TRUST THE ROUTER: ADAPTIVE RANK-SPACE SPARSIFICATION FOR EFFICIENT MIXTURE-OF-EXPERTS

Anonymous authors
Paper under double-blind review

ABSTRACT

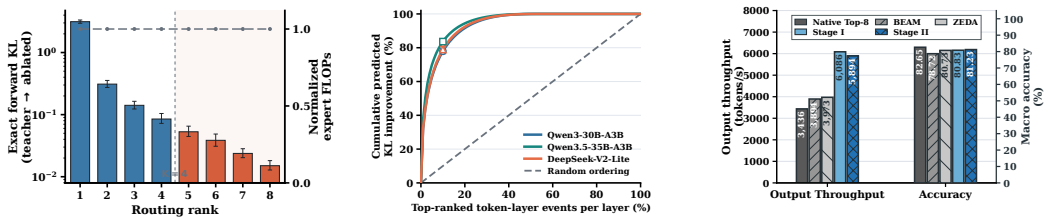
Mixture-of-Experts (MoE) models already learn to rank experts through their pretrained routers, yet existing expert-reduction methods often underutilize this existing routing knowledge when reducing computation. We argue that efficient MoE inference should leverage the expert-ranking structure encoded by the pretrained router, rather than selecting experts without explicitly preserving the ranked prefix structure. The central question is no longer merely which experts to select, but how many of the top-ranked experts should actually be executed for each token. Based on this insight, we propose TRUST-MoE, a coarse-to-fine framework that preserves the ranking structure provided by the pretrained router while learning adaptive expert execution. In the first stage, we introduce continuous rank-wise masks together with multi-level knowledge distillation, gradually adapting the original Top-K model toward a shorter Top-$\tilde{K}$ routing prefix instead of abruptly truncating lower-ranked experts. In the second stage, with the adapted model frozen, a lightweight boundary predictor estimates the counterfactual utility of the next-ranked expert and dynamically decides whether it should be restored for each token-layer pair. This reduces dynamic expert allocation to a simple truncate-or-restore decision, without searching over arbitrary expert subsets. In this way, the MoE itself determines whether a token requires Top-$\tilde{K}$ or Top-$(\tilde{K}+1)$ computation. Experiments across multiple MoE model families and downstream benchmarks show that TRUST-MoE generally improves performance over static expert-reduction and dynamic expert-skipping baselines while maintaining improved inference efficiency.

1 INTRODUCTION

Mixture-of-Experts (MoE) language models scale model capacity by routing each token to only a small subset of experts (Shazeer et al., 2017; Artetxe et al., 2021; Yang et al., 2024). However, the activated Top-K experts still dominate the computation of each MoE layer, making expert activation a major source of inference cost. Therefore, reducing the number of executed experts provides a direct way to lower the inference cost of MoE models (He et al., 2023; Jin et al., 2026). Importantly, the pretrained router provides not only expert selection but also a meaningful ranking: as shown in Figure 1a, removing higher-ranked experts causes substantially larger output discrepancy despite identical per-expert cost. This motivates preserving the pretrained ranking while reducing the number of executed experts. Yet directly switching from Top-K to Top-$\tilde{K}$ can cause substantial degradation (Chen & Yao, 2026; Wang et al., 2025). Moreover, Figure 1b shows that the benefit of restoring additional computation is concentrated in only a small fraction of token–layer pairs.

Existing expert-reduction methods can be broadly categorized into training-free and training-based approaches. Training-free methods directly modify the expert execution strategy of a pretrained MoE without updating its parameters. For example, fixed Top-$\tilde{K}$ uniformly reduce the activated experts number, LExI (Chitty-Venkata et al., 2025) assigns different expert budgets across layers. NAEE (Lu et al., 2024) dynamically skips less important experts based on their relative routing weights and layer-specific thresholds, while MC-MoE (Huang et al., 2025) and ACE (Xu et al., 2026) dynamically determine which expert computations can be skipped during inference. In contrast, training-based methods, such as AdaMoE (Zeng et al., 2024), DynMoE (Guo et al., 2025), BEAM (Wu et al., 2026), and ZEDA (Lv et al., 2026), introduce additional optimization to learn

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.



(a) Rank contribution and cost. (b) Concentrated boundary utility. (c) Throughput–accuracy trade-off.

Figure 1: **Empirical observations motivating TRUST-MoE.** (a) On Qwen3-30B-A3B, ablating higher-ranked routing positions produces substantially larger exact forward KL divergence, despite identical normalized expert FLOPs across ranks. This motivates preserving the pretrained router ordering when reducing expert activation. (b) The predicted refinement utility is highly concentrated across token–layer pairs: the top 10% of events account for 77.9–83.6% of the cumulative predicted KL improvement across three MoE models, motivating selective rather than uniform boundary refinement. (c) On Qwen3-30B-A3B, Stage I and Stage II substantially increase output throughput over Native Top-8, while maintaining higher accuracy than BEAM and ZEDA. Stage II retains most of the throughput gain of Stage I while recovering additional accuracy.

token-dependent expert allocation or dynamical skipping policies. Despite their different mechanisms, these approaches primarily focus on determining which experts to execute or how many experts to allocate, rather than explicitly exploiting the ranked structure already provided by the router.

To address these limitations, we propose **TRUST-MoE**, a two-stage coarse-to-fine framework that reduces expert computation while preserving the ranking structure learned by the pretrained router. In Stage I, TRUST-MoE introduces continuous learnable masks over routing ranks and combines them with task supervision and multi-level distillation, gradually adapting the original Top-K model to a shorter Top-$\tilde{K}$ routing prefix rather than abruptly truncating lower-ranked experts. However, a fixed Top-$\tilde{K}$ prefix can not be equally sufficient for all token-layer pairs. Stage II therefore freezes the adapted model and performs a lightweight refinement at the truncation boundary: a predictor estimates the counterfactual utility of the next-ranked expert and determines whether the Rank-$(\tilde{K}+1)$ expert should be restored. In this way, TRUST-MoE retains the pretrained expert ordering while restricting token-wise adaptation to a simple choice between Top-$\tilde{K}$ and Top-$(\tilde{K}+1)$ execution. As shown in Figure 1c, this design translates into substantial serving-efficiency gains while preserving strong downstream accuracy. Our main contributions are as follows:

- We introduce a new perspective on reducing active experts in MoE models: instead of relearning expert selection, we preserve the ranking encoded by the router and determine how many top-ranked experts to activate for each token. We then adopt a coarse-to-fine strategy that establishes a compact default execution pattern and selectively adds computation only for token–layer pairs that require further refinement.
- We propose **TRUST-MoE**, a two-stage framework that preserves the expert ordering produced by the pretrained router. In Stage I, continuous rank-wise masks and multi-level distillation gradually adapt Top-K execution to Top-$\tilde{K}$. In Stage II, a lightweight predictor determines when one-step adjustment is beneficial for each token-layer pair, without retraining the router or searching over alternative expert subsets.
- We conduct extensive evaluations across diverse MoE architectures, downstream tasks, and expert-activation budgets. TRUST-MoE improves inference speed while preserving strong performance, and achieves a more competitive performance-efficiency trade-off than existing fixed and dynamic expert-reduction methods.

