expert-count recording is on in every run. At t=2, energy per token is 9–36% below top-8's on every pair. But answers are longer on ten of the twelve, and energy per response is higher on six, three of them beyond noise. The largest accuracy gain, Gemma-4 MATH-500 at t=2, costs 3.3× top-8's energy per response. The clearest saving is on Qwen3.6 MATH-500 (Table 3). There the dial at t=2 scores 92.3% versus 91.3% for top-8, with 16% shorter answers and 47% less energy per response. Fixed k=3 saves 38% on this nearly flat curve; it is the one comparison on the same routing path, so part of the dial's advantage may come from the faster fused routing rather than from allocation (Table 2). On Qwen3.6 MBPP+, the dial settings up to t=7 write 34–88% longer answers than top-8 with no detected difference in accuracy.

Against the interpolated fixed-k reference at matched average expert count (Table 16), t=2 uses less energy per response on 10 of the 12 pairs, six of them beyond noise, by up to 42% (Laguna-XS MuSR). Fewer of its answers hit the token cap on 11 of the 12 pairs (none do on Qwen3.6 MuSR either way; 17% against 34% on Gemma-4 MATH-500). From t=5 up, the dial uses more energy than fixed k on 22 of 48 comparisons, seven of them beyond noise and all on Qwen3.6, and free mode uses more on 9 of 12, one beyond noise.

5 Discussion

How the allocator spends its experts. A simple way to save experts would be to learn which layers matter and give every token the same count at each layer. Ada-K ends up close to that: the layer alone explains about 90% of the variation in its k (Figure 3). ECCoMoE does not. It has the weakest depth trend of the three methods on every backbone and the widest spread across tokens within a layer, and at t=2 the layer explains only 6–10% of its k on MATH-500 and MuSR answer tokens. To see what the rest is, we split the variation in each trace into a per-layer part, a per-token

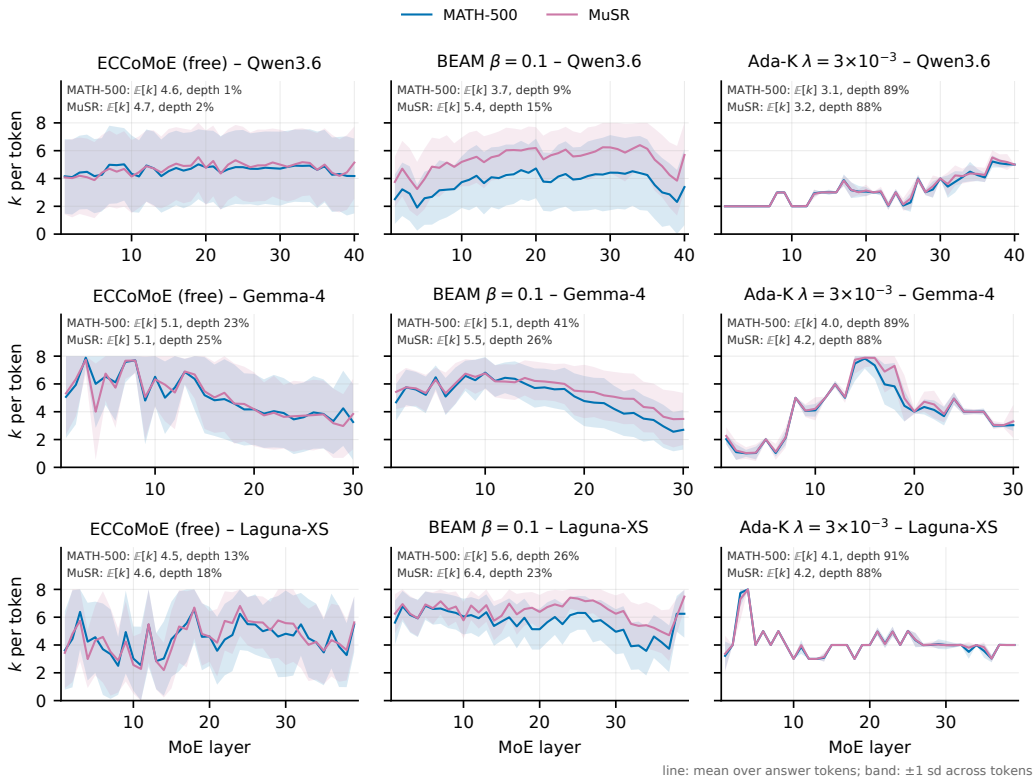

Figure 3: Expert count per MoE layer for ECCoMoE in free mode, BEAM (β=0.1) and Ada-K (λ=3×10^{-3}), each on teacher-forced traces of its own answers. Depth: share of the variation in k explained by the layer alone (100% is a pure depth schedule).

part (a level a token keeps through the whole stack) and a remainder. Most of it is remainder: a token that gets many experts at one layer is not especially likely to get many at the next, so the allocator seems to decide layer by layer rather than give each token a budget up front. The per-token part is small but real, several times what the same counts give when shuffled among tokens. BEAM sits in between, with a depth trend and a stronger per-token level than ECCoMoE.

Do the per-token decisions matter? Structure alone does not show that these choices help, so we swap them out at serving time while keeping the budget (Table 4). Each replacement removes one thing. E draws each token's count at random from the allocator's own per-layer distribution, so it keeps the depth profile but ignores the token. S gives every token the allocator's rounded per-layer mean, a pure depth schedule. D hands each token another token's full recorded profile, which keeps the depth profile and the way counts hang together across layers, but on the wrong token. At a tight budget all three hurt on every model; on MuSR at $t = 2$ they cost 2 to 14 points. D (one run each) costs as much as a random draw, or more on Qwen3.6, so a coherent profile helps only on the token it came from. The effect fades as the budget grows. By t=4 only Qwen3.6 still loses clearly under E and S, and in free mode a depth schedule does as well as the allocator, or better, on all but one setting.

Where the gain over fixed k comes from. Fixed k differs from the allocator in two ways: every layer gets the same count, and so does every token. S keeps the allocator's per-layer counts but gives every token the same count, so it removes only the first difference; comparing fixed k, S and the allocator splits the dial's gain (Table 1) into a depth part and a per-token part. The split is rough, because S's rounding spends a few percent fewer experts at t=2, but the pattern on Gemma-4 is clear. At t=2 the per-token part is larger than the whole gain, and S on its own falls below fixed k. At t=4 we detect no difference between S and the allocator, so the learned depth profile can carry the gain there. On Qwen3.6, per-token matching supplies part of the gain at t=2 and all of it at t=4.

Table 4: Replacing the allocator's per-token choices (A) at serving time, at about the same $\mathbb{E}[k]$. E: a random k per token and layer from A's per-layer distribution. S: A's rounded per-layer mean for every token (a depth schedule). D: another token's recorded profile. EH, SH: E and S fitted on GPQA-D. Column A: accuracy ($\mathbb{E}[k]$) [runs]. Other columns: change from A in points [runs], plus the change in $\mathbb{E}[k]$ when above 2%. **Bold**: the 95% interval (pooled seed spread) excludes zero.

Model	Setting	A	E	S	D	EH	SH
Qwen3.6	MuSR t=2	64.9 (2.33) [3]	**−7.5** [3]	**−2.1** [3] −7%	**−13.9** [1]	**−4.7** [1]	−0.2 [1]
	MuSR t=4	66.6 (3.90) [3]	**−2.9** [3]	**−2.3** [3]	**−3.7** [1]		
	MuSR free	65.2 (4.68) [1]	+0.8 [1]	+0.5 [1]			
	GPQA-D free	77.9 (4.64) [3]	−1.0 [3]	+0.7 [3]			
Gemma-4	MuSR t=2	60.8 (2.05) [3]	**−8.6** [3]	**−5.2** [3] −4%	**−6.9** [1]	**−7.7** [1]	**−4.2** [1] −4%
	MuSR t=4	66.2 (3.77) [3]	0.0 [3]	+0.4 [3]	**−3.0** [1]		
	MuSR free	59.2 (5.14) [2]	**+7.6** [2]	**+9.5** [2]			
	GPQA-D t=4	74.2 (3.79) [3]	−3.5 [1]	−4.5 [1]			
	GPQA-D free	77.1 (5.08) [3]	**−6.6** [3]	**−4.0** [3]			
Laguna-XS	MuSR t=2	62.4 (2.07) [3]	**−2.4** [3]	**−2.2** [3]	**−4.4** [1]	−0.9 [1] +4%	+0.7 [1] +5%
	MuSR t=4	63.4 (3.73) [4]	−0.6 [3]	+0.2 [3]	−0.3 [1]		
	MuSR free	62.4 (4.32) [5]	−1.3 [5]	**+1.9** [2]			
	GPQA-D free	53.5 (4.58) [3]	+1.5 [3]	−1.7 [3]			

The advantage seems to come from both sources, in proportions that vary with model and budget; the causal evidence for the per-token part rests on E and D, which keep the depth profile intact.

When elastic k does not help. If fixed k barely suffers from dropping experts, there is little for the allocator to recover. Laguna-XS MATH-500, fixed k loses only about 4 points from k=8 to k=2, and the dial beats it at a single budget. Free mode can also fall short, most visibly on Gemma-4 MuSR, where it trails fixed k by 8 points. The problem there is not how many experts it chose, since its own depth schedule at almost the same $\mathbb{E}[k]$ recovers the whole gap (Table 4). Without a budget to steer it, the allocator puts its experts in the wrong places.

Limitations. Energy and speed from the accuracy runs compare routing paths as well as allocations (our fused kernel for the dial and free mode, our slower PyTorch routing for fixed k), and our stock-vLLM timing is one run per setting on short generations. Each allocator was trained once, so our intervals cover sampling seeds but not training variance, and our comparisons are nominal. Our BEAM and Ada-K runs are frozen-backbone adaptations, not the published methods. We did not ablate the mutual-information bonus, the budget dropout or the soft relaxation. A server folds one budget into the allocator when it loads, so per-request budgets are untested. Our three backbones all route to 8 experts, our four benchmarks all reward long reasoning, and we measured neither safety behaviour nor long-context tasks.

6 Conclusion

ECCoMoE chooses, for each token at each layer, how many of the router's top experts to run, so one checkpoint of a frozen MoE model serves both a continuous budget dial and a free mode. Across three backbones and four benchmarks, at matched average expert count, the dial gains most over interpolated fixed top-k where fixed k degrades fastest, up to 20 points at about two experts per token, and on MuSR at that budget, swapping in random counts or another token's profile costs accuracy. With the usual renormalization, fixed top-k collapses on two of the three backbones, so how kept experts are weighted can decide a comparison on its own. Because the allocator also changes how much the model writes, adaptive-compute methods should report energy per response, not only expert counts.

