STAVER: A PROBABILISTIC FORMULATION FOR STABLE ROUTING IN MIXTURE-OF-EXPERTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture-of-Experts (MoE) models scale capacity efficiently by routing each token to a small subset of experts. A key challenge in MoE models is routing instability, which can reduce training efficiency and impair model performance. Through empirical studies and theoretical analysis, we identify the discontinuity of hard Top-k selection as a source of this instability: perturbations can abruptly change the selected expert set, leading to performance degradation. To address this issue, we propose StaVER, a unified Bayesian framework that formulates MoE routing as a probabilistic inference problem. StaVER models expert assignment as a latent variable and performs routing by sampling experts from its distribution. Under this probabilistic formulation, the expected expert-selection frequencies vary continuously to perturbations, smoothing the abrupt changes in expert selection induced by hard Top-k routing. To compute this intractable posterior distribution over expert assignment, we follow variational inference to derive a unified training objective. This yields a prior-matching term that naturally promotes load balancing, thus eliminating the need for an additional heuristic auxiliary loss. Through extensive experiments on MoE models ranging from 1.8B to 6.6B parameters, we demonstrate that StaVER achieves more stable routing, more balanced expert utilization, and consistent performance gains on both upstream and downstream tasks. Across eight downstream benchmarks, StaVER improves average accuracy up to 2.6 percentage points over the strongest baseline. In the pre-training evaluation, it obtains the lowest PPL, while reducing estimated training FLOPs by approximately 18%.

1 INTRODUCTION

Recent advances in large language models (LLMs) have enabled remarkable success across diverse domains Ma et al. (2026); Wei et al. (2026), but their training costs grow rapidly with model scale. Mixture-of-Experts (MoE) architectures address this challenge by activating only a small subset of experts for each input token via a learnable router Guo et al. (2026); Lv et al. (2025); Kang et al. (2024). This sparse activation increases model capacity without a commensurate increase in computational overhead Liu et al. (2024a); Team (2026); Li et al. (2025), enabling efficient training of trillion-parameter language models Cai et al. (2025); Mu & Lin (2025); Artetxe et al. (2021).

Central to MoE models is the routing mechanism Chi et al. (2022). It is responsible for (i) computing routing logits that reflect token-to-expert affinity and (ii) selecting experts based on routing logits Muennighoff et al. (2024). One of the prominent approaches is to select the k highest-scoring experts, known as Top-k selection Lepikhin et al. (2020). This selection enables sparse activation by mapping the routing logits to a discrete, k-hot expert selection mask. However, while the logits can vary continuously with token representations or router parameters, the selection mask changes through abrupt switches between expert sets. Consequently, small changes in routing preferences can produce abrupt changes in token-to-expert assignments, making Top-k routing sensitive to perturbations. Such reassignments can redirect tokens to experts insufficiently trained on similar inputs, impairing model performance. As shown in Figure 1, perplexity increases substantially for tokens with changed expert sets under perturbations, whereas it remains largely unchanged for those with unchanged expert sets.

This routing instability highlights a key challenge in MoE training: token-to-expert assignments can fluctuate under perturbations. Existing methods rely on training constraints to suppress routing fluctuations. StableMoE Dai et al. (2022) distills the routing strategy into a separate router and freezes it in the second training stage to enforce fixed token-to-expert assignments. TGR-MoE Kada et al. (2026) uses a distillation loss to guide the router toward a stable teacher. While these methods reduce routing fluctuations, their constraints on routing adaptation can interfere with task-driven router

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.

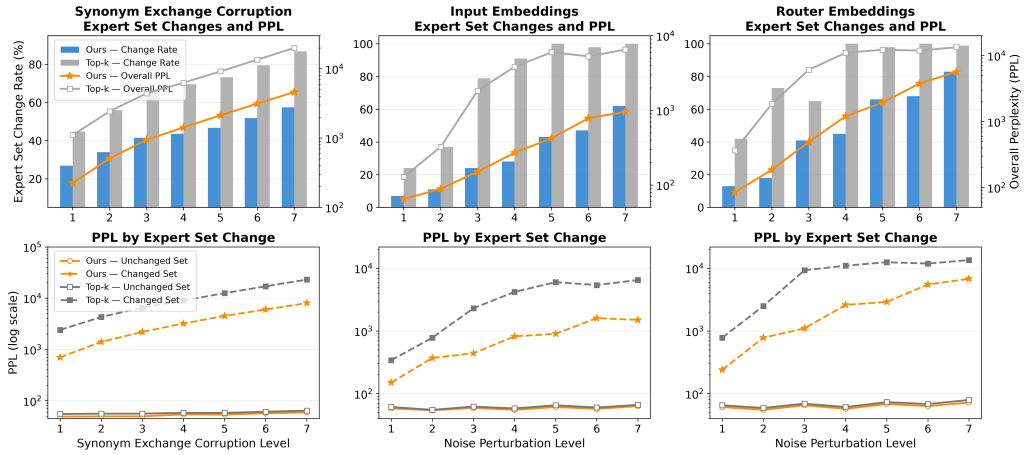


Figure 1: **Effect of routing instability.** Perturbations simulate meaning-preserving input variations at inference (**left**) and gradient-space variations affecting the router during pretraining (**middle and right**). Routing selection changes occur when a token's selected expert set differs between the unperturbed and perturbed conditions. **Top:** StaVER maintains lower routing changed rates and lower perplexity than Top-k method. **Bottom:** perplexity degradation is concentrated in tokens with changed routing.

learning and optimization, potentially limiting model quality. More importantly, these methods retain the discontinuous Top-k selection mechanism without directly addressing the router's underlying sensitivity to perturbations. Consequently, the trained routers remain susceptible to perturbation-induced changes in expert selection at post-training and inference stages. These limitations highlight the need for an explicit mechanism that addresses the instability arising from discontinuous Top-k selection, enabling stable routing while preserving sparse expert activation.

To this end, we propose StaVER (Stable Variational Expert Routing), a Bayesian framework for stable and efficient routing. Specifically, StaVER introduces a latent variable to model expert assignment, with its distribution parameterized by token-to-expert affinity scores. Routing is then performed through Monte Carlo sampling from this distribution, allowing empirical expert selection to approximate the distribution over the latent variable while achieving dynamic routing through adaptive sampling. Theoretical analysis on this routing mechanism shows that perturbations to the routing logits induce correspondingly small changes in expected expert selection frequencies, avoiding abrupt changes in the expert selection set. We adopt a variational inference scheme to approximate the intractable posterior distribution over expert assignments. The resulting objective consists of a language modeling term and a prior-matching term, allowing task-driven router optimization while encouraging balanced routing. This formulation therefore integrates routing stability and load balancing within a unified probabilistic framework, eliminating the need for a heuristic load-balancing loss. To summarize, our contribution is threefold:

- We identify the discontinuity of Top-k selection as a fundamental source of routing instability and propose a unified probabilistic formulation for the routing process. Our framework improves routing stability by addressing perturbation sensitivity, while its sampling-based routing reduces expert activations and improves overall training efficiency (Sec. 2.2).
- From this probabilistic formulation, we derive a prior-matching term that promotes balanced expert utilization. This principled objective achieves better load balancing than both heuristic auxiliary-loss methods and auxiliary-loss-free approaches (Sec. 3.2).
- In pre-training experiments across three model scales, StaVER (1) improves training efficiency, reducing estimated training FLOPs by approximately 18% while outperforming the strongest baseline by up to 2.6 percentage points in average accuracy; (2) maintains more stable routing than the stabilization baseline under different perturbations; and (3) achieves more balanced expert utilization than both heuristic auxiliary-loss and auxiliary-loss-free approaches (Sec. 4).

2 Motivation

2.1 Preliminaries of MoE

Mixture-of-Experts routing. A typical MoE layer consists of N experts and a router that determines which experts process each input token (Lepikhin et al., 2020; Fedus et al., 2022). Given a token

representation x, the router assigns a normalized score for each expert,

$$\pi_\phi(x) = \mathrm{softmax}(r_\phi(x)), \tag{1}$$

where r_ϕ denotes the router logits parameterized by ϕ, and $\pi_{\phi,i}(x)$ is the routing score for expert i. The router usually activates the k ($k < N$) experts with the highest affinity score,

$$\mathcal{S}_k(x) = \mathrm{TopK}(\pi_\phi(x)), \tag{2}$$

and computes the MoE output as

$$f_{\mathrm{MoE}}(x) = \textstyle\sum_{i\in\mathcal{S}_k(x)} \tilde{\pi}_{\phi,i}(x) E_i(x), \tag{3}$$

where $\tilde{\pi}_{\phi,i}(x)$ is the aggregation weight by renormalizing routing scores over the selected experts, and $E_i(x)$ represents the output computed by expert i. Thus, Top-k routing makes a deterministic selection according to the ordering of its affinity scores.

