LESS CAN BE MORE: LEARNING PRUNED TOP-K IN SMOE-BASED LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse Mixture of Experts (SMoEs) typically retain the fixed Top-K expert budget chosen during pretraining when deployed on downstream tasks. But is this computation budget always necessary or even optimal? Through an oracle analysis across nine reasoning and non-reasoning SMoE models, we find substantial redundancy in expert activation: smaller budgets can preserve correct predictions, while some inputs that fail under the native Top-K become solvable when fewer experts are activated. Overall, the oracle reduces the average expert budget by **40.8%** while yielding a **26.8% relative accuracy improvement**. Motivated by these observations, we introduce **LEPO (Learning Expert-Pruning Policies)**, which adapts both *which experts to favor* and *how many experts to execute* while keeping pretrained model parameters frozen. LEPO combines lightweight expert-preference adaptation with grouped expert-budget policies and learns from the available context through guidance from the original model, without requiring task labels or oracle budget annotations. Across test-time adaptation, offline budget learning, and permanent expert pruning, LEPO consistently improves the accuracy and efficiency trade-off over fixed and adaptive expert-allocation baselines. In the settings, LEPO uses only **75–87%** of the native routed-expert budget while largely preserving or improving accuracy. Our results suggest that downstream Top-K should be treated as an adaptive computation-allocation problem rather than a fixed consequence of pretraining.

1 INTRODUCTION

Sparse Mixture of Experts (SMoE) models have achieved substantial success across natural language processing (NLP) and visual representation learning tasks (Du et al., 2022; Fedus et al., 2022b; Riquelme et al., 2021; Shen et al., 2023; DeepSeek-AI et al., 2026; Team et al., 2026). These advances build upon the Transformer architecture (Vaswani et al., 2017) and its variants (Child et al., 2019; Dai et al., 2019), enabling model capacity to scale while activating only a subset of parameters for each token. Although SMoE models are substantially more computationally efficient than equivalently sized dense models(Wang et al., 2024a), their inference costs remain considerable, particularly as model scale, context length, and the number of activated experts increase. Consequently, improving the inference efficiency of MoE models while preserving their capabilities remains an important open problem (Jiang et al., 2025).

Dynamic Top-K expert activation in SMoE models has been explored in prior work Yue et al. (2025); Zeng et al. (2024). However, these approaches typically introduce a trade-off between computational efficiency and model performance. As a result, most existing MoE models still follow the conventional paradigm Chi et al. (2022); Do et al. (2023); Jiang et al. (2024a); Do et al. (2025), using a fixed number of activated experts during both training and inference. In this work, we revisit this design choice and ask a fundamental question: ***Is activating all Top-K experts necessary for solving a task at test time?***

To answer this question, we conduct an oracle analysis, illustrated in Figure 1, to quantify the potential for reducing the expert budget at test time. Across nine advanced SMoE models, including both reasoning and non-reasoning models, on three MMLU-Pro subjects (Wang et al., 2024b), we search for the smallest Top-K budget that still yields a correct answer. Surprisingly, this oracle, termed *SearchTopK*, reduces the number of activated experts by **40.8%** on average while achieving a **26.8% relative improvement in accuracy** over the native Top-K configuration. These results re-

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.

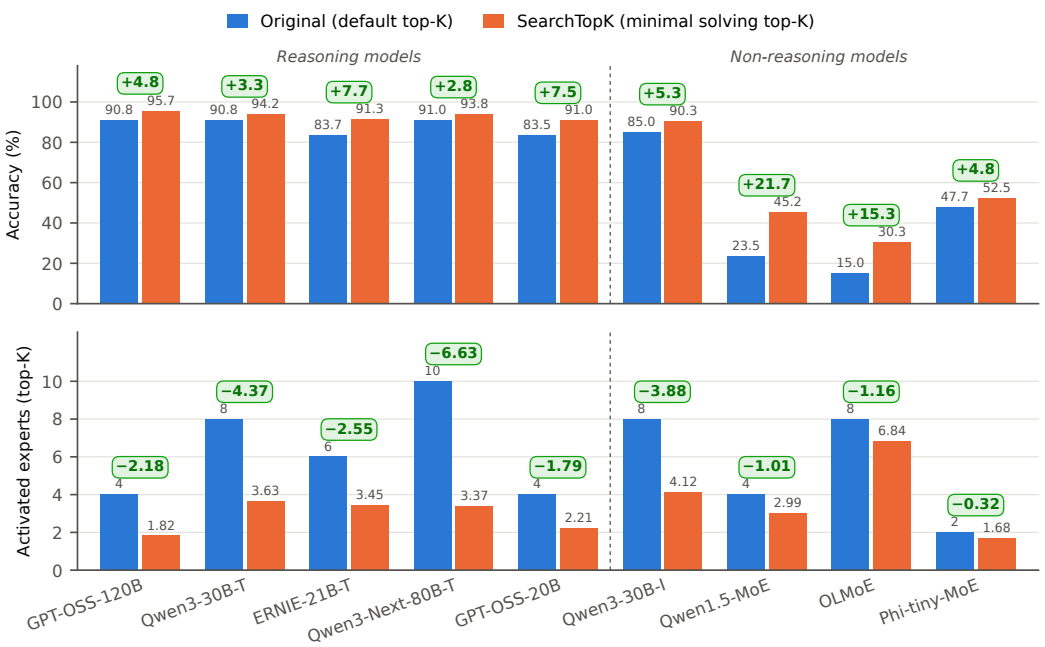


Figure 1: **Potential accuracy gains and expert-budget reductions across nine MoE models over MMLU-Pro .** Blue bars use the native Top-K; orange bars show *SearchTopK*, an oracle that selects the smallest tested k producing a correct answer for each question. Across five reasoning and four non-reasoning models, oracle accuracy increases by 26.8% (relative) and the question-averaged expert budget decreases by 40.8%.

veal substantial redundancy in the default expert budget and motivate adaptive, instance-dependent expert allocation at test time.

These observations reveal two key opportunities. First, many questions can be solved with fewer experts, suggesting substantial redundancy in the native computation budget. Second, the higher oracle accuracy indicates that some questions answered incorrectly with the native K become correct when fewer experts are activated. Thus, activating more experts does not necessarily lead to better predictions under the evaluated decoding setting. However, identifying the minimum sufficient Top-K at test time is non-trivial: exhaustive search is computationally expensive, and no direct supervision is available for the optimal expert budget.

In this work, we introduce **LEPO (Learning Expert-Pruning Policies)**, a test-time learning framework for identifying a minimum sufficient expert budget. LEPO freezes all pretrained model parameters and augments each router with a lightweight expert-preference vector, together with a compact budget policy shared across groups of layers as Figure 2b. To overcome the absence of budget labels at test time, the original model operating with its native Top-K serves as a frozen teacher. LEPO adapts to each input using self-supervised signals from the available context, including next-token prediction, output-distribution matching, and hidden-representation alignment, while learning the expert-budget policy with Group Relative Policy Optimization (GRPO) Shao et al. (2024). By alternating between expert-preference adaptation and budget-policy optimization, LEPO learns to preserve the behavior of the full-budget model while progressively reducing unnecessary expert computation. Importantly, LEPO operates entirely at test time using only the input prompt and generated context, requiring no external training data, task labels, or oracle budget annotations.

We evaluate LEPO across three complementary settings: request-local test-time adaptation, offline expert-budget learning, and calibration-based routing adaptation with permanent expert pruning. Across these settings and advanced SMoE models, LEPO consistently reduces expert computation while preserving or improving model performance, outperforming static Top-k allocation and competitive adaptive-routing and pruning baselines. In the request-local setting, LEPO retains only 75–87% of the native expert budget while matching or exceeding the original model's average accuracy on nearly all evaluated models. These results demonstrate that learned, input-dependent expert

pruning can recover a substantial portion of the efficiency suggested by our oracle analysis without requiring task labels or oracle budget supervision.

Our contributions are threefold. First, we reveal substantial test-time redundancy in native Top-K expert activation through an oracle analysis across nine reasoning and non-reasoning SMoE models, showing that reduced expert budgets can not only preserve correct predictions but also recover correct solutions for inputs that fail at the native budget. Second, we introduce **LEPO**, a lightweight framework that combines expert-preference adaptation, grouped budget policies, and full-budget teacher guidance to learn adaptive expert pruning without labels. Third, we demonstrate the generality of LEPO across request-local adaptation, offline budget learning, and permanent expert pruning, consistently improving the accuracy and efficiency trade-off over fixed and adaptive expert-allocation baselines.

2 Related Work

Sparse Mixture of Experts (SMoE). Sparse Mixture of Experts (SMoE), an extension of the Mixture-of-Experts framework (Jacobs et al., 1991; Jordan & Jacobs, 1994), has become a prominent architecture for scaling large language models and has also been widely adopted in domains such as computer vision and speech recognition (Zhou et al., 2022b; Riquelme et al., 2021). Beyond conventional fixed Top-K routing, prior work has explored more flexible expert allocation. Expert Choice assigns tokens to experts under fixed expert capacities, allowing each token to be processed by a variable number of experts (Zhou et al., 2022a), while AdaMoE introduces null experts to dynamically adjust the number of computationally active experts (Zeng et al., 2024). Most SMoE research, however, has focused on improving training, routing, and post-training effectiveness (Dai et al., 2022; Do et al., 2026b; Muennighoff et al., 2024; Do et al., 2026a), leaving efficient test-time adaptation for SMoE comparatively underexplored. Recent methods such as C3PO (Li et al., 2025) and Rewiring-MoE (Su et al., 2025) extend MoE adaptation to fast or test-time settings, but primarily target predictive performance rather than reducing inference-time expert computation. In contrast, our work focuses on learning adaptive expert budgets at test time to jointly preserve model capability and improve inference efficiency. Further discussion of related work can be found in Section A.1.

3 Method

3.1 Preliminaries: Sparse Mixture of Experts

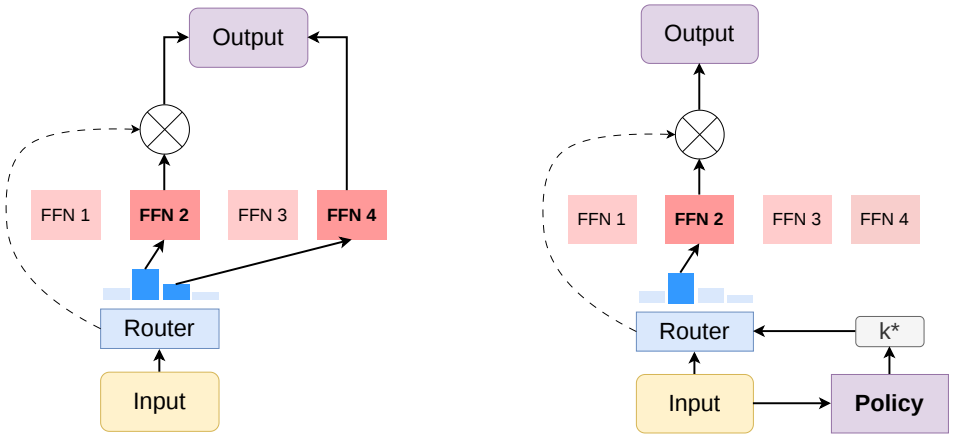
A Sparse Mixture of Experts (SMoE) layer replaces the dense feed-forward block of a Transformer with N_l experts, of which only a few execute for each token (Shazeer et al., 2017; Fedus et al., 2022a). Consider a model with L routed layers. For token t with input $h_{l,t} \in \mathbb{R}^{d_l}$ to routed layer l, the router $g_l(h) = W_l h$ with $W_l \in \mathbb{R}^{N_l \times d_l}$ gives logits $z_{l,t} = g_l(h_{l,t})$ and probabilities $p_{l,t} = \mathrm{softmax}(z_{l,t})$. The layer executes the K_l highest-scoring experts, $\mathcal{E}_{l,t} = \mathrm{TopK}(z_{l,t}, K_l)$, and outputs

$$o_{l,t} = \sum_{j \in \mathcal{E}_{l,t}} w_{l,t,j}\, E_{l,j}(h_{l,t}), \tag{1}$$

where each expert $E_{l,j} : \mathbb{R}^{d_l} \to \mathbb{R}^{d_l}$ is a feed-forward network and $w_{l,t,j}$ is the router probability $p_{l,t,j}$, renormalized over $\mathcal{E}_{l,t}$ in some models.

3.2 Learning Expert-Pruning Policies (LEPO)

Unlike conventional SMoE as Figure 2, which keeps the number of active experts fixed at the native K, LEPO jointly adapts *which experts to favor* and *how many experts to execute*. Specifically, a trainable preference vector adjusts expert rankings at each routed layer, while a lightweight policy selects the expert budget shared within each layer group. The chosen budget k^* is the policy action from Equation equation 4, conditioned on router statistics in Equation equation 3, and the adjusted ranking follows Equation equation 2. Both components are adapted using only the current context and guidance from a frozen full-Top-K teacher, requiring no external training data, task labels, or oracle budget annotations.



(a) Conventional MoE: fixed Top-K.

(b) LEPO: policy-selected Top-k^*.

Figure 2: **Conventional MoE routing and LEPO's adaptive expert execution.** (a) The pretrained router activates a fixed number of experts for each token. (b) LEPO's policy chooses the active-expert budget.

3.2.1 Lightweight routing adaptation

Let $M_T = M_{\theta,K}$ denote the pretrained teacher, with L routed layers, N_l experts at layer l, and native budget K_l. The student uses the same pretrained parameters θ, all of which remain frozen, including the original routers, attention blocks, experts, shared experts, and language-model head. The only trainable parameters are expert-preference vectors $\phi = \{r_l\}_{l=1}^{L}$ and budget-policy parameters ψ. Teacher and student can share the frozen weights: a teacher pass disables the routing adaptations and restores the native budgets.