2 Related Work

Mixture-of-Experts for LLMs. Mixture-of-Experts (MoE) models scale model capacity through sparse conditional computation, where a router activates only a small subset of experts for each

token (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022; Du et al., 2022). Early large-scale sparse language models, such as GLaM (Du et al., 2022) and ST-MoE (Zoph et al., 2022), demonstrated that MoE architectures can substantially expand model capacity without a proportional increase in per-token computation. This design has been widely adopted in recent MoE LLMs, including Mixtral (Jiang et al., 2024), DeepSeekMoE (Dai et al., 2024), DeepSeek-V2 (Liu et al., 2024a), DeepSeek-V3 (Liu et al., 2024b), Qwen3 (Yang et al., 2025), OpenMoE (Xue et al., 2024) and OLMoE (Muennighoff et al., 2025). By activating only a small subset of top-ranked experts from expert pool, these models substantially reduce per-token computation, making the number of activated experts a key determinant of MoE inference efficiency.

Expert Reduction in Mixture-of-Experts. Although sparse MoE models activate only a small subset of experts for each token, the executed experts still account for a substantial portion of the computation in each MoE layer. Reducing the number of executed experts therefore provides a direct way to further lower MoE inference cost. Training-free methods modify expert execution without updating the pretrained model. LExI (Chitty-Venkata et al., 2025) allocates different expert budgets across layers, Alloc-MoE (Liu et al., 2026) coordinates expert activation budgets across layers and tokens, while MC-MoE (Huang et al., 2025) and ACE (Xu et al., 2026) perform token-adaptive expert skipping based on importance estimates. In contrast, training-based methods learn adaptive expert execution through additional optimization. DynMoE (Guo et al., 2025) replaces fixed Top-K routing with dynamic expert activation, BEAM (Wu et al., 2026) learns token-dependent binary masks over routed experts, and ZEDA (Lv et al., 2026) introduces zero-output experts and self-distillation to convert post-trained MoE models into dynamic ones. Different from these approaches, TRUST-MoE retains the ranked Top-K routing structure and formulates activated expert reduction as routing-prefix adaptation. Stage I jointly adapts the router and the model to a shorter routing prefix, while Stage II restricts token-wise flexibility to the truncation boundary, deciding only whether to execute the next-ranked expert rather than dynamically reselecting experts from the full expert pool.

3 METHOD

3.1 OVERVIEW

TRUST-MOE adopts a coarse-to-fine two-stage strategy to reduce expert activation while retaining the rank-based structure of Top-K routing. Stage I progressively adapts the model to a shorter Top-$\tilde{K}$ routing prefix, establishing a strong default execution pattern. Building on this adapted model, Stage II performs token-wise refinement at the truncation boundary, deciding only whether the Rank-$(\tilde{K}+1)$ expert should be additionally executed. As a result, the model follows Top-$\tilde{K}$ execution for most token-layer pairs and introduces the extra expert only when it is predicted to be beneficial. Figure 2 summarizes the overall workflow of our method.

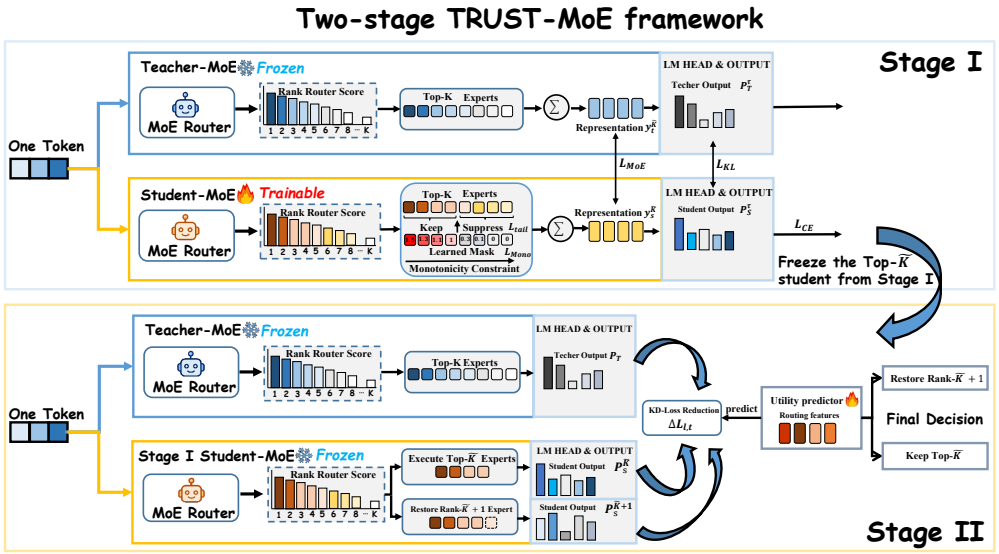


Figure 2: Overview of the two-stage TRUST-MoE framework.

3.2 STAGE I: RANK-WISE PREFIX ADAPTATION

Stage-I training objective. Stage I adapts the original pretrained Top-K model toward Top-$\tilde{K}$ computation while preserving the behavior learned by an SFT-trained teacher. Specifically, we use the SFT-trained Top-K model as the frozen teacher, while the student is initialized from the original pretrained Top-K model. The student is optimized with a masking objective together with distillation losses on the output distributions and intermediate representations. The overall Stage-I objective can be represented as:

$$\mathcal{L}_{\text{Stage,I}} = \mathcal{L}_{\text{distill}} + \mathcal{L}_{\text{mask}}. \tag{1}$$

Learnable rank-wise masking. To progressively reduce the number of experts executed for each token, we introduce learnable scaling coefficients for different positions in the Top-K routing order. For each token, the K activated experts are ranked by their router scores, and experts at the same routing rank share the same coefficient across tokens. In this way, even when different tokens activate different experts, their contributions are consistently modulated according to their relative routing ranks. For the ℓ-th MoE layer, we can define a learnable rank-wise mask vector as:

$$\tilde{\mathbf{m}}_\ell = \text{ReLU}[m_\ell^1, \ldots, m_\ell^K] \in \mathbb{R}^K, \tag{2}$$

where $\tilde{m}_\ell^r$ denotes the non-negative scaling coefficient applied to the expert occupying the r-th routing position. Accordingly, the output of the ℓ-th MoE layer is given by:

$$\widetilde{\mathbf{y}}_\ell(x) = \sum_{r=1}^{K} \tilde{m}_\ell^r p_\ell^r(x) E_\ell^r(x), \tag{3}$$

where $p_\ell^r(x)$ denotes the router weight associated with the expert at routing rank r. All rank-wise masks are initialized to 1, such that the student initially retains the original Top-K computation. During training, these coefficients are progressively adjusted, allowing the contributions of lower-ranked routing positions to be gradually suppressed rather than removed abruptly.