Load balancing. To prevent expert collapse, MoE models usually augment the language modeling loss with an auxiliary load-balancing loss (Shazeer et al., 2017; Fedus et al., 2022):

$$\mathcal{L} = \mathcal{L}_{\mathrm{task}} + \alpha\mathcal{L}_{\mathrm{aux}}, \qquad \mathcal{L}_{\mathrm{aux}} = N \textstyle\sum_{i=1}^{N} f_i P_i. \tag{4}$$

Here f_i denotes the fraction of routing assignments received by expert i, and P_i is the average routing probability assigned to expert i. Since the actual expert load f_i is determined by the discrete Top-k selection, it is non-differentiable. Therefore, the auxiliary loss couples f_i with the differentiable routing probability P_i. Minimizing this loss adjusts routing probabilities according to the actual expert load, encouraging more uniform token allocation across experts.

2.2 Instability of Hard Top-k Routing

We present how perturbations to the router gradient affect token-to-expert assignments under hard Top-k routing. Since softmax preserves logit ordering, expert selection can be analyzed directly in logit space. Let a and b denote the experts with the k-th and $(k+1)$-th largest routing logits, respectively. We define the boundary margin as the difference between their routing logits:

$$\Delta_k(x) = r_{\phi,a}(x) - r_{\phi,b}(x). \tag{5}$$

This margin quantifies the change in the logit gap required to reverse the ordering of the two experts.

During training, stochastic mini-batch sampling Wu et al. (2020), contaminated examples Yang et al. (2022), nonstationary distribution shifts Baby et al. (2025), and gradient quantization Alistarh et al. (2017) can perturb the router gradient. These perturbations affect the router parameter update and can therefore alter the logit gap between the two boundary experts. To examine how these changes affect expert selection, consider a linear router $r_\phi(x) = Wx$ for a fixed token representation $x \neq 0$. Let G denote the router gradient and ξ an additive gradient perturbation. Starting from the same parameters $W^{(n)}$, the updates with and without gradient perturbations are given by

$$W^{(n+1)} = W^{(n)} - \eta G, \qquad \widehat{W}^{(n+1)} = W^{(n)} - \eta(G+\xi) = W^{(n+1)} - \eta\xi, \tag{6}$$

where $\eta > 0$ is the learning rate. We use $W^{(n+1)}$ as the reference router and evaluate a, b, and $\Delta_k(x)$ at this update. With the perturbed logits defined as $\widehat{r}(x) = \widehat{W}^{(n+1)}x$, the gap between the two boundary experts becomes

$$\widehat{r}_a(x) - \widehat{r}_b(x) = \Delta_k(x) - \eta \left\langle \xi, (e_a - e_b)x^\top \right\rangle_F,$$

where e_i is the i-th standard basis vector and $\langle\cdot,\cdot\rangle_F$ denotes the Frobenius inner product. The second term is the perturbation-induced reduction in the logit gap. Once this reduction exceeds the original margin, the ordering of a and b reverses, changing the selected expert set. Thus, a sufficient condition for a routing change is

$$\eta \left\langle \xi, (e_a - e_b)x^\top \right\rangle_F > \Delta_k(x). \tag{7}$$

When the boundary margin $\Delta_k(x)$ is small, even a small gradient perturbation can reverse the ordering of the two boundary experts and change the selected expert set. To quantify the instability of this selection, let $\rho_{\mathrm{grad}}(x)$ denote the minimum Frobenius norm of an additive gradient perturbation that changes the selected set. For a positive boundary margin, every selected–unselected expert pair has a logit gap at least $\Delta_k(x)$. Applying the Cauchy–Schwarz inequality to these gaps, and considering perturbations aligned with $(e_a - e_b)x^\top$, gives

$$\rho_{\mathrm{grad}}(x) = \tfrac{\Delta_k(x)}{\sqrt{2}\eta\|x\|_2}, \tag{8}$$

where the factor $\sqrt{2}\|x\|_2$ is the Frobenius norm of $(e_a - e_b)x^\top$.

Eq. 8 reveals that, for fixed x and η, the perturbation threshold scales linearly with the boundary margin. Although routing logits vary continuously under gradient perturbations, as the boundary

margin approaches zero, *arbitrarily small gradient perturbations can induce small logit changes that trigger abrupt changes in expert selection.* Such discontinuity in routing change is undesirable: when related tokens are reassigned, a previously frequently selected expert may lose updates from those tokens. Meanwhile, a newly selected expert may not yet be well trained to process them. Such reassignment can therefore leave experts with insufficient training on the relevant tokens, hindering specialization and finally degrading model performance.

3 METHODOLOGY

Motivated by the unstable nature of hard Top-k routing, we model expert assignments probabilistically and propose a routing mechanism that approximates the marginalized output through Monte Carlo sampling. To optimize the router, we use variational inference to derive a training objective that unifies language-model fitting with expert load balancing.

3.1 MONTE CARLO STOCHASTIC ROUTING

To address routing instability, we first reformulate the MoE output as a marginalization over expert assignments. Given a token x, let z denote its latent expert assignment. We interpret the routing scores $\pi_\phi(x)$ over N experts as the probabilities of a categorical distribution $q_\phi(z \mid x) = \mathrm{Cat}(\pi_\phi(x))$, which we refer to as the *routing distribution*. The marginalized output is

$$f_\phi(x) = \mathbb{E}_{z\sim q_\phi(z|x)}\left[E_z(x)\right] = \textstyle\sum_{i=1}^{N} \pi_{\phi,i}(x) E_i(x), \tag{9}$$

where $E_i(x)$ represents the representation computed by expert i. This formulation weights expert outputs directly by their routing probabilities, which vary continuously with the routing logits. A reversal in the ordering of two experts therefore changes the relative weights without abruptly including or excluding either expert from the marginalized output.

Sparse Expert Activation. Exact evaluation of Eq. 9 requires computing the outputs of all N experts. To retain sparse expert activation, we approximate this expectation through Monte Carlo sampling. For each token x, we draw K expert indices with replacement, $z_{1:K} \overset{\text{i.i.d.}}{\sim} q_\phi(z \mid x)$ and compute

$$\hat{f}_K(x; z_{1:K}) = \tfrac{1}{K} \textstyle\sum_{j=1}^{K} E_{z_j}(x). \tag{10}$$

Thus, only the sampled experts need to be evaluated, with at most K distinct experts activated for each token. Repeated selections of the same expert are aggregated through their empirical frequency n_i/K, where $n_i = \sum_{j=1}^{K} \mathbf{1}\{z_j = i\}$. Since $\mathbb{E}[n_i/K] = \pi_{\phi,i}(x)$, Eq. 10 is an unbiased estimator of the marginalized output in Eq. 9. *We refer to this sampling-based mechanism as stochastic routing.* To relax the constraints of discrete sampling we use the Straight-Through Gumbel-Softmax trick Jang et al. (2017). This allows us to efficiently back-propagate the parameter gradients of the router while sampling expert candidates. Detailed discussion is provided in Appendix B.1.

Stable Routing. Since each expert's sampling frequency is an unbiased estimator of its routing probability, stochastic routing preserves the continuous dependence of expected selection frequencies on the routing logits, avoiding the discontinuity at Top-k selection boundaries identified in Sec. 2.2. To quantify this stability under gradient perturbations, consider the same linear router $r_\phi(x) = Wx$. Let W and $\widehat{W} = W - \eta\xi$ denote the parameters after the unperturbed and perturbed updates, respectively, and write $\pi_W(x) = \mathrm{softmax}(Wx)$. We measure the perturbation-induced shift in expert selection using the total variation distance between routing distributions:

$$D_{\mathrm{stoch}}(x) := \tfrac{1}{2}\left\|\pi_{\widehat{W}}(x) - \pi_W(x)\right\|_1. \tag{11}$$

Proposition 3.1. *For the stochastic router, the shift in the expert routing distribution satisfies the following bound:*

$$D_{\mathrm{stoch}}(x) \le \tfrac{\eta}{2}\|\xi\|_F \|x\|_2. \tag{12}$$

Moreover, for $x \neq 0$, $1 \le k < N$, and any $\epsilon > 0$, there exists a gradient perturbation satisfying Eq. 8 that changes the Top-k selection:

$$\|\xi\|_F = \rho_{\mathrm{grad}}(x) + \epsilon. \tag{13}$$

Substituting Eq. 13 into Eq. 12 yields the following bound on the change in the stochastic routing distribution under the same perturbation:

$$D_{\mathrm{stoch}}(x) \le \tfrac{\Delta_k(x)}{2\sqrt{2}} + \tfrac{\eta\epsilon}{2}\|x\|_2. \tag{14}$$

Proposition 3.1 bounds the change in the stochastic routing distribution in terms of the gradient perturbation norm. As $\Delta_k(x) \to 0$ and $\epsilon \to 0$, the perturbation in Eq. 13 becomes arbitrarily small

while still changing the Top-k selection set. In contrast, for a fixed $x \neq 0$ and $\eta > 0$, both terms on the right-hand side of Eq. 14 vanish, implying that the change in the stochastic routing distribution under the same perturbation also tends to zero. The proof is provided in Appendix B.2.