7 AI use statement

In this work, we used generative AI tools for implementing methods, cleaning data and research methodology. We have not used generative AI tools for generating hypotheses or mathematical

claims, and proofs and theoretical models do not apply to this work. Separately from this assistance, the allocator training data are synthetic by design as they are each backbone's own responses to a prompt pool (Section 3.3, Appendix A). According to their dataset cards and the CAMEL paper, the CAMEL math, physics, chemistry and biology problems in that pool were themselves generated by GPT-4; we use only the problems, not GPT-4's solutions. Due to unavailability of Ada-K's code, we used Anthropic's Claude Opus 5 Anthropic (2026) through Claude Code, an agentic coding harness. These models wrote and debugged code for our frozen-backbone adaptations of Ada-K. The authors designed the methods, ran the experiments and analyses and made all research decisions. The authors reviewed the code. No AI was used to generate references. The authors manually verified every cited work and bibliography entry. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

8 Ethics statement

This work involves no human subjects or annotators and releases no new dataset. The allocators are trained on the backbones' responses to publicly available prompts: the problem statements of the CAMEL math, physics, chemistry and biology sets and of AQuA-RAT, and first user turns from WildChat-1M (Hugging Face revision `7d6490e`, 837,989 conversations). We use none of the original chatbot responses. WildChat-1M consists of real users' conversations with GPT-3.5 and GPT-4 chatbots that its authors offered for free; users could open the chat only after accepting an agreement for the collection, use and sharing of their conversations. Its authors de-identified the data with Microsoft Presidio and hand-written rules and hashed IP addresses, and the release we use excludes conversations flagged as toxic by the OpenAI Moderation API or Detoxify and conversations a later study flagged as containing personal information. We filtered the WildChat prompts only by length, character set and exact duplicates (Appendix A). Automatic de-identification and toxicity filtering are imperfect, so the traces may still carry personal or offensive content. We therefore release code to regenerate the traces rather than the traces themselves. The backbones are released under Apache 2.0 (Qwen3.6-35B-A3B, Gemma-4-26B-A4B) and OpenMDW-1.1 (Laguna-XS-2.1), which place no restriction on using their outputs. AQuA-RAT is licensed Apache 2.0 and WildChat-1M ODC-BY. The CAMEL sets are licensed CC BY-NC 4.0 and gated for research use; we use them only for non-commercial research. ECCoMoE changes how much computation a frozen model spends on each token. We measure task accuracy but not safety behaviour such as refusals, which may degrade at low budgets as accuracy does, so it should be evaluated before deployment. Fewer experts per token also do not guarantee lower energy per response (Section 4.5).

9 Reproducibility statement

Sections 3.2 and 3.3 define the allocator, its training relaxation and objective, and its use at inference. Appendix A gives the backbones and their sampling settings (Table 5), the prompt pool and its decontamination, the trace acceptance rules (Table 6), the complete training recipe and seed (Table 7), checkpoint selection and compute. Appendix B describes the fused routing kernels on each backbone, the vLLM patch that every reported evaluation depends on, the routing path of every setting and the kernel parity checks (Table 9). Section 4.1 and Appendix C give the prompts and grading of each benchmark, the serving and sampling configuration and software versions, how $\mathbb{E}[k]$ and energy are measured, the noise bands (Table 11), and the interpolated fixed-k reference at matched average expert count and its confidence interval. Appendix E documents our frozen-backbone adaptations of BEAM and Ada-K and how they depart from the published methods (Tables 18 and 19). Appendix D reports every configuration, and Appendices F and G give the protocols of the controls and analyses. The anonymized supplementary material contains the code for trace generation and filtering, allocator training, the baselines, the vLLM integration with its Triton kernels and patch, and the evaluation harness. We will release the code, kernels, and the three allocator checkpoints publicly with the final version.

References

Julien Abadji, Marah Abdin, Connor Adams, Eric Alcaide, Mustafa Altun, Michele Artoni, Junze Bao, Uday Barar, Vassilis Bekiaris, Arkadii Bessonov, Benjamin Bütikofer, Jonathan Chang,

Yen-Chun Chen, Dmitry Chernenkov, Yang Chi, Filippos Christianos, Fenia Christopoulou, Razvan-Andrei Ciocoiu, Tzachi Cohen, Yohann Coppel, Dmitrii Emelianenko, Brandon Fergerson, Brian Fitzgerald, Matthias Gallé, Alex Golonzovskyi, George Grigorev, Yiyang Hao, Christian Hensel, Jan Huenermann, Ye Ji, Sarthak Joshi, Eiso Kant, Kabir Khandpur, Seonghyeon Kim, Vladimir Kirichenko, Umut Kocasarac, Ilya Kochik, Ivan Komarov, Chaerin Kong, Anurag Koul, François-Joseph Lacroix, Sergei Laktionov, Waren Long, Quentin Malartic, Vadim Markovtsev, Afonso Marques, Robert McHardy, Carlos Mocholí, Dmitry Monakhov, Adam Morris, Martin Muller, Christian Mürtz, Robin Nabel, Thien Nguyen, Rok Novosel, Szymon Ozog, Aalhad Patankar, Aleksei Petrov, Alexandre Piché, Arthur Pignet, Teodor Poncu, Phil Potter, Alexander Rakowski, Pierre-Yves Ritschard, Jay Roberts, Joe Rowell, Piotr Sarna, Pierre-André Savalle, Uladzislau Sazanovich, Nikita Shapovalov, Arsenii Shevchenko, Mikhail Shilkov, Andrei Sokol, Mohamed Soliman, Jack Stephenson, Victor Storchan, Dragos-Constantin Tantaru, Artem Tyurin, Adrian Wälchli, Pengming Wang, Jianxiao Yang, Renat Zayashnikov, Alexander Zelenka Martin, Nikolay Zinov, Caroline Bercier, José Caldeira, Margarida Garcia, Tom George, Kabeer Gharzai, Glenn Hitchcock, Carson Klingenberg, Ivo Pinto, Varun Randery, Noah Smith, Arina Sugako, and Jason Warner. Laguna m.1/xs.2 technical report, 2026. URL https://arxiv.org/abs/2605.27605.

Maryam Akhavan Aghdam, Hongpeng Jin, and Yanzhao Wu. Da-moe: Towards dynamic expert allocation for mixture-of-experts models. *arXiv preprint arXiv:2409.06669*, 2024.

Anthropic. Claude opus 5 system card. https://www.anthropic.com/claude/opus, 2026.

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

Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder. *arXiv preprint arXiv:2107.05407*, 2021.

Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. Flextron: Many-in-one flexible large language model. *arXiv preprint arXiv:2406.10260*, 2024.

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

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

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *CoRR*, abs/2101.03961, 2021. URL https://arxiv.org/abs/2101.03961.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Yongxin Guo, Zhenglin Cheng, Xiaoying Tang, Zhaopeng Tu, and Tao Lin. Dynamic mixture of experts: An auto-tuning approach for efficient transformer models. In *International Conference on Learning Representations*, volume 2025, pp. 79643–79672, 2025.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

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. *arXiv preprint arXiv:2103.03874*, 2021.

Quzhe Huang, Zhenwei An, Nan Zhuang, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. Harder task needs more experts: Dynamic routing in moe models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12883–12895, 2024.

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, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Peng Jin, Bo Zhu, Yuan Li, and Shuicheng Yan. Moe++: Accelerating mixture-of-experts methods with zero-computation experts. In *International Conference on Learning Representations*, volume 2025, pp. 50832–50856, 2025.

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.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding, 2020. URL https://arxiv.org/abs/2006.16668.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008, 2023.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 158–167, 2017.

Baihui Liu, Kaiyuan Tian, Wei Wang, Zhaoning Zhang, Linbo Qiao, and Dongsheng Li. Allocmoe: Budget-aware expert activation allocation for efficient mixture-of-experts inference. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9653–9667, 2026.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in neural information processing systems*, 36:21558–21572, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6159–6172, 2024.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, April 2026. URL https://qwen.ai/blog?id=qwen3.6-35b-a3b.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472, 2022.

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

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. Musr: Testing the limits of chain-of-thought with multistep soft reasoning. In *International Conference on Learning Representations*, volume 2024, pp. 14670–14728, 2024.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, Mayank Chaturvedi, Aditya Chawla, Victor Cotruta, Alice Coucke, Phil Culliton, Robert Dadashi, Lucas Dixon, Mohamed Elhawaty, Utku Evci, Clément Farabet, Johan Ferret, Filippo Galgani, Sertan Girgin, Jean-Bastien Grill, Maarten Grootendorst, Jiaxian Guo, Cassidy Hardin, Yanzhang He, Steven M. Hernandez, Omri Homburger, Léonard Hussenot, Juyeong Ji, Armand Joulin, Aishwarya Kamath, Parnian Kassraie, Olivier Lacombe, Preethi Lahoti, Gaël Liu, Gus Martins, Luciano Martins, Tatiana Matejovicova, Ramona Merhej, Nikola Momchev, Sneha Mondal, Ryan Mullins, Sindhu Raghuram Panyam, Shreya Pathak, Sarah Perrin, André Susano Pinto, Etienne Pot, Angéline Pouget, Alexandre Ramé, Sabela Ramos, Douglas Reid, David Rim, Morgane Rivière, Karsten Roth, Louis Rouillard, Omar Sanseviero, Pier Giuseppe Sessa, Shane Settle, Danila Sinopalnikov, Sara Smoot, Piotr Stanczyk, Andreas Steiner, Lawrence Stewart, Ilya Tolstikhin, Michael Tschannen, Anton Tsitsulin, Nino Vieillard, Renjie Wu, Pingmei Xu, Haichuan Yang, Edouard Yvinec, Biao Zhang, Li Zhang, Joe Zou, Nicolas Aagnes, Abdelrahman Abdelhamed, Jakub Adamek, Shivani Agrawal, Shubham Agrawal, Ibrahim Alabdulmohsin, Jean Baptiste Alayrac, Uri Alon, Chandramouli Amarnath, Ankesh Anand, Chrysovalantis Anastasiou, Setareh Ariafar, François-Xavier Aubet, Kyriakos Axiotis, Federico Barbero, Joelle Barral, Alexei Bendebury, Urs Bergmann, Stanley Bileschi, Kat Black, Mathieu Blondel, Sebastian Borgeaud, Arthur Bražinskas, Ryan Burnell, Robert Busa-Fekete, Mu Cai, Daniele Calandriello, Glenn Cameron, Charlotte Caucheteux, Rahma Chaabouni, Garima Chadha, Jetha Chan, Blake Jianhang Chen, Jesse Chen, Lin Chen, Xu Chen, Derek Cheng, Tzu hsiang Chien, Nikolai Chinaev, Yi Chou, Zhaohui Chu, Benjamin Coleman, Pooja Consul, Sam Conway-Rahman, Scott Crowell, Dylan Cutler, Vivek Dani, Samira Daruki, Anil Das, Daniel Deutsch, Nishanth Dikkala, Li Ding, Qiuhan Ding, Shenil Dodhia, Konstantin Donhauser, Tulsee Doshi, Anca Dragan, Alex Druinsky, Sahil Dua, Zoltan Egyed, Danielle Eisenbud, Daniel Eppens, Cindy Fan, Bahare Fatemi, Yassir Fathullah, Vlad Feinberg, Milen Ferev, Sebastian Flennerhag, Takumi Fujimoto, João Gabriel Oliveira, Isaac Galatzer-Levy, João Gante, Simon Geisler, Soham Ghosal, Antonious M. Girgis, Tamara von Glehn, Alec Go, Alhaad Gokhale, Alex Grills, Yiming Gu, Mayank Gupta, Pramod Gupta, Guru Guruganesh, Raia Hadsell, Hamza Harkous, Jitendra Harlalka, Demis Hassabis, Anja Hauth, Joe Heyward, Arian Hosseini, Chih-Yang Hsia, I-Hung Hsu, Xiaopeng Huang, Yangsibo Huang, Kevin Hui, Adrian Hutter, Te I, Fotis Iliopoulos, Advait Jain, Ganesh Jawahar, Ziwei Ji, Qilin Jin, Melvin Johnson, Kandarp Joshi, Arun Kandoor, Wang-Cheng Kang, Koray Kavukcuoglu, Mehran Kazemi, Kathleen Kenealy, Amr Khalifa, Phoebe Kirk, Ivan Korotkov, Suraj Kothawade, Vitaly Kovalev, Neel Kovelamudi, Adam Kraft, Ravin Kumar, Vivek Kumar, Harish Kuppam, Justin Lannin, Chen-Yu Lee, Seungji Lee, Dmitry Lepikhin, Alon Levkovitch, Dongdong Li, Qiujia Li, Valentin Liévin, Ethan Lin, Ziqian Lin, Casper Liu, Tianlin Liu, Tianqi Liu, Xin Liu, Ivan Lobov, Mayank Lunayach, Min Ma, Gagan Madan, Andrii Maksai, Eric Malmi, Michal Matuszak, Daniel McDuff, Gaurav Menghani, Maciej Mikuła, Daniil Mirylenka, Karolis Misiunas, Vedant Misra, Andreea Mitran, Kareem Mohamed, Maksim Mukha, Eric Noland, James