Tail-rank suppression and order regularization. While the continuous rank-wise masks provide a smooth mechanism for adjusting the contribution of each routing position, they alone do not guarantee that the learned masks converge to the desired Top-$\tilde{K}$ prefix structure. To explicitly guide the masks toward a prefix-shaped solution, we introduce two complementary regularization terms. First, we suppress the routing positions beyond the target prefix by penalizing the overall magnitude of their mask values:

$$\mathcal{L}_{\text{tail}} = \frac{1}{L} \sum_{\ell=1}^{L} \sqrt{\frac{1}{K - \tilde{K}} \sum_{r=\tilde{K}+1}^{K} \tilde{m}_{\ell,r}^2 + \epsilon}, \tag{4}$$

where $\epsilon = 10^{-12}$ is a small constant. This term progressively suppresses routing positions beyond the target Top-$\tilde{K}$ prefix. Besides, to preserve the ranking induced by the router, we impose a monotonicity constraint on the rank-wise masks:

$$\mathcal{L}_{\text{mono}} = \frac{1}{L(K-1)} \sum_{\ell=1}^{L} \sum_{r=1}^{K-1} \left[\text{ReLU}\left(\tilde{m}_{\ell,r+1} - \tilde{m}_{\ell,r}\right)\right]^2. \tag{5}$$

This term discourages a lower-ranked routing position from receiving a larger retention coefficient than a higher-ranked one, thereby keeping the learned mask structure consistent with the original router ordering. Therefore, the overall mask regularization is

$$\mathcal{L}_{\text{mask}} = \lambda_{\text{mask}} s(t) \left(\lambda_{\text{tail}} \mathcal{L}_{\text{tail}} + \lambda_{\text{mono}} \mathcal{L}_{\text{mono}}\right), \tag{6}$$

where $s(t)$ gradually increases from 0 to 1, progressively enforcing the prefix constraint during training. Together, these terms guide the rank-wise masks toward a Top-$\tilde{K}$ prefix structure. Therefore, we further incorporate multi-level knowledge distillation to progressively compensate for the impact of reduced lower-ranked expert contributions on the student during training.

Multi-level distillation for student-teacher alignment. Mask regularization progressively reduces the student's reliance on lower-ranked experts, but this structural adaptation can also shift the intermediate representations and final predictions of the student model. To mitigate this shift, we supervise the student at both the output-distribution and intermediate MoE representation levels, encouraging it to remain aligned with the teacher as expert contributions are progressively reduced. Specifically, we optimize the following objective:

$$\mathcal{L}_{\text{distill}} = \lambda_{\text{CE}}\mathcal{L}_{\text{CE}} + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}} + \lambda_{\text{MoE}}\mathcal{L}_{\text{MoE}}. \tag{7}$$

The output-level distillation loss $\mathcal{L}_{\text{KL}}$ is

$$\mathcal{L}_{\text{KL}} = \tau^2 D_{\text{KL}}\left(P_T^{(\tau)} \| P_S^{(\tau)}\right), \tag{8}$$

where τ is the distillation temperature, and $P_T^{(\tau)}$ and $P_S^{(\tau)}$ denote the teacher and student output distributions softened by temperature τ, respectively. Because the rank-wise masks act inside each MoE layer, we additionally match the corresponding post-MoE hidden states by using the equation:

$$\mathcal{L}_{\text{MoE}} = \frac{1}{L}\sum_{\ell=1}^{L} \left\| \mathbf{y}_\ell^T - \widetilde{\mathbf{y}}_\ell^S \right\|_2^2, \tag{9}$$

Combining the distillation terms with the mask regularization gives:

$$\mathcal{L}_{\text{Stage,I}} = \lambda_{\text{CE}}\mathcal{L}_{\text{CE}} + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}} + \lambda_{\text{MoE}}\mathcal{L}_{\text{MoE}} + \lambda_{\text{mask}} s(t) \left(\mathcal{L}_{\text{tail}} + \lambda_{\text{mono}}\mathcal{L}_{\text{mono}}\right). \tag{10}$$

During Stage I, all routing candidates remain active so that every rank-wise mask receives gradients. Therefore, Stage I gradually reduces the contribution of lower-ranked experts while training the student to match the SFT teacher.

3.3 Stage II: Token-Wise Boundary Refinement

For most tokens, the selected experts are already sufficient to closely match the teacher, while a small subset can still benefit from one further refinement. Stage II therefore performs a *one-step refinement* of the Stage-I decision, determining for each token-layer pair whether the next-ranked expert should also be executed.

Token-wise one-step refinement. We retain the expert ranking produced by the Stage-I student router and restrict refinement to the next-ranked expert rather than reselecting from the remaining expert pool. Specifically, the Rank-$(\tilde{K}+1)$ expert is the highest-ranked expert outside the current Top-$\tilde{K}$ set, making it the natural candidate for one-step refinement. Its weighted contribution is given by:

$$\boldsymbol{\delta}_{l,t}^{\tilde{K}+1} = p_{l,t}^{\tilde{K}+1} E_{l,t}^{\tilde{K}+1}(x), \tag{11}$$

where $p_{l,t}^{\tilde{K}+1}$ and $E_{l,t}^{\tilde{K}+1}$ denote the routing weight and expert at Rank-$(\tilde{K}+1)$, respectively. When this refinement is applied, the MoE output $\mathbf{y}_{l,t}^{\tilde{K}+1}$ becomes:

$$\mathbf{y}_{l,t}^{\tilde{K}+1} = \mathbf{y}_{l,t}^{\tilde{K}} + \boldsymbol{\delta}_{l,t}^{\tilde{K}+1}. \tag{12}$$

Counterfactual utility of one-step refinement. For each token-layer pair (l,t), we further consider a counterfactual case in which the Rank-$(\tilde{K}+1)$ expert is additionally executed. Specifically, the corresponding MoE output is changed from $\mathbf{y}_{l,t}^{\tilde{K}}$ to $\mathbf{y}_{l,t}^{\tilde{K}} + \boldsymbol{\delta}_{l,t}^{\tilde{K}+1}$. We then define the utility of the Rank-$(\tilde{K}+1)$ expert as the resulting reduction in KD loss:

$$\Delta\mathcal{L}_{l,t} = \Phi_{l,t}\left(\mathbf{y}_{l,t}^{\tilde{K}}\right) - \Phi_{l,t}\left(\mathbf{y}_{l,t}^{\tilde{K}} + \boldsymbol{\delta}_{l,t}^{\tilde{K}+1}\right), \tag{13}$$

where

$$\Phi_{l,t}(\mathbf{y}) = D_{\text{KL}}\left(P_T \| F_{l,t}(\mathbf{y})\right). \tag{14}$$

$F_{l,t}(\mathbf{y})$ denotes the final output distribution obtained by replacing the MoE output of token t at layer l with $\mathbf{y}$. Computing this quantity exactly for every token-layer pair requires a separate downstream forward pass for each candidate refinement, resulting in substantial computational overhead. We approximate the second term using a first-order Taylor expansion around the current Top-$\tilde{K}$ execution:

$$\Phi_{l,t}\left(\mathbf{y}_{l,t}^{\tilde{K}}+\boldsymbol{\delta}_{l,t}^{\tilde{K}+1}\right)\approx\Phi_{l,t}\left(\mathbf{y}_{l,t}^{\tilde{K}}\right)+\left\langle\frac{\partial\Phi_{l,t}}{\partial\mathbf{y}_{l,t}^{\tilde{K}}},\boldsymbol{\delta}_{l,t}^{\tilde{K}+1}\right\rangle. \tag{15}$$

Substituting this Formulation (15) into the Equation (13) for $\Delta\mathcal{L}_{l,t}$, we obtain the following formulation and define the approximate utility of the Rank-$(\tilde{K}+1)$ expert as:

$$\Delta\mathcal{L}_{l,t}\approx-\left\langle\frac{\partial\Phi_{l,t}}{\partial\mathbf{y}_{l,t}^{\tilde{K}}},\boldsymbol{\delta}_{l,t}^{\tilde{K}+1}\right\rangle=u_{l,t}. \tag{16}$$

We define $u_{l,t}$ as the approximate utility of the Rank-$(\tilde{K}+1)$ expert. A larger value indicates a greater expected reduction in KD loss from executing this additional expert, suggesting that the corresponding token-layer pair is more likely to benefit from one-step refinement.