For token t, let $h_{l,t}^S$ be the student's input to routed layer l, and let g_l be its original router. LEPO modifies the routing logits as

$$z_{l,t}^S = g_l(h_{l,t}^S) + \widetilde{r}_l, \qquad \widetilde{r}_l = r_l - \frac{\mathbf{1}^\top r_l}{N_l}\mathbf{1}, \qquad r_l \in \mathbb{R}^{N_l}. \tag{2}$$

For a linear router, $g_l(h) = W_l h$, the preference vector shifts expert scores for the current request while preserving the frozen router's token-dependent behavior. Mean centering removes the common-logit component, which does not affect softmax probabilities or expert ranking. This lightweight adapter adds only $\sum_l N_l$ parameters, compared with $\sum_l N_l d_l$ for learning an additional router matrix. We initialize $r_l = 0$, so the student initially matches the teacher under the native expert budget up to numerical precision. After adaptation, teacher agreement is encouraged through the training objective rather than guaranteed.

3.2.2 Grouped adaptive expert budgets

We partition the routed layers into B consecutive, nonempty groups $\{\mathcal{G}_b\}_{b=1}^{B}$. One budget decision per token is shared across all layers in a group. Let $a_b = \min \mathcal{G}_b$ and $e_b = \max \mathcal{G}_b$ denote its first and last routed layers. For clarity, we describe groups whose layers share a native budget K_b; layers with different native budgets can be placed in separate groups. Our default uses four approximately equal groups and $k_{\min,b} = \max(1, \lfloor K_b/2 \rfloor)$, with action set $\mathcal{A}_b = \{k_{\min,b}, \ldots, K_b\}$. Per-layer control is recovered by setting $B = L$.

At the first layer of group b, we compute $p_{a_b,t}^S = \mathrm{softmax}(z_{a_b,t}^S)$ and sort its entries as $p_{(1)} \geq \cdots \geq p_{(N_{a_b})}$. The policy receives the compact state

$$s_{b,t} = \left[p_{(1)}, \ldots, p_{(K_b)},\ H(p),\ p_{(1)} - p_{(2)},\ p_{(K_b)} - p_{(K_b+1)},\ \sum_{j=1}^{K_b} p_{(j)} \right], \tag{3}$$

where $H(p) = -\sum_j p_j \log p_j$ and unavailable entries beyond the expert count are set to zero. These statistics describe routing concentration and the separation between selected and unselected experts. A linear categorical head predicts

$$\pi_{\psi_b}(\cdot \mid s_{b,t}) = \mathrm{softmax}(U_b s_{b,t} + v_b), \qquad k_{b,t} \sim \pi_{\psi_b}(\cdot \mid s_{b,t}), \qquad k_{l,t} = k_{b,t} \quad (l \in \mathcal{G}_b). \tag{4}$$

Let $\mathcal{E}^S_{l,t} = \mathrm{TopK}(z^S_{l,t}, k_{l,t})$. The routed contribution is

$$o^S_{l,t} = \sum_{j \in \mathcal{E}^S_{l,t}} w^S_{l,t,j} E_{l,j}(h^S_{l,t}), \tag{5}$$

where $w^S_{l,t,j}$ follows the model's native weighting, normalization, and scaling conventions on the retained experts. Shared-expert contributions and native dispatch rules are preserved. Only retained experts execute, so decreasing $k_{l,t}$ reduces actual routed-expert work.

3.2.3 Context adaptation with teacher fidelity

For a current context $x = (x_1, \ldots, x_T)$ with $T \geq 2$, teacher and student process exactly the same causal prefixes. All teacher targets are detached. LEPO combines three signals that serve complementary roles.

Context next-token prediction. The observed context supplies its own next-token targets:

$$\mathcal{L}_{\mathrm{ctx}} = -\frac{1}{T-1} \sum_{t=1}^{T-1} \log P_S(x_{t+1} \mid x_{\leq t}). \tag{6}$$

This term adapts the student to the available sequence without accessing a reference answer.

Output-distribution preservation. Let u^T_t and u^S_t be teacher and student vocabulary logits, distinct from the expert-routing logits in Equation equation 2. With temperature $\tau > 0$, define $q^T_t = \mathrm{softmax}(u^T_t/\tau)$ and $q^S_t = \mathrm{softmax}(u^S_t/\tau)$. We match the full predictive distributions using

$$\mathcal{L}_{\mathrm{KD}} = \frac{\tau^2}{T} \sum_{t=1}^{T} D_{\mathrm{KL}}(q^T_t \| q^S_t). \tag{7}$$

Unlike context prediction, this loss preserves probability mass across alternative continuations. It also provides supervision at the final context position, where an observed next token is unavailable.

Hidden-representation preservation. Let $v^T_{e_b,t}$ and $v^S_{e_b,t}$ denote aligned hidden states after group b. We apply parameter-free layer normalization LN_0 and minimize

$$\mathcal{L}_{\mathrm{hid}} = \frac{1}{BT} \sum_{b=1}^{B} \sum_{t=1}^{T} \frac{\| \mathrm{LN}_0(v^T_{e_b,t}) - \mathrm{LN}_0(v^S_{e_b,t}) \|_2^2}{d_{e_b}}. \tag{8}$$

Matching group boundaries provides a representation target after each budget-controlled region. Averaging over hidden dimensions keeps the loss scale comparable across model widths.

For fixed budget decisions, the router vectors minimize

$$\mathcal{L}_r = w_c \mathcal{L}_{\mathrm{ctx}} + w_o \mathcal{L}_{\mathrm{KD}} + w_h \mathcal{L}_{\mathrm{hid}} + \frac{w_r}{L} \sum_{l=1}^{L} \|\widetilde{r}_l\|_2^2, \tag{9}$$

with nonnegative weights. The anchor discourages unnecessary departures from the pretrained routing. Gradients update only ϕ, through the selected experts' routing weights and the frozen computation graph; discrete budget actions and Top-k indices are held fixed within each backward pass. Expert selections are recomputed on subsequent forwards.

3.2.4 LEARNING BUDGETS WITH GRPO

Let c_l approximate the execution cost of one routed expert at layer l. For a budget trajectory, the normalized routed-expert cost and savings are

$$C = \frac{\sum_{t=1}^{T}\sum_{l=1}^{L} c_l k_{l,t}}{T\sum_{l=1}^{L} c_l K_l}, \qquad S = 1 - C. \tag{10}$$

Setting $c_l = 1$ measures the fraction of native expert-token assignments. The controller seeks greater savings while maintaining agreement with the teacher, which can be expressed as

$$\max_{\psi} \mathbb{E}[S] \quad \text{subject to} \quad \mathbb{E}[\mathcal{L}_{\mathrm{KD}}] \le \epsilon_o, \quad \mathbb{E}[\mathcal{L}_{\mathrm{hid}}] \le \epsilon_h. \tag{11}$$

We optimize a penalized reward that also includes context prediction. Write $\ell_{\mathrm{ctx},t} = -\log P_S(x_{t+1} \mid x_{\le t})$ for $t < T$ and $\ell_{\mathrm{ctx},T} = 0$, and let $\ell_{\mathrm{KD},t}$ and $\ell_{\mathrm{hid},t}$ be the per-position terms whose averages give Equations equation 7 and equation 8. Then

$$R_t = \left(1 - \frac{\sum_l c_l k_{l,t}}{\sum_l c_l K_l}\right) - \lambda_c \ell_{\mathrm{ctx},t} - \lambda_o \ell_{\mathrm{KD},t} - \lambda_h \ell_{\mathrm{hid},t}, \qquad R = \frac{1}{T}\sum_t R_t. \tag{12}$$

Consequently, the context penalty in R is $\lambda_c(T-1)\mathcal{L}_{\mathrm{ctx}}/T$, reflecting the absence of a next-token label at the last position. The teacher penalties measure fidelity on the available context; satisfying them does not establish a downstream accuracy guarantee.

With router vectors fixed, we sample G student budget trajectories on the same context from a frozen behavior policy π_{old}. Teacher targets are computed once and reused. For rollout g, our implementation assigns credit at each token by normalizing its reward across rollouts:

$$\mu_t = \frac{1}{G}\sum_g R_{g,t}, \qquad \sigma_t^2 = \frac{1}{G}\sum_g (R_{g,t} - \mu_t)^2, \qquad A_{g,t} = \mathrm{stopgrad}\left(\frac{R_{g,t} - \mu_t}{\max(\sigma_t, \delta)}\right). \tag{13}$$

Here $\delta > 0$ ensures numerical stability. All layer-group decisions at that position receive the same advantage. This supplies local credit without fitting a value model. A sequence-level variant instead normalizes R_g across rollouts and broadcasts a single advantage to all decisions; it is an alternative credit-assignment choice.

For a stored action $k_{g,b,t}$ and detached state $s_{g,b,t}$, define

$$\rho_{g,b,t}(\psi) = \frac{\pi_{\psi_b}(k_{g,b,t} \mid s_{g,b,t})}{\pi_{\mathrm{old},b}(k_{g,b,t} \mid s_{g,b,t})}. \tag{14}$$

We maximize the clipped group-relative policy optimization (GRPO) objective

$$\begin{aligned}\mathcal{J}(\psi) = \frac{1}{GBT}\sum_{g,b,t}\Big[&\min(\rho_{g,b,t}A_{g,t}, \mathrm{clip}(\rho_{g,b,t}, 1-\varepsilon, 1+\varepsilon)A_{g,t}) \\ &+ \alpha H(\pi_{\psi_b}(\cdot \mid s_{g,b,t})) - \beta D_{\mathrm{KL}}(\pi_{\psi_b}(\cdot \mid s_{g,b,t}) \| \pi_{\mathrm{init},b}(\cdot \mid s_{g,b,t}))\Big].\end{aligned} \tag{15}$$

Entropy and reference-policy regularization are optional ($\alpha, \beta \ge 0$). GRPO updates only the budget policy; expert preferences are learned through Equation equation 9. Fidelity multipliers can be fixed or, when explicit tolerances are supplied, updated as dual variables:

$$\lambda_j \leftarrow \Big[\lambda_j + \eta_j(\widehat{\mathcal{L}}_j - \epsilon_j)\Big]_+, \qquad j \in \{o, h\}, \tag{16}$$

where $\mathcal{L}_o = \mathcal{L}_{\mathrm{KD}}$, $\mathcal{L}_h = \mathcal{L}_{\mathrm{hid}}$, and hats denote rollout averages. Fixed penalties are the default; dual updates implement the optional fidelity-constrained variant.

4 EXPERIMENTS

We test LEPO in three settings: Section 4.1 studies test-time adaptation on generative tasks, following Rewiring (Su et al., 2025). Section 4.2 studies adaptive expert budgets, following Ada-K routing (Yue et al., 2025). Section 4.3 studies expert pruning, following STEP (Zong et al., 2026). In every setting, we report accuracy together with the number of routed experts that each method activates.

Table 1: **Generative tasks under the test-time adaptation protocol of Su et al. (2025).** Accuracy (%; pass@1 for HumanEval and MBPP) with the average number of activated routed experts per token in grey. $\bar{K}$ averages the five tasks; the percentage is relative to the native Top-K. ICL and C3PO use 3/5 demonstrations or 100 labelled reference samples per task; all other rows are zero-shot and use no external data. Best result per model in **bold**.

Method	Setting	HumanEval	MBPP	GSM8K	MATH500	MMLU	Avg	$\bar{K}$ (% of K)
DeepSeek-V2-Lite (64 routed experts, $K = 6$)								
Original	0-shot	53.66 6	**58.88** 6	70.43 6	26.20 6	51.97 6	52.23	6.00 (100%)
ICL	3-shot	53.25 6	49.94 6	**71.72** 6	25.80 6	51.17 6	50.38	6.00 (100%)
ICL	5-shot	50.81 6	54.73 6	70.74 6	**28.20** 6	52.07 6	51.31	6.00 (100%)
C3PO	100-ref	53.46 6	57.33 6	68.31 6	27.60 6	50.90 6	51.52	6.00 (100%)
Rewiring	0-shot	50.81 6	58.63 6	69.07 6	26.00 6	51.47 6	51.20	6.00 (100%)
Ada-K	0-shot	43.70 2.68	52.53 2.52	61.56 2.62	19.00 2.44	48.47 2.59	45.05	2.57 (43%)
LEPO (ours)	0-shot	**55.49** 4.51	57.46 4.90	70.73 4.52	27.86 4.54	**53.23** 4.89	**52.95**	4.67 (78%)
OLMoE-1B-7B (64 routed experts, $K = 8$)								
Original	0-shot	33.94 8	**42.80** 8	42.00 8	**12.60** 8	42.87 8	**34.84**	8.00 (100%)
ICL	3-shot	27.24 8	**42.80** 8	45.34 8	11.20 8	46.90 8	34.70	8.00 (100%)
ICL	5-shot	26.83 8	41.25 8	**45.41** 8	10.60 8	**48.90** 8	34.60	8.00 (100%)
C3PO	100-ref	**34.76** 8	41.12 8	41.24 8	10.00 8	41.70 8	33.76	8.00 (100%)
Rewiring	0-shot	31.10 8	40.99 8	40.86 8	9.40 8	42.33 8	32.94	8.00 (100%)
Ada-K	0-shot	29.27 5.70	34.11 5.74	37.45 5.86	7.40 5.83	40.67 5.88	29.78	5.80 (73%)
LEPO (ours)	0-shot	33.82 6.85	40.59 6.26	43.59 7.68	11.18 7.09	43.91 6.94	34.62	6.97 (87%)
Qwen1.5-MoE-A2.7B (60 routed experts, $K = 4$)								
Original	0-shot	45.93 4	43.32 4	57.16 4	21.40 4	52.20 4	44.00	4.00 (100%)
ICL	3-shot	45.33 4	**48.12** 4	59.74 4	**23.20** 4	54.90 4	46.26	4.00 (100%)
ICL	5-shot	47.15 4	44.10 4	**60.96** 4	21.20 4	**55.77** 4	45.84	4.00 (100%)
C3PO	100-ref	43.09 4	43.45 4	56.56 4	22.80 4	51.20 4	43.42	4.00 (100%)
Rewiring	0-shot	41.06 4	44.88 4	56.79 4	22.00 4	50.37 4	43.02	4.00 (100%)
Ada-K	0-shot	44.11 3.08	44.62 3.10	55.12 3.12	14.20 3.11	50.63 3.11	41.74	3.10 (78%)
LEPO (ours)	0-shot	**53.25** 3.18	45.14 3.09	60.71 3.40	20.40 2.66	53.69 2.59	**46.64**	2.98 (75%)