O'Donnell, Brendan O'Donoghue, Kate Olszewska, Bernett Orlando, Wanqiong Pan, Rina Panigrahy, Unnati Parekh, Nicolas Perez-Nieves, Chunjong Park, Eric Paskie, Liqian Peng, Bryce Petrini, Slav Petrov, Jonas Pfeiffer, Bilal Piot, Martyna Plomecka, Siim Poder, Octavio Ponce, Arijit Pramanik, David Racz, Anish Rajan, Michelle Ramanovich, Anand Rao, Marvin Ritter, Vitor Rodrigues, Evan Rosen, Mikołaj Rybiński, Noveen Sachdeva, Michaël E. Sander, Rohit Sathyanarayana, Sagar Savla, Samuel Schmidgall, Tal Schuster, George Scrivener, Benoit Seguin, Andrew Sellergren, Aliaksei Severyn, Izhak Shafran, Dhruv Shah, Bobak Shahriari, Yuan Shangguan, Ashish Shenoy, Pradeep Shenoy, Rakesh Shivanna, Pauline Sho, Lucas Spangher, Wojciech Stokowiec, Tim Strother, Yao Su, Yinghao Sun, Mukund Sundararajan, Andrea Tacchetti, Mor Hazan Taege, Pouya Tafti, Jean Tarbouriech, Chetan Tekur, Shantanu Thakoor, Rahul Thapa, Madeleine Traverse, Lenart Treven, Tao Tu, Chien Te Tung, Çağlar Ünlü, Petar Veličković, Malini Pooni Venkat, Sagar Gubbi Venkatesh, Vidya Venkiteswaran, Francesco Visin, Alex Vitvitskyi, Kiran Vodrahalli, Weiyi Wang, Xin Wang, Tris Warkentin, Jan Wassenberg, John Wieting, Cindy Wu, Lechao Xiao, Hao Xu, Yuhui Xu, Fuzhao Xue, Arun Yadav, Jun Yan, Antoine Yang, Lin Yang, Ming-Hsuan Yang, Ziyu Ying, Jae Hyeon Yoo, Morteza Zadimoghaddam, Sajjad Zafar, Fred Zhang, Jiageng Zhang, Jianyi Zhang, Xiaofan Zhang, Chao Zhao, David Zhou, and Chen Zou. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Philippe Tillet, Hsiang-Tsung Kung, and David D. Cox. Triton: an intermediate language and compiler for tiled neural network computations. *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, 2019. URL https://api.semanticscholar.org/CorpusID:184488182.

Ziteng Wang, Jun Zhu, and Jianfei Chen. Remoe: Fully differentiable mixture-of-experts with relu routing. In *International Conference on Learning Representations*, volume 2025, pp. 59486–59507, 2025.

Juntong Wu, Jialiang Cheng, Qishen Yin, Yue Dai, Yuliang Yan, Fuyu Lv, Ou Dan, and Li Yuan. Beam: Binary expert activation masking for dynamic routing in moe. *arXiv preprint arXiv:2605.14438*, 2026.

Yuanhang Yang, Shiyi Qi, Wenchao Gu, Chaozheng Wang, Cuiyun Gao, and Zenglin Xu. Xmoe: Sparse models with fine-grained and adaptive expert selection, 2024. URL https://arxiv.org/abs/2403.18926.

Zihao Zeng, Yibo Miao, Hongcheng Gao, Hao Zhang, and Zhijie Deng. Adamoe: Token-adaptive routing with null experts for mixture-of-experts language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 6223–6235, 2024.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatgpt interaction logs in the wild. In *International Conference on Learning Representations*, volume 2024, pp. 34590–34605, 2024.

Zijia Zhao, Longteng Guo, Jie Cheng, Xuange Gao, Hua Huang, and Jing Liu. Ada-k routing: Boosting the efficiency of moe-based llms. In *International Conference on Learning Representations*, 2025.

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 Models and training

Backbones. See Table 5. Qwen3.6 and Gemma-4 use standard softmax routing, while Laguna-XS weights experts by its unbiased sigmoid scores normalized over its top 8 and multiplies the MoE output by a routed scaling factor. Qwen3.6 and Laguna-XS add a shared expert; Gemma-4 runs a dense MLP in parallel with the MoE block and RMS-normalizes the routed output.

Table 5: Backbones. E: routed experts. $k_{\max}$: default top-k. d: hidden size. Allocator: trained parameters over all MoE layers. Sampling (temperature / top-p / top-k / presence penalty) is used for the training traces and every evaluation.

Model	Router	E	$k_{\max}$	d	MoE layers	Allocator	Sampling
Qwen3.6-35B-A3B	softmax	256	8	2048	40/40	21.1M	1.0 / 0.95 / 20 / 1.5
Gemma-4-26B-A4B	softmax	128	8	2816	30/30	10.8M	1.0 / 0.95 / 64 / –
Laguna-XS-2.1	sigmoid + bias	256	8	2048	39/40	20.5M	1.0 / 1.00 / 20 / –

Prompts. All backbones share one pool of 495,060 first user turns: 346,542 from WildChat (10–2,000 characters, at least 90% ASCII, exact duplicates removed) and 148,518 math and science problems (30%): 48,801 CAMEL math, about 20,000 each of CAMEL physics, chemistry and biology, and 39,998 AQuA-RAT algebra word problems. Decontamination by 13-gram word-shingle collision against the evaluation benchmarks removed 3 math and science prompts.

Traces. Each backbone answers pool prompts with its own chat template, thinking on and the sampling of Table 5, until it has about 20M accepted response tokens (Table 6). A response is accepted if it stopped on its own with a closed reasoning section (on Gemma-4, its thought channel) and a non-empty answer; rejected answers are not retried. The output cap is 32,768 tokens on Qwen3.6 and Gemma-4; Laguna-XS runs in a 16,384-token context with an unrecorded cap (script default 12,288; longest accepted response 12,268). After holding out complete validation traces, the rest are packed in order (prompt, then response) into one 20.1M-token stream read as non-overlapping 2,048-token windows. The loss covers response tokens only.

Table 6: Training traces. Train and validation tokens include prompts; scored tokens are the response tokens in the loss. Generation time excludes failed attempts. Accepted traces beyond the 20.1M-token cap are not used.

	Qwen3.6	Gemma-4	Laguna-XS
Prompts tried / accepted	5,888 / 5,824	10,240 / 10,224	8,456 / 7,846
Accepted response tokens	20.70M	20.96M	20.64M
Train tokens (scored)	20.10M (19.67M)	20.10M (19.30M)	20.10M (19.29M)
Train documents	5,543	9,461	7,345
Validation tokens / documents	231k / 64	144k / 64	113k / 32
Math and science share (examples / tokens)	29.5% / 43.3%	30.1% / 40.0%	25.6% / 42.4%
Generation time (GPU-hours)	2.8	2.8	2.3

Training. Only the allocators are trained, in fp32, with the objective of Section 3.3 and the recipe of Table 7. The initialization picks $k{=}8$ in every layer under a hard argmax without a budget. Before training, the wrapped model at $k{=}8$ reproduced the unmodified model's logits exactly on 128 validation tokens on all backbones (training wrapper only, not the served path; Appendix B). We use a budget dropout of 0.25, but in practice 289 of the 1,220 steps withheld the budget (free mode).

Checkpoint selection. We keep the checkpoint with the lowest validation perplexity at budget 3.5, except on Qwen3.6, where it (step 100) still overspends (4.9 experts per token at budget 3.5), so we take the final step. The evaluated checkpoints are steps 1,220 (Qwen3.6), 1,150 (Gemma-4) and 850 (Laguna-XS).

Compute. Training is data-parallel on H200 GPUs, a full bf16 backbone per GPU: 4 GPUs $\times$ 2 accumulation steps on Qwen3.6 and Gemma-4, 2 GPUs $\times$ 4 accumulation steps on Laguna-XS. It took 81 min (5.4 GPU-hours) on Qwen3.6, 21 min (1.4 GPU-hours) on Gemma-4 and 31 min (1.0 GPU-hours) on Laguna-XS.

Table 7: Allocator training recipe, shared by all three backbones.

Optimizer	AdamW, learning rate 10^{-3}, weight decay 0.01, default betas
Schedule	50-step linear warm-up multiplied into a cosine decay to 10^{-4}
Gradient clipping	global norm 1.0
Steps and batch	1,220 steps of $8 \times 2{,}048$ tokens (19.99M tokens, under one epoch)
Count set	$k \in \{1, \ldots, 8\}$
Temperature τ	cosine from 1.0 to 0.3 over the run
Budget penalty λ_s	linear from 0 to 0.5 over the first 100 steps, then fixed
Mutual-information weight β	0.2
Budget curriculum	steps 1–300: $t \in \{2, 7\}$; then $t \sim \mathcal{U}(2, 7)$
Budget withheld	probability 0.25 per step
Initialization	$W^{(2)}{=}0$, $\mathbf{b}^{(2)}{=}(0, \ldots, 0, 2)$, $\gamma_\ell{=}1$
Validation	every 50 steps, 16 held-out windows, hard argmax, budgets 1, 2, 3.5, 4, 6, 7 and free
Seed	1234

B SERVING IMPLEMENTATION

Headroom. Let f be the routed-expert share of the unmodified model's forward pass. With expert time linear in active experts and a zero-cost allocator, running k of $k_{\max}$ experts gives

$$S(k) = \frac{1}{(1-f) + f\,k/k_{\max}}, \qquad S(1) = \frac{1}{1 - f(1 - 1/k_{\max})}, \qquad S_\infty = \frac{1}{1-f}, \tag{6}$$

with S_∞ the Amdahl ceiling. Decoding does worse, since a batch still touches many distinct experts (weight traffic, underfilled GEMM tiles, dispatch overhead). Table 8 gives f from kernel profiles (16 short GSM8K prompts, repeated four times for 64 requests, prefilled and then greedily decoded to 128 tokens; summed GPU event time, not wall-clock). These values indicate headroom but neither bound nor predict our measured speedups; at $f{=}0.44$ (Qwen3.6, stock model, 64 requests), an eightfold cut in experts gives only $S(1){\approx}1.6$.

Table 8: Routed-expert GEMM share of GPU event time in short, decode-dominated runs, current serving code with every counter off. Stock (share f): the unmodified model on vLLM 0.28.0 with only the buffer-zeroing patch of the fixed-width routing paragraph below; fixed k runs through our PyTorch routing.

	stock	fixed k			
Model	batch 64	$k{=}8$, batch 16	$k{=}8$, batch 64	$k{=}4$, batch 64	$k{=}1$, batch 64
Qwen3.6	43.9%	46.3%	39.9%	30.8%	16.1%
Gemma-4	52.5%	50.4%	47.1%	39.0%	20.5%
Laguna-XS	49.4%	46.6%	44.3%	34.1%	18.6%

Fused routing. On Qwen3.6 and Gemma-4, the router weight W^R is stacked with $W^{(1)}$ at load, so one GEMM yields both the router logits and the allocator's pre-activations (on Gemma-4 the allocator rows are rescaled at load to absorb its router's input normalization). One Triton epilogue then computes, per token, the fp32 softmax and top-8, the allocator's SiLU and second layer, and the argmax over counts (first maximum on ties), and writes the kept expert ids with their native weights. Routing thus costs one GEMM and one kernel per MoE layer, plus count logging when enabled. A fixed budget's term (Eq. 3) is folded into $\mathbf{b}_\ell^{(2)}$ at load; free mode skips the fold. The GEMM runs in the router's precision, fp32 on Gemma-4 and bf16 on Qwen3.6, so Qwen3.6's allocator pre-activations are bf16 rather than fp32 as in training. On Laguna-XS, vLLM runs the sigmoid router and selection bias unchanged and returns the top-8 candidates; the allocator's first layer is a separate fp32 linear, and a per-token Triton kernel applies SiLU, the second layer and the budget term, takes the argmax, re-ranks the candidates by biased score and keeps the first k at native weights. Free mode runs the same kernel with the budget term's weight set to zero; Table 9 does not cover it, but it matches the PyTorch path's k on all 479,232 random (not captured) test inputs. Every evaluated run uses the kernel source and checkpoint that passed Table 9.