Utility-guided selective refinement. Although $u_{l,t}$ estimates the reduction in KD loss obtained by executing the Rank-$(\tilde{K}+1)$ expert, computing it requires the actual candidate-expert output $\boldsymbol{\delta}_{l,t}^{\tilde{K}+1}$. It therefore cannot be directly used to make the refinement decision at inference process. Instead, we compute $u_{l,t}$ only during offline calibration and use it as a supervision signal to train a lightweight predictor that estimates the refinement utility from routing information available before the candidate expert is executed. For each token-layer pair (l,t), we can construct the routing descriptor feature:

$$\mathbf{r}_{l,t}=\left[p_{l,t}^{\tilde{K}+1},\ \tilde{m}_l^{\tilde{K}}p_{l,t}^{\tilde{K}}-p_{l,t}^{\tilde{K}+1},\ \sum_{j=1}^{\tilde{K}}\tilde{m}_l^j p_{l,t}^j,\ \tilde{m}_l^{\tilde{K}+1}p_{l,t}^{\tilde{K}+1}\right]. \tag{17}$$

The four components respectively capture the routing weight of the candidate Rank-$(\tilde{K}+1)$ expert, the routing-weight gap across the truncation boundary, the cumulative routing mass covered by the current Top-$\tilde{K}$ expert, and the RankMask coefficient learned at Rank-$(\tilde{K}+1)$. The routing weights are taken from the original router outputs before truncation or renormalization. Based on these features, we use a lightweight linear predictor to estimate the refinement utility:

$$\hat{u}_{l,t}=\boldsymbol{\theta}^\top\mathbf{r}_{l,t}+c_l, \tag{18}$$

where $\boldsymbol{\theta}\in\mathbb{R}^4$ denotes the feature weights shared across all MoE layers, and c_l is a layer-specific bias. The predictor is trained to regress the first-order utility $u_{l,t}$ obtained during offline calibration:

$$\mathcal{L}_{\text{pred}}=\frac{1}{N}\sum_{l,t}\left(\hat{u}_{l,t}-u_{l,t}\right)^2, \tag{19}$$

where N denotes the total number of token-layer pairs used for predictor training. In this way, $\hat{u}_{l,t}$ estimates the expected reduction in KD loss without requiring the Rank-$(\tilde{K}+1)$ expert to be executed beforehand. At inference process, we determine a threshold q_ρ such that approximately a fraction ρ of token-layer pairs are selected for one-step refinement. The one-step refinement decision is then given by:

$$z_{l,t}=\mathbb{I}\left[\hat{u}_{l,t}\geq q_\rho\right]. \tag{20}$$

Therefore, the resulting MoE output can be represented as:

$$\hat{\mathbf{y}}_{l,t}=\mathbf{y}_{l,t}^{\tilde{K}}+z_{l,t}\boldsymbol{\delta}_{l,t}^{\tilde{K}+1}. \tag{21}$$

When $z_{l,t}=0$, the token-layer pair retains the Stage-I Top-$\tilde{K}$ execution; when $z_{l,t}=1$, the routing prefix is extended by one position to include the Rank-$(\tilde{K}+1)$ expert.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Models and Training Data. We evaluate different methods on different representative MoE model such as Qwen3.5-35B-A3B, Qwen3-30B-A3B, and DeepSeekV2-Lite. To provide a diverse set of training data, we collect a mixture of around 80K samples from NVIDIA Nemotron Post-Training Dataset (Basant et al., 2025) and CodeAlpaca (Chaudhary, 2023), which covering reasoning, coding, STEM, and general instruction-following tasks. For training-based methods, including ZEDA, BEAM, and our approach, we use the same constructed dataset for training to ensure a fair comparison across these methods.

Baselines. We compare TRUST-MoE with representative activate expert reduction baselines. Top-K Pruning directly reduces the number of activated experts, while Top-K Distillation further applies knowledge distillation to the activated expert pruned model. For MC-MoE, we use only its Online Dynamic Pruning component to isolate dynamic expert skipping. For ZEDA, we evaluate its first training stage and calibrate the shared logit bias of the zero expert so that its average number of activated real experts matches the target computation budget. For our method, we set the tail mask coefficients to zero after training to enforce the predefined expert-activation budget. Besides, all training-based methods are trained for a single epoch using the same training data. Moreover, we fine-tune all model parameters for ZEDA and BEAM, whereas our method updates only the parameters associated with the MoE modules.

Datasets and Evaluation. We evaluate our method on a diverse set of language understanding, reasoning, instruction-following, and code-generation datasets, including MMLU (Hendrycks et al., 2020), C-Eval (Huang et al., 2023), CMMLU (Li et al., 2024), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), GSM8K (Cobbe et al., 2021), IFEval (Zhou et al., 2023), HumanEval (Chen et al., 2021), and MATH (Hendrycks et al., 2021). For choice-based benchmarks, including MMLU(5-shot), C-Eval(5-shot), CMMLU(5-shot), BoolQ(0-shot), and HellaSwag(0-shot), we use the LM Evaluation Harness for evaluation. For generation-based datasets, including GSM8K(5-shot), IFEval(0-shot), HumanEval(0-shot), and MATH(4-shot), we use EvalScope with vLLM and access the models through the OpenAI Chat Completions interface. All models use their native chat templates with thinking mode disabled. The maximum generation length is set to 4096 tokens for GSM8K, IFEval, MATH and HumanEval.

Hyperparameters Setting. In Stage I, we use AdamW optimization, with learning rates of 1×10^{-5} and 5×10^{-2} for the trainable MoE-related parameters and rank-wise mask parameters. $\lambda_{\mathrm{CE}}, \lambda_{\mathrm{KL}}, \lambda_{\mathrm{MoE}}$ are 1.0 and temperature τ is 2.0. The tail-sparsity λ_{tail} and monotonicity weights λ_{mono} are set to 1.0 and 0.1. The overall RankMask regularization coefficient λ_{mask} is set to 1.0 for Qwen3-30B-A3B and 0.5 for Qwen3.5-35B-A3B and DeepSeek-V2-Lite. In Stage II, we freeze all Stage-I model parameters and optimize only the lightweight boundary-admission gate. We use 512 training sequences from the training dataset and train the gate with AdamW using a learning rate of 3×10^{-4}. All experiments are conducted on NVIDIA A800 GPUs with 80 GB memory.

4.2 RESULTS ON LLMS

Choice Benchmarks Results. As shown in Table 1, directly reducing the number of activated experts through Top-K pruning leads to clear performance degradation on both models, suggesting that the pretrained models do not readily accommodate an abrupt reduction in expert activation. Standard distillation partially recovers the lost performance. In contrast, our Stage-I model maintains strong performance under a strict four-expert budget. Notably, on Qwen3.5-35B-A3B, Stage I achieves an average score of 84.70, only 0.31 points below the original Top-8 model (85.01) while halving the number of activated experts. Building on this adapted model, Stage II further improves the overall performance on both backbones with only a marginal increase in the average number of activated experts. These results highlight the effectiveness of progressive rank-wise adaptation for establishing a strong compact execution, while selective one-step refinement further improves performance with limited additional computation.

Generation Task Results. We further evaluate generation performance on GSM8K, MATH, IFEval, and HumanEval. As shown in Table 2, Stage I achieves the strongest average performance

Table 1: Choice benchmarks results of different activated expert reduction methods. Bold and underlined values indicate the best and second-best results among expert-reduction methods, respectively.