Summary. Our stochastic router learns a probabilistic selection strategy that aligns empirical expert selection with the routing distribution over the latent variable. While Top-k routing commits to the best-ranked expert set for each input, our probabilistic strategy allows routing preferences to change continuously with perturbations on router, even when the expert rankings change. Expert selection therefore follows these continuously varying preferences, smoothing the abrupt routing changes caused by switching between competing Top-k sets.

3.2 Variational Bayesian Inference for Expert Selection

We then consider how to learn the routing distribution from training data. For each training pair (x, y), the prediction target y provides evidence for inferring the latent variable given input x. Learning the routing distribution can thus be viewed as posterior inference for expert assignments given the observed training pairs. We therefore use Bayesian inference to learn the routing distribution. Specifically, for sampled assignments $z_{1:K}$, let $\hat{f}_K(x; z_{1:K})$ denote the corresponding MoE representation. The likelihood is defined as

$$p_\theta(y \mid x, z_{1:K}) = p_\theta\left(y \mid \hat{f}_K(x; z_{1:K})\right). \tag{15}$$

A factorized prior $p(z_{1:K}) = \prod_{j=1}^{K} p(z_j)$ is placed over expert assignments. By Bayes' rule, the posterior is

$$p_\theta(z_{1:K} \mid x, y) = \frac{p_\theta(y \mid x, z_{1:K})p(z_{1:K})}{p_\theta(y \mid x)}. \tag{16}$$

This posterior characterizes the distribution over expert assignments conditioned on the observed token and its prediction target. However, exact computation for the posterior is generally intractable, as evaluating $p_\theta(y \mid x)$ requires marginalizing over all possible expert assignments. We therefore propose a variational distribution $q_\phi(z_{1:K} \mid x)$ as an approximation for the posterior in amortized variational inference Margossian & Blei (2024):

$$q_\phi(z_{1:K} \mid x) = \prod_{j=1}^{K} q_\phi(z_j \mid x), \quad \text{where } q_\phi(z_j \mid x) = \mathrm{Cat}(z_j; \pi_\phi(x)). \tag{17}$$

For each training pair (x, y), we seek a variational distribution $q_\phi(z_{1:K}|x)$ that approximates the corresponding posterior $p_\theta(z_{1:K}|x, y)$ by minimizing the Kullbeck-Leibler (KL) divergence with respect to variational parameter ϕ:

$$\min_{\phi} \quad \mathbb{E}_{x,y}\mathrm{KL}\big(q_\phi(z_{1:K} \mid x)\|p(z_{1:K} \mid x, y)\big), \tag{18}$$

Equivalently, we can optimize an evidence lower bound (ELBO) Kingma & Welling (2014) instead, yielding the following tractable objective.

Proposition 3.2 (Variational objective)**.** *Let $\bar{q}_\phi(z) = \frac{1}{M}\sum_{m=1}^{M} q_\phi(z \mid x_m)$ denote the aggregated routing distribution. We derive the following training objective from the ELBO associated with Eq. 18 for $K \geq 1$:*

$$\mathcal{L}_{\text{train}} = \underbrace{-\frac{1}{M}\sum_{m=1}^{M} \mathbb{E}_{z_{1:K}\sim q_\phi(\cdot|x_m)} \log p_\theta\left(y_m \mid \hat{f}_K(x_m; z_{1:K})\right)}_{\mathcal{L}_{\text{LM}}} + \alpha \underbrace{D_{\text{KL}}\left(\bar{q}_\phi(z)\|p(z)\right)}_{\mathcal{L}_{\text{prior}}}. \tag{19}$$

Here, $z_{1:K}$ denotes K i.i.d. samples from $q_\phi(z \mid x)$, $\hat{f}_K(x; z_{1:K}) = \frac{1}{K}\sum_{j=1}^{K} E_{z_j}(x)$ is the MoE output, and α is a scaling parameter for the regularization magnitude. The proof is in Appendix B.3.

LLM fitting. The first term $\mathcal{L}_{\text{LM}}$ corresponds to the language modeling loss under stochastic expert routing. For each sampled configuration $z_{1:K}$, the model computes its prediction from $\hat{f}_K(x; z_{1:K})$ and evaluates the standard NLL loss. Taking the expectation over sampled expert assignments therefore preserves the original language modeling objective, i.e., training the router to favor expert assignments with better predictions.

Prior matching. The second term $\mathcal{L}_{\text{prior}}$ regularizes the average routing probability assigned to each expert across the batch toward a prior $p(z)$. Importantly, the prior is imposed on the routing of the batch of tokens across experts, individual tokens remain free to select the most suited experts.

Since the prior-matching term minimizes the discrepancy between the routing distribution and the prior, the prior provides a natural way to guide the routing distribution. To encourage balanced

expert utilization, we adopt a *uniform prior* over experts, $p(z = i) = \frac{1}{N}, i = 1, \ldots, N$. Then, the prior-matching term becomes

$$D_{\mathrm{KL}}\left(\bar{q}_\phi(z) \parallel p(z)\right) = \sum_{n=1}^{N} \bar{\pi}_{\phi,n} \log \frac{\bar{\pi}_{\phi,n}}{1/N}. \tag{20}$$

Uniform Prior as Load Balancing. Here, Eq. 20 is minimized when $\bar{\pi}_{\phi,i} = 1/N$. Therefore, minimizing this term encourages a uniform expert distribution, resulting in more balanced expert utilization. Compared with the auxiliary load balancing loss Eq. 4 introduced in conventional MoEs, this term achieves load balancing by matching the routing distribution to a uniform prior.

Summary. Proposition 3.2 shows that language-model fitting and load balancing follow from a single variational objective. Conventional MoE balance expert loads by adding a heuristic auxiliary loss to the language-modeling objective. This loss can produce interference gradients that conflict with language-model optimization Wang et al. (2024); Guo et al. (2026). Our formulation instead derives both terms from the same variational objective in a principled manner, with load balancing provided by prior matching.

4 EXPERIMENTS

In this section, we evaluate our proposed `StaVER` by answering the following four questions:

- **RQ1**: Does our proposed `StaVER` improve overall performance across both upstream and downstream tasks compared to Top-k baselines?
- **RQ2**: Is the router of `StaVER` robust to different types of perturbations?
- **RQ3**: How does the derived prior-matching term influence expert utilization and routing entropy?
- **RQ4**: What is the impact of different design choices, including the stochastic router and $\mathcal{L}_{\mathrm{prior}}$, on model performance?

4.1 EXPERIMENTAL SETUP

Datasets. The proposed method is evaluated on a total of 8 benchmarks, covering three major aspects of downstream tasks: *(1) Commonsense Reasoning:* PIQA Bisk et al. (2019), SIQA Sap et al. (2019), WinoGrande Sakaguchi et al. (2019), and HellaSwag Zellers et al. (2019), *(2) Science Question Answering:* ARC-Easy Clark et al. (2018); *(3) Natural Language Understanding:* MNLI Williams et al. (2018), QNLI Wang et al. (2018), and SST-2 Socher et al. (2013)

Baselines. We compare our method with five representative Top-k MoE pre-training methods to evaluate model performance and stability. GShard Lepikhin et al. (2020) provides a conventional Top-k routing baseline, while DeepSeekMoE Dai et al. (2024) incorporates shared experts that continuously learn common knowledge from all tokens. Switch Transformer Fedus et al. (2022) selectively uses higher precision in the router to improve stability. StableMoE Dai et al. (2022) stabilizes the router with a two stage training. We also include Auxiliary-FreeMoE Wang et al. (2024) as a load-balancing baseline, which balances expert load by adding biases to routing logits.