4.1 Data-free test-time adaptation on generative tasks

RQ1. *Without labels, demonstrations or reference data, can LEPO match or exceed the original Top-K model and existing test-time adaptation methods on generative tasks, while activating fewer experts than the pretraining budget?*

Results. Table 1 gives the results. *(i) LEPO matches the original model at a smaller expert budget.* On DeepSeek-V2-Lite, LEPO averages 52.95 against 52.23 for the original model ($+0.72$) with $\bar{K} = 4.67$ of 6 (78%). On Qwen1.5-MoE, it averages 46.64 against 44.00 ($+2.64$) with 2.98 of 4 experts (75%). Most of the Qwen gain is on HumanEval (53.25 vs. 45.93). On OLMoE, LEPO is within 0.22 points of the original model (34.62 vs. 34.84) and uses 87% of $K = 8$. *(ii) LEPO is the strongest zero-shot method.* Its average is above Rewiring on all three models ($+1.75$, $+1.68$, $+3.62$), even though Rewiring runs at the full budget. It is also above C3PO ($+1.43$, $+0.86$, $+3.22$), which uses 100 labelled references. On two of the three models, LEPO has the best average of any method, including 3- and 5-shot ICL. ICL remains stronger on single tasks, mostly GSM8K and MMLU on OLMoE and Qwen. Demonstrations are orthogonal to LEPO and could be combined with it, as Su et al. (2025) do for Rewiring. *(iii) Removing experts is not enough on its own.* Ada-K also lowers the budget, to 43–78% of K, but loses 2.3–7.2 points of average accuracy against the original model. LEPO loses at most 0.22 points in this setting. *(iv) The learned budget depends on the task.* Within one model, LEPO chooses different budgets for different tasks. On OLMoE it keeps more experts for GSM8K ($\bar{K} = 7.68$) than for MBPP (6.26). On Qwen1.5-MoE it keeps 3.40 experts for GSM8K and 2.59 for MMLU.

Table 2: **Comparison with Ada-K routing under its evaluation protocol (Yue et al., 2025).** Leaderboard tasks with the Open LLM Leaderboard shot counts (ARC-C 25, HellaSwag 10, MMLU 5, GSM8K 5, TruthfulQA 0, WinoGrande 5). Act is the average number of activated routed experts per token. Δ_{line} is the average accuracy minus the static Top-k line linearly interpolated at the same Act (positive: above the line); it is undefined for DeepSeek-MoE-16B, where no static row lies below Act 3. **Bold**: better of the two adaptive methods.

Model	Method	ARC-C	HellaSwag	MMLU	GSM8K	TruthfulQA	WinoGrande	Avg	Act	Δ_{line}
Mixtral-8x22B ($K = 2$ of 8)	Top-k ($k = 1$)	65.78	86.25	72.01	65.50	48.86	80.27	69.78	1.00	–
	Top-k ($k = 2$)	70.90	88.44	77.43	78.62	51.05	84.53	75.16	2.00	–
	Ada-K	67.15	86.52	72.50	68.39	49.21	81.85	70.93	1.19	+0.13
	LEPO (ours)	**68.26**	**87.83**	**75.76**	**74.30**	**51.14**	**83.58**	**73.48**	1.38	**+1.66**
Mixtral-8x7B ($K = 2$ of 8)	Top-k ($k = 1$)	60.32	82.52	64.58	42.53	45.87	75.30	61.85	1.00	–
	Top-k ($k = 2$)	66.13	86.17	70.30	58.91	46.86	82.79	68.53	2.00	–
	Ada-K	63.14	83.32	66.35	46.17	**46.10**	77.66	63.79	1.27	+0.14
	LEPO (ours)	**65.27**	**85.36**	**68.69**	**57.39**	45.53	**80.74**	**67.16**	1.43	**+2.44**
Qwen1.5-MoE-A2.7B ($K = 4$ of 60)	Top-k ($k = 2$)	51.96	78.36	60.06	54.36	46.76	71.74	60.54	2.00	–
	Top-k ($k = 3$)	54.10	79.02	60.45	59.67	49.29	72.30	62.47	3.00	–
	Top-k ($k = 4$)	55.12	79.38	61.15	60.58	50.08	72.85	63.19	4.00	–
	Ada-K	53.41	78.85	60.35	**59.06**	47.60	71.27	61.76	2.62	+0.02
	LEPO (ours)	**53.50**	**78.92**	**60.85**	58.61	**49.16**	**72.45**	**62.25**	2.70	**+0.36**
DeepSeek-MoE-16B ($K = 6$ of 64)	Top-k ($k = 3$)	51.37	78.49	40.26	11.75	34.84	73.09	48.30	3.00	–
	Top-k ($k = 4$)	52.39	79.59	42.68	14.78	34.92	73.80	49.69	4.00	–
	Top-k ($k = 5$)	52.47	79.62	44.36	14.56	35.61	74.35	50.16	5.00	–
	Top-k ($k = 6$)	52.82	79.99	44.72	16.60	36.09	75.30	50.92	6.00	–
	Ada-K	49.32	77.64	38.40	12.51	33.82	**72.14**	47.31	2.59	–
	LEPO (ours)	**50.94**	**78.01**	**39.75**	**12.96**	**34.57**	71.98	**48.04**	2.65	–

Answer to RQ1. Yes on DeepSeek-V2-Lite and Qwen1.5-MoE; on OLMoE it is within 0.22 points of the original model. LEPO reaches or exceeds the accuracy of the original Top-K model with 75–87% of its routed experts.

4.2 Adaptive expert budgets at matched compute

RQ2. *At a matched average number of activated experts, does LEPO allocate experts better than a learned adaptive-k router (Ada-K) and than simply lowering k uniformly (static Top-k)?*

Results. Table 2 gives the results. *(i) LEPO beats Ada-K on all four models.* The average gains are $+2.55$ (Mixtral-8x22B), $+3.37$ (Mixtral-8x7B), $+0.49$ (Qwen1.5-MoE), and $+0.73$ (DeepSeek-MoE-16B). LEPO also wins 21 of the 24 individual task columns. LEPO does activate slightly more experts than Ada-K, between 0.06 and 0.19 more per token. We therefore also compare both methods with the static line. *(ii) Ada-K sits on the static line, and LEPO lies above it.* Ada-K's gap to the line is only $+0.02$ to $+0.14$ points. Its accuracy is therefore what a uniform Top-k would give at the same compute. LEPO lies above the line on all three models where the line is defined: $+1.66$ on Mixtral-8x22B, $+2.44$ on Mixtral-8x7B, and $+0.36$ on Qwen1.5-MoE. On both Mixtral models, the gain over Ada-K is largest on GSM8K, the only generative task. On Mixtral-8x22B, LEPO reaches 74.30 with 1.38 experts; Ada-K reaches 68.39, and static Top-2 reaches 78.62 with two experts. On Mixtral-8x7B, LEPO keeps 98% of the Top-2 average (67.16 vs. 68.53) with 72% of its routed experts. *(iii) Aggressive budgets are harder on DeepSeek-MoE-16B.* At 44% of K, both adaptive methods fall below static Top-3 (48.30 at Act 3.00). LEPO narrows that gap to 0.26 points while using 0.35 fewer experts per token. Ada-K is 0.99 points below Top-3.

Answer to RQ2. Yes. At nearly the same Act, LEPO beats Ada-K on every model. Unlike Ada-K, whose accuracy equals the static Top-k line at its Act, LEPO lies above that line on every model where the line is defined.

Table 3: **Expert pruning under the STEP protocol** on OLMoE-1B-7B, keeping 48 of 64 experts per layer (75%). Zero-shot accuracy (%) on eight tasks with lm-evaluation-harness. Baseline rows are from the STEP paper. Among pruned models, the best result is in **bold** and the second best is underlined.

Method	Experts	Kept	ARC-E	ARC-C	BoolQ	PIQA	Wino	HellaSwag	MMLU	OBQA	Avg
Original	64	100%	77.23	46.84	70.15	78.62	69.06	58.44	53.54	44.20	62.26
D2-MoE	48	75%	71.21	37.46	66.67	75.35	66.85	49.63	28.39	41.20	54.60
MC-SMoE	48	75%	70.12	38.91	52.26	66.32	61.96	41.47	37.83	35.90	50.60
HC-SMoE	48	75%	71.25	39.59	65.78	72.42	66.22	51.69	42.65	29.40	54.88
Frequency	48	75%	62.37	34.04	62.94	75.89	67.32	53.11	31.13	41.00	53.48
Gate Score	48	75%	54.88	28.58	62.11	73.12	63.30	50.22	24.50	39.90	49.58
MoNE	48	75%	62.37	33.28	65.84	77.09	67.32	54.88	23.27	42.00	53.26
STEP	48	75%	73.82	44.03	**68.35**	77.53	65.51	55.24	39.70	43.20	58.42
LEPO (ours)	48	75%	**73.95**	**44.50**	67.10	**78.27**	**67.84**	**55.30**	**43.37**	**44.50**	**59.35**

4.3 Complementarity with expert pruning

RQ3. *When some experts are removed permanently, does LEPO's routing adaptation preserve more of the original model's accuracy than dedicated pruning and merging methods?*

Results. Table 3 gives the results. LEPO is the strongest 75% model, with an average of 59.35 against 58.42 for STEP, the best baseline ($+0.93$). It retains 95.3% of the original average, against 93.8% for STEP, and closes 24% of STEP's remaining gap to the unpruned model. LEPO is best or tied for best on seven of the eight tasks. The largest gain is on MMLU, where it beats STEP by $+3.67$ (43.37 vs. 39.70) and HC-SMoE by $+0.72$. MMLU is also the task on which every pruning method loses the most accuracy. On OpenBookQA, the pruned LEPO model is slightly above the unpruned model (44.50 vs. 44.20). The only task where LEPO is not best is BoolQ, where it is 1.25 points below STEP.

Answer to RQ3. Yes. LEPO's learned expert preferences combine well with permanent pruning. Under the same data and the same sparsity, the result is the best 48-expert OLMoE model among eight methods, and it keeps the most accuracy on knowledge-heavy tasks such as MMLU.

5 Conclusion

We presented **LEPO (Learning Expert-Pruning Policies)**, a framework that treats expert selection and expert budget as learnable downstream decisions while keeping pretrained MoE parameters frozen. Motivated by an oracle study across nine reasoning and non-reasoning MoE models, LEPO combines lightweight expert-preference adaptation, grouped budget policies, full-budget teacher guidance, and GRPO to reduce routed-expert computation without requiring task labels or oracle budget annotations. Across request-local adaptation, offline budget learning, and permanent pruning, LEPO consistently improves the accuracy and efficiency trade-off, using only 75–87% of the native expert budget on generative tasks while largely preserving or improving accuracy. These results suggest that the pretraining Top-K need not remain fixed at deployment, and that learning which experts to favor and how many to activate can provide a practical path toward more efficient MoE inference.

Limitations

This work investigates LEPO as a framework for learning both which experts an SMoE model should prioritize and how many experts it should activate, covering request-local test-time adaptation, offline budget learning, and permanent expert pruning. Although LEPO preserves or improves the original model's accuracy while using fewer routed experts, and achieves a better accuracy–efficiency trade-off than uniform Top-k and Ada-K under the evaluated settings, several limitations remain. Due to computational constraints, our test-time adaptation experiments are limited to small-

and medium-scale MoE models, including DeepSeek-V2-Lite, OLMoE-1B-7B, and Qwen1.5-MoE-A2.7B, while offline budget learning is evaluated on models up to Mixtral-8x22B. Future work should investigate more efficient adaptation mechanisms, extend evaluation to larger and reasoning-oriented MoE models, and develop systems-level support that can better realize the computational benefits of adaptive expert budgets in practice.

IMPACT STATEMENT

This paper presents foundational contributions to machine learning, evaluated entirely in an academic setting using standard public benchmarks without involving human subjects, personally identifiable information, or proprietary data. While our research focuses on core methodological advancements rather than direct applications, we acknowledge broader systemic concerns inherent to contemporary machine learning. In particular, models trained on large-scale web corpora risk inheriting and perpetuating societal biases, such as those concerning race, gender, and socioeconomic background. Additionally, the intensive computational demands of large-scale model training and evaluation contribute to environmental and energy concerns. We consider proactive bias mitigation, fair algorithmic representation, and computational efficiency to be crucial imperatives for future work, and we urge researchers to evaluate downstream deployments in light of these considerations.

USE OF LARGE LANGUAGE MODELS

In accordance with the ICLR 2027 LLM Policy and Code of Ethics, we disclose our use of Large Language Models (LLMs). The authors take full responsibility for the entirety of this work. LLM assistance was strictly limited to: (i) scenario rewriting and recognition verification, and (ii) minor text editing for grammar and phrasing. LLMs were not used to generate or manipulate empirical findings, summarize related work without human verification, or formulate conclusions.

REFERENCES

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

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

I-Chun Chen, Hsu-Shen Liu, Wei-Fang Sun, Chen-Hao Chao, Yen-Chang Hsu, and Chun-Yi Lee. Retraining-free merging of sparse moe via hierarchical clustering, 2025. URL https://arxiv.org/abs/2410.08589.

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

Zewen Chi, Li Dong, Shaohan Huang, Damai Dai, Shuming Ma, Barun Patra, Saksham Singhal, Payal Bajaj, Xia Song, Xian-Ling Mao, Heyan Huang, and Furu Wei. On the representation collapse of sparse mixture of experts, 2022.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers, 2019.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

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

Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. Stablemoe: Stable routing strategy for mixture of experts, 2022.

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

Damai Dai, Chengqi Deng, Chenggang Zhao, R.x. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y.k. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1280–1297, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.70. URL https://aclanthology.org/2024.acl-long.70/.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1285. URL https://aclanthology.org/P19-1285.