Model	Method	Activate Expert	MMLU	CEVAL	CMMLU	BoolQ	HellaSwag	Avg
QWEN3.5-35B-A3B	ORIGNAL	8.00	84.83	84.60	85.29	87.89	82.45	85.01
	ORIGNAL+SFT(TEACHER)	8.00	84.96	85.14	85.56	88.72	82.66	85.41
	TOP-K PRUNING	4.00	74.95	82.32	83.00	84.47	81.17	81.18
	TOP-K DISTILLATION	4.00	77.61	**84.25**	84.25	87.16	**81.47**	82.95
	MC-MOE	4.49	63.48	75.93	76.95	75.50	78.62	74.10
	LEXI	4.00	69.32	79.57	80.57	83.15	59.55	74.43
	ACE	4.02	66.15	79.49	81.58	82.14	80.77	78.03
	NAEE	4.08	71.37	81.95	83.06	83.52	81.38	80.26
	BEAM	4.25	78.05	75.93	77.94	88.44	80.05	80.08
	ZEDA	4.12	**84.28**	82.69	84.36	88.17	80.80	84.06
	OURS(STAGE I)	4.00	84.23	83.95	84.84	89.57	80.93	84.70
	OURS(STAGE II)	4.11	84.24	83.65	**85.02**	**90.06**	80.97	**84.79**
QWEN3-30B-A3B	ORIGNAL	8.00	79.68	83.80	83.73	88.50	77.54	82.65
	ORIGNAL+SFT(TEACHER)	8.00	80.05	83.88	84.14	88.04	78.04	82.83
	TOP-K PRUNING	4.00	74.28	77.12	78.34	82.72	73.38	77.17
	TOP-K DISTILLATION	4.00	75.91	78.53	80.30	84.77	74.98	78.90
	MC-MOE	4.28	65.35	64.12	64.23	73.49	69.22	67.28
	LEXI	4.00	67.18	67.31	70.35	77.46	69.21	70.30
	ACE	3.91	72.06	73.40	74.36	83.09	72.05	74.99
	NAEE	3.90	73.75	74.44	75.95	82.66	73.16	75.99
	BEAM	3.99	75.89	76.60	76.98	85.78	**78.35**	78.72
	ZEDA	3.98	77.51	81.87	82.19	**87.31**	74.77	80.73
	OURS(STAGE I)	4.00	77.68	82.39	82.53	86.64	74.91	80.83
	OURS(STAGE II)	4.12	**78.10**	**83.06**	**82.70**	86.85	75.46	**81.23**

among the compared expert-reduction methods on both Qwen models under comparable activation budgets. Notably, on Qwen3.5-35B-A3B, Stage I reaches an average score of 92.46 with only four activated experts, surpassing the original Top-8 model (91.76) while halving the expert-activation budget. Building on this compact execution, Stage II further improves the average performance with only a small increase in expert activation.

Table 2: Generation task performance of different methods in the Qwen MoE family.

Model	Method	Activate Expert	GSM8K	IFEval	HumanEval	Math	Avg
QWEN3.5-35B-A3B	ORIGNAL	8.00	96.66	85.58	93.29	91.52	91.76
	ORIGNAL+SFT(TEACHER)	8.00	96.51	85.40	95.12	93.56	92.65
	TOP-K PRUNING	4.00	95.83	80.78	89.02	85.58	87.80
	TOP-K DISTILLATION	4.00	96.29	83.36	92.07	92.52	91.06
	ACE	4.33	94.69	77.45	91.46	84.28	86.97
	NAEE	4.27	95.30	80.59	86.59	86.24	87.18
	BEAM	4.15	94.47	65.43	89.02	83.74	83.17
	ZEDA	4.02	93.18	64.88	86.59	78.80	80.86
	OURS(STAGE I)	4.00	96.29	85.40	94.51	**93.62**	92.46
	OURS(STAGE II)	4.10	**96.59**	**85.58**	**95.12**	93.52	**92.70**
QWEN3-30B-A3B	ORIGNAL	8.00	94.69	84.10	92.07	85.32	89.05
	ORIGNAL+SFT(TEACHER)	8.00	95.00	83.92	92.68	85.54	89.29
	TOP-K PRUNING	4.00	90.83	73.57	81.71	81.12	81.81
	TOP-K DISTILLATION	4.00	90.90	74.68	82.93	80.40	82.23
	ACE	4.18	88.55	71.90	76.83	80.70	79.50
	NAEE	4.16	89.08	75.23	75.00	80.20	79.88
	BEAM	3.99	89.31	65.99	78.66	63.02	74.25
	ZEDA	4.01	91.28	76.16	84.15	80.44	83.01
	OURS(STAGE I)	4.00	**93.71**	80.04	84.76	80.94	84.86
	OURS(STAGE II)	4.09	**93.71**	**81.33**	**85.98**	**82.18**	**85.80**

Different Architecture MoE Model. To evaluate whether our approach generalizes beyond the Qwen MoE family, we further conduct experiments on DeepSeek-V2-Lite, reducing its default expert activation from 6 to approximately 3 experts per token. As shown in Table 3, direct Top-K pruning and NAEE suffer substantial performance degradation, indicating that DeepSeek-V2-Lite is particularly sensitive to abrupt reductions in activate experts. MC-MoE and ZEDA preserve substantially more of the original model performance. In contrast, TRUST-MOE retains substantially more of the original model capability under a comparable activation budget. The strong performance on DeepSeek-V2-Lite suggests that the proposed routing adaptation is not specific to the Qwen MoE family and remains effective across different MoE architectures under an approximately halved expert-activation budget.

Table 3: Performance of different activated expert reduction methods on DeepSeek-V2-Lite.

Model	Method	Activate Expert	MMLU	CEVAL	CMMLU	BoolQ	HellaSwag	Avg
DEEPSEEK-V2-LITE	ORIGNAL	6.00	58.04	60.55	64.07	79.91	77.93	68.10
	ORIGNAL+SFT(TEACHER)	6.00	57.56	60.33	63.81	80.28	78.80	68.16
	TOP-K PRUNING	3.00	24.81	24.37	24.65	44.71	35.71	30.85
	NAEE	3.00	24.33	25.93	25.82	44.62	38.82	31.90
	LEXI	3.00	55.55	56.61	61.83	73.98	71.40	63.87
	MC-MOE	3.00	52.02	53.71	56.26	68.75	74.75	61.10
	ZEDA	3.01	55.31	55.13	60.79	**78.59**	75.21	65.01
	OURS(STAGE I)	3.00	55.73	55.65	61.48	76.79	76.66	65.26
	OURS(STAGE II)	3.12	**56.12**	**57.50**	**61.90**	78.35	**77.15**	**66.20**

Effectiveness of Utility-Guided Refinement. We further compare Stage II with random Rank-$(\tilde{K}+1)$ restoration under matched average expert-activation budgets. As shown in Table 4, utility-guided refinement consistently outperforms random restoration across all evaluated budgets and benchmarks. For example, with 4.12 activated experts on average, the average score improves from 80.89 to 81.23. This consistent advantage under matched computation budgets supports the effectiveness of selectively allocating additional expert computation to token-layer pairs with higher predicted refinement utility.

Table 4: Effectiveness of Utility-Guided Refinement on Qwen3-30B-A3B.