Fixed-width routing. All paths return $T \times 8$ routing tensors, as default top-8 does; dropped slots get weight 0 and id -1. Shapes are data-independent with no host sync, so every evaluation except the eager replacement controls runs routing in CUDA graphs. vLLM 0.28.0's Triton fused-MoE kernels skip id -1, but their aligned path reuses one workspace for both expert GEMMs, so skipped rows keep stale values that are summed into the output. All reported evaluations, stock-model runs included, use a one-line patch that zeroes this buffer; without it, variable k gives degenerate repetitions while fixed $k{=}8$ (no -1 ids) looks normal. The fixed width keeps vLLM's small-batch path, which skips the expert sort, at stock sizes (up to 8 tokens on Qwen3.6 and Laguna-XS, 4 on Gemma-4), half the range a true $T \times 4$ tensor would get at fixed $k{=}4$. In synthetic batch-16 generation on Qwen3.6 at $k{=}4$, which only the narrower tensor would cover, the sort takes under 2% of GPU time.

Correctness. On the softmax path (Table 9), kernel and reference read the same widened logits, so the check excludes the widened GEMM, and the reference is our fp32 implementation, not the PyTorch routing used for fixed k; the captured Qwen3.6 tokens chose $k{=}3$ to 6. Expert ids can differ only on bit-identical probabilities, which the kernel breaks toward the higher id and PyTorch's top-k, like the pretrained layer, leaves undefined; such ties are common on Qwen3.6 (kept sets agree on 98.2% of tokens) and absent from the Gemma-4 capture, so the softmax path matches up to tie-breaking. The $k{=}k_{\max}$ identity was checked logit for logit only in the training wrapper (Appendix A). The served $k{=}8$, our PyTorch routing, was compared with the unmodified model only in earlier runs at temperature 0.6, on GPQA-D and MuSR among our benchmarks, three seeds each: within noise on Qwen3.6 and Gemma-4 (means at most 1.3 points apart), not run on Laguna-XS. Laguna-XS's PyTorch free-mode path, used in the eager replacement controls, was compared with the free-mode kernel only on the random inputs above, never token by token on captured inputs.

Table 9: Agreement of the served routing kernels with unfused references. Both softmax rows use layer 19.

Path	Inputs	Criterion	Result
Softmax, Qwen3.6	1,536 captured tokens, one layer, $t{=}4$	counts exact; kept weights $< 5{\times}10^{-7}$; ids equal except exact ties	max error $4.5{\times}10^{-8}$ (ties on 61% of tokens)
Softmax, Gemma-4	1,024 captured tokens, one layer, $t{=}4$	as above	max error $8.9{\times}10^{-8}$; router fold within $1.5{\times}10^{-6}$
Sigmoid, Laguna-XS	39 layers $\times$ $t \in \{1, 2, 4, 8\}$, 39,936 tokens	counts, ids and weights exact	0 mismatches

Cost of the allocator. The allocator's first layer has the router projection's shape, so gating costs $2dE{+}2Ek_{\max}$ FLOPs per token and MoE layer, against $2dE$ for the router (Table 10). The epilogue takes 2–6 μs of GPU time per layer for 1–1,536 tokens.

Table 10: FLOPs per token and MoE layer. Routed experts are three-matrix gated FFNs (SwiGLU (Shazeer, 2020) on Qwen3.6 and Laguna-XS, GELU-tanh-gated (Hendrycks & Gimpel, 2016) on Gemma-4) of width $d_{\text{ff}}{=}512$, 704, 512, so one expert costs $6\,d\,d_{\text{ff}}$.

Model	Allocator	Router	One expert	Routed, $k{=}8$	Allocator / routed ($k{=}8$)	($k{=}2$)
Qwen3.6	1.05M	1.05M	6.29M	50.3M	2.1%	8.4%
Gemma-4	0.72M	0.72M	11.9M	95.2M	0.8%	3.0%
Laguna-XS	1.05M	1.05M	6.29M	50.3M	2.1%	8.4%

C EVALUATION PROTOCOL

Prompts and grading. One zero-shot chat request per item (system and user message), thinking enabled; the reasoning parser strips the reasoning and only the final answer is graded. Truncated and empty answers stay in the denominator; unparseable ones count as wrong. MATH-500: last `\boxed{}` expression, Hendrycks normalization, exact string match. GPQA-Diamond: options in a question-hashed order, answer as `Answer: <letter>`. MuSR: all three splits (murder mysteries, object placements, team allocation; 756 items, two to five options), original option order. Both multiple-choice benchmarks take the last *answer is (X)*, else the last `Answer: X` line, else

the last standalone option letter. MBPP+: one self-contained Python function in one code block, with the task's visible tests in the prompt; the longest code block must pass every EvalPlus extended test in an isolated interpreter (10 s CPU, 4 GB memory, 30 s wall clock).

Serving and sampling. Each run (one configuration, benchmark and seed) gets a fresh vLLM 0.28.0 server on one H200 (bf16, CUDA graphs, chunked prefill, Triton MoE backend, prefix caching off) with 64 requests in flight, each model's recommended sampling on every benchmark (Table 5), a 32,768-token cap and seed 0, 1 or 2 passed to server and requests. Three warm-up requests and 10 s idle precede the measured window; the GPU runs nothing else and starts stable at or below 60°C.

Measuring $\mathbb{E}[k]$. Inside every MoE layer, a CUDA-graph-compatible counter records how many experts each token is sent to, as one histogram per layer. It counts every token the model processes during the measured window, once per MoE layer: each prompt token once, at prefill (prefix caching is off), and each generated token, reasoning included; warm-up requests and CUDA-graph padding are not counted. Prompt tokens are thus 0.5–2.6% of the count on MATH-500 and, on MuSR, whose prompts are ∼1k tokens, 16–23% on Gemma-4, 23–36% on Laguna-XS and 34–44% on Qwen3.6 (three-seed means over the main grid). In offline per-token traces, leaving out prompt tokens changes $\mathbb{E}[k]$ by at most 0.01 on MATH-500 and at most 0.22 on MuSR. For one run, $\mathbb{E}[k]$ is the number of expert activations divided by the number of (token, layer) pairs; we report its mean over seeds. Under fixed k every counted pair falls in bin k on all three backbones, so the counter reads exactly k.

Energy and speed. Energy $\mathcal{E}$ is GPU board energy from NVML's cumulative counter over the window (readings aligned to counter updates): prefill, decoding, allocator and idle floor, but not CPU, memory or the rest of the node. Over window duration T with n items and O generated tokens (reasoning included) we report $\mathcal{E}/n$, O/T, n/T and O/n. Latency is client-side, request sent to full response; p90 is over items. Values are three-seed means unless marked.

Noise bands. Every main-grid configuration has three seeds, and the spread across seeds sets our noise level. For each benchmark and metric we take each configuration's standard deviation over its three seeds, in points for accuracy and relative to the configuration's mean for the other metrics, and pool these by root mean square into $\bar{\sigma}$, assuming comparable seed variance across configurations. The difference of two three-seed means then has standard error $\bar{\sigma}\sqrt{2/3}$, and Table 11 lists the 95% threshold $1.96\,\bar{\sigma}\sqrt{2/3}$. When the two sides have n_A and n_X seeds, the threshold is $1.96\,\bar{\sigma}\sqrt{1/n_A + 1/n_X}$, i.e. the table value times $\sqrt{(1/n_A + 1/n_X)\cdot 3/2}$. The pool holds the 15 main-grid configurations of each reported backbone (fixed k=2–8, dial t=2–8, free mode), 45 per benchmark; on GPQA-D and MuSR it also holds the seven three-seed configurations of a fourth, unreported backbone, giving 52. Without them, the GPQA-D and MuSR accuracy thresholds would be 3.66 and 1.63 points instead of 3.73 and 1.71. BEAM, Ada-K and renormalized runs are not pooled.

Table 11: Noise bands: 95% thresholds on the difference of two three-seed means (accuracy in points, other metrics relative). Configs: number of three-seed configurations pooled.

Benchmark	Configs	Acc	J/resp	Items/s	p90 lat.	Tok/resp
MATH-500	45	1.25	7.8%	6.0%	7.3%	4.2%
GPQA-D	52	3.73	20.0%	17.0%	32.8%	18.9%
MuSR	52	1.71	10.9%	12.5%	7.9%	5.0%
MBPP+	45	1.60	13.7%	12.0%	12.6%	12.3%

Matched average expert count. With $x = \mathbb{E}[k]$, $k_0 = \lfloor x \rfloor$ and $w = x - k_0$, the interpolated fixed-k reference is $(1-w)\,a_{k_0} + w\,a_{k_0+1}$ over native-weight fixed-k accuracies. The difference $\Delta_{\mathbb{E}}$ has standard error $\sigma\sqrt{1/n + (1-w)^2/n_{k_0} + w^2/n_{k_0+1}}$, with $\sigma = \bar{\sigma}$, seed counts n, n_{k_0}, n_{k_0+1}, and w treated as fixed. A win or loss is a 95% interval excluding zero (nominal, uncorrected for multiple comparisons); otherwise no difference is detected under our uncertainty estimate. Our BEAM and

Ada-K adaptations are compared the same way against both this reference and the dial's curve (dial points at measured $\mathbb{E}[k]$). Below a curve's lowest point the comparison is left open.

D FULL RESULTS

Tables 12–14 list every configuration on every benchmark: fixed k from 2 to 8, the dial from $t{=}2$ to 8, free mode, and two settings each of our frozen-backbone adaptations of BEAM and Ada-K. Figure 4 plots accuracy against measured $\mathbb{E}[k]$ for all twelve model–benchmark pairs.

Three patterns in these tables help in reading the main results. First, response length moves with the expert count. At fixed $k{=}2$ answers are longer than at top-8 on ten of the twelve pairs, most on Gemma-4, where MATH-500 answers grow fivefold and energy per response nearly as much. The dial at $t{=}2$ writes shorter answers than fixed $k{=}2$ on ten of the twelve pairs, which is why its energy per response at matched average expert count is lower on most of them (Table 16). Second, the baselines' failures are often failures of length as well as accuracy: Ada-K at $\lambda{=}10^{-2}$ on Qwen3.6 MATH-500 runs 2.6 experts per token but writes 17k-token answers, spending about 1.5 times top-8's energy per response at 43.8% accuracy. BEAM at $\beta{=}1.0$ on Qwen3.6 runs below one routed expert per token on three benchmarks and still keeps 56–80% accuracy, a regime below the dial's trained range. Third, at the top of the dial ($t{=}8$, $\mathbb{E}[k]$ 6.9–7.0), accuracy and energy per response stay within noise of top-8 on every pair.

Table 15 extends the renormalization comparison of Figure 2 to all four benchmarks at the dial's own $\mathbb{E}[k]$: at $t{=}2$, renormalized fixed k scores 18.5–22.0% on Qwen3.6 and 1.8–11.5% on Laguna-XS, while on Gemma-4 it stays within about 3 points of native weights. Table 16 adds energy and throughput to the matched-count comparison of Table 1: at $t{=}2$ the dial uses less energy per response than the interpolated fixed-k reference on ten of the twelve pairs and serves up to $1.96\times$ as many responses per second (Laguna-XS MuSR). Savings against top-8 are in Table 3.

Table 12: Full results on Qwen3.6-35B-A3B, three-seed means. Acc: accuracy (%); $\mathbb{E}[k]$: measured experts per token; Tok/resp: tokens per response; J/resp: GPU energy per response (J).