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chengyu Hou, Chenhao Xu, Chenze Shao, Chong Ruan, Conner Sun, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Donghao Li, Dongjie Ji, Erhang Li, Fang Wei, Fangyun Lin, Fangzhou Yuan, Feiyu Xia, Fucong Dai, Guangbo Hao, Guanting Chen, Guoai Cao, Guolai Meng, Guowei Li, Han Yu, Han Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoling Zhang, Haoming Luo, Haoran Wei, Haotian Yuan, Haowei Zhang, Haowen Luo, Haoyu Chen, Haozhe Ji, Hengqing Zhang, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, J Yang, JQ Zhu, Jia Luo, Jia Song, Jia Yu, Jialiang Huang, Jialu Cai, Jian Liang, Jiangting Zhou, Jiasheng Ye, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jieyu Yang, Jin Chen, Jin Yan, Jingchang Chen, Jingli Zhou, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jingzi Zhou, Jinhua Zhu, Jiping Yu, Joseph Sun, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junmin Zheng, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Leyi Xia, Li Zhang, Liang Zhao, Lihua Guo, Lingxiao Luo, Linwang Ma, Linyan Zhu, Litong Wang, Liyu Cai, Liyue Zhang, Longhao Chen, MS Di, MY Xu, Max Mei, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Mingxu Zhou, Minmin Han, Ning Wang, Panpan Huang, Panpan Wang, Peixin Cong, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Qiwei Jiang, Rui Tian, Ruifan Xu, Ruijie Lu, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang,

Runqian Chen, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, Ruyi Chen, SH Liu, Shanghao Lu, Shangmian Sun, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoheng Nie, Shaoqing Wu, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Shuying Yu, Songyang Zhou, Tao Ni, Tao Yun, Tian Jin, Tian Pei, Tian Ye, Tianle Lin, Tianran Ji, Tianyi Cui, Tianyuan Yue, Tingting Yu, Tun Wang, W Zhang, WL Xiao, Wangding Zeng, Wei An, Weilin Zhao, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjing Yao, Wenjun Gao, Wenkai Yang, Wenlve Huang, Wenqing Hou, Wentao Zhang, Wenting Ma, Xi Gao, Xiang He, Xiangwen Wang, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingchen Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyu Zhang, Xu Chen, Xuanyu Wang, Xuecheng Su, Xueyin Chen, Xuheng Lin, Xuwei Fu, YC Yan, YQ Wang, YW Ma, Yanfeng Luo, Yang Zhang, Yanhong Xu, Yanru Ma, Yanwen Huang, Yao Li, Yao Li, Yao Xu, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Shao, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yijia Wu, Yiliang Xiong, Yiling Ma, Ying He, Ying Tang, Ying Zhou, Yingjia Luo, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiang Zhang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, YuKun Li, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuanhao Li, Yuduan Wang, Yuehan Yang, Yuer Xu, Yuhan Wu, Yuhao Meng, Yuheng Zou, Yukun Zha, Yunfan Xiong, Yupeng Chen, Yuping Lin, Yuqian Cao, Yuqian Wang, Yushun Zhang, Yuting Yan, Yutong Lin, Yuxian Gu, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuxuan Zhou, Yuyang Zhou, Yuzhen Huang, ZF Wu, Zehao Wang, Zehua Zhao, Zehui Ren, Zekai Zhang, Zhangli Sha, Zhe Fu, Zhe Ju, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zheren Gao, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhixuan Chen, Zhiyu Wu, Zhizhou Ren, Zhongyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihua Qu, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Ziyi Wan, Zizheng Pan, and Zongqing Yao. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Giang Do, Hung Le, and Truyen Tran. SimSMoE: Toward efficient training mixture of experts via solving representational collapse. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 2012–2025, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.107. URL https://aclanthology.org/2025.findings-naacl.107/.

Giang Do, Hung Le, and Truyen Tran. Eigenvectors of experts are training-free non-collapsing routers, 2026a. URL https://arxiv.org/abs/2605.30992.

Giang Do, Hung Le, and Truyen Tran. Rethinking sparse mixture of experts from a unified perspective, 2026b. URL https://arxiv.org/abs/2503.22996.

Truong Giang Do, Le Khiem, Quang Pham, TrungTin Nguyen, Thanh-Nam Doan, Binh Nguyen, Chenghao Liu, Savitha Ramasamy, Xiaoli Li, and Steven Hoi. HyperRouter: Towards efficient training and inference of sparse mixture of experts. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5754–5765, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.351. URL https://aclanthology.org/2023.emnlp-main.351/.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P Bosma, Zongwei Zhou, Tao Wang, Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. GLaM: Efficient Scaling of Language Models with Mixture-of-Experts. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 5547–5569. PMLR, July 2022. URL https://proceedings.mlr.press/v162/du22c.html.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2022a.

William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022b. URL http://jmlr.org/papers/v23/21-0998.html.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Hao Gu, Wei Li, Lujun Li, Qiyuan Zhu, Mark Lee, Shengjie Sun, Wei Xue, and Yike Guo. Delta decompression for moe-based llms compression, 2025. URL https://arxiv.org/abs/2502.17298.

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, 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, 2024a. URL https://arxiv.org/abs/2401.04088.

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, 2024b.

Yinsicheng Jiang, Yao Fu, Yeqi Huang, Ping Nie, Zhan Lu, Leyang Xue, Congjie He, Man-Kit Sit, Jilong Xue, Li Dong, Ziming Miao, Dayou Du, Tairan Xu, Kai Zou, Edoardo Ponti, and Luo Mai. Moe-cap: Benchmarking cost, accuracy and performance of sparse mixture-of-experts systems, 2025. URL https://arxiv.org/abs/2412.07067.

Michael Jordan and Robert Jacobs. Hierarchical mixtures of experts and the. *Neural computation*, 6:181–, 01 1994.

Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. Merge, then compress: Demystify efficient smoe with hints from its routing policy, 2024. URL https://arxiv.org/abs/2310.01334.

Zhongyang Li, Ziyue Li, and Tianyi Zhou. C3po: Critical-layer, core-expert, collaborative pathway optimization for test-time expert re-mixing, 2025. URL https://arxiv.org/abs/2504.07964.

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

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. Olmoe: Open mixture-of-experts language models, 2024. URL https://arxiv.org/abs/2409.02060.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023. URL https://arxiv.org/abs/1910.10683.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 8583–8595. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/48237d9f2dea8c74c2a72126cf63d933-Paper.pdf.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets, 2015. URL https://arxiv.org/abs/1412.6550.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Sheng Shen, Zhewei Yao, Chunyuan Li, Trevor Darrell, Kurt Keutzer, and Yuxiong He. Scaling vision-language models with sparse mixture of experts. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 11329–11344, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.758. URL https://aclanthology.org/2023.findings-emnlp.758.

Guinan Su, Yanwu Yang, Li Shen, Lu Yin, Shiwei Liu, and Jonas Geiping. Rewiring experts on the fly: Continuous rerouting for better online adaptation in mixture-of-expert models. *arXiv preprint arXiv:2510.14853*, 2025.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts, 2020. URL https://arxiv.org/abs/1909.13231.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, M. C., Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y. Charles, H. S. Che, Guanduo Chen, Guangyu Chen, Guanzheng Chen, Huarong Chen, Jia Chen, Jianlong Chen, Jun Chen, Kexin Chen, Peng Chen, Ruijue Chen, Wentao Chen, Xin Chen, Yang Chen, Yanru Chen, Yifei Chen, Yingjiang Chen, Yuankun Chen, Yujie Chen, Yutian Chen, Zhirong Chen, Dazhi Cheng, Yean Cheng, Jialei Cui, Jingbing Cui, Anqi Dai, Jiaqi Deng, Hao Ding, Rui Ding, Shaofeng Ding, Mengfan Dong, Mengnan Dong, Yuhao Dong, Yuxin Dong, Angang Du, Chenzhuang Du, Dikang Du, Jusen Du, Yulun Du, Yu Fan, Jing Feng, Qiulin Feng, Yichen Feng, Kelin Fu, Qiang Fu, Fuxuan Gao, Hongcheng Gao, Jingyue

Gao, Tong Gao, Weijia Gao, Shangyi Geng, Jie Gong, Linhu Gong, Shengao Gong, Xiaochen Gong, Qizheng Gu, Yicheng Gu, Shuhao Guan, Haiqing Guo, Shiqi Guo, Xiang Guo, Zhengyan Guo, Beixi Hao, Wenxin Hao, Xiaoru Hao, Dailan He, Haotian He, Lehan He, Qi He, Weiran He, Xinran He, Xinyi He, Yibo He, Yunjia He, Chao Hong, Tiange Hong, Hao Hu, Jiaxi Hu, Ruikun Hu, Weiming Hu, Yangyang Hu, Zhenxing Hu, Liang Hua, Jinbin Huang, Ke Huang, Ruiyuan Huang, Siying Huang, Weixiao Huang, Yan Huang, Zhengjie Huang, Zhiqi Huang, Yulong Hui, Chaobo Jia, Yutong Jiang, Zhejun Jiang, Zuoyou Jiang, Wenyi Jin, Xinyi Jin, Yu Jing, Huanjun Kong, Guokun Lai, Aidi Li, Cheng Li, Chengyuan Li, Cong Li, Fang Li, Guanyu Li, Haoyang Li, Jia Li, Junxiong Li, Lei Li, Letian Li, Lincan Li, Weihong Li, Wentao Li, Xintong Li, Yang Li, Yishen Li, Yiwei Li, Yuxiao Li, Zhaowei Li, Zhaoxi Li, Zheming Li, Zhengxiao Li, Zhiyuan Li, Jiawei Lin, Xiaohan Lin, Yibo Lin, Zichao Lin, Ziyan Lin, Bill Liu, Boxiao Liu, Chuan Liu, Liang Liu, Shaowei Liu, Shudong Liu, Shuran Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yanming Liu, Yibo Liu, Yipeng Liu, Zhengying Liu, Zhiheng Liu, Enzhe Lu, Haoyu Lu, Linqiang Lu, Tingzhan Lu, Zhiyuan Lu, Aotian Luo, G. Luo, Junyu Luo, Yifan Luo, B. Lyu, Wenzhou Lyu, Shaoguang Mao, Yuan Mei, Xin Men, Minqing Ni, Yixuan Niu, Siyuan Pan, Shujun Peng, Zhangyang Qi, Ruoyu Qin, ZeChao Qin, Zeyu Qin, Haiquan Qiu, Jianxin Qiu, Jiezhong Qiu, Bowen Qu, Yuhao Qu, Zeyu Shang, Youbo Shao, Han Shen, Jincheng Shi, Juanfeng Shi, Lidong Shi, Shengyuan Shi, Wingchun Siu, Pengwei Song, Xiaoxi Song, Jianlin Su, Yunfeng Su, Zhaochen Su, Lin Sui, Jingsong Sun, Junyao Sun, Shaoning Sun, Shuzhe Sun, Tongyu Sun, Yujun Sun, Yunpeng Tai, Chuning Tang, Heyi Tang, Sirui Tang, Zecheng Tang, Chaoran Tian, Rongpeng Tian, Yu Tian, Wei Tu, Chensi Wang, Chuang Wang, Chunjie Wang, Dinglu Wang, Feng Wang, Hailong Wang, Haiming Wang, Hao Wang, Hao Wang, Huaqing Wang, Hui Wang, Jiayi Wang, Jinglong Wang, Jinhong Wang, Jiuzheng Wang, Linian Wang, Shaobo Wang, Shenzhi Wang, Shuyi Wang, Si Wang, Siyuan Wang, Tianfu Wang, Wenjue Wang, Xingran Wang, Xinmei Wang, Xinyuan Wang, Xusheng Wang, Yalin Wang, Yangkun Wang, Yao Wang, Yaoyu Wang, Yejie Wang, Yiqin Wang, Yucheng Wang, Yuzhi Wang, Zhaoji Wang, Zhaowei Wang, Zhengtao Wang, Zhenhao Wang, Zhongsheng Wang, Zifan Wang, Chu Wei, Ming Wei, Shouxin Wei, Zichen Wen, Fan Wu, Haoning Wu, Rucong Wu, Wenhao Wu, Xiaoxue Wu, Yingcong Wu, Yongqi Wu, Yuxin Wu, Zijian Wu, Xinglang Xian, Chenxuan Xiang, Yuye Xiang, Bocheng Xiao, Chenjun Xiao, Xin Xiao, Jin Xie, Xiaotong Xie, Yifeng Xie, Zhe Xie, Bowei Xing, Yiming Xiong, Baosheng Xu, Boyu Xu, Jiale Xu, Jianfan Xu, Jing Xu, Jinjing Xu, L. H. Xu, Qingtao Xu, Shuyao Xu, Suting Xu, Tiantian Xu, Tianxiang Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ye Xu, Yueni Xu, Ziyao Xu, Haonan Xue, Junjie Yan, Yaoyao Yan, Fan Yang, Guangyao Yang, Hao Yang, Junwei Yang, Ruoyu Yang, Wenjie Yang, Xiaofei Yang, Xinyu Yang, Yi Yang, Yiling Yang, Ying Yang, Yuchen Yang, Zhen Yang, Zhilin Yang, Zian Yang, Zuhao Yang, Haotian Yao, Dan Ye, Haoran Ye, Wenjie Ye, Zhanbo Ye, Bohong Yin, Haoxiang Yin, Xietong Yin, Chengzhen Yu, Haozhen Yu, Longhui Yu, Shengnan Yu, Shuying Yu, Tianxiang Yu, Enming Yuan, Mengjie Yuan, Tongtian Yue, Wei Yue, Yang Yue, Dunyuan Zha, Haobing Zhan, B. H. Zhang, Dehao Zhang, Fei Zhang, Hao Zhang, Haoyuan Zhang, Huanyu Zhang, Jiapei Zhang, Jiaxuan Zhang, Jin Zhang, Kaiyi Zhang, Miaozhen Zhang, Puqi Zhang, Qinglei Zhang, Rong Zhang, Rui Zhang, Shaoshuai Zhang, Shiyi Zhang, Xiaobin Zhang, Xiaoyun Zhang, Y. Zhang, Yangkun Zhang, Ye Zhang, Yichi Zhang, Yikun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Zijing Zhang, Bin Zhao, Chenguang Zhao, Feifan Zhao, Jinglun Zhao, Jinxiang Zhao, Shuai Zhao, Wenshuo Zhao, Xiangyu Zhao, Xuanle Zhao, Yikai Zhao, Zijia Zhao, Haozhi Zheng, Huabin Zheng, Ruihan Zheng, Shaojie Zheng, Tengyang Zheng, Haofeng Zhong, Lei Zhong, Longguang Zhong, M. Zhou, Qiankang Zhou, Runjie Zhou, Ruozhang Zhou, Xinyu Zhou, Yiqiao Zhou, Zaida Zhou, Jinguo Zhu, Liya Zhu, Xinhao Zhu, Yangjunfeng Zhu, Yuxuan Zhu, Zhen Zhu, Chen Zhuang, Weiyu Zhuang, and Xinxing Zu. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Siqi Wang, Zhengyu Chen, Bei Li, Keqing He, Min Zhang, and Jingang Wang. Scaling laws across model architectures: A comparative analysis of dense and moe models in large language models, 2024a. URL https://arxiv.org/abs/2410.05661.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024b. URL https://arxiv.org/abs/2406.01574.