Model	Method	Activate Expert	MMLU	CEVAL	CMMLU	BoolQ	HellaSwag	Avg
QWEN3-30B-A3B	RANDOM	4.12	77.82	82.32	82.52	86.61	75.16	80.89
	OURS(STAGE II)	4.12	**78.10**	**83.06**	**82.70**	**86.85**	**75.46**	**81.23**
	RANDOM	4.25	78.12	82.69	82.58	86.88	75.58	81.17
	OURS(STAGE II)	4.25	**78.14**	**83.06**	**82.80**	**86.94**	**75.67**	**81.32**
	RANDOM	4.37	78.32	82.32	82.72	86.54	75.69	81.12
	OURS(STAGE II)	4.37	**78.41**	**82.99**	**82.84**	**87.00**	**75.94**	**81.44**

Results with Fewer Activated Experts. To further evaluate different methods under a more constrained computation budget, Table 5 reduces the activation width to approximately two activated experts per token. Under this aggressive setting, several expert-reduction baselines experience pronounced performance degradation, particularly on Qwen3-30B-A3B. In contrast, TRUST-MoE remains considerably more robust, achieving the best performance on Qwen3.5-35B-A3B and Qwen3-30B-A3B. This behavior highlights the robustness of the our proposed method as the expert-activation budget becomes increasingly constrained.

Table 5: Performance of different methods with fewer activated experts.

Model	Method	Activate Expert	MMLU	CEVAL	CMMLU	BoolQ	HellaSwag	Avg
QWEN3.5-35B-A3B	TOP-K PRUNING	2.00	47.77	63.37	66.63	69.08	69.78	63.33
	NAEE	2.07	50.22	62.18	63.89	68.90	69.79	63.00
	ZEDA	2.03	**82.44**	**81.13**	**82.29**	79.85	73.67	79.88
	LEXI	2.00	60.76	64.56	66.28	69.57	69.68	66.17
	OURS(STAGE I)	2.00	81.34	80.98	82.13	83.61	75.98	80.81
	OURS(STAGE II)	2.10	81.51	**81.13**	82.08	**86.30**	**76.78**	**81.56**
QWEN3-30B-A3B	TOP-K PRUNING	2.00	34.25	30.09	30.45	52.91	44.77	38.49
	NAEE	2.00	31.04	26.08	27.41	52.02	43.22	35.95
	ZEDA	2.01	45.10	55.94	72.47	79.76	63.59	63.37
	LEXI	2.00	60.44	28.97	28.35	50.52	43.05	42.27
	OURS(STAGE I)	2.00	71.34	72.81	73.90	81.84	65.66	73.11
	OURS(STAGE II)	2.10	**71.70**	**73.77**	**74.31**	**83.42**	**67.18**	**74.08**

5 CONCLUSION

In this paper, we propose TRUST-MoE, an efficient activated expert reduction framework that exploits the routing structure already learned by pretrained MoE models to reduce activated expert computation. TRUST-MoE formulates expert activation reduction as a coarse-to-fine adaptation process, enabling the model to reduce its default routing width while selectively allocating additional expert computation when needed. Moreover, we evaluate our method across different MoE architectures, downstream benchmarks, comparing it with both static expert-reduction and dynamic expert-skipping methods. The experimental results show that TRUST-MoE better preserves model performance under reduced expert activation than the compared baselines, while end-to-end inference evaluations further demonstrate higher throughput and lower latency, highlighting the effectiveness of exploiting routing decisions for efficient MoE inference.

AI USE STATEMENT

We use large language models solely for grammar correction and language polishing to improve the clarity and readability of the manuscript. The core ideas, method design, technical development, and experimental framework are independently conceived and developed by the authors, without relying on generative AI tools.

REFERENCES

Mikel Artetxe, Shruti Bhosale, Naman Goyal, Todor Mihaylov, Myle Ott, Sam Shleifer, Xi Victoria Lin, Jingfei Du, Srinivasan Iyer, Ramakanth Pasunuru, et al. Efficient large scale language modeling with mixtures of experts. *arXiv preprint arXiv:2112.10684*, 2021.

Aarti Basant, Abhijit Khairnar, Abhijit Paithankar, Abhinav Khattar, Adithya Renduchintala, Aditya Malte, Akhiad Bercovich, Akshay Hazare, Alejandra Rico, Aleksander Ficek, et al. Nvidia nemotron nano 2: An accurate and efficient hybrid mamba-transformer reasoning model. *arXiv preprint arXiv:2508.14444*, 2025.

Sahil Chaudhary. Code alpaca: An instruction-following llama model for code generation. `https://github.com/sahil280114/codealpaca`, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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

Krishna Teja Chitty-Venkata, Sandeep Madireddy, Murali Emani, and Venkatram Vishwanath. Lexi: Layer-adaptive active experts for efficient moe model inference. *arXiv preprint arXiv:2509.02753*, 2025.

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 *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, 2019.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 1280–1297, 2024.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International conference on machine learning*, pp. 5547–5569. PMLR, 2022.

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

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

Shwai He, Run-Ze Fan, Liang Ding, Li Shen, Tianyi Zhou, and Dacheng Tao. Merging experts into one: Improving computational efficiency of mixture of experts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 14685–14691, 2023.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Wei Huang, Yue Liao, Jianhui Liu, Ruifei He, Haoru Tan, Shiming Zhang, Hongsheng Li, Si Liu, and Xiaojuan Qi. Mixture compressor for mixture-of-experts llms gains more. In *International Conference on Learning Representations*, volume 2025, pp. 68628–68648, 2025.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, et al. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in neural information processing systems*, 36:62991–63010, 2023.

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

Zewen Jin, Shen Fu, Chengjie Tang, Youhui Bai, Shengnan Wang, Jiaan Zhu, Chizheng Fang, Ping Gong, and Cheng Li. Smidt: High-performance inference framework for moe models with dynamic top-k routing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 22444–22453, 2026.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. Cmmlu: Measuring massive multitask language understanding in chinese. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 11260–11285, 2024.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024b.

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

Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models, 2024. URL https://arxiv.org/abs/2402.14800.

Xingtai Lv, Li Sheng, Kaiyan Zhang, Yichen You, Siyan Gao, Xueheng Luo, Yuxin Zuo, Yuchen Fan, Junlin Yang, Ganqu Cui, et al. Post-trained moe can skip half experts via self-distillation. *arXiv preprint arXiv:2605.18643*, 2026.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. Olmoe: Open mixture-of-experts language models. In *International Conference on Learning Representations*, volume 2025, pp. 62061–62121, 2025.

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

Yaoxiang Wang, Qingguo Hu, Yucheng Ding, Ruizhe Wang, Yeyun Gong, Jian Jiao, Yelong Shen, Peng Cheng, and Jinsong Su. Training matryoshka mixture-of-experts for elastic inference-time expert utilization. *arXiv preprint arXiv:2509.26520*, 2025.

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

Zukang Xu, Zhixiong Zhao, Xing Hu, Jiangyong Yu, Houji Wen, Jun Li, Zhe Jiang, and Dawei Yang. Ace: Adaptive calibration-free expert skipping for moe-based llms. *arXiv preprint arXiv:2609.05228*, 2026.

Fuzhao Xue, Zian Zheng, Yao Fu, Jinjie Ni, Zangwei Zheng, Wangchunshu Zhou, and Yang You. Openmoe: An early effort on open mixture-of-experts language models. *arXiv preprint arXiv:2402.01739*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yuanhang Yang, Shiyi Qi, Wenchao Gu, Chaozheng Wang, Cuiyun Gao, and Zenglin Xu. Xmoe: Sparse models with fine-grained and adaptive expert selection. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 11664–11674, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

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

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

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

A Ablation Studies

Stage I training objective ablation. To analyze the contribution of different Stage I objectives, we start from a base objective consisting of the language-modeling loss, output-level distillation, and tail regularization, and then independently introduce monotonicity regularization and MoE representation distillation. As shown in Table 6, both components consistently improve the base objective. The monotonicity constraint encourages the learned masks to maintain an ordered routing prefix, while MoE representation distillation mitigates the intermediate representation shift caused by suppressing lower-ranked experts. The highest overall performance is obtained when both components are jointly applied, highlighting their complementary roles in preserving routing structure and representation quality under reduced expert activation.