Setup. All models are pretrained from scratch on 30B tokens from FineWeb-Edu Penedo et al. (2024). We consider three model sizes, ranging from 1.8B to 6.6B parameters. The models comprise 24 Transformer layers with a hidden size of 1,024 and 16 attention heads. Each MoE layer has an intermediate size of 4,096 per expert. For each token, Top-k baselines activate K experts, while our method draws K samples, with $K = \{2, 4, 8\}$ for each model sizes. Detailed in Appendix F.

4.2 OVERALL PERFORMANCE (RQ1)

To demonstrate the effectiveness of stable routing in enhancing model performance, we evaluate our method on both upstream and downstream tasks. For downstream tasks, we compare our method with all baselines across eight benchmarks. For upstream evaluation, we compare PPL, NLL loss and computation FLOPs to assess training efficiency.

Finding 1. `StaVER` consistently improves downstream task performance. As shown in Table 1, `StaVER` achieves the highest accuracy on all eight downstream benchmarks, and its advantage over the baselines persists as model size increases. The 1.8B `StaVER` model reaches an average accuracy of 47.7%, matching the strongest 3.4B baseline with roughly half the total parameters. These results support the effectiveness of enhancing routing stability in improving downstream performance across model sizes. More results for models pretrained on another dataset are provided in Appendix G.2.

Finding 2. `StaVER` achieves better training efficiency. In Table 2, we provide the model performance and computational costs of different MoE baselines. `StaVER` achieves the lowest training

Table 1: **Performance on different downstream tasks.** We provide accuracy across eight downstream benchmarks for different MoE routing methods and model scales. Bold values denote the best performance, underlined values denote the second-best performance, and arrows show the difference relative to GShard.

Setting	Method	ARC-E	PIQA	SIQA	WINO	HELLA	MNLI	QNLI	SST-2	Avg.
8 Experts *1.8B params*	GShard	41.4	59.1	39.0	50.1	32.0	34.0	49.7	55.3	45.1
	Switch Transformer	38.8↓2.6	60.6↑1.5	38.0↓1.0	50.7↑0.6	32.2↑0.2	34.6↑0.6	47.5↓2.2	59.3↑4.0	45.2↑0.1
	StableMoE	41.6↑0.2	59.6↑0.5	38.7↓0.3	50.6↑0.5	32.7↑0.7	34.9↑0.9	51.5↑1.8	51.0↓4.3	45.1→0.0
	Auxiliary-FreeMoE	41.5↑0.1	59.7↑0.6	38.8↓0.2	50.5↑0.4	32.6↑0.6	34.8↑0.8	51.3↑1.6	52.0↓3.3	45.2↑0.1
	DeepSeekMoE	40.9↓0.5	59.5↑0.4	39.0→0.0	49.0↓1.1	32.6↑0.6	33.1↓0.9	47.4↓2.3	60.9↑5.6	45.3↑0.2
	Ours	**44.4**↑3.0	**61.3**↑2.2	**39.2**↑0.2	**51.2**↑1.1	**33.2**↑1.2	**35.7**↑1.7	**52.4**↑2.7	**64.0**↑8.7	**47.7**↑2.6
16 Experts *3.4B params*	GShard	43.0	60.5	38.8	50.2	32.9	35.5	52.5	64.8	47.3
	Switch Transformer	41.5↓1.5	60.7↑0.2	39.4↑0.6	50.4↑0.2	32.1↓0.8	36.5↑1.0	52.9↑0.4	64.0↓0.8	47.2↓0.1
	StableMoE	42.9↓0.1	60.8↑0.3	38.9↑0.1	50.3↑0.1	33.0↑0.1	35.8↑0.3	52.8↑0.3	61.7↓3.1	47.0↓0.3
	Auxiliary-FreeMoE	42.8↓0.2	60.9↑0.4	39.0↑0.2	50.2→0.0	33.1↑0.2	35.9↑0.4	52.7↑0.2	62.0↓2.8	47.1↓0.2
	DeepSeekMoE	43.2↑0.2	61.4↑0.9	39.2↑0.4	50.5↑0.3	33.3↑0.4	36.2↑0.7	53.3↑0.8	64.2↓0.6	47.7↑0.4
	Ours	**47.2**↑4.2	**63.9**↑3.4	**40.7**↑1.9	**52.8**↑2.6	**35.6**↑2.7	**38.3**↑2.8	**55.4**↑2.9	**68.6**↑3.8	**50.3**↑3.0
32 Experts *6.6B params*	GShard	47.2	64.5	41.7	53.2	36.9	38.9	56.2	70.8	51.2
	Switch Transformer	46.1↓1.1	64.8↑0.3	41.1↓0.6	53.6↑0.4	36.6↓0.3	39.2↑0.3	55.8↓0.4	70.2↓0.6	50.9↓0.3
	StableMoE	45.6↓1.6	63.5↓1.0	40.8↓0.9	52.9↓0.3	36.1↓0.8	38.5↓0.4	56.0↓0.2	66.7↓4.1	50.0↓1.2
	Auxiliary-FreeMoE	45.8↓1.4	63.7↓0.8	40.9↓0.8	52.8↓0.4	36.2↓0.7	38.6↓0.3	56.1↓0.1	67.0↓3.8	50.1↓1.1
	DeepSeekMoE	46.8↓0.4	64.2↓0.3	41.5↓0.2	53.0↓0.2	36.4↓0.5	38.8↓0.1	56.5↑0.3	68.6↓2.2	50.7↓0.5
	Ours	**50.1**↑2.9	**66.7**↑2.2	**42.3**↑0.6	**54.5**↑1.3	**38.1**↑1.2	**41.0**↑2.1	**58.6**↑2.4	**73.4**↑2.6	**53.1**↑1.9

Table 2: **Upstream task evaluation.** Comparison of total and active parameter counts, estimated training and inference costs, and perplexity on the training and test sets.

Method	Parameters (B)		Training cost	Inference cost	Language modeling	
	Total	Active	FLOPs ↓	FLOPs/token ↓	Train PPL ↓	Test PPL ↓
GShard	1.8	0.56	9.57×10^{19}	1.31×10^{9}	26.52	26.79
Switch Transformer	1.8	0.36	6.62×10^{19}	9.10×10^{8}	27.44	27.69
StableMoE	1.8	0.36	6.62×10^{19}	9.10×10^{8}	31.00	30.27
DeepSeekMoE	1.8	0.56	9.57×10^{19}	1.31×10^{9}	28.10	28.47
Auxilary-FreeMoE	1.8	0.56	9.57×10^{19}	1.31×10^{9}	30.51	30.75
Ours	1.8	0.44	7.80×10^{19}	1.07×10^{9}	**25.74**	**25.98**

and test perplexity among all compared methods. For Switch Transformer and StableMoE, we retain their original Top-1 routing designs, accounting for their lower FLOPs. Compared with other Top-k baselines, our sampling-based routing enables `StaVER` to activate fewer unique experts, reducing estimated training and inference FLOPs by approximately 18%. With this reduced computation, `StaVER` lowers test perplexity from 26.79 to 25.98 relative to the strongest baseline in model performance. More evidence of `StaVER`'s training efficiency, including NLL loss curves, and different model scale, is provided in the Appendix G.

4.3 Routing Stability (RQ2)

To examine whether `StaVER`'s router remains stable under perturbations, we consider three perturbation types: synonym substitution Wang et al. (2023); Singh et al. (2025), input embedding perturbations Zoph et al. (2022), and router embedding perturbations Agarwal et al. (2022); Wu et al. (2022). These settings simulate variations that the router may encounter during pretraining and downstream stages. Performance under different perturbation levels is evaluated using perplexity for pretraining and accuracy for downstream tasks. Detailed settings are provided in Appendix C.

Finding 3. `StaVER` maintains more stable routing on the upstream task. In Figure 1, `StaVER` consistently exhibits lower routing change rates and lower overall perplexity than Top-k routing across all three perturbation types. The lower panels show that both methods retain similarly low perplexity on tokens with unchanged routing. In contrast, tokens with changed routing exhibit much higher perplexity, with the gap exceeding an order of magnitude under stronger perturbations. *The pronounced perplexity degradation is concentrated among tokens whose routing changes.* `StaVER` reduces the proportion of these tokens and achieves lower perplexity within this group, jointly limiting the degradation in overall language modeling performance.