Tongtian Yue, Longteng Guo, Jie Cheng, Xuange Gao, Hua Huang, and Jing Liu. Ada-k routing: Boosting the efficiency of moe-based LLMs. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=9CqkpQExe2.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Zihao Zeng, Yibo Miao, Hongcheng Gao, Hao Zhang, and Zhijie Deng. Adamoe: Token-adaptive routing with null experts for mixture-of-experts language models, 2024. URL https://arxiv.org/abs/2406.13233.

Geng Zhang, Yuxuan Han, Yuxuan Lou, Yiqi Zhang, Wangbo Zhao, and Yang You. Mone: Replacing redundant experts with lightweight novices for structured pruning of moe, 2026. URL https://arxiv.org/abs/2507.00390.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. Mixture-of-experts with expert choice routing, 2022a. URL https://arxiv.org/abs/2202.09368.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, zhifeng Chen, Quoc V Le, and James Laudon. Mixture-of-Experts with Expert Choice Routing. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 7103–7114. Curran Associates, Inc., 2022b. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/2f00ecd787b432c1d36f3de9800728eb-Paper-Conference.pdf.

Zeliang Zong, Kai Zhang, Yarong Wang, Wenming Tan, Ye Ren, and Jilin Hu. Less token, more signal: Moe expert pruning via critical token selection. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=4iupzej9nT.

A Appendix

Supplementary Material for "Less Can Be More: Learning Pruned Top-K in MoE-based LLMs"

The appendix is organized as follows. Section A.1 provides additional related work, while Section A.2 presents supplementary experiments, including test-time adaptation, matched-compute expert-budget evaluation, expert pruning, ablation studies, in-depth analysis, and efficiency analysis. Sections A.6–A.10 provide model configurations, dataset statistics, experimental settings, hyperparameters, and hardware details. Finally, Section A.11 presents implementation details and pseudocode for LEPO and SearchTopK, with the full LEPO algorithm described in Section A.12.

A.1 Related Work (Cont.)

Test-Time Training (TTT). Test-time training adapts a model to unlabeled test inputs using self-supervised signals available at inference time (Sun et al., 2020). For SMoE LLMs, C3PO (**?**) adapts expert mixing weights in critical layers using surrogate objectives constructed from successful examples in a reference set. Rewiring-MoE (Su et al., 2025) removes the need for reference data by learning lightweight additive router-logit vectors through next-token prediction on the available context, alternating router adaptation with generation. LEPO builds on this data-free,

context-based adaptation paradigm but extends it from *which experts to favor* to *how many experts to activate*. Specifically, LEPO jointly adapts expert preferences and expert budgets under guidance from the frozen full-Top-K model, using context prediction, output-distribution distillation, and hidden-representation matching for capability preservation, while a lightweight policy learns to reduce expert computation subject to fidelity constraints.

Teacher Guidance and Policy Optimization. Knowledge distillation transfers predictive distributions from a teacher to a student (Hinton et al., 2015), while intermediate-feature matching provides additional supervision (Romero et al., 2015). LEPO adopts both signals using a shared frozen backbone, where the full-budget teacher guides a reduced-budget student and only routing preferences are updated. For discrete expert-budget selection, LEPO uses group-relative policy optimization (GRPO) (Shao et al., 2024), with rewards balancing expert savings against prediction fidelity. This unifies routing adaptation and compute allocation under a common teacher-guided objective.

The work most closely related to ours is Ada-K (Yue et al., 2025), which augments pretrained routers with lightweight per-token budget allocators learned jointly during fine-tuning. However, this approach requires supervised fine-tuning data and, in our experiments, exhibits a trade-off between efficiency and predictive performance. LEPO shares the goal of adapting computation to each input but is designed to preserve model capability while reducing expert activation. Notably, LEPO can operate in a test-time training setting without additional labels or task-specific training data. Under matched-compute evaluation (Section 4.2), LEPO consistently achieves a better accuracy–efficiency trade-off than both Ada-K and uniform Top-k reduction.

A.2 Experiments (Cont.)

We test LEPO in three settings: Section 4.1 studies test-time adaptation on generative tasks, following Rewiring (Su et al., 2025). Section 4.2 studies adaptive expert budgets, following Ada-K routing (Yue et al., 2025). Section 4.3 studies expert pruning, following STEP (Zong et al., 2026). In every setting, we report accuracy together with the number of routed experts that each method activates.

A.2.1 Data-free test-time adaptation on generative tasks

Setup. We follow the protocol of Su et al. (2025). We use three MoE models: DeepSeek-V2-Lite (Dai et al., 2024a) (64 routed experts, $K = 6$), OLMoE-1B-7B (**?**) (64 experts, $K = 8$), and Qwen1.5-MoE-A2.7B (Team, 2024) (60 experts, $K = 4$). There are five tasks: HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) for code, GSM8K (Cobbe et al., 2021) and MATH500 (Lightman et al., 2023) for mathematics, and MMLU (Hendrycks et al., 2021) for knowledge. Every question is decoded greedily with one sample and at most 512 new tokens. We report pass@1 for code and exact-match accuracy of the extracted answer otherwise. We compare LEPO with both supervised and data-free baselines. The supervised baselines include in-context learning (ICL) with 3 or 5 demonstrations and C3PO (Li et al., 2025), which adapts expert mixtures using 100 labeled reference examples per task. The data-free baselines include Rewiring-MoE (Su et al., 2025), which adapts routing at test time while retaining the native Top-K, and Ada-K (Yue et al., 2025), which learns a per-token expert budget.

Adaptive expert budgets at matched compute

Setup. We follow the main comparison of Yue et al. (2025) (their Table 3). It covers four MoE models: Mixtral-8x22B and Mixtral-8x7B (Jiang et al., 2024b) ($K = 2$ of 8), Qwen1.5-MoE-A2.7B ($K = 4$ of 60), and DeepSeek-MoE-16B (Dai et al., 2024b) ($K = 6$ of 64). Evaluation uses six Open LLM Leaderboard tasks with the leaderboard's shot counts and metrics: ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), MMLU, GSM8K, TruthfulQA (**?**) and WinoGrande (**?**). All numbers come from lm-evaluation-harness (Gao et al., 2024). Ada-K is retrained with the recipe in its paper. LEPO's router preferences and budget policy are trained once on the same training data, then frozen for evaluation. Its budget is set near Ada-K's measured Act so that the two methods are compared at about the same compute. We also evaluate static Top-k for every k between the Ada-K budget and K. Static Top-k is the simplest way to trade accuracy for compute, and it gives the reference line. We report Δ_{line}, the gap between a method's average and

Table 4: **LEPO ablations on OLMoE/GSM8K ($n = 200$).** Accuracy is exact-match accuracy (%). $\bar{K}$ averages active experts over the student's prefill and decoding forwards; $\bar{K}_{\mathrm{dec}}$ counts decoding only. Both exclude adaptation work. Parentheses in panel (a) give input dimensions. Defaults are repeated because the three studies were run separately.

Configuration	Accuracy	$\bar{K}$	$\bar{K}_{\mathrm{dec}}$
Original Top-8	39.50	8.00	8.00
Static Top-7	39.50	7.00	7.00
Static Top-6	38.00	6.00	6.00
Static Top-5	37.50	5.00	5.00
Static Top-4	35.50	4.00	4.00
(a) Policy inputs; $B = 4$			
Router statistics (12; default)	41.50	7.46	7.20
Sorted Top-K probabilities (8)	41.00	7.33	7.01
Confidence summaries (4)	38.50	7.44	7.17
Router entropy (1)	40.00	7.35	7.05
Input-independent policy (0)	39.50	7.37	7.07
Full router probabilities (64)	39.50	7.21	6.84
Projected hidden state (32)	40.50	6.95	6.64
Router statistics + hidden state (44)	42.50	7.03	6.77
(b) Budget granularity; default inputs and reward			
One group ($B = 1$)	42.00	7.44	7.18
Four groups ($B = 4$; default)	40.50	7.47	7.21
One group per layer ($B = 16$)	38.00	7.86	7.79
(c) Budget-policy reward; $B = 4$, default inputs			
Full reward (default)	40.50	7.47	7.22
Without hidden-state penalty ($\lambda_h = 0$)	44.50	7.42	7.14
Without output distillation ($\lambda_o = 0$)	30.50	5.98	5.00

this line interpolated at the method's Act. A positive gap means the method spends its experts better than a uniform reduction of k.

Complementarity with expert pruning

Setup. We follow the protocol of STEP (Zong et al., 2026) on OLMoE-1B-7B. Every MoE layer keeps 48 of its 64 experts, and calibration uses 128 unlabelled C4 (Raffel et al., 2023) blocks of 2,048 tokens. Models are evaluated zero-shot on eight tasks with lm-evaluation-harness: ARC-Easy and ARC-Challenge, BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2019), WinoGrande, HellaSwag, MMLU, and OpenBookQA (Mihaylov et al., 2018). The baselines, with numbers taken from the STEP paper, are of three kinds. D2-MoE (Gu et al., 2025) compresses experts by delta decomposition. MC-SMoE (Li et al., 2024) and HC-SMoE (Chen et al., 2025) merge experts. Frequency, Gate Score (He et al., 2025), MoNE (Zhang et al., 2026), and STEP prune experts by a score. In this setting, LEPO's expert-preference vectors are amortised over the same unlabelled C4 calibration blocks rather than reset for each request. They are then folded into the router as a bias, and STEP's pruning and gate fine-tuning are applied to prune to 48 experts. LEPO therefore uses no data beyond what STEP already uses.

A.3 Ablation studies

We ablate LEPO's policy inputs, budget granularity, and reward on the same first 200 GSM8K test problems using OLMoE-1B-7B-0924-Instruct ($L = 16$, $K = 8$). Unless varied, we use four layer groups, budgets $k \in \{4, \ldots, 8\}$, the router statistics in Equation equation 3, and reward weights $(\lambda_c, \lambda_o, \lambda_h) = (0.1, 10, 0.3)$. Each request uses two alternating adaptation steps per round, four GRPO trajectories, and generation chunks of 50 tokens, with greedy decoding capped at 512 new tokens. Table 4 reports each study's separately evaluated default; these runs are not pooled. Accuracy differences below are in percentage points, with approximate 95% paired normal confidence intervals across problems; they do not measure variation across random seeds.

Policy inputs. The default state combines sorted Top-K probabilities with four confidence summaries: entropy, the top-two probability gap, the Top-K boundary gap, and Top-K probability mass.

Using only the sorted probabilities gives 41.0% accuracy, close to the default's 41.5%, whereas the confidence summaries alone give 38.5%. Providing all 64 probabilities in expert order gives 39.5%, offering no observed accuracy advantage from explicit expert identity. Removing input conditioning yields 39.5% at a similar budget ($\bar{K} = 7.37$ versus 7.46); its difference from the default is -2.0 points (CI $[-8.05, 4.05]$). Adding a fixed 32-dimensional random projection of the layer-normalized MoE input to the default statistics gives the highest observed feature-ablation accuracy, 42.5%, while reducing $\bar{K}$ to 7.03. However, its gain over the default is only $+1.0$ point (CI $[-5.81, 7.81]$), and no feature variant has a confidence interval excluding zero against the static Top-k accuracy line interpolated at its own $\bar{K}$. These results motivate further evaluation of hidden-state inputs but do not establish a superior feature set.

Budget granularity. A single group still chooses a different budget for each token, but shares that decision across all layers. It reaches 42.0% accuracy at $\bar{K} = 7.44$, compared with 40.5% at 7.47 for four groups; the paired gain is $+1.5$ points (CI $[-5.09, 8.09]$). Independent policies for all 16 layers retain nearly the native budget ($\bar{K} = 7.86$) and reach 38.0%, a -2.5-point difference from four groups (CI $[-8.78, 3.78]$). The four-group policy allocates mean decoding budgets of 5.97, 7.21, 7.81, and 7.86 from early to late groups, showing that it learns different allocations across depth. Nevertheless, this experiment does not establish an accuracy benefit from layer-specific budgets, and finer granularity does not improve the observed accuracy–budget trade-off.

Reward terms. We remove teacher penalties only from the budget-policy reward in Equation equation 12; the router-vector loss retains both output and hidden-state matching in every variant. Removing output distillation lowers accuracy from 40.5% to 30.5% (difference -10.0, CI $[-16.95, -3.05]$), reduces $\bar{K}$ from 7.47 to 5.98, and increases teacher–student output KL from 0.0475 to 0.1647. It also falls 7.49 points below the interpolated static Top-k line at the same average budget (CI $[-14.76, -0.22]$), suggesting that the degradation is not explained by budget reduction alone. Output distillation therefore provides the clearest evidence of a useful reward component in this setting. Removing the hidden-state penalty instead gives 44.5% at $\bar{K} = 7.42$, but its $+4.0$-point gain over the full reward remains inconclusive (CI $[-2.19, 10.19]$). Thus, these runs do not demonstrate an additional accuracy benefit from hidden-state matching in the policy reward; they do not test its contribution to router adaptation.