	MATH-500				GPQA-D				MuSR				MBPP+			
Setting	Acc	$\mathbb{E}[k]$	Tok/resp	J/resp	Acc	$\mathbb{E}[k]$	Tok/resp	J/resp	Acc	$\mathbb{E}[k]$	Tok/resp	J/resp	Acc	$\mathbb{E}[k]$	Tok/resp	J/resp
fixed $k{=}2$	89.1	2.00	7634	1226	61.6	2.00	9431	1717	57.6	2.00	1296	190	74.6	2.00	2331	487
fixed $k{=}3$	92.2	3.00	7629	1364	74.6	3.00	5844	1229	63.5	3.00	1426	200	81.0	3.00	2270	461
fixed $k{=}4$	92.4	4.00	8149	1533	78.3	4.00	5575	1198	65.0	4.00	1678	250	81.7	4.00	1888	324
fixed $k{=}5$	92.0	5.00	8427	1755	79.3	5.00	5231	1206	65.5	5.00	1907	302	81.4	5.00	2086	401
fixed $k{=}6$	92.1	6.00	8681	1845	78.8	6.00	5523	1375	65.0	6.00	1873	306	82.0	6.00	2049	424
fixed $k{=}7$	91.8	7.00	8812	2013	79.6	7.00	5128	1302	65.2	7.00	1960	336	83.2	7.00	2074	418
fixed $k{=}8$	91.3	8.00	8890	2204	80.3	8.00	5468	1458	65.2	8.00	2003	367	81.4	8.00	2088	448
dial $t{=}2$	92.3	2.30	7466	1179	76.4	2.35	7065	1284	64.9	2.33	1512	196	80.3	2.34	2956	505
dial $t{=}3$	92.3	2.84	7949	1356	74.2	2.85	7126	1397	66.0	2.77	1541	212	81.6	2.86	3239	577
dial $t{=}4$	92.4	3.95	8551	1582	78.8	3.95	6088	1352	66.6	3.90	1477	222	81.7	3.97	3275	654
dial $t{=}5$	91.7	5.07	9035	1892	78.5	5.08	5603	1281	66.4	5.08	1689	268	81.0	5.07	3929	790
dial $t{=}6$	91.9	6.15	9332	2039	79.3	6.15	6772	1672	65.8	6.18	1836	315	82.8	6.13	3144	696
dial $t{=}7$	92.3	6.82	9227	2110	78.5	6.80	6587	1685	64.9	6.84	1884	329	82.7	6.79	2789	669
dial $t{=}8$	91.6	7.01	9228	2109	79.1	7.00	6411	1683	65.7	7.04	1914	330	82.5	6.98	2113	457
free mode	92.2	4.60	8924	1753	77.9	4.64	5903	1389	66.6	4.68	1629	255	83.2	4.61	3169	664
BEAM $\beta{=}0.1$	92.1	3.71	7097	1271	77.8	4.61	12806	2771	66.5	5.42	2148	335	81.7	4.37	5131	926
BEAM $\beta{=}1.0$	80.0	0.75	7858	1138	56.2	0.98	7975	1299	63.5	1.28	1999	260	68.7	0.93	2591	459
Ada-K $\lambda{=}3{\times}10^{-3}$	74.3	3.13	15939	3095	59.9	3.15	9783	1954	60.9	3.19	2671	417	74.1	3.14	5099	910
Ada-K $\lambda{=}10^{-2}$	43.8	2.58	17174	3251	38.0	2.59	5693	1091	34.2	2.59	1797	297	40.7	2.58	3540	629

Table 13: Full results on Gemma-4-26B-A4B; columns as in Table 12.

	MATH-500				GPQA-D				MuSR				MBPP+			
Setting	Acc	$\mathbb{E}[k]$	Tok/resp	J/resp	Acc	$\mathbb{E}[k]$	Tok/resp	J/resp	Acc	$\mathbb{E}[k]$	Tok/resp	J/resp	Acc	$\mathbb{E}[k]$	Tok/resp	J/resp
fixed k=2	33.2	2.00	21387	4252	43.9	2.00	13666	2598	57.4	2.00	5478	927	42.3	2.00	8400	1613
fixed k=3	79.6	3.00	9597	1861	66.5	3.00	12744	2476	66.1	3.00	4347	732	78.0	3.00	3250	611
fixed k=4	90.7	4.00	5476	1018	74.2	4.00	11321	2301	62.7	4.00	4017	706	82.7	4.00	2274	477
fixed k=5	92.4	5.00	4644	888	77.3	5.00	10797	2229	67.5	5.00	3828	685	82.3	5.00	1993	456
fixed k=6	91.9	6.00	4470	871	78.8	6.00	10477	2252	68.3	6.00	3856	709	83.9	6.00	1934	474
fixed k=7	92.4	7.00	4349	883	77.8	7.00	10439	2340	69.0	7.00	3557	667	83.2	7.00	1975	510
fixed k=8	92.5	8.00	4299	891	79.3	8.00	10163	2371	69.4	8.00	3492	675	82.9	8.00	1910	545
dial t=2	55.5	2.05	15819	2971	46.8	2.06	12799	2387	60.8	2.05	5236	854	50.0	2.04	6898	1260
dial t=3	85.2	2.87	6672	1221	72.1	2.89	12028	2336	66.1	2.87	4650	795	70.9	2.83	2703	493
dial t=4	89.6	3.75	5196	935	74.2	3.79	11101	2204	66.2	3.77	3787	667	79.6	3.72	2339	484
dial t=5	91.9	4.87	4603	864	77.1	4.91	10619	2243	67.9	4.88	3684	666	81.9	4.85	2020	449
dial t=6	92.7	5.97	4377	854	79.0	5.96	10526	2272	69.4	5.96	3591	675	83.3	5.93	1964	485
dial t=7	92.5	6.73	4256	879	77.6	6.74	10318	2270	69.1	6.71	3365	645	83.2	6.72	1901	491
dial t=8	92.0	7.03	4250	873	77.6	7.05	10325	2328	69.7	7.00	3416	643	82.7	7.04	1868	484
free mode	91.5	5.11	4844	918	77.1	5.08	10747	2254	59.6	5.14	3569	643	77.1	5.05	2152	489
BEAM β=0.1	91.9	5.06	4383	880	76.8	5.33	11352	2612	67.5	5.37	3161	599	82.6	5.27	2063	517
BEAM β=1.0	87.6	3.14	4998	894	73.2	3.37	9467	1921	66.6	3.40	2947	541	80.7	3.33	2656	562
Ada-K λ=3×10^{-3}	87.8	4.02	4916	976	72.1	4.10	10247	2103	41.8	4.25	3341	595	83.1	4.04	2149	476
Ada-K λ=10^{-2}	41.3	2.44	17407	3409	46.3	2.44	14089	2764	59.2	2.46	7591	1326	50.1	2.44	6973	1259

Table 14: Full results on Laguna-XS-2.1; columns as in Table 12.

	MATH-500				GPQA-D				MuSR				MBPP+			
Setting	Acc	$\mathbb{E}[k]$	Tok/resp	J/resp	Acc	$\mathbb{E}[k]$	Tok/resp	J/resp	Acc	$\mathbb{E}[k]$	Tok/resp	J/resp	Acc	$\mathbb{E}[k]$	Tok/resp	J/resp
fixed k=2	87.4	2.00	6688	1159	49.7	2.00	11663	2175	59.3	2.00	3488	555	75.6	2.00	6417	1107
fixed k=3	90.9	3.00	5040	909	53.4	3.00	9116	1718	63.6	3.00	2269	336	80.2	3.00	4361	837
fixed k=4	90.9	4.00	4633	884	51.3	4.00	8610	1712	63.1	4.00	2023	289	82.5	4.00	3807	735
fixed k=5	90.6	5.00	4507	943	52.5	5.00	8142	1667	64.8	5.00	1985	311	81.2	5.00	3570	759
fixed k=6	91.4	6.00	4292	890	50.5	6.00	8242	1864	64.2	6.00	1924	311	81.3	6.00	3552	830
fixed k=7	91.5	7.00	4211	967	46.3	7.00	8245	1945	64.7	7.00	1953	335	81.5	7.00	3551	841
fixed k=8	91.5	8.00	4161	975	51.2	8.00	8310	2031	64.8	8.00	1935	353	81.7	8.00	3357	818
dial t=2	87.1	2.07	5584	946	50.3	2.15	9067	1527	62.4	2.07	2391	311	77.4	2.13	5064	868
dial t=3	90.5	2.73	4957	849	51.9	2.81	8908	1580	63.3	2.67	2094	275	78.0	2.81	4333	768
dial t=4	91.2	3.79	4540	856	56.1	3.93	8450	1598	63.8	3.73	1860	262	81.3	3.89	3779	722
dial t=5	92.1	4.91	4308	875	53.9	4.99	8429	1738	63.1	4.79	1874	277	81.1	4.98	3569	767
dial t=6	91.4	6.12	4365	956	55.7	6.10	8356	1811	63.8	5.90	1960	327	82.4	6.13	3594	815
dial t=7	91.8	6.71	4230	917	49.5	6.69	8304	1849	66.0	6.61	1952	348	82.1	6.72	3416	790
dial t=8	92.3	6.92	4227	923	52.0	6.92	8421	1905	65.2	6.88	1909	317	81.5	6.94	3541	853
free mode	90.8	4.44	4466	875	53.5	4.58	8521	1783	63.5	4.33	1900	298	80.4	4.53	3622	773
BEAM β=0.1	90.8	5.59	5097	1086	55.2	6.20	10129	2401	65.5	6.48	2204	372	77.1	5.90	4016	903
BEAM β=1.0	85.4	3.38	6667	1216	44.9	3.75	9629	1887	60.3	4.14	1746	255	65.9	3.55	6127	1167
Ada-K λ=3×10^{-3}	84.0	4.14	7802	1627	33.2	4.13	9995	2009	56.4	4.14	3304	512	75.0	4.15	6441	1298
Ada-K λ=10^{-2}	70.4	3.25	11455	2308	28.3	3.25	12376	2445	48.8	3.26	5290	801	61.4	3.25	10635	2105

Table 15: Accuracy (%) of the dial and of native-weight and renormalized fixed k at the dial's measured $\mathbb{E}[k]$ for each budget t; fixed-k columns are interpolated between the bracketing integer k. Top-8: the unmodified model, which all three equal at k=8. Three seeds throughout.

			t=2				t=4				t=6			
Model	Benchmark	Top-8	$\mathbb{E}[k]$	Dial	Native	Renorm	$\mathbb{E}[k]$	Dial	Native	Renorm	$\mathbb{E}[k]$	Dial	Native	Renorm
Qwen3.6	MATH-500	91.3	2.30	92.3	90.0	18.5	3.95	92.4	92.4	83.1	6.15	91.9	92.0	91.0
	GPQA-D	80.3	2.35	76.4	66.2	22.0	3.95	78.8	78.1	73.5	6.15	79.3	78.9	79.3
	MuSR	65.2	2.33	64.9	59.5	20.3	3.90	66.6	64.8	61.3	6.18	65.8	65.0	65.1
	MBPP+	81.4	2.34	80.3	76.7	20.5	3.97	81.7	81.6	77.4	6.13	82.8	82.2	81.0
Gemma-4	MATH-500	92.5	2.05	55.5	35.4	33.6	3.75	89.6	87.9	87.7	5.97	92.7	91.9	92.2
	GPQA-D	79.3	2.06	46.8	45.3	42.3	3.79	74.2	72.6	72.6	5.96	79.0	78.7	78.7
	MuSR	69.4	2.05	60.8	57.8	59.2	3.77	66.2	63.5	62.9	5.96	69.4	68.3	69.0
	MBPP+	82.9	2.04	50.0	43.9	42.8	3.72	79.6	81.4	81.7	5.93	83.3	83.7	83.1
Laguna-XS	MATH-500	91.5	2.07	87.1	87.6	2.0	3.79	91.2	90.9	61.1	6.12	91.4	91.4	91.4
	GPQA-D	51.2	2.15	50.3	50.2	11.5	3.93	56.1	51.5	31.6	6.10	55.7	50.1	43.0
	MuSR	64.8	2.07	62.4	59.6	10.5	3.73	63.8	63.2	51.7	5.90	63.8	64.3	61.6
	MBPP+	81.7	2.13	77.4	76.2	1.8	3.89	81.3	82.3	42.9	6.13	82.4	81.3	81.3