Finding 4. `StaVER` reduces downstream performance degradation under perturbations. To further validate the stability of the proposed method, we compare it with StableMoE under perturbations. Figure 2 shows that `StaVER` maintains higher accuracy and exhibits smaller overall accuracy

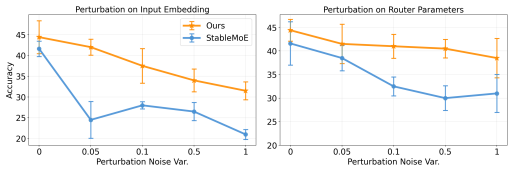


Figure 2: **Robustness under perturbations.** This figure compares accuracy drop under input-embedding perturbations and router-parameter perturbations. StaVER shows smaller performance drops and lower variance than StableMoE across perturbation levels.

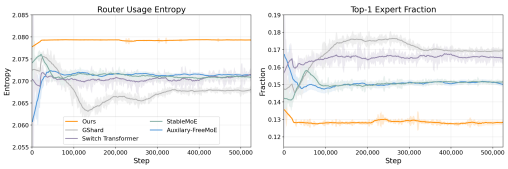


Figure 3: **Load balancing behavior during pretraining.** We compare RUE and Top-1 EF across our method and various baselines. Higher RUE indicates more balanced expert activation, while a lower Top-1 EF suggests less load concentration.

Table 3: **Ablation of StaVER components.** Downstream accuracy of models trained by replacing R_{stoch} with Top-k routing and $\mathcal{L}_{\text{prior}}$ with $\mathcal{L}_{\text{aux}}$, individually or jointly ("w/o all").

Variant	ARC-E	PIQA	SIQA	WINO	HELLA	MNLI	QNLI	SST-2	AVG ↑
StaVER (full)	**44.4**	**61.3**	**39.2**	**51.2**	**33.2**	**35.7**	**52.4**	**64.0**	**47.7**
w/o R_{stoch}	42.4$^{\downarrow 2.0}$	59.4$^{\downarrow 1.9}$	38.4$^{\downarrow 0.8}$	49.9$^{\downarrow 1.3}$	32.6$^{\downarrow 0.6}$	34.8$^{\downarrow 0.9}$	47.7$^{\downarrow 4.7}$	57.3$^{\downarrow 6.7}$	45.3$^{\downarrow 2.4}$
w/o $\mathcal{L}_{\text{prior}}$	40.5$^{\downarrow 3.9}$	60.7$^{\downarrow 0.6}$	38.0$^{\downarrow 1.2}$	50.7$^{\downarrow 0.5}$	32.7$^{\downarrow 0.5}$	35.0$^{\downarrow 0.7}$	48.5$^{\downarrow 3.9}$	60.4$^{\downarrow 3.6}$	45.8$^{\downarrow 1.9}$
w/o all	38.8$^{\downarrow 5.6}$	60.6$^{\downarrow 0.7}$	38.0$^{\downarrow 1.2}$	50.7$^{\downarrow 0.5}$	32.2$^{\downarrow 1.0}$	34.6$^{\downarrow 1.1}$	47.5$^{\downarrow 4.9}$	59.2$^{\downarrow 4.8}$	45.2$^{\downarrow 2.5}$

drops under both perturbation types. The accuracy gap between the two methods is larger under perturbations than without perturbations, indicating that StaVER's advantage extends beyond its higher initial accuracy to greater robustness. These results further support the effectiveness of the proposed stochastic router in improving robustness to perturbations.

4.4 Prior Matching Loss (RQ3)

To verify that the prior matching term $\mathcal{L}_{\text{prior}}$ in our unified variational objective promotes load balancing, we analyze expert utilization from two perspectives. We first track Routing Usage Entropy (RUE) and Top-1 Expert Fraction (Top-1 EF) during pretraining. We then examine expert load distributions on SST-2, measured by the percentage of tokens processed by each expert. Detailed metric definitions are provided in Appendix D.

Finding 5. $\mathcal{L}_{\text{prior}}$ improves overall load balance and reduces load concentration. We compare our prior matching term with four baselines covering auxiliary-loss-based and auxiliary-free balancing. Figure 3 shows that StaVER achieves the highest RUE throughout training, indicating more uniform expert usage in expectation. The lower Top-1 Expert Fraction further indicates reduced concentration of token assignments on the most frequently activated expert. Concentrating training tokens on a few experts leaves the others under-trained and model capacity underutilized. These improvements support more balanced allocation of training tokens across experts.

Finding 6. $\mathcal{L}_{\text{prior}}$ promotes more consistent load balancing. We further observe that the baselines exhibit higher variance in load-balancing performance, whereas StaVER achieves more consistent results (Figure 3). Conventional auxiliary losses adjust routing probabilities based on observed expert load, but hard Top-k selection makes actual load respond discontinuously to these adjustments. Auxiliary-Free balancing uses fixed bias updates that can amplify small load differences into larger changes in expert assignments. In contrast, $\mathcal{L}_{\text{prior}}$ directly regularizes the routing distribution governing stochastic routing in StaVER. The resulting empirical expert usage is an unbiased estimate of the routing probabilities, allowing prior matching to regulate expected expert load directly.

Finding 7. $\mathcal{L}_{\text{prior}}$ enables balanced expert utilization across all layers. Figure 4 compares expert load distributions on SST-2. Top-k without $\mathcal{L}_{\text{aux}}$ concentrates activations on a few experts, leaving several others almost unused. Adding $\mathcal{L}_{\text{aux}}$ substantially improves balance, yielding load entropy between 2.04 and 2.07. Auxiliary-FreeMoE achieves comparable balance in some layers but retains near-zero expert usage in others, with entropy falling to 1.79. Meanwhile, our method maintains load entropy near the maximum $\ln 8 \approx 2.08$ in all 12 layers. Expert activation shares remain close to $1/8$, indicating balanced utilization of all eight experts at every depth.

4.5 Ablation Study (RQ4)

To further evaluate the contributions of stochastic routing and prior term to model performance, we train models with stochastic router R_{stoch} replaced by deterministic Top-k routing, $\mathcal{L}_{\text{prior}}$ replaced by the conventional auxiliary loss $\mathcal{L}_{\text{aux}}$, or both. We then evaluate the model on downstream tasks.

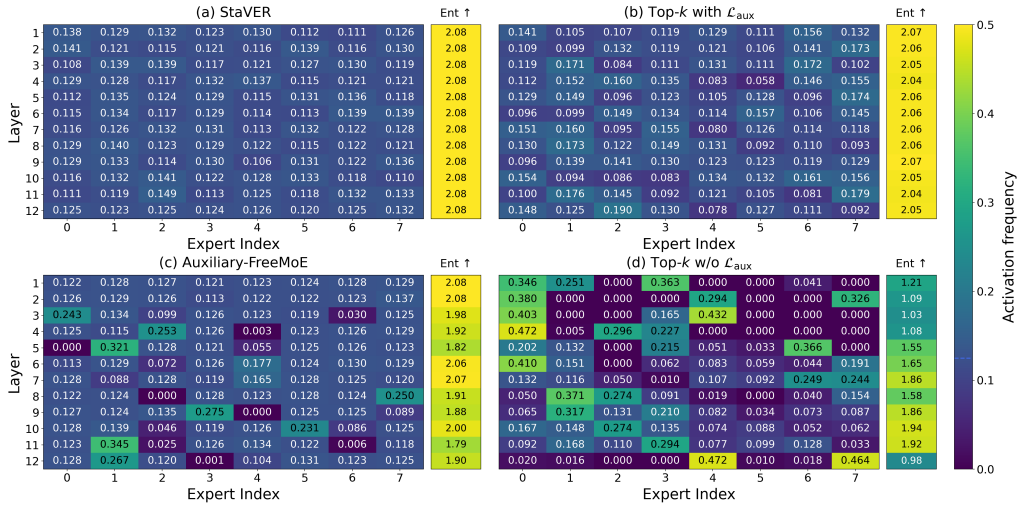


Figure 4: **Statistical analysis of expert load.** Each row reports normalized expert activation frequencies within a layer. The adjacent column gives the corresponding load entropy (Ent ↑), with higher values indicating more balanced expert utilization. StaVER maintains near-uniform expert usage across all layers.