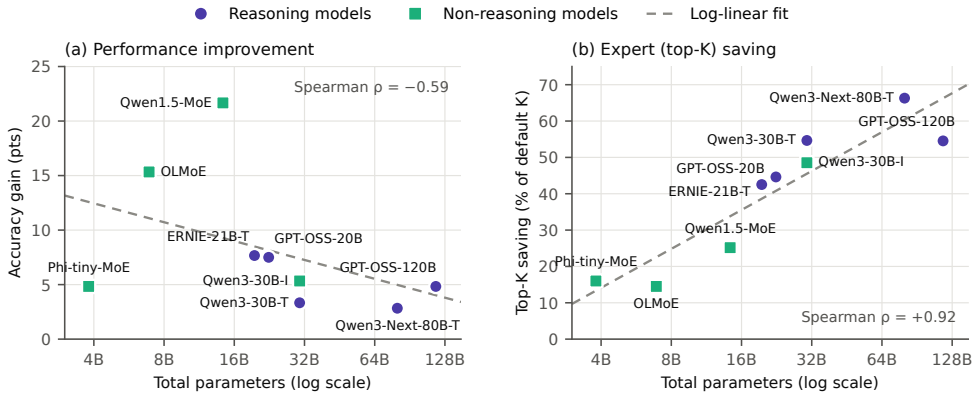
Scope and compute. These findings concern one model and a 200-problem subset. Even static Top-7 matches the native model's observed accuracy while using fewer experts than the default LEPO runs. The expert counts above describe student execution, not total savings: including teacher passes, rollouts, repeated prefills, and router updates, the recorded routed-expert compute proxy is 31.2–44.6$\times$ the native reference across the adaptive variants (charging a backward pass as two forwards). Consequently, these ablations characterize routing behavior and accuracy trade-offs, without establishing an end-to-end efficiency gain.

A.4 In-depth analysis

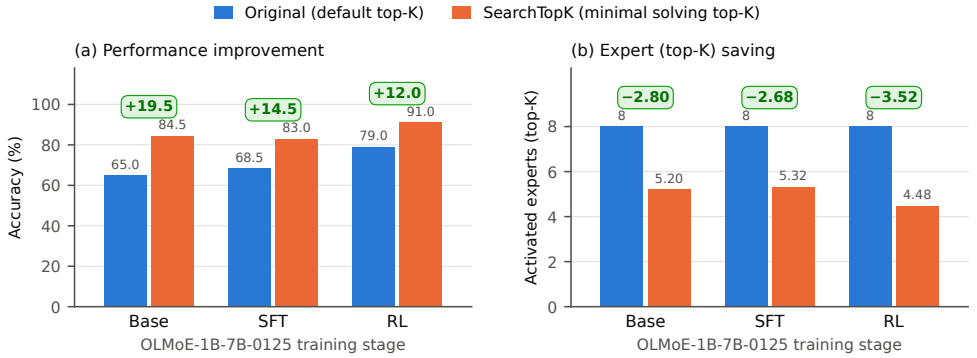
We investigate three questions behind adaptive expert execution: how much room exists to improve on the native Top-K, whether local routing statistics identify that opportunity, and why learning to reduce expert counts can still hurt downstream performance. We distinguish oracle budget search, diagnostics of pretrained checkpoints, and observed LEPO/Ada-K behavior; each provides a different kind of evidence.

How much potential does a fixed Top-K leave unused? We first evaluate *SearchTopK*, an oracle that greedily decodes each question at every budget in a predefined grid and selects the smallest budget producing a correct answer. Questions unsolved at every budget remain incorrect and are charged the native K. Thus, the reported accuracy is the fraction solved by at least one budget, and the expert count averages the selected budgets across questions. This diagnostic uses reference answers and multiple generations; its reported expert count excludes the search itself and is not the cost of an executable label-free policy.

We extend the nine-model comparison in Figure 1, using 200 validation questions from each of three MMLU-Pro domains: mathematics, physics, and biology. For GPT-OSS-120B, oracle accuracy rises from 90.83% to 95.67% while the mean selected budget falls from 4 to 1.82 experts. For



(a) Oracle potential versus model size on MMLU-Pro.



(b) Oracle potential across OLMoE-0125 training stages on GSM8K.

Figure 3: **Accuracy and expert-budget potential of SearchTopK.** SearchTopK selects the smallest successful budget using answer labels; unsolved questions retain native-K cost. Panel (a) averages three domains, each with 200 validation questions, for nine models; dashed lines are descriptive fits against log model size. Panel (b) uses the same 200 GSM8K questions across checkpoints and the grid $\{1, 2, 4, 6, 8\}$. These are oracle results, not LEPO measurements; expert savings exclude search overhead and differences in generation length.

Qwen3-Next-80B-Thinking, accuracy rises from 91.00% to 93.83% at 3.37 rather than 10 experts. Across the five reasoning models, the gains range from 2.83 to 7.67 percentage points, with 42.5–66.3% fewer experts under this accounting. The opportunity spans domains: median gains across the nine models are 4.0, 5.5, and 7.0 points for mathematics, physics, and biology, respectively. Across these checkpoints, larger total parameter counts correlate with greater expert savings ($\rho = 0.92$) but smaller accuracy gains ($\rho = -0.59$; Figure 3(a)). These descriptive trends involve only nine heterogeneous models, with differences in architecture, training, and generation limits; they do not establish a scaling law. These are results for the displayed models, not a universal guarantee: an excluded DeepSeek-MoE-16B-base sweep showed no oracle gain, while the large gains for the Qwen1.5 and OLMoE-0924 base models are affected by frequent generation truncation. Generation length also matters: Qwen3-30B-Thinking's selected-budget ratio is 45.3%, but its token-weighted expert-work ratio is 90.8%. Expert-count reductions therefore need not translate proportionally into total computation or latency savings.

Insight 1: The native budget can miss both accuracy and sparsity opportunities. Changing only the expert budget can recover answers missed at the native Top-K while using fewer experts on other questions. The oracle quantifies an opportunity for budget selection; it does not show that LEPO, or any label-free selector, can recover all of it.

Does post-training remove this opportunity? Figure 3(b) compares the Base, SFT, and final Instruct checkpoints of OLMoE-1B-7B-0125; the figure labels the last checkpoint "RL." Their native accuracies are 65.0%, 68.5%, and 79.0%, whereas oracle selection reaches 84.5%, 83.0%, and 91.0% at mean budgets of 5.20, 5.32, and 4.48 instead of 8. Native Top-8 is also the best single fixed budget on this grid for each checkpoint, so the oracle gain comes from selecting budgets by question rather than finding a better global constant. For the final checkpoint, the gain is 12.0 points (paired-bootstrap 95% CI $[7.5, 16.5]$), with a 44.0% expert-count reduction (CI $[40.9, 47.1]$). This comparison establishes remaining potential within each checkpoint; differences across stages are not an isolated training intervention, since the Base evaluation uses a four-shot plain prompt and a 512-token cap, whereas the later checkpoints use a chat template and a 1,024-token cap.

Insight 2: Better task accuracy does not imply an optimal expert budget. Substantial oracle gains remain in the final post-trained checkpoint, even though native Top-8 is its strongest fixed budget. Adapting the budget to the request is therefore a distinct opportunity from improving the model at a fixed budget.

Can router concentration or expert similarity identify the right budget? We measure both on identical teacher-forced GSM8K question-and-gold-solution text across the three OLMoE-0125 checkpoints. These are offline diagnostics, not features computed from reference solutions at LEPO inference time. Normalized entropy of the selected Top-8 mixing weights is approximately 0.946–0.947 in all three checkpoints. Despite this similarity, their oracle accuracy gains and expert savings differ substantially. At the question level, raw entropy correlates with the oracle budget (Spearman $\rho = 0.223$, 0.372, and 0.304 for Base, SFT, and RL). After subtracting the stage-and-layer-specific mean entropy at each token position, these correlations fall to -0.046, 0.130, and 0.118; all three 95% bootstrap intervals include zero (Figure 4). The apparent association is therefore sensitive to token-position effects, including the unusually concentrated routing of the first token.

We also test whether the next ranked expert adds a representation already supplied by the selected experts. On layer inputs from the unmodified native-budget forward, define $A_k = \sum_{j \le k} w_j E_j(h)$ and $B_k = w_{k+1} E_{k+1}(h)$, and measure direction difference $D = 1 - \cos(A_k, B_k)$ and relative magnitude $M = \|B_k\|/\|A_k\|$ for $k = 1, \ldots, 7$. Mean D is 0.928, 0.930, and 0.927 across the three checkpoints; these values do not indicate near-collinear expert outputs. After controlling for sequence length, the question-level correlations of D with the oracle budget are 0.005, 0.069, and -0.067; those of M are -0.087, 0.002, and -0.010. All six 95% intervals include zero (Figure 5). These aggregated local measurements neither reliably identify the required budget nor establish that the discarded experts are functionally redundant throughout generation.

Insight 3: Pruning potential is not captured by a single local statistic. Neither mean router entropy nor the tested expert-similarity measures reliably predicts the oracle budget after the relevant controls. This motivates LEPO's use of student behavior under sampled budgets and teacher-fidelity feedback, while leaving open whether richer combinations of local features can be predictive.

Why can learning to prune still trade performance for compute? We track OLMoE-1B-7B-0924 during 200 training steps on the same 6,400 SFT sequences, comparing native Top-8 router fine-tuning with Ada-K allocator training. Both train approximately 2.1 million parameters, but use their respective objectives and optimizers. Ada-K uses 50 warm-start steps followed by PPO, with activation penalties $\lambda \in \{0.001, 0.003, 0.01\}$. Every five steps, we evaluate the same 256 held-out sequences; static Top-k evaluations provide a reference at matched average expert counts (Figure 6).

At step 200, Ada-K uses 7.83, 6.31, and 4.20 experts, respectively, while held-out log-likelihood changes by -0.0110, -0.0312, and -0.1390 nats per token relative to the untouched Top-8 model. The corresponding gaps below the interpolated static Top-k line are 0.0099, 0.0122, and 0.0278 nats per token. By contrast, native-budget router fine-tuning improves log-likelihood by 0.0763 nats per token. Downstream accuracy, averaged over the first 500 questions of ARC-Challenge, HellaSwag, and WinoGrande, changes from 65.07% to 65.47%, 64.60%, and 61.60% for the three Ada-K penalties, versus 66.13% for router fine-tuning. Thus, stronger compute pressure is associated with larger

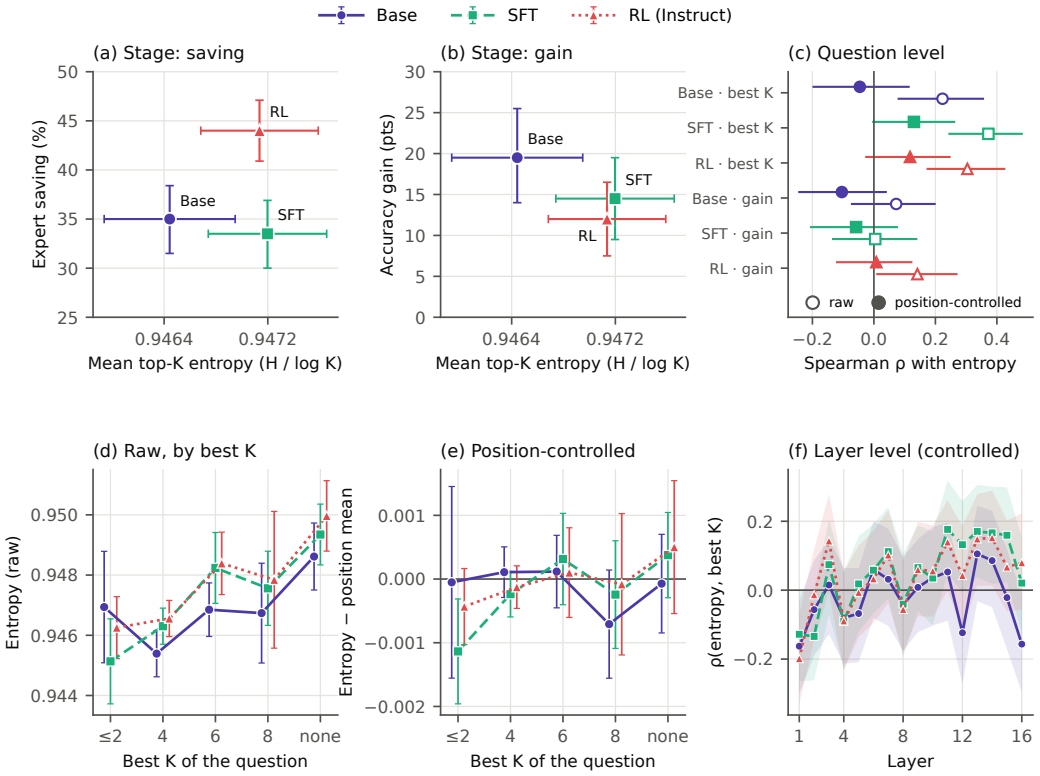


Figure 4: **Router entropy is a weak standalone guide to oracle budgets.** Entropy is computed over the renormalized Top-8 routing weights on identical teacher-forced text, with 200 questions per checkpoint. Panels (a–b) compare checkpoint means; (c–f) examine question and layer statistics. Filled markers in (c) subtract the mean entropy at each token position before averaging; the question-level budget correlations then have intervals spanning zero. "None" denotes questions unsolved by the search; these receive budget 8 in correlations. Correlation intervals use bootstrap resampling of questions.

performance losses in this experiment, although the weakest penalty slightly improves the downstream average. The fine-tuning control shows that degradation is not inevitable when training on these batches; it does not isolate the allocator objective from all other optimization differences.