Table 6: Ablation of Stage-I training objectives on Qwen3.5-35B-A3B

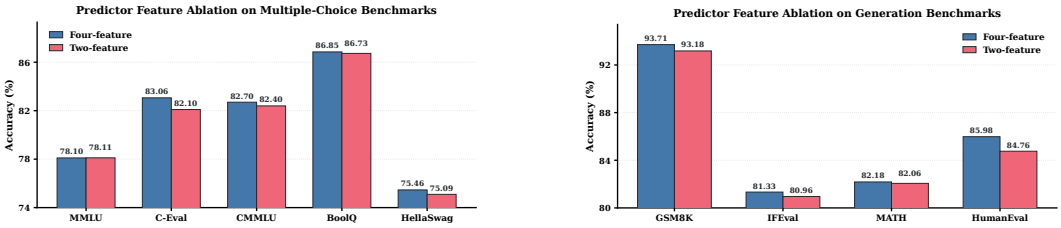
Different Training Objective	GSM8K	IFEval	HumanEval	Math	Avg
BASE-LOSS($\mathcal{L}_{\text{CE}}, \mathcal{L}_{\text{KL}}, \mathcal{L}_{\text{tail}}$)	95.75	83.55	91.46	93.28	91.01
BASE-LOSS + MONO LOSS($\mathcal{L}_{\text{CE}}, \mathcal{L}_{\text{KL}}, \mathcal{L}_{\text{tail}}, \mathcal{L}_{\text{mono}}$)	96.29	84.84	93.90	93.48	92.13
BASE-LOSS + MOE LOSS($\mathcal{L}_{\text{CE}}, \mathcal{L}_{\text{KL}}, \mathcal{L}_{\text{tail}}, \mathcal{L}_{\text{moe}}$)	96.21	85.03	93.29	93.32	91.96
BASE-LOSS + MOE LOSS + MONO LOSS($\mathcal{L}_{\text{CE}}, \mathcal{L}_{\text{KL}}, \mathcal{L}_{\text{tail}}, \mathcal{L}_{\text{moe}}, \mathcal{L}_{\text{mono}}$)	**96.29**	**85.40**	**94.51**	**93.62**	**92.46**

Effect of Learned Rank-Wise Masks. The rank-wise masks learned during Stage I can remain useful after the adaptation process. We compare two deployment settings that share the same mask-training procedure. In the fixed-mask variant, the mask values of the retained Ranks 1-4 are reset to 1 after training, whereas our method preserves the mask values learned for Ranks 1-4 during Stage I. As shown in Table 7, retaining the learned rank-wise masks achieves higher average performance on both models. This indicates that the learned masks not only facilitate the suppression of lower-ranked experts during training, but also capture useful adjustments to the relative contributions of the retained routing positions.

Table 7: Comparison between fixed and learned rank-wise masks.

Model	Method	GSM8K	IFEval	HumanEval	Math	Avg
Qwen3.5-35B-A3B	Fixed Mask(=1)	95.38	85.03	92.07	**93.68**	91.54
	Learned Rank-Wise Mask (Ours)	**96.29**	**85.40**	**94.51**	93.62	**92.46**

Predictor feature ablation. We compare the full four-feature predictor with a reduced two-feature variant ($p_{l,t}^{\tilde{K}+1}, m_l^{\tilde{K}} p_{l,t}^{\tilde{K}} - p_{l,t}^{\tilde{K}+1}$) to examine whether the additional routing statistics provide useful supervision for Stage II in Figure 3. The full feature set yields consistently comparable or better performance across both generation and multiple-choice benchmarks, with more visible gains on HumanEval, C-Eval, and HellaSwag. Additional routing statistics complement the candidate-expert weight and boundary score gap, improving refinement-utility estimation and helping the predictor identify token-layer pairs that benefit from executing the Rank-$(\tilde{K}+1)$ expert.

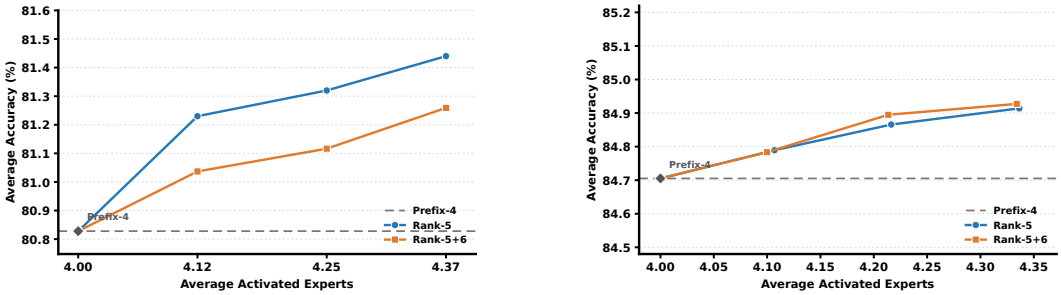


(a) Multiple-choice benchmarks (b) Generation benchmarks

Figure 3: Ablation of the Stage-II predictor features on Qwen3-30B-A3B. (a) Performance on multiple-choice benchmarks. (b) Performance on generation benchmarks. The four-feature predictor achieves comparable or better performance than the reduced two-feature variant across the evaluated tasks, suggesting that the additional routing statistics provide complementary information for refinement-utility prediction.

Effect of Restoration Width in Stage II. To study how the additional expert budget should be allocated in Stage II, we compare restoring only the Rank-5 expert with jointly restoring the Rank-5

and Rank-6 experts under matched average activation budgets. As shown in Figure 4, both strategies consistently improve over the Prefix-4 baseline as the average number of active experts increases. Overall, Rank-5 and Rank-5+6 achieve comparable performance across the two models. However, Rank-5 restoration involves only one additional candidate expert, making the refinement decision and execution simpler and more favorable for practical inference acceleration. We therefore adopt Rank-5 as the default restoration boundary, balancing performance and inference efficiency.



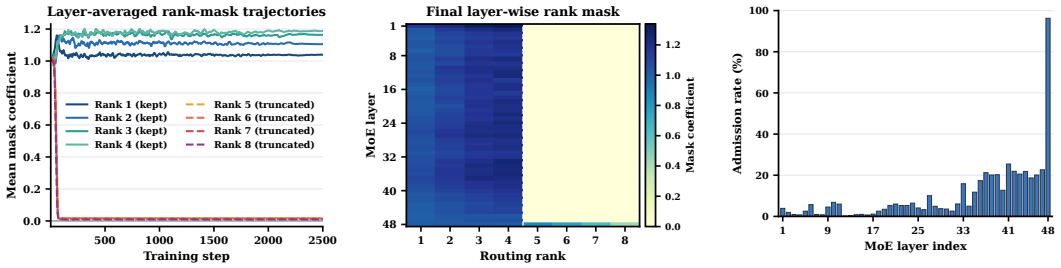
(a) Qwen3-30B-A3B (b) Qwen3.5-35B-A3B

Figure 4: Effect of restoration width in Stage II under matched average expert-activation budgets. We compare Rank-5 restoration with joint Rank-5+6 restoration on (a) Qwen3-30B-A3B and (b) Qwen3.5-35B-A3B. Both strategies improve over the Prefix-4 baseline and exhibit comparable overall performance-computation trade-offs, although their relative gains vary across models.