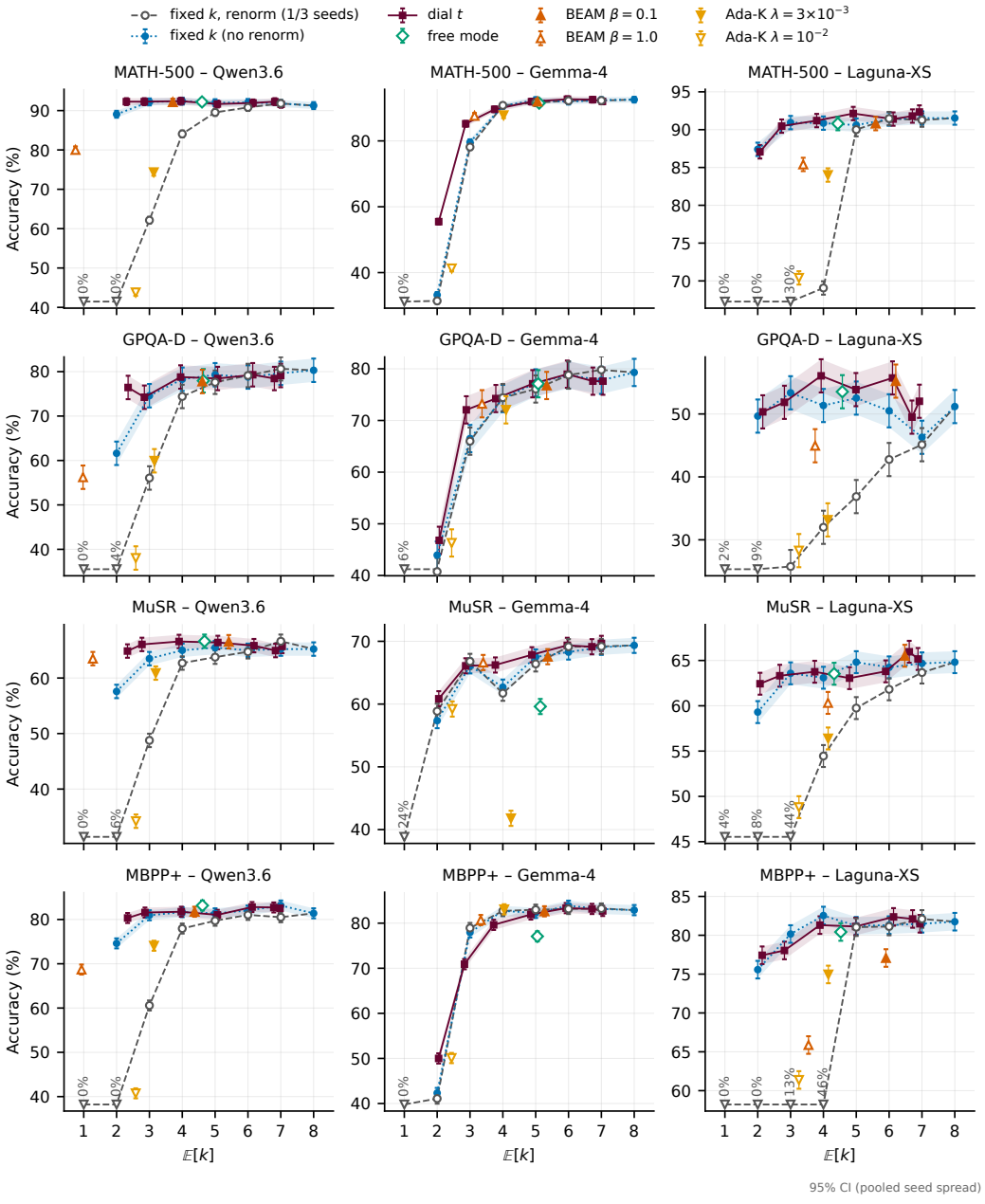

Figure 4: Accuracy against measured $\mathbb{E}[k]$: dial (red), native-weight fixed k (blue), renormalized fixed k (grey), free mode (green), BEAM (orange) and Ada-K (yellow) adaptations, filled and hollow markers for their two settings. Fixed-k lines between integer k are interpolations. Bars and bands: 95% CI from the benchmark's pooled seed spread (comparable seed variance assumed, $\mathbb{E}[k]$ treated as fixed). Three seeds per point.

Table 16: The dial against native-weight fixed k at matched average expert count, $t \in \{2, 4, 8\}$. $\Delta_{\mathbb{E}}$: dial minus the interpolated fixed-k reference at the dial's measured $\mathbb{E}[k]$, as points (Acc), relative energy per response (J/resp) and ratio of responses per second (Items/s); tokens per response are not matched. Energy and items/s come from the accuracy runs (fixed k on PyTorch routing, the dial on the fused kernel, count recording on). **Bold**: accuracy interval excludes zero, as in Table 1; energy or items/s beyond the noise band for two three-seed means (Table 11); nominal, uncorrected for multiple comparisons. At t=8, $\mathbb{E}[k]$ is 6.9–7.0, 12–14% below top-8. Three seeds; accuracy deltas for every budget are in Table 1.

			t=2				t=4				t=8			
Model	Benchmark	Top-8 Acc	$\mathbb{E}[k]$	$\Delta_{\mathbb{E}}$Acc	$\Delta_{\mathbb{E}}$J/resp	$\Delta_{\mathbb{E}}$Items/s	$\mathbb{E}[k]$	$\Delta_{\mathbb{E}}$Acc	$\Delta_{\mathbb{E}}$J/resp	$\Delta_{\mathbb{E}}$Items/s	$\mathbb{E}[k]$	$\Delta_{\mathbb{E}}$Acc	$\Delta_{\mathbb{E}}$J/resp	$\Delta_{\mathbb{E}}$Items/s
Qwen3.6	MATH-500	91.3	2.30	**+2.3**	−7%	**1.08×**	3.95	0.0	+4%	0.98×	7.01	−0.2	+5%	0.98×
	GPQA-D	80.3	2.35	**+10.3**	−17%	1.16×	3.95	+0.7	+13%	0.92×	7.00	−0.5	**+29%**	**0.79×**
	MuSR	65.2	2.33	**+5.3**	+1%	1.05×	3.90	**+1.7**	−9%	1.11×	7.04	+0.5	−2%	1.04×
	MBPP+	81.4	2.34	**+3.6**	+6%	0.99×	3.97	+0.1	**+99%**	**0.52×**	6.98	−0.6	+9%	0.94×
Gemma-4	MATH-500	92.5	2.05	**+20.1**	**−28%**	**1.30×**	3.75	**+1.7**	**−24%**	**1.27×**	7.03	−0.4	−1%	1.03×
	GPQA-D	79.3	2.06	+1.5	−8%	1.08×	3.79	+1.7	−6%	1.04×	7.05	−0.2	−1%	1.03×
	MuSR	69.4	2.05	**+3.1**	−7%	**1.14×**	3.77	**+2.8**	−6%	1.10×	7.00	+0.6	−4%	1.06×
	MBPP+	82.9	2.04	**+6.1**	**−20%**	**1.18×**	3.72	**−1.8**	−6%	1.03×	7.04	−0.4	−5%	1.09×
Laguna-XS	MATH-500	91.5	2.07	−0.6	**−17%**	**1.22×**	3.79	+0.3	−4%	**1.09×**	6.92	+0.8	−4%	**1.08×**
	GPQA-D	51.2	2.15	+0.1	**−27%**	**1.66×**	3.93	**+4.6**	−7%	**1.30×**	6.92	**+5.4**	−2%	**1.23×**
	MuSR	64.8	2.07	**+2.8**	**−42%**	**1.96×**	3.73	+0.5	**−13%**	**1.33×**	6.88	+0.5	−4%	**1.15×**
	MBPP+	81.7	2.13	+1.2	**−19%**	**1.24×**	3.89	−1.0	−3%	1.05×	6.94	0.0	+1%	1.04×

Table 17: Our frozen-backbone adaptations of BEAM and Ada-K at their own measured $\mathbb{E}[k]$, against the dial (Δ_t) and native-weight fixed k (Δ_k): the baseline's accuracy minus each curve, interpolated linearly at the baseline's measured $\mathbb{E}[k]$ (points). **Bold**: the 95% CI excludes zero. open: the baseline runs below both frontiers' lowest points. Each row is one training run with three seeds.

		MATH-500			GPQA-D			MuSR			MBPP+		
Model	Setting	$\mathbb{E}[k]$	Δ_t	Δ_k	$\mathbb{E}[k]$	Δ_t	Δ_k	$\mathbb{E}[k]$	Δ_t	Δ_k	$\mathbb{E}[k]$	Δ_t	Δ_k
Qwen3.6	BEAM β=0.1	3.71	−0.2	−0.2	4.61	−0.8	−1.1	5.42	+0.3	+1.3	4.37	+0.3	+0.2
	BEAM β=1.0	0.75	open	open	0.98	open	open	1.28	open	open	0.93	open	open
	Ada-K λ=3×10^{-3}	3.13	**−18.0**	**−17.9**	3.15	**−15.6**	**−15.2**	3.19	**−5.4**	**−2.9**	3.14	**−7.5**	**−7.0**
	Ada-K λ=10^{-2}	2.58	**−48.5**	**−47.1**	2.59	**−37.3**	**−31.2**	2.59	**−31.3**	**−26.9**	2.58	**−40.2**	**−37.6**
Gemma-4	BEAM β=0.1	5.06	−0.1	−0.4	5.33	−1.1	−1.0	5.37	−1.0	−0.2	5.27	+0.2	−0.1
	BEAM β=1.0	3.14	+1.1	**+6.5**	3.37	0.0	**+3.9**	3.40	+0.5	**+1.9**	3.33	**+4.9**	+1.2
	Ada-K λ=3×10^{-3}	4.02	**−2.4**	**−2.9**	4.10	−3.0	−2.5	4.25	**−25.1**	**−22.1**	4.04	**+2.8**	+0.4
	Ada-K λ=10^{-2}	2.44	**−28.6**	**−12.5**	2.44	**−12.1**	**−7.6**	2.46	**−4.3**	**−2.2**	2.44	**−10.3**	**−7.8**
Laguna-XS	BEAM β=0.1	5.59	−0.9	−0.3	6.20	+0.5	**+5.6**	6.48	0.0	+1.1	5.90	**−5.0**	**−4.2**
	BEAM β=1.0	3.38	**−5.5**	**−5.5**	3.75	**−10.4**	**−6.9**	4.14	**−3.2**	**−3.0**	3.55	**−14.4**	**−15.6**
	Ada-K λ=3×10^{-3}	4.14	**−7.5**	**−6.8**	4.13	**−22.5**	**−18.3**	4.14	**−7.1**	**−6.9**	4.15	**−6.3**	**−7.4**
	Ada-K λ=10^{-2}	3.25	**−20.4**	**−20.5**	3.25	**−25.2**	**−24.6**	3.26	**−14.7**	**−14.6**	3.25	**−18.0**	**−19.4**

E ADAPTIVE-ROUTING BASELINES

We adapt BEAM and Ada-K to a frozen backbone and train them on the allocator's data: the same token caches in the same order, loss on response tokens only, about 20M tokens on one H200, two settings per backbone (Tables 18 and 19). Each checkpoint is evaluated at its final step with ECCoMoE's vLLM 0.28.0 setup and protocol, routed in PyTorch.