Insight 4: A smaller budget is useful only if the allocation preserves task behavior. In these Ada-K runs, lower expert counts do not yield a likelihood advantage over uniform Top-k at the same count. LEPO addresses this design challenge by coupling budget savings to teacher fidelity and adapting expert preferences alongside budgets; the diagnostic motivates this coupling rather than proving that any one component causes LEPO's gains.

What does the trade-off look like in a generated solution? Figure 7 illustrates GSM8K question 804 with OLMoE-0924-Instruct. Two cars travel at 60 and 70 miles per hour for two hours, giving a correct separation of 20 miles. The original model computes $140 - 120 = 20$ with 8 experts per answer token. LEPO reaches the same answer with 6.95 experts on average: it computes a signed difference of -20 and correctly interprets its magnitude. Ada-K uses 5.70 experts on average but adds the signed difference to itself, producing 40 miles. The failure concerns how intermediate quantities are combined, despite locally valid multiplication.

Across the 12 selected examples, the mean answer budgets are 8.00 for the original model, 5.76 for Ada-K, and 7.19 for LEPO; the reported mean fractions of answer tokens below six experts are 0%, 77.9%, and 1.4%. Ada-K's digit/operator tokens and other answer tokens receive similar budgets (5.70 versus 5.75), as do LEPO's (7.16 versus 7.21). The traces therefore show broadly different pruning levels, without establishing preferential protection of arithmetic tokens. The examples are

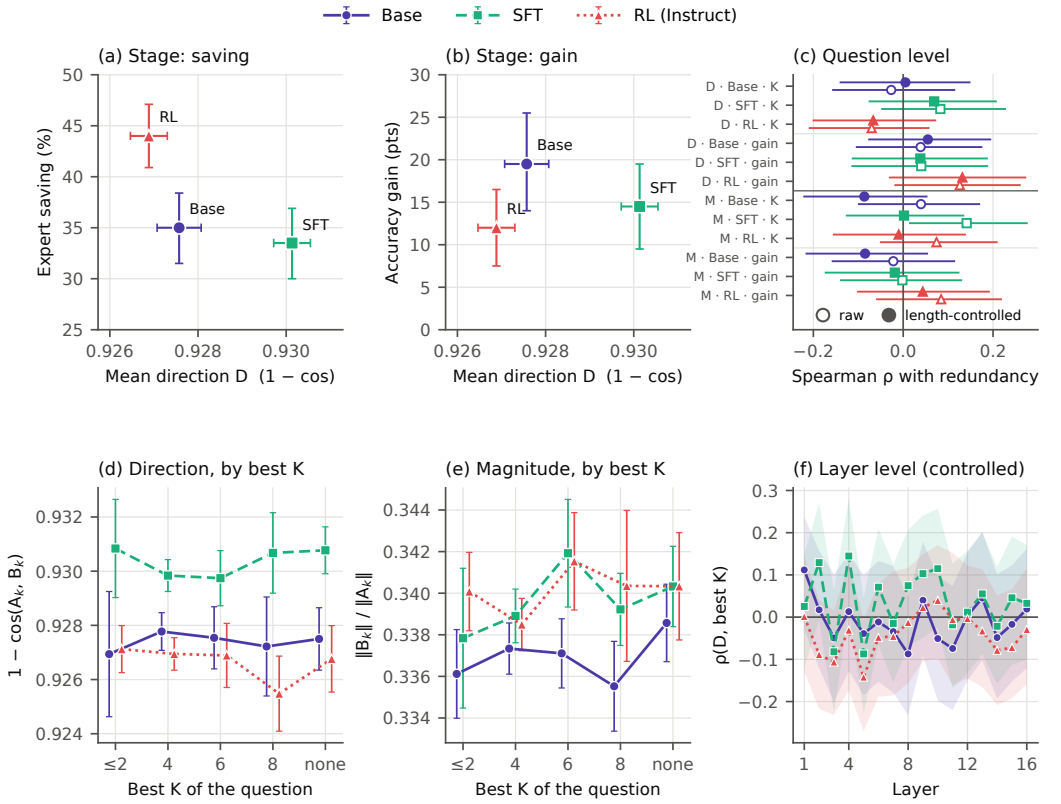


Figure 5: **Local expert-output geometry does not reliably predict pruning potential.** $D = 1 - \cos(A_k, B_k)$ measures direction difference and $M = \|B_k\|/\|A_k\|$ measures relative magnitude, averaged over native-forward layer inputs and $k = 1, \ldots, 7$. Filled markers in (c) show partial Spearman correlations controlling for text length; error bars are 95% bootstrap intervals over 200 questions. Layer-local measurements do not include the changes in later representations caused by actually pruning earlier layers.

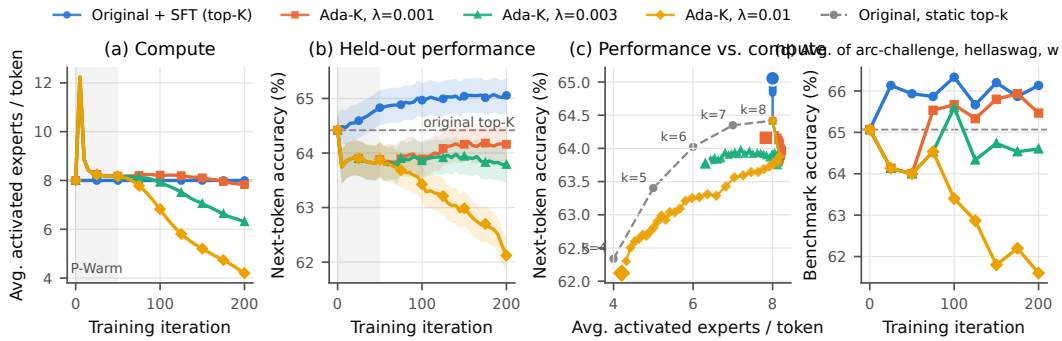


Figure 6: **Reducing expert counts does not guarantee better allocation.** OLMoE-0924 with native-budget router SFT or Ada-K at three activation penalties, trained on the same batches. Shading in (a–b) marks the 50-step warm start. Panel (b) reports token-weighted likelihood changes on 256 held-out sequences, with approximate paired 95% intervals over sequences, not training seeds. Panel (c) compares these changes with static Top-k at matched held-out expert counts; panel (d) averages three downstream benchmarks. Compute axes count routed experts, excluding training overhead.

selected from questions where the original model and LEPO succeed and Ada-K fails (82 of 1,319 candidates before length filtering). Moreover, original and LEPO decoding is greedy whereas Ada-K uses temperature 0.3; LEPO is rerun for tracing, and its displayed answer for question 804 differs

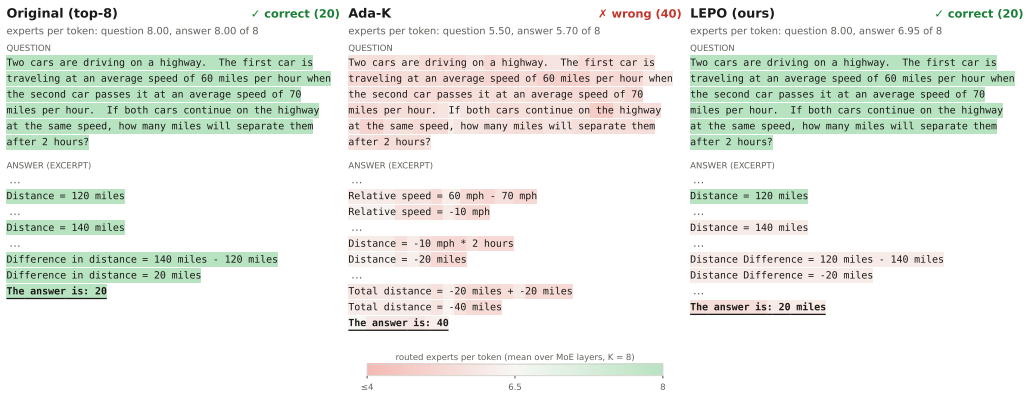


Figure 7: **A worked example of solution preservation under reduced expert execution.** OLMoE-0924-Instruct on GSM8K question 804; answer excerpts are shown, and panel averages use the full answers. Token backgrounds encode mean expert counts across the 16 MoE layers: red denotes fewer experts and green denotes counts near native Top-8. Original and LEPO return 20 miles; Ada-K returns 40. Original/LEPO counts are recorded during generation; Ada-K counts are reconstructed by a causal teacher-forced pass over its stored completion. This is a selected qualitative example with different decoding settings, not a controlled attribution of the error to particular token budgets.

Table 5: **Computational efficiency on DeepSeek-V2-Lite / HumanEval.** Average total FLOPs and inference time per sample. Rows without † are reported by Su et al. (2025); † denotes estimates obtained by scaling the corresponding reference method using cost ratios measured in our implementation.

Method	Total FLOPs	Time (s)
Baseline	4.71e+11	10.71
ICL (3-shot)	1.93e+12	12.17
ICL (5-shot)	3.05e+12	12.53
Self-Consistency (3)	1.68e+12	34.20
C3PO (100-reference)	2.63e+12	26.20
Rewiring	1.96e+12	20.12
Ada-K†	3.30e+11	9.10
LEPO (ours)†	1.27e+12	19.06

from the stored answer but remains correct under rescoring. These examples illustrate behavior rather than provide a matched-budget or causal comparison.

Insight 5: Preserving a valid solution is the relevant outcome of pruning. LEPO can retain a correct solution while executing fewer experts than the native model in these examples. More aggressive reduction can accompany errors in combining intermediate results, but token shading alone cannot identify which expert decisions caused an error.

A.5 Efficiency Analysis

Following the efficiency protocol of Rewiring-MoE (Su et al., 2025), we evaluate average total FLOPs and wall-clock time per HumanEval sample using DeepSeek-V2-Lite. Unlike the routed-expert budget $\bar{K}$, which measures only student generation, this analysis accounts for the complete test-time computation, including adaptation, teacher passes, policy rollouts, and generation.

Setup. All methods are evaluated with batch size 1 and the same decoding configuration as Section 6.3 of Su et al. (2025). We report total FLOPs and inference time per sample. For LEPO, FLOPs include teacher inference, $G = 4$ budget rollouts, router adaptation, and generation. Since FLOP accounting differs across implementations, we primarily interpret costs relative to the corresponding baseline.

Results. Table 5 shows that LEPO requires less total computation than Rewiring while retaining the benefits of test-time routing adaptation. However, its per-request cost remains higher than vanilla inference because teacher passes, policy rollouts, and router optimization dominate the savings obtained from activating fewer experts. In contrast, when LEPO adaptation is performed once and amortized across requests, as in Sections 4.2 and 4.3, inference requires only the learned reduced expert budget and incurs no per-request adaptation overhead. Thus, LEPO's expert-budget reduction translates most directly into end-to-end efficiency when the learned routing policy can be reused across inputs.

A.6 Models Configurations

Table 6 lists every checkpoint used in the paper, with the short name we use for it in the appendix. The three protocols use different checkpoints of the same families. The Rewiring protocol (Su et al., 2025) uses the chat or instruction-tuned release of each model, and prompts it with its chat template. The Ada-K protocol (Yue et al., 2025) uses the pretrained base releases, as in their Table 3. The STEP protocol (**?**) uses the base OLMoE release. All routers select experts by a softmax over the routed experts followed by Top-K. Mixtral renormalises the weights of the selected experts; the other families keep the unnormalised softmax weights, and LEPO keeps each family's convention (Equation equation 5).

Table 6: **Model configurations.** Layers counts all decoder layers; the routed (MoE) layers are in parentheses, since DeepSeek models replace the first MoE layer by a dense FFN. d: hidden size. d_e: intermediate size of one routed expert. N: routed experts per layer. K: native Top-K. Shared: always-active shared experts, which LEPO never prunes (Qwen1.5-MoE has one gated shared expert of width $4d_e$). Renorm: whether the selected routing weights are renormalised to sum to one. $|\phi| = \sum_l N_l$ and $|\psi| = B(K+5)|\mathcal{A}|$ are LEPO's trainable parameters (expert-preference vectors and budget heads) with $B = 4$ groups and $\mathcal{A} = \{\lfloor K/2 \rfloor, \ldots, K\}$. Protocols: R = Rewiring (Section 4.1), A = Ada-K (Section 4.2), S = STEP (Section 4.3).

Short name	Hugging Face checkpoint	Prot.	Params (total / active)	Layers (MoE)	d	d_e	N	K	Shared	Renorm	$\lvert\phi\rvert$	$\lvert\psi\rvert$
DS-V2-Lite	`deepseek-ai/DeepSeek-V2-Lite-Chat`	R	15.7B / 2.4B	27 (26)	2048	1408	64	6	2	no	1,664	176
OLMoE-I	`allenai/OLMoE-1B-7B-0924-Instruct`	R	6.9B / 1.3B	16 (16)	2048	1024	64	8	0	no	1,024	260
Qwen-MoE-C	`Qwen/Qwen1.5-MoE-A2.7B-Chat`	R	14.3B / 2.7B	24 (24)	2048	1408	60	4	1	no	1,440	108
Qwen-MoE	`Qwen/Qwen1.5-MoE-A2.7B`	A	14.3B / 2.7B	24 (24)	2048	1408	60	4	1	no	1,440	108
DS-MoE	`deepseek-ai/deepseek-moe-16b-base`	A	16.4B / 2.8B	28 (27)	2048	1408	64	6	2	no	1,728	176
Mixtral-7	`mistralai/Mixtral-8x7B-v0.1`	A	46.7B / 12.9B	32 (32)	4096	14336	8	2	0	yes	256	56
Mixtral-22	`mistralai/Mixtral-8x22B-v0.1`	A	141B / 39B	56 (56)	6144	16384	8	2	0	yes	448	56
OLMoE	`allenai/OLMoE-1B-7B-0924`	S	6.9B / 1.3B	16 (16)	2048	1024	64	8	0	no	1,024	–

LEPO adds at most 1,904 trainable parameters to a model (DS-MoE), about 10^{-7} of its weights. The budget heads are small because their input is the $(K+4)$-dimensional router summary of Equation equation 3, not the hidden state. On the two Mixtral models, $K = 2$ gives $\mathcal{A} = \{1, 2\}$, so each head chooses between one and two experts. Because Mixtral renormalises the kept weights, a token run with one expert gives that expert weight one whatever the preference vector is; such tokens give r_l no gradient, and the vectors learn only from tokens run with two experts.