Finding 8. The combination of stochastic routing with $\mathcal{L}_{\text{prior}}$ enhances model performance. As shown in Table 3, StaVER achieves the highest accuracy on all eight benchmarks. Replacing R_{stoch} reduces average accuracy from 47.7 to 45.3, leaving only a marginal improvement over the conventional baseline of 45.2. This indicates that retaining the prior alone with deterministic routing recovers little of StaVER's average performance gain. Replacing $\mathcal{L}_{\text{prior}}$ while retaining stochastic routing lowers average accuracy to 45.8. Although this variant outperforms the conventional baseline on every benchmark, it consistently falls below the full model, further supporting the benefit of the prior regularization analyzed earlier. Jointly replacing both components reduces average accuracy by 2.5 points to 45.2. These comparisons support using stochastic routing together with prior regularization to achieve the strongest downstream performance. We further conduct sensitivity analyses of hyperparameters in StaVER, with detailed discussions provided in Appendix G.5.

5 Related Work

Probabilistic Routing. Shazeer et al. (2017) introduce noise into routing scores before Top-k selection. To address the non-differentiability of Top-k routing, SparseMixer Liu et al. (2023) uses a midpoint approximation to estimate routing gradients, while GRIN Liu et al. (2024b) extends this approach with masked expert sampling and higher-order gradient estimation. ProbMoE Zhao et al. (2026) samples cardinality-constrained expert subsets and estimates routing gradients through exact expert-selection marginals. DirMoE Vahidi et al. (2026) models expert activation and combination weights separately with Bernoulli and Dirichlet distributions. More recently, Li & Wicker (2026) introduce Bayesian inference over selection-space through an additional inference network to estimate predictive uncertainty. We instead formulate routing itself as variational inference to improve routing stability during pretraining, without an additional inference module or extra expert computation.

Load Balancing. Load-balancing losses encourage balanced expert utilization in MoEs Lepikhin et al. (2020); Fedus et al. (2022). Strong balancing penalties can interfere with task optimization, motivating alternatives such as load-dependent routing biases Wang et al. (2024) and additional objectives for expert specialization Guo et al. (2026). Our method applies prior matching to the batch-averaged distribution used for Monte Carlo expert sampling. This regularizes expected expert selection frequencies while allowing individual tokens to favor different experts.

6 Conclusion

In this work, we present StaVER, a unified probabilistic framework for stable and balanced MoE routing. Through variational inference, StaVER addresses instability induced by hard Top-k selection and promotes load balancing without the need of heuristic auxiliary loss. Experiments show consistent performance gains and improved routing stability with reduced training FLOPs, supporting the proposed method as an effective approach towards stable and efficient MoE training.

7 AI USE STATEMENT

In this work, we used generative AI tools to implement methods and to design or provide feedback on research methodology or experiments. We have not used generative AI tools to generate synthetic datasets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in writing proofs, propose or refine hypotheses, assist with translation, clean and reformat datasets, support qualitative and thematic data analysis, or interpret results. Additionally, we used generative AI tools to draft parts of the research paper, edit the paper to improve readability, identify relevant literature, source and search for information, create or edit software code, and create or modify scientific figures or images.

We have reviewed all AI-assisted work. We evaluated AI-assisted methodological and experimental suggestions for scientific validity and relevance to our research objectives. We checked the manuscript for accuracy and clarity, verified citations against the original sources, and confirmed that cited works support the associated statements. We inspected and tested AI-assisted code for correctness and consistency with the described methods. We checked scientific visualizations against the underlying data and results, including their labels, scales, and captions. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Chirag Agarwal, Daniel D'souza, and Sara Hooker. Estimating example difficulty using variance of gradients, 2022.

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

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.

Dheeraj Baby, Boran Han, Shuai Zhang, Cuixiong Hu, Bernie Wang, and Yu-Xiang Wang. Adapting to online distribution shifts in deep learning: A black-box approach. In *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pp. 3853–3861, 2025.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019.

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A survey on mixture of experts in large language models. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

Zewen Chi, Li Dong, Shaohan Huang, Damai Dai, Shuming Ma, Barun Patra, Saksham Singhal, Payal Bajaj, Xia Song, Xian-Ling Mao, et al. On the representation collapse of sparse mixture of experts. *Advances in Neural Information Processing Systems*, 35:34600–34613, 2022.

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. *arXiv:1803.05457v1*, 2018.

Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. StableMoE: Stable Routing Strategy for Mixture of Experts, April 2022. arXiv:2204.08396 [cs].

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.

William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity, June 2022. arXiv:2101.03961 [cs].

Hongcan Guo, Haolang Lu, Guoshun Nan, Bolun Chu, Jialin Zhuang, Yuan Yang, Wenhao Che, Xinye Cao, Sicong Leng, Qimei Cui, and Xudong Jiang. Advancing expert specialization for better moe, 2026.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax, August 2017. arXiv:1611.01144 [stat].

Masahiro Kada, Ryota Yoshihashi, Satoshi Ikehata, Rei Kawakami, and Ikuro Sato. Teacher-guided routing for sparse vision mixture-of-experts. *arXiv preprint arXiv:2604.21330*, 2026.

Junmo Kang, Leonid Karlinsky, Hongyin Luo, Zhen Wang, Jacob Hansen, James Glass, David Cox, Rameswar Panda, Rogerio Feris, and Alan Ritter. Self-moe: Towards compositional large language models with self-specialized experts. *arXiv preprint arXiv:2406.12034*, 2024.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

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.

Albus Yizhuo Li and Matthew Wicker. Variational routing: A scalable Bayesian framework for calibrated mixture-of-experts transformers. In *International Conference on Machine Learning*, 2026.

Aonian Li, Bangwei Gong, Bo Yang, Boji Shan, Chang Liu, Cheng Zhu, Chunhao Zhang, Congchao Guo, Da Chen, Dong Li, et al. Minimax-01: Scaling foundation models with lightning attention. *arXiv preprint arXiv:2501.08313*, 2025.

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*, 2024a.

Liyuan Liu, Jianfeng Gao, and Weizhu Chen. Sparse backpropagation for MoE training, 2023.

Liyuan Liu, Young Jin Kim, Shuohang Wang, Chen Liang, Yelong Shen, Hao Cheng, Xiaodong Liu, Masahiro Tanaka, Xiaoxia Wu, Wenxiang Hu, Vishrav Chaudhary, Zeqi Lin, Chenruidong Zhang, Jilong Xue, Hany Awadalla, Jianfeng Gao, and Weizhu Chen. GRIN: GRadient-INformed MoE, 2024b.

Ang Lv, Jin Ma, Yiyuan Ma, and Siyuan Qiao. Coupling experts and routers in mixture-of-experts via an auxiliary loss. *arXiv preprint arXiv:2512.23447*, 2025.

Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun Ma, and Wenhu Chen. General-reasoner: Advancing llm reasoning across all domains. *Advances in Neural Information Processing Systems*, 38:56596–56618, 2026.

Charles C. Margossian and David M. Blei. Amortized variational inference: When and why? In *Proceedings of the Fortieth Conference on Uncertainty in Artificial Intelligence*, volume 244 of *Proceedings of Machine Learning Research*, pp. 2434–2449. PMLR, 2024.

Siyuan Mu and Sen Lin. A Comprehensive Survey of Mixture-of-Experts: Algorithms, Theory, and Applications, April 2025. arXiv:2503.07137 [cs].

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, K Lo, J Morrison, S Min, W Shi, P Walsh, O Tafjord, N Lambert, et al. Olmoe: Open mixture-of-experts language models. *URL https://arxiv.org/abs/2409.02060*, 2024.

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

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. In *EMNLP*, 2019.

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.

Ayush Singh, Navpreet Singh, and Shubham Vatsal. Robustness of large language models to perturbations in text, 2025.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Evan Walsh, Luke Zettlemoyer, Noah Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. Dolma: an open corpus of three trillion tokens for language model pretraining research. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15725–15788, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.840.

Qwen Team. Qwen3. 5-omni technical report. *arXiv preprint arXiv:2604.15804*, 2026.

Amirhossein Vahidi, Hesam Asadollahzadeh, Navid Akhavan Attar, Marie Moullet, Kevin Ly, Xingyi Yang, and Mohammad Lotfollahi. DirMoE: Dirichlet-routed mixture of experts. In *International Conference on Learning Representations*, 2026.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Tal Linzen, Grzegorz Chrupała, and Afra Alishahi (eds.), *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://aclanthology.org/W18-5446/.

Haoyu Wang, Guozheng Ma, Cong Yu, Ning Gui, Linrui Zhang, Zhiqi Huang, Suwei Ma, Yongzhe Chang, Sen Zhang, Li Shen, et al. Are large language models really robust to word-level perturbations? *arXiv preprint arXiv:2309.11166*, 2023.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv preprint arXiv:2408.15664*, 2024.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *Advances in Neural Information Processing Systems*, 38:78500–78525, 2026.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122. Association for Computational Linguistics, 2018.