BEAM. A mask head reads the MoE input and gives each of the router's top 8 experts a logit; an expert is kept if its logit is non-negative, with a straight-through sigmoid for the gradient. The penalty is β times the mean sigmoid over the eight slots. The original method fine-tunes the whole model on the Tülu 3 mixture; we freeze the backbone and train only the head on our data. With the original learning rate and a positive mask bias the heads stayed almost dense ($\mathbb{E}[k] \approx 7.8$–$7.95$), so we use zero initialization and the allocator's learning rate. We chose $\beta \in \{0.1, 1.0\}$ from 250-step Gemma-4 pilots to spread the two settings across budgets. At β=1.0, Qwen3.6 runs no routed expert on 41–61% of token–layer pairs; its shared expert still runs.

Ada-K. A per-layer linear policy picks $k \in \{1, \ldots, 8\}$ and keeps the first k of the top 8, renormalized to sum to one on Qwen3.6 and Gemma-4 (with Gemma-4's per-expert scale re-applied) and to the native top-8 total on Laguna-XS. A warm-up fits the router's nucleus counts, with the threshold chosen so the mean count is closest to 8. Each PPO step compares a teacher-forced rollout that samples k at every token and layer with one at top-8; the per-token advantage is their difference in response-token log-likelihood, discounted by $0.99^{L-\ell}$ at layer ℓ (clipped surrogate, ϵ=0.2, two epochs per step). λ=3×10^{-3} is the original paper's setting and 10^{-2} a more aggressive one. Serving takes the argmax count, which barely changes across benchmarks; at λ=10^{-2} some layers always pick one expert.

Table 18: Baseline recipes. Both train only the added module on a frozen bf16 backbone with AdamW (learning rate 10^{-3}, weight decay 0.01, gradient clipping 1.0) and $8 \times 2{,}048$ tokens per step.

	BEAM	Ada-K
Added module	$\mathrm{Linear}(d, E)$ mask head, zero-initialized	$\mathrm{Linear}(d, 8)$ policy over k
Parameters (Qwen3.6 / Gemma-4 / Laguna-XS)	21.0M / 10.8M / 20.5M	0.66M / 0.68M / 0.64M
Decision	keep a top-8 expert if its logit ≥ 0	first k of the top 8, argmax at serving
Kept weights	native, not renormalized	renormalized
Fewest routed experts	0	1
Objective	LM loss + β· soft fraction kept	PPO on log-likelihood + $\lambda \cdot \mathbb{E}[k]$
Settings	$\beta \in \{0.1, 1.0\}$	$\lambda \in \{3{\times}10^{-3}, 10^{-2}\}$
Schedule	37-step warm-up, linear decay to 0	97 warm-up steps, then 1,123 PPO steps (cosine)
Steps (tokens)	1,221 (20.0M)	1,220 (19.99M)

Table 19: Baseline checkpoints: validation $\mathbb{E}[k]$ at the end of training and training time on one H200. Served $\mathbb{E}[k]$ per benchmark is in Table 17.

Model	Setting	Val. $\mathbb{E}[k]$	GPU-hours
Qwen3.6	BEAM β=0.1	4.78	4.0
	BEAM β=1.0	1.36	4.0
	Ada-K λ=3×10^{-3}	3.16	2.0
	Ada-K λ=10^{-2}	2.59	2.0
Gemma-4	BEAM β=0.1	5.87	1.0
	BEAM β=1.0	3.92	1.0
	Ada-K λ=3×10^{-3}	4.19	0.8
	Ada-K λ=10^{-2}	2.47	0.8
Laguna-XS	BEAM β=0.1	6.10	0.8
	BEAM β=1.0	3.97	0.8
	Ada-K λ=3×10^{-3}	4.14	0.9
	Ada-K λ=10^{-2}	3.26	0.8

Other differences from the originals. Both train on the backbone's own traces rather than the original instruction data and are served with fixed-width routing rather than their original kernels. BEAM has no load-balancing term, and on Gemma-4 its head reads the experts' input rather than the router's normalized copy of the residual stream. Ada-K's actions are capped at eight experts (the original can add experts), its warm-up labels are clipped to 8, and its penalty averages over all tokens while its policy gradient uses response tokens only; the original does not state its discount and clipping constants. We implemented AdaMoE (null experts) but did not retrain or evaluate it.

F RENORMALIZATION CONTROL

Every fixed-k run in the paper, like the allocator and our BEAM adaptation, keeps the router's first k experts at native weights, scaling the routed output by their share of the top-$k_{\max}$ weight,

$$s_{\ell,i}(k) = \frac{\sum_{m=1}^{k} g_{\ell,i,j_m}}{\sum_{m=1}^{k_{\max}} g_{\ell,i,j_m}}. \tag{7}$$

Standard top-k instead multiplies the survivors by $1/s_{\ell,i}(k)$, about 2.4–2.9 at k=2 (Table 20). We rerun fixed k=1, . . . , 7 with renormalization and nothing else changed (model, protocol, seeds, sampling, serving path), dividing by the kept sum on Qwen3.6 and Gemma-4 and rescaling to the native top-8 total, which its router normalizes to one, on Laguna-XS.

Table 20: Mean kept share $s(k)$ of the native top-8 weight on the unmodified routers, over response tokens and MoE layers of the allocator's free-mode answers (teacher-forced); ranges over the four benchmarks. k=1, 2, 4 use the full traces; k=3, 5, 6, 7 use the first 60–105k response tokens per benchmark, on which k=1, 2, 4 agree with the full traces within 0.003.

Model	k=1	2	3	4	5	6	7
Qwen3.6	.22–.23	.39–.40	.52–.53	.64	.74–.75	.83–.84	.92
Gemma-4	.25–.26	.41–.42	.54–.56	.66–.67	.76–.77	.85	.93
Laguna-XS	.20	.35–.36	.48–.49	.60–.61	.71–.72	.81–.82	.91

Qwen3.6 and Laguna-XS. Renormalization is worse at every small k (Table 21), and at k=2 the two fail differently. Qwen3.6's answers stop after about 150–340 tokens (median 74–251), typically after one or two degraded sentences, and two thirds are empty. Laguna-XS keeps writing, often longer than with native weights, but incoherently; 52–76% of its answers are empty. At k=3 both still lose 15–67 points, and at k=4 Qwen3.6 writes 1.4–2.1× the native-weight tokens. From k=5 up renormalization stays within 3 points on Qwen3.6, though 5 of its 12 comparisons remain below native weights beyond noise, while Laguna-XS still loses 16 and 8 points on GPQA-D and 5 and 2 on MuSR at k=5 and 6. Renormalization is never better beyond noise.

Gemma-4. Gemma-4 applies an RMSNorm to the routed output before adding it to the dense branch and the residual stream. Renormalization scales a token's kept weights by one scalar $1/s_{\ell,i}(k) > 0$, and $\mathrm{RMSNorm}(c\,\mathbf{x}) = \mathrm{RMSNorm}(\mathbf{x})$ for $c > 0$ up to its ϵ, so the variants match up to ϵ and rounding. With three seeds at every k from 2 to 7, no difference is detected under our uncertainty estimate in 22 of 24 comparisons; the exceptions are MATH-500 at k=2 (-1.8; the native-weight seeds span 30.6–36.4) and k=3 (-1.5). At k=1, where native weights have one seed, renormalization is 2.5 points higher on MuSR (23.8 vs 21.3), just beyond noise, and within noise elsewhere. Response lengths differ by at most 3.4%. Qwen3.6 and Laguna-XS add the MoE and shared-expert output to the residual stream unnormalized, so the scale change reaches the next layer.

Table 21: Fixed k with renormalized survivors (R) against native weights (N): accuracy in %. R is the three-seed mean; N is the three-seed mean for $k \geq 2$ and seed 0 at k=1.

Model	k	MATH-500 R / N	GPQA-D R / N	MuSR R / N	MBPP+ R / N
Qwen3.6	1	0.0 / 3.4	0.2 / 3.0	0.3 / 8.1	0.0 / 5.0
	2	0.0 / 89.1	3.7 / 61.6	6.2 / 57.6	0.2 / 74.6
	3	62.1 / 92.2	56.1 / 74.6	48.8 / 63.5	60.6 / 81.0
	4	84.1 / 92.4	74.4 / 78.3	62.7 / 65.0	78.0 / 81.7
	5	89.5 / 92.0	77.6 / 79.3	63.8 / 65.5	79.7 / 81.4
	6	90.8 / 92.1	79.1 / 78.8	64.7 / 65.0	81.0 / 82.0
	7	91.8 / 91.8	80.6 / 79.6	66.6 / 65.2	80.5 / 83.2
Gemma-4	1	0.1 / 0.0	5.9 / 8.6	23.8 / 21.3	0.5 / 0.8
	2	31.4 / 33.2	40.7 / 43.9	58.9 / 57.4	41.1 / 42.3
	3	78.1 / 79.6	66.0 / 66.5	66.8 / 66.1	78.9 / 78.0
	4	90.8 / 90.7	74.4 / 74.2	61.7 / 62.7	82.7 / 82.7
	5	91.6 / 92.4	76.1 / 77.3	66.4 / 67.5	83.0 / 82.3
	6	92.2 / 91.9	78.8 / 78.8	69.1 / 68.3	83.2 / 83.9
	7	92.3 / 92.4	79.8 / 77.8	69.2 / 69.0	83.2 / 83.2
Laguna-XS	1	0.0 / 6.2	3.4 / 22.7	4.6 / 48.0	0.0 / 4.2
	2	0.0 / 87.4	8.9 / 49.7	7.9 / 59.3	0.0 / 75.6
	3	30.0 / 90.9	25.8 / 53.4	44.2 / 63.6	13.5 / 80.2
	4	69.1 / 90.9	32.0 / 51.3	54.5 / 63.1	46.4 / 82.5
	5	90.0 / 90.6	36.9 / 52.5	59.7 / 64.8	81.0 / 81.2
	6	91.5 / 91.4	42.8 / 50.5	61.8 / 64.2	81.1 / 81.3
	7	91.3 / 91.5	45.1 / 46.3	63.7 / 64.7	82.1 / 81.5

G MECHANISM ANALYSES

Teacher-forced traces. Per model and benchmark: whole items, in one stratified order shared by all models, to about 500k tokens, each the served prompt (system and user message in the evaluation chat template) plus the model's seed-0 answer at that budget. We teacher-force them through the Hugging Face backbone with the allocator as served (hard argmax) and record $k_{\ell,i}$ at every token and MoE layer. Figure 3 uses answer tokens, the analyses below all tokens unless stated.

Variance decomposition. The layer-mean, token-mean and residual sums of squares of the token-by-layer matrix k add exactly to the total (Table 22). The chance allocation draws each entry independently from the same per-layer distribution, giving expected token share $T_0 = (1 - \text{layer share})/L$. Since $T/T_0 = 1 + (L-1)\bar{r}$, with $\bar{r}$ the mean correlation between two layers' k on the same token, a ratio of 4 is $\bar{r} \approx 0.08$ (Qwen3.6, Laguna-XS) to 0.10 (Gemma-4). Adjacent layers correlate at 0.09–0.44, layers twenty apart hardly at all.

Table 22: Share of the variance of $k_{\ell,i}$ explained by depth (one schedule for all tokens) and by token (one level per token across layers), with the token share over chance in parentheses. Ranges over the four benchmarks, about 500k tokens each.

	free		t=2	
Model	depth	token (×chance)	depth	token (×chance)
Qwen3.6	1–2%	9–11% (3.7–4.4)	3–6%	18–24% (7.6–10.2)
Gemma-4	20–25%	6.5–7.5% (2.6–2.9)	6–7%	10–11% (3.3–3.6)
Laguna-XS	12–14%	8–10% (3.8–4.3)	7–12%	9–13% (3.9–5.4)

Layer profiles and token maps. Under the dial the per-layer mean k changes shape with the budget rather than scaling one depth schedule, and the layer explains 1–26% of the variation in k at every budget (Figure 5). Figure 6 shows the token-by-layer choices behind these averages on one MATH-500 problem.