A.7 Datasets Statistics

Table 7 summarises the evaluation data of the three protocols. LEPO reads no label from any of these sets. In the Rewiring protocol it adapts on each test prompt and its own generated tokens only.

A.8 Experimental settings

Table 9 contrasts the three protocols. Each follows the paper it is taken from, so the compute measure, the data a method may use, and the baselines differ between tables. Numbers should therefore be compared within a table, not across tables.

A.9 LEPO hyperparameters

Table 10 lists LEPO's hyperparameters. The same values are used for every model and task. All frozen model weights run in bfloat16, while the preference vectors, the budget heads, and all losses are kept in float32.

Table 7: **Evaluation datasets.** #Eval is the number of test items scored. Shots are in-context demonstrations in the prompt. The generation cap is the maximum number of new tokens; "LL" marks tasks scored by log-likelihood over the answer options, which generate nothing.

Protocol	Task	Source	Split	#Eval	Shots	Metric	Gen. cap
Rewiring (Table 1)	HumanEval (**?**)	`openai/openai_humaneval`	test	164	0	pass@1 (unit tests)	512
	MBPP (**?**)	`google-research-datasets/mbpp`	sanitized test	257	0	pass@1 (unit tests)	512
	GSM8K (**?**)	`openai/gsm8k`	main test	1,319	0	exact match	512
	MATH500 (**?**)	`HuggingFaceH4/MATH-500`	test	500	0	exact match (boxed)	1,024
	MMLU-Redux (**?**)	`edinburgh-dawg/mmlu-redux`	30 subjects × 100	3,000	0	accuracy (letter)	512
Ada-K (Table 2)	ARC-Challenge (**?**)	`arc_challenge`	test	1,172	25	acc_norm	LL
	HellaSwag (**?**)	`hellaswag`	validation	10,042	10	acc_norm	LL
	MMLU (**?**)	`mmlu`	test (57 subjects)	14,042	5	acc	LL
	GSM8K (**?**)	`gsm8k`	test	1,319	5	exact match (strict)	256
	TruthfulQA (**?**)	`truthfulqa_mc2`	validation	817	0	mc2	LL
	WinoGrande (**?**)	`winogrande`	validation (XL)	1,267	5	acc	LL
STEP (Table 3)	ARC-Easy (**?**)	`arc_easy`	test	2,376	0	accuracy	LL
	ARC-Challenge (**?**)	`arc_challenge`	test	1,172	0	accuracy	LL
	BoolQ (**?**)	`boolq`	validation	3,270	0	accuracy	LL
	PIQA (**?**)	`piqa`	validation	1,838	0	accuracy	LL
	WinoGrande (**?**)	`winogrande`	validation (XL)	1,267	0	accuracy	LL
	HellaSwag (**?**)	`hellaswag`	validation	10,042	0	accuracy	LL
	MMLU (**?**)	`mmlu`	test (57 subjects)	14,042	0	accuracy	LL
	OpenBookQA (**?**)	`openbookqa`	test	500	0	accuracy	LL

Table 8: **Instruction-tuning mixture used to train Ada-K** (after Table 12 of Yue et al. (2025)). Sources with fewer rows than the per-source quota are taken in full.

Source	Hugging Face dataset	Available	Used
Alpaca-GPT4	`vicgalle/alpaca-gpt4`	52,002	1,660
UltraChat	`HuggingFaceH4/ultrachat_200k` (train_sft)	207,843	1,660
LIMA	`64bits/lima_vicuna_format`	1,030	1,030
Open-Platypus	`garage-bAInd/Open-Platypus`	24,926	1,659
CodeAlpaca	`sahil2801/CodeAlpaca-20k`	20,016	1,659
WikiQA	`microsoft/wiki_qa`	873	873
MathInstruct	`TIGER-Lab/MathInstruct`	262,038	1,659
Total			10,200

Table 9: **Summary of the three evaluation protocols.**

	Rewiring (Su et al., 2025)	Ada-K (Yue et al., 2025)	STEP (**?**)
Question	RQ1: test-time adaptation	RQ2: adaptive budgets at matched compute	RQ3: complement to expert pruning
Models	DS-V2-Lite, OLMoE-I, Qwen-MoE-C	Mixtral-22, Mixtral-7, Qwen-MoE, DS-MoE	OLMoE
Tasks	5 generative (code, math, knowledge)	6 leaderboard (5 LL + GSM8K)	8 zero-shot LL
Harness	chat template, greedy, one sample	lm-evaluation-harness	lm-evaluation-harness
Data used by LEPO	the test prompt and its own generated tokens	see Section **??**	128 unlabelled C4 blocks
Compute reported	$\bar{K}$: mean routed experts per token and layer	Act: mean routed experts per token and layer	experts kept per layer (48 of 64)
Reference line	original model at native K	static Top-k line (Δ_{line})	unpruned model
Baselines	ICL, C3PO, Rewiring, Ada-K	static Top-k, Ada-K	D2-MoE, MC-/HC-SMoE, Frequency, Gate Score, MoNE, STEP

Table 10: **LEPO hyperparameters** (shared by all models and tasks). Equation numbers refer to Section 3.

Symbol	Meaning	Value
Budget policy (Sections 3.2.2, 3.2.4)		
B	consecutive layer groups	4
$k_{\min,b}$	smallest budget	$\max(1, \lfloor K_b/2 \rfloor)$
η	initial off-K probability mass	0.2
G	rollouts per GRPO step	4
–	PPO epochs per GRPO step	4
ε	ratio clip	0.2
–	policy optimiser, learning rate	Adam, 0.1
$\alpha,\ \beta$	entropy bonus, KL to π_{init}	0, 0
δ	advantage standard-deviation floor	10^{-6}
–	credit assignment	per position (Eq. 13)
c_l	per-expert cost	1
$\lambda_c,\ \lambda_o,\ \lambda_h$	reward weights (Eq. 12)	0.1, 10, 0.3
$\epsilon_o,\ \epsilon_h$	fidelity tolerances (dual update, Eq. 16)	off
Expert preferences (Sections 3.2.1, 3.2.3)		
–	initialisation	$r_l = 0$
$w_c,\ w_o,\ w_h$	router-loss weights (Eq. 9)	1, 1, 1
w_r	anchor weight	10^{-3}
τ	distillation temperature	1
–	router optimiser, learning rate, weight decay	AdamW, 0.02, 0
–	hidden states matched	last layer of each group
Schedule (Section A.12)		
–	alternating (policy, router) steps per round	2
–	tokens generated per round (chunk)	50
q	budget-perturbation range in round n	$\min(1+n,\ K-k_{\min})$
–	decoding	greedy
–	state reset	every request

With these values one adaptation round costs one teacher forward, $2G = 8$ policy rollouts, and two router steps over the current context. Counting a backward pass as two forwards, this is $1 + 2(G + 3) = 15$ forward-equivalents per round, before the student re-reads the context to rebuild its key–value cache. The rounds are what make LEPO more expensive end to end than a single native-K pass, which is why we report $\bar{K}$ and Act for the deployed forwards separately from adaptation cost (Section A.2).

A.10 Hardware and software

All experiments run on four H200 (141 GB) GPUs with PyTorch and Hugging Face Transformers. Models that fit on one GPU run one copy per GPU; Mixtral-22 is sharded layer-wise over several GPUs per copy. Requests are independent in the Rewiring protocol, so they are distributed over copies in shards and merged afterwards; the result does not depend on the sharding.

A.11 Implementation

Listing 1 summarises LEPO for one request in Python-style pseudocode. It follows Algorithm 1 and omits engineering details such as key–value caching, batching, and cost accounting. `model.forward(x, prefs, budget)` runs the frozen MoE with `routed_layer` in place of each MoE layer; `prefs=None` with the native budget is the original model, which serves as the teacher.

Listing 1: LEPO (pseudocode).

```
def routed_layer(l, h, prefs, budget):              # one MoE layer, Eqs. equation 2 and equation 5
    z = router[l](h) + center(prefs[l])             # rewired routing logits
    p = softmax(z)
    if l == first_layer_of_group(l):                # one decision per layer group
        budget.k[group(l)] = budget.choose(group(l), state(p))   # state of Eq. equation 3
    k = budget.k[group(l)]
    experts = topk(z, k)                            # only these experts are executed
    return sum(weight(p, experts)[j] * expert[l][j](h) for j in experts) + shared[l](h)

def lepo_generate(model, prompt, K, max_new, B=4, steps=2, G=4, chunk=50):
    prefs  = zeros(L, N)                            # expert preferences: r_l = 0 is the original router
    policy = BudgetPolicy(B, actions=range(max(1, K // 2), K + 1), init_mass_on_K=0.8)
    x = prompt
    while len(x) - len(prompt) < max_new and not ends_with_eos(x):
        teacher = model.forward(x, prefs=None, budget=native(K))        # frozen targets, no gradient
        for _ in range(steps):
            # (A) budget policy, preferences fixed: GRPO over G sampled budget trajectories
            rollouts = [model.forward(x, prefs, policy.sample) for _ in range(G)]
            R = [savings(o) - lam_c * ctx_ce(o, x) - lam_o * kl(teacher, o)
                 - lam_h * hid(teacher, o) for o in rollouts]   # per-position reward, Eq. equation 12
            A = normalize_across_rollouts(R)        # group-relative advantage, Eq. equation 13
            policy.clipped_update(rollouts, A)      # clipped objective, Eq. equation 15
            # (B) expert preferences, policy fixed: one gradient step on the fidelity loss
            o = model.forward(x, prefs, perturbed(policy.sample))   # sampled budgets, capped at random
            loss = (w_c * ctx_ce(o, x) + w_o * kl(teacher, o) + w_h * hid(teacher, o)
                    + w_r * norm2(center(prefs)))   # loss of Eq. equation 9
            prefs = center(prefs - lr * grad(loss, prefs))           # gradient step, then re-centre
        x = x + model.generate(x, prefs, policy.argmax, n_tokens=chunk)   # deploy with fewer experts
    return x[len(prompt):]
```

Listing 2 gives SearchTopK, the oracle used in Figure 1 and described in Section 1. It changes only the number of selected experts; the checkpoint, its router, and its expert weighting are unchanged.

Listing 2: SearchTopK oracle (pseudocode).

```
def search_topk(model, questions, grid, K, max_new):
    assert K in grid and max(grid) == K             # the native budget is always tested
    correct = {}
    for k in grid:
```

Algorithm 1 LEPO: test-time expert preferences and adaptive budgets

Require: Frozen MoE $M_{\theta,K}$, prompt x, layer groups, adaptation and generation budgets

Initialize $\phi = 0$ and $\pi_\psi = \pi_{\text{init}}$ for this request
while generation budget remains and no end-of-sequence token is produced **do**
  Run the native-budget teacher on x; cache detached output and hidden targets
  for each adaptation step **do**
    Freeze ϕ and snapshot $\pi_{\text{old}} \leftarrow \pi_\psi$
    Sample G student budget trajectories on x and compute rewards
    Compute group-relative advantages and update ψ using Equation equation 15
    Update fidelity multipliers if the constrained variant is enabled
    Freeze ψ; run the student with sampled, perturbed budgets
    Update ϕ using Equation equation 9; recenter every r_l
  end for
  Rebuild the student key–value cache on x under the adapted routing
  Generate a chunk using $k_{b,t} = \arg\max_{k \in \mathcal{A}_b} \pi_{\psi_b}(k \mid s_{b,t})$
  Append the generated tokens to x
end while
return Generated continuation

```
        set_topk(model, k)                          # same k in every MoE layer and at every token
        for q in questions:
            y = greedy_generate(model, build_prompt(q), max_new)
            correct[q, k] = extract_choice(y) == q.answer
    set_topk(model, K)
    # smallest correct budget; a question no budget solves is charged the native K
    k_star = {q: min([k for k in grid if correct[q, k]], default=K) for q in questions}
    acc_native = mean(correct[q, K] for q in questions)
    acc_oracle = mean(any(correct[q, k] for k in grid) for q in questions)
    k_bar = mean(k_star.values())
    return acc_native, acc_oracle, k_bar, 1 - k_bar / K          # last value: budget reduction

def set_topk(model, k):
    for gate in moe_gates(model):                   # the attribute each family's router reads
        gate.top_k = k                              # e.g. mlp.top_k, mlp.gate.top_k, mlp.router.top_k
```

A.12 LEPO Algorithm

LEPO resets the router vectors and policies for each independent request. During an adaptation round, it caches the teacher distributions and group-boundary representations on the current context, then alternates between policy updates with ϕ fixed and router-vector updates with ψ fixed. Fresh trajectories are collected after each vector update because changing expert preferences changes both policy features and student behavior.

Router updates include reduced budgets. At zero initialization and native K, teacher and student coincide, so the two fidelity losses provide no corrective signal. We therefore sample budgets from the current policy and cap them with a random budget from $\{\max(k_{\min,b}, K_b - q), \ldots, K_b\}$. The perturbation range $q \geq 1$ grows across rounds toward the allowed minimum budget. We take a gradient step on $\mathcal{L}_r$ and recenter each vector afterward.

After adaptation, the student uses deterministic argmax budgets to generate the next chunk. We rebuild its key–value cache under the updated routing so that cached states correspond to the adapted model. Generated tokens extend the context for the next round; no reference continuation is used. Teacher computation is needed during adaptation, while chunk generation uses only the student.

Equation equation 10 measures routed-expert savings for a student forward. End-to-end cost also includes teacher passes, policy rollouts, router backpropagation, and repeated prefills. These costs must be accounted for separately from generation-time expert savings, which alone do not imply lower total computation or latency.