Boxi Wu, Jie Jiang, Haidong Ren, Zifan Du, Wenxiao Wang, Zhifeng Li, Deng Cai, Xiaofei He, Binbin Lin, and Wei Liu. Towards in-distribution compatibility in out-of-distribution detection, 2022.

Jingfeng Wu, Wenqing Hu, Haoyi Xiong, Jun Huan, Vladimir Braverman, and Zhanxing Zhu. On the noisy gradient descent that generalizes as sgd. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10367–10376, 2020.

Yu Yang, Tian Yu Liu, and Baharan Mirzasoleiman. Not all poisons are created equal: Robust training against data poisoning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 25154–25165, 2022.

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*, 2019.

Heng Zhao, Zilei Shao, Guy Van den Broeck, and Zhe Zeng. ProbMoE: Differentiable probabilistic routing for mixture-of-experts. In *Forty-third International Conference on Machine Learning*, 2026.

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.

Appendix

A Algorithm **15**

B Detailed Methods **16**

B.1 Instability of hard Top-k routing 16

B.2 Stability of stochastic routing under gradient perturbations 17

B.3 Variational motivation for the training objective 19

B.4 Straight-Through Gumbel-Softmax 22

C Perturbing Types **22**

D Evaluation Metrics **23**

E Baselines **23**

F Implementation Details **24**

G Additional Experiments **24**

G.1 Training Dynamics and Convergence Comparison 24

G.2 Performance across Pretraining Corpora 25

G.3 Performance at Larger Model Scales 26

G.4 Training Efficiency of Sampling-Based Routing 26

G.5 Hyperparameter Sensitivity 27

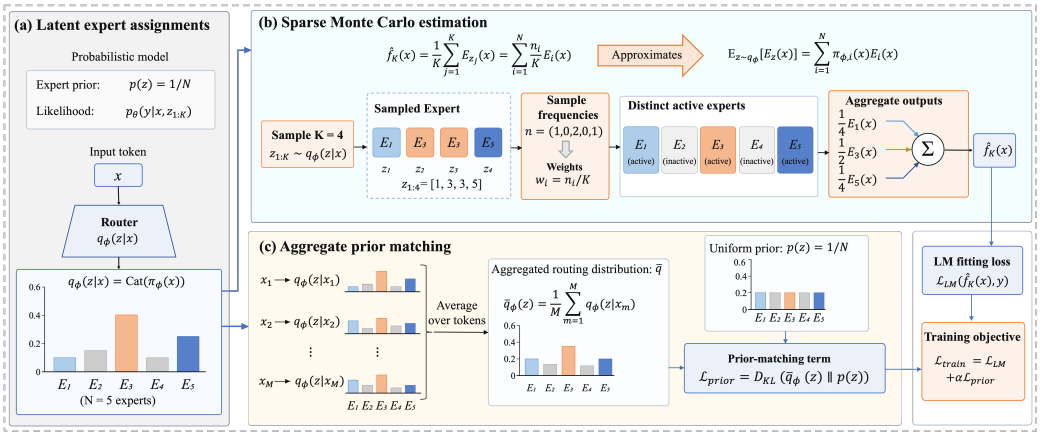


Figure 5: **Overview of our MoE framework from a Bayesian perspective.** **(a)** Expert assignments are modeled as latent variables with a uniform prior, and the router parameterizes a variational distribution $q_\phi(z \mid x)$. **(b)** We sample K expert indices with replacement and aggregate distinct expert outputs by relative sampling frequency, yielding a sparse Monte Carlo estimate of the expected MoE output. **(c)** Aggregate prior matching regularizes the routing distribution averaged across tokens toward the uniform prior. Training combines the language modeling and prior-matching losses.

A Algorithm

The forward pass of `StaVER` is illustrated in Algorithm 1. During training, this procedure is applied at each MoE layer to compute both the language modeling loss and the prior-matching loss. In the backward pass, gradients are propagated through the soft Gumbel logits, enabling efficient end-to-end training.

Algorithm 1: Forward Pass of a `StaVER` Layer

Input: Token representations $X = (x_b)_{b=1}^{B}$; number of experts N; number of samples K; temperature $\tau > 0$; router $r_\phi(\cdot)$; experts $\{E_n(\cdot)\}_{n=1}^{N}$

Output: Layer outputs $\{\hat{f}_K(x_b)\}_{b=1}^{B}$ and prior-matching loss $\mathcal{L}_{\text{prior}}$

for $b \leftarrow 1$ **to** B **do**
- $x \leftarrow x_b$;
- Compute router logits $\ell_\phi(x) \leftarrow r_\phi(x) \in \mathbb{R}^N$;
- Compute routing probabilities $\pi_\phi(x) \leftarrow \text{softmax}\big(\ell_\phi(x)\big) \in \Delta^{N-1}$;
- **for** $j \leftarrow 1$ **to** K **do**
  - Sample $g_{j,n} \sim \text{Gumbel}(0,1)$ independently for $n = 1, \ldots, N$;
  - Compute the relaxed categorical sample

$$\tilde{z}_{j,n} \leftarrow \frac{\exp\big(\big(\log \pi_{\phi,n}(x) + g_{j,n}\big)/\tau\big)}{\sum_{m=1}^{N} \exp\big(\big(\log \pi_{\phi,m}(x) + g_{j,m}\big)/\tau\big)}, \qquad n = 1, \ldots, N.$$

  - Compute the hard assignment $z_j^{\text{hard}} \leftarrow \text{onehot}(\arg\max_n \tilde{z}_{j,n})$.
  - Apply the straight-through estimator $z_j \leftarrow \text{sg}\big(z_j^{\text{hard}} - \tilde{z}_j\big) + \tilde{z}_j$;
- Compute the Monte Carlo MoE output $\hat{f}_K(x_b) \leftarrow \frac{1}{K}\sum_{j=1}^{K}\sum_{n=1}^{N} z_{j,n} E_n(x_b)$.

Compute the aggregated routing probabilities $\bar{\pi}_{\phi,n} \leftarrow \frac{1}{B}\sum_{b=1}^{B} \pi_{\phi,n}(x_b), n = 1, \ldots, N$.

Compute the prior-matching loss $\mathcal{L}_{\text{prior}} \leftarrow \sum_{n=1}^{N} \bar{\pi}_{\phi,n} \log\left(\frac{\bar{\pi}_{\phi,n}}{1/N}\right)$.

return $\{\hat{f}_K(x_b)\}_{b=1}^{B}$ *and* $\mathcal{L}_{\text{prior}}$;

B Detailed Methods

B.1 Instability of hard Top-k routing

Hard Top-k routing selects experts according to their rank. Its behavior therefore depends most strongly on the two experts at the selection boundary, i.e., the lowest-ranked selected expert and the highest-ranked unselected expert. If their logits are close, a small change in the router can exchange their positions, even when the remaining logits change only slightly. This section makes that observation precise for perturbations to a router gradient.

As in Section 2.2, fix a token representation $x \neq 0$ and consider the linear router $r_W(x) = Wx$. Let $\mathcal{S}_k(W, x)$ denote its selected expert set. Write a for the expert with the k-th largest logit and b for the expert with the $(k+1)$-th largest logit, and assume their boundary margin is positive:

$$\Delta_k(x) = (e_a - e_b)^\top Wx > 0.$$

The positive margin makes the selected set unambiguous at the reference router. Since softmax preserves the ordering of logits, the same selected set would be obtained by ranking routing probabilities.

To connect this margin to training, recall the two updates in Eq. 6. We take the update without gradient perturbation, $W = W^{(n+1)}$, as the reference router. An additive gradient perturbation ξ then produces $\widehat{W} = W - \eta\xi$, where $\eta > 0$. We first ask when this perturbation is sufficient to exchange the two boundary experts. We then ask for the smallest perturbation norm that can change the selected set at all.

Lemma B.1 (Sufficient condition for a Top-k routing change)**.** *Under the definitions above, if*

$$\eta\langle \xi, (e_a - e_b)x^\top \rangle_F > \Delta_k(x),$$

then $\mathcal{S}_k(\widehat{W}, x) \neq \mathcal{S}_k(W, x)$.

The condition compares the original boundary margin with the component of the gradient perturbation that closes it. A perturbation in another direction may have little effect on this particular pair, regardless of its overall norm.