Sources of the per-token structure. Of the token share's excess over chance, the prompt/answer split, the drift of k along the answer and item means explain 5–13%; shuffling k among the 16 neighbouring tokens removes 66–91%. Token identity (out of sample: items split by parity, estimates shrunk toward the global mean) explains about a third on Qwen3.6, a fifth to two fifths on Gemma-4 and over half on Laguna-XS. For router confidence we use the nucleus count n_p, the fewest of the top-8 candidates holding a share $p \in \{0.5, 0.7, 0.9\}$ of their renormalized weight, and the entropy of those weights (on Gemma-4 including its per-expert scale). Within a layer, k correlates with $n_{0.9}$ at -0.08 to $+0.02$ on Qwen3.6 and Gemma-4 and at -0.20 to -0.11 on Laguna-XS, so the allocator is not a top-p rule. Position, item, token identity and router confidence together leave about half the excess unexplained on Qwen3.6 and Gemma-4 and a fifth on Laguna-XS.

Does k follow expert demand? On MuSR and MATH-500 (200k tokens each), with all other allocations frozen as served, a random 2% of answer positions at one layer (rotating over eight layers per sequence) get k=8 in one pass and k=1 in another; their demand is the KL divergence between the two next-token distributions. As many unperturbed positions, scored in the same passes, measure leakage through attention. We report the within-layer, tie-aware rank correlation of k with demand, averaged over layers, with bootstrap intervals over items. In free mode it is $+0.09$ to $+0.15$ on every model and benchmark (router confidence: -0.06 to -0.28); at fixed next-token entropy, 0.00 to $+0.04$ on Qwen3.6 and Gemma-4 and $+0.07$ to $+0.10$ on Laguna-XS. The dial raises it to $+0.25$/$+0.29$ on Qwen3.6 at t=2 ($+0.11$/$+0.12$ beyond entropy; free: $+0.09$/$+0.10$) and to $+0.25$ on Laguna-XS MATH-500 (free: $+0.14$); Gemma-4 peaks at t=4. At unperturbed positions, k's correlation with the leaked divergence is 0.63–0.86 of that with demand on five of the six free-mode model–benchmark pairs (0.14 on Laguna-XS MuSR), similar under the dial and for our adaptation of BEAM. We therefore treat these correlations as confounded associations, not measures of aim.

Learned-k baselines. On the twelve checkpoints of our frozen-backbone adaptations of BEAM and Ada-K, Ada-K is a depth schedule: the layer is 86–99% of its variance, its token share at most

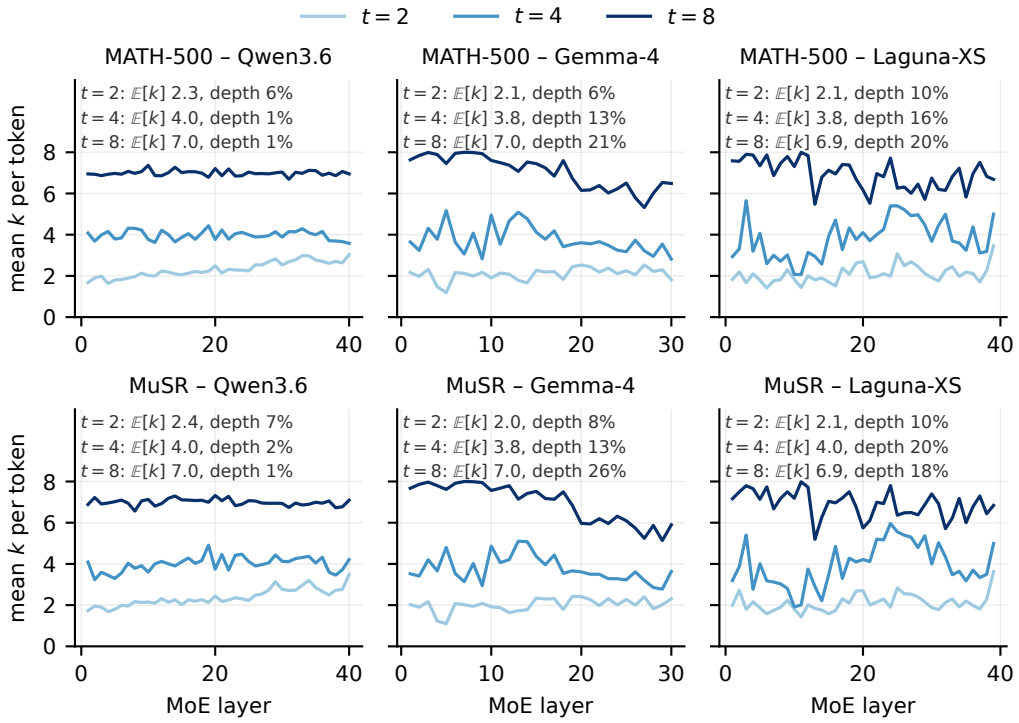

Figure 5: Mean k per MoE layer under the dial (t=2, 4, 8), over answer tokens of teacher-forced traces of each setting's own answers, on MATH-500 and MuSR. Each panel gives $\mathbb{E}[k]$ and the share of the variation in k explained by the layer alone (depth; 100% is a pure depth schedule).

1% (1.0–2.3× chance), its correlation with demand -0.01 to $+0.08$. BEAM learns a per-token level (6–35% of its variance, 4–16× chance) and correlates more with demand than the free-mode allocator ($+0.28$ to $+0.40$ against $+0.09$ to $+0.10$ on Qwen3.6; $+0.15$ to $+0.24$ against $+0.12$ to $+0.15$ on Gemma-4). The allocator matches it only under the dial: Gemma-4 at t=4, and Laguna-XS at t=2, where it exceeds BEAM on MATH-500 ($+0.25$ against $+0.20$).

Token kinds. Relative to the answer average on Qwen3.6 and Laguna-XS, in free mode and at t=2, content words draw 6–11% more experts, LaTeX 13–28% fewer and whitespace 10–17% fewer; at t=2 numbers and operators also draw 8–14% fewer. On Gemma-4 every kind is within 7%. The top tokens are hedges and checks on Qwen3.6 (*sometimes*, *unless*, *assume*, *verify*) and problem terms on Laguna-XS. Ada-K's top-token lists span ±9% with no consistent split by kind.

Replacement controls. Table 4 follows the main protocol but routes in eager PyTorch, since a per-token random draw cannot be captured in a CUDA graph. E's histogram is from A's run on the same benchmark and seed; EH and SH use A's GPQA-D run at the same budget. S's rounding (half up) does not preserve the total (MuSR t=2: $\mathbb{E}[k]$ 2.18 against 2.33 on Qwen3.6, 1.97 against 2.05 on Gemma-4); we ran no budget-balanced rounding, so part of S's cost there may reflect its lower count. D gives each token, per forward pass, a random row of the teacher-forced trace of the allocator's own answers at the same budget, applied at every layer. A is the main evaluation's run (CUDA graphs; fused routing on all three backbones) for the dialled settings and GPQA-D, and an eager rerun for free-mode MuSR; where both exist they are close (rerun first: 59.2/59.6 on Gemma-4, 62.4/63.5 on Laguna-XS, 65.2/66.6 on Qwen3.6). Seed 0 of the dialled E and S ran on a second machine (4×H200 NVL), seeds 1–2 on the main H200 node, with close per-seed accuracies (e.g. Qwen3.6 E at t=2: 58.5, 57.7, 55.8; Gemma-4 S: 56.6, 54.8, 55.6). Energy is compared only within one machine and execution mode. Intervals scale the noise bands of Table 11 (pooled seed spreads; comparable seed variance and fixed $\mathbb{E}[k]$ assumed) by $\sqrt{(1/n_A + 1/n_X) \cdot 3/2}$ for n_A and n_X runs of A and the control.

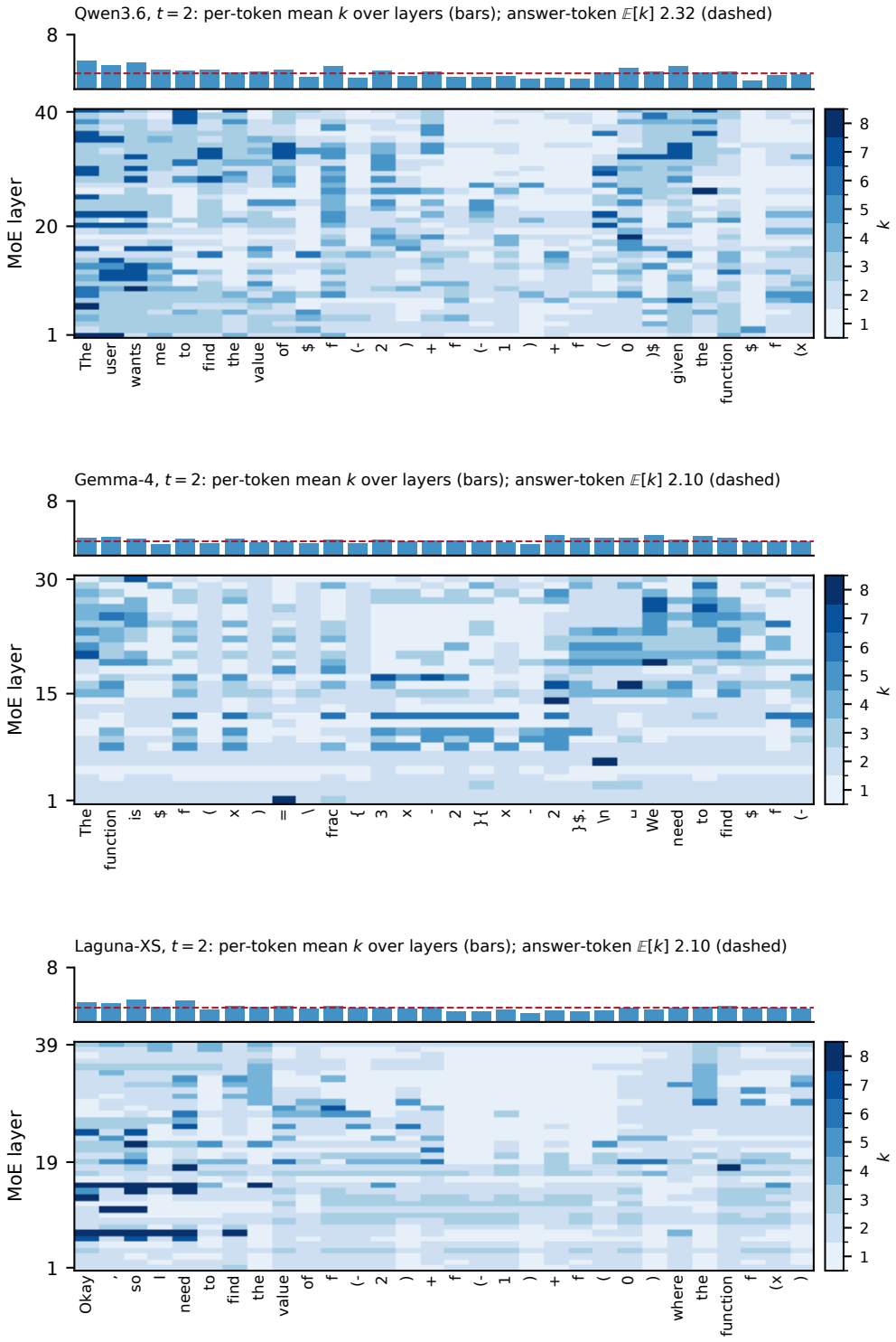

Figure 6: The allocator's k at each token (column) and MoE layer (row) at t=2, for the first 30 answer tokens of the same MATH-500 problem (item 002) on each backbone; the span was fixed by rule before looking at k. Bars: each token's mean k over layers; dashed line: the trace's answer-token $\mathbb{E}[k]$.