B Experimental Results Analysis

B.1 Optimization Dynamics and Learned Routing Structure

Figure 5 illustrates how the two stages shape the execution boundary on Qwen3-30B-A3B. During Stage I, the mask coefficients of Ranks 5–8 rapidly decay toward zero, while those of Ranks 1–4 undergo gradual adaptation before stabilizing, as shown in Figure 5a. The final layer-wise masks exhibit a clear separation between the retained and suppressed routing positions across most MoE layers, forming a compact Top-4 prefix rather than uniformly shrinking all routing ranks in Figure 5b. Stage II further refines this boundary by selectively restoring the Rank-5 expert. As shown in Figure 5c, the recovery decisions are highly non-uniform across depth and are concentrated mainly in the later layers. Overall, Stage I establishes a compact default routing boundary, while Stage II selectively expands this boundary for token-layer pairs predicted to benefit from the additional Rank-5 expert.

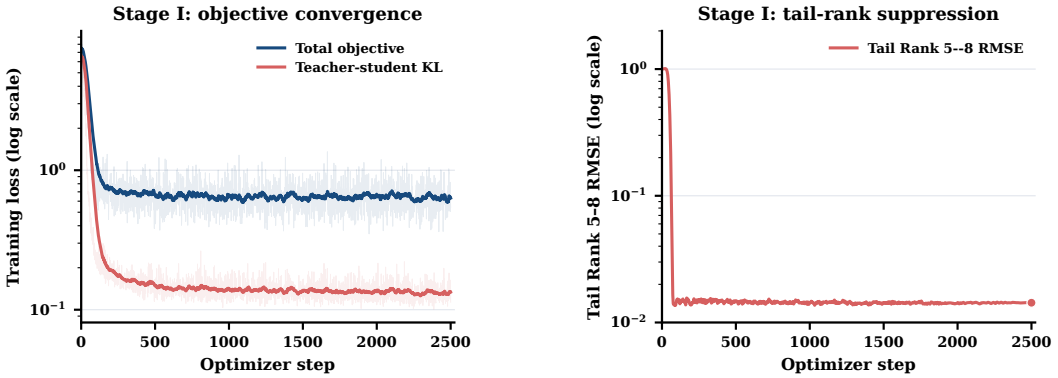


(a) RankMask Training Dynamics (b) Final RankMask Structure (c) Layer-wise Rank-5 Recovery

Figure 5: Optimization dynamics and adaptive boundary refinement on Qwen3-30B-A3B. (a) Layer-averaged RankMask trajectories during Stage I. Ranks 5–8 are rapidly suppressed, while Ranks 1–4 remain stable. (b) Final layer-wise mask coefficients, showing a clear Top-4 prefix structure across most layers. (c) Layer-wise fraction of token decisions for which the Rank-5 expert is restored during Stage II.

B.2 Training Loss Convergence

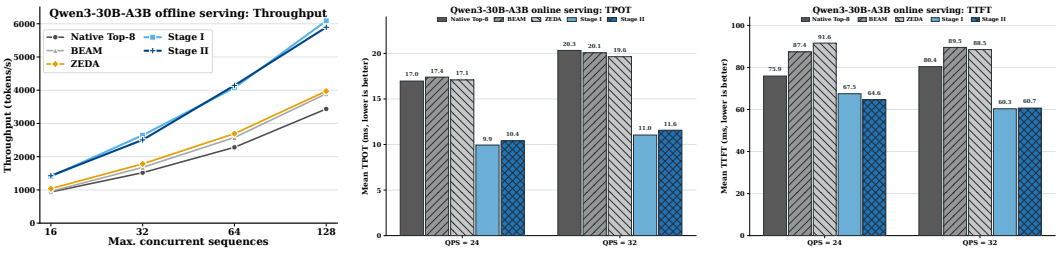
In Figure 6a, both the overall training objective and the teacher-student KL decrease rapidly during the early stage of optimization and stabilize thereafter. The RMSE of Ranks 5-8 decreases even more rapidly, reaching a low level early in training and remaining stable throughout the subsequent updates in Figure 6b. This behavior indicates that Stage I quickly suppresses the routing positions beyond the target Top-$\tilde{K}$ boundary while maintaining stable optimization. The earlier stabilization of the tail-rank RMSE further suggests that the suppression pattern is established early, whereas the remaining optimization continues to reduce the overall training objective under the resulting compact routing structure.



(a) Training Objective Convergence (b) Tail-Rank Suppression Convergence

Figure 6: Empirical convergence behavior of Stage I on Qwen3-30B-A3B. (a) Convergence of the overall training objective and teacher–student KL divergence. (b) Convergence of the tail-rank RMSE for Ranks 5–8. Both quantities stabilize early in training, indicating rapid formation of the compact Top-4 routing structure.

C Acceleration Comparison



(a) Offline Throughput (b) Online Time per Output Token (c) Online Time to First Token

Figure 7: Serving efficiency on Qwen3-30B-A3B. (a) Offline generation throughput under different maximum numbers of concurrent sequences. (b) Mean time per output token (TPOT) under different offered loads. (c) Mean time to first token (TTFT) under different offered loads. Stage II retains most of the serving-efficiency gain achieved by Stage I while selectively introducing additional expert computation.

Experiment Settings. We implement our method, BEAM, and ZEDA within the same vLLM serving stack and use the same downstream sparse-execution backend. For our method, a fused Triton routing kernel performs Prefix-4 routing together with utility prediction and conditional Rank-5 admission. For BEAM and ZEDA, fused Triton post-processing kernels convert their routing decisions into the same sparse execution format. In all cases, inactive routes are removed before the expert GEMMs, ensuring that only physically retained routes incur expert computation. We further use method-specific, autotuned Triton MoE configurations for BEAM and ZEDA, and include all method-specific routing overheads in the end-to-end serving measurements. Therefore, Figure 7

provides a fair comparison of the three routing strategies under a shared serving backend and consistent sparse-execution mechanism.

Offline Inference Throughput. We evaluate offline throughput using 5,000 fixed-length requests, each with 128 input and 32 output tokens. As shown in Figure 7a, Stage II consistently achieves higher throughput than Native Top-8, BEAM, and ZEDA across different concurrency levels, reaching up to $1.82\times$ the throughput of Native Top-8 with a maximum of 64 concurrent sequences. Meanwhile, its throughput remains close to Stage I, indicating that selective Rank-$(\tilde{K}+1)$ refinement preserves most of the efficiency gained from the compact Top-$\tilde{K}$ execution.

Online Inference Latency. We further evaluate online inference under Poisson request arrivals at 24 and 32 queries per second (QPS). As shown in Figure 7b 7c, Stage I and Stage II both substantially reduce serving latency compared with Native Top-8 and the competing expert-reduction methods. In particular, Stage II reduces TPOT by 38.8% and 42.9% relative to Native Top-8 at QPS 24 and QPS 32, respectively, while introducing only 5.1% and 5.5% TPOT overhead over Stage I. The time to first token (TTFT) is also reduced under both offered loads. These results show that Stage I provides the primary efficiency gain, while Stage II retains most of this benefit while selectively allocating additional computation for token-wise refinement.