Proof. We examine the logits of the two experts on opposite sides of the reference selection boundary. At the perturbed router, their gap is

$$\begin{aligned}(e_a - e_b)^\top \widehat{W}x &= (e_a - e_b)^\top (W - \eta\xi)x \\ &= \Delta_k(x) - \eta\langle \xi, (e_a - e_b)x^\top \rangle_F.\end{aligned}$$

The stated inequality makes this gap negative, so b ranks above a after the perturbation. Because a was selected and b was not, they cannot both retain their original membership in a Top-k set. Hence the selected set changes. This gives the sufficient condition stated as Eq. 7 in the main text. □

Equation 7 identifies a direction that causes a change, but it does not yet quantify how large a perturbation must be. For that purpose, we measure perturbations using the Frobenius norm. Every change to the selected set requires at least one expert selected at W to exchange rank with an expert unselected at W. The smallest reference logit gap among such pairs is $\Delta_k(x)$, attained by (a, b). The next lemma shows that this pair determines the distance to the nearest selection boundary.

Lemma B.2 (Distance to a Top-k selection boundary)**.** *For the fixed $x \neq 0$ and $\eta > 0$ above, let $\rho_{\mathrm{grad}}(x)$ be the infimum of $\|\xi\|_F$ over gradient perturbations for which $\mathcal{S}_k(W - \eta\xi, x) \neq \mathcal{S}_k(W, x)$. Then*

$$\rho_{\mathrm{grad}}(x) = \frac{\Delta_k(x)}{\sqrt{2}\eta\|x\|_2}.$$

The same value is the minimum norm required to reach a boundary between a selected and an unselected expert.

Proof. We first establish a lower bound that applies to every possible selected–unselected exchange. Take $i \in \mathcal{S}_k(W, x)$ and $j \notin \mathcal{S}_k(W, x)$, and define their reference gap by

$$d_{ij}(x) = (e_i - e_j)^\top Wx.$$

By the definitions of a and b, $d_{ij}(x) \geq \Delta_k(x)$. To make i and j tie at the perturbed router, the perturbation must satisfy

$$\eta\langle \xi, (e_i - e_j)x^\top \rangle_F = d_{ij}(x).$$

Cauchy–Schwarz, together with $\|(e_i - e_j)x^\top\|_F = \sqrt{2}\|x\|_2$, therefore gives

$$\|\xi\|_F \geq \frac{d_{ij}(x)}{\sqrt{2}\eta\|x\|_2} \geq \frac{\Delta_k(x)}{\sqrt{2}\eta\|x\|_2}.$$

Any change to the selected set must cross at least one such boundary, so this is also a lower bound on the required norm.

We next show that the bound can be reached. Choose a perturbation aligned with the boundary pair:

$$\xi^\star = \frac{\Delta_k(x)}{2\eta\|x\|_2^2}(e_a - e_b)x^\top.$$

Its effect on the gap is

$$\eta\langle \xi^\star, (e_a - e_b)x^\top \rangle_F = \Delta_k(x),$$

so a and b tie. Moreover,

$$\|\xi^\star\|_F = \frac{\Delta_k(x)}{\sqrt{2}\eta\|x\|_2},$$

which attains the lower bound.

For every sufficiently small $\delta > 0$, $\xi^\star + \delta$ strictly reverses the ordering of a and b. Its norm approaches $\|\xi^\star\|_F$ as $\delta \to 0$. Consequently, the minimum distance to a selection boundary and the infimum norm for a strict support change have the same value. This is Eq. 8 in the main text. □

B.2 Stability of stochastic routing under gradient perturbations

Section B.1 shows that hard Top-k routing can change its selected expert set discontinuously when a perturbation crosses a selection boundary. We now examine how the same gradient perturbation affects stochastic routing.

As in Section B.1, fix a token representation $x \neq 0$ and consider the linear router $r_W(x) = Wx$. Let

$$\pi_W(x) = \mathrm{softmax}(Wx)$$

denote the corresponding routing distribution. The perturbed routing distribution is therefore $\pi_{\widehat{W}}(x) = \mathrm{softmax}(\widehat{W}x)$.

Stochastic routing samples expert assignments from these distributions. Consequently, its expected expert-selection frequencies change from $\pi_W(x)$ to $\pi_{\widehat{W}}(x)$. We measure this change using their total variation distance:

$$D_{\mathrm{stoch}}(x) := \frac{1}{2}\left\|\pi_{\widehat{W}}(x) - \pi_W(x)\right\|_1.$$

This quantity concerns the underlying routing distributions rather than the random difference between two finite collections of sampled experts. It therefore captures how the expected selection frequencies respond to the perturbation.

The argument proceeds in three parts. We first establish a Lipschitz property of the softmax mapping. We then use it to control $D_{\mathrm{stoch}}(x)$ in terms of $\|\xi\|_F$. Finally, we apply the boundary construction from Section B.1 to compare the continuous change in stochastic routing with the discrete change in hard Top-k routing.

Lemma B.3 (Softmax stability under logit perturbations)**.** *For any $u, v \in \mathbb{R}^N$,*

$$\|\mathrm{softmax}(v) - \mathrm{softmax}(u)\|_1 \leq \|v - u\|_\infty.$$

This lemma states that softmax does not amplify a change in logits when the input and output changes are measured using the ℓ_∞ and ℓ_1 norms, respectively. It provides the continuity needed to distinguish stochastic routing from hard Top-k selection.

Proof. We connect u and v by the line segment

$$u_t = u + t(v - u), \qquad t \in [0, 1],$$

and define

$$p_t = \mathrm{softmax}(u_t).$$

The softmax Jacobian along this path is

$$J_t = \mathrm{diag}(p_t) - p_t p_t^\top.$$

For any $h \in \mathbb{R}^N$, let $\mu_t = p_t^\top h$. The i-th coordinate of $J_t h$ is $p_{t,i}(h_i - \mu_t)$, and hence

$$\|J_t h\|_1 = \sum_{i=1}^{N} p_{t,i}|h_i - \mu_t|.$$

The expression on the right is the mean absolute deviation of a random variable that takes the value h_i with probability $p_{t,i}$. Since each h_i lies in $[-\|h\|_\infty, \|h\|_\infty]$, its mean absolute deviation is at most $\|h\|_\infty$. Therefore,

$$\|J_t h\|_1 \le \|h\|_\infty.$$

Applying the fundamental theorem of calculus along the path gives

$$\mathrm{softmax}(v) - \mathrm{softmax}(u) = \int_0^1 J_t(v - u)\, dt.$$

Taking the ℓ_1 norm and using the preceding Jacobian bound, we obtain

$$\begin{aligned} \|\mathrm{softmax}(v) - \mathrm{softmax}(u)\|_1 &\le \int_0^1 \|J_t(v - u)\|_1\, dt \\ &\le \int_0^1 \|v - u\|_\infty\, dt \\ &= \|v - u\|_\infty. \end{aligned}$$

This proves the result. □

We can now translate this general softmax property into a bound on the change in the stochastic routing distribution. The next lemma makes explicit how the router perturbation ξ changes the logits.

Lemma B.4 (Stability of the stochastic routing distribution)**.** *For the reference router W and perturbed router $\widehat{W} = W - \eta\xi$,*

$$D_{\mathrm{stoch}}(x) \le \frac{\eta}{2}\|\xi\|_F \|x\|_2.$$

The bound is linear in the perturbation norm. Thus, with x and η fixed, the change in the stochastic routing distribution converges to zero whenever the gradient perturbation converges to zero.

Proof. Let

$$u = Wx, \qquad \widehat{u} = \widehat{W}x.$$

Because $\widehat{W} = W - \eta\xi$, their difference is

$$\widehat{u} - u = -\eta\xi x.$$

Applying Lemma B.3 yields

$$\begin{aligned} D_{\mathrm{stoch}}(x) &= \frac{1}{2}\,\|\mathrm{softmax}(\widehat{u}) - \mathrm{softmax}(u)\|_1 \\ &\le \frac{1}{2}\|\widehat{u} - u\|_\infty = \frac{\eta}{2}\|\xi x\|_\infty \\ &\le \frac{\eta}{2}\|\xi x\|_2 \le \frac{\eta}{2}\|\xi\|_F \|x\|_2. \end{aligned}$$

The final inequality follows from the compatibility of the Frobenius and Euclidean norms. This proves Eq. 12. □

The bound applies to any perturbation. To establish the remaining claims of Proposition 3.1, we choose one that crosses the nearest hard Top-k boundary. Let a, b, $\Delta_k(x)$, and $\rho_{\mathrm{grad}}(x)$ be as defined in Section B.1. In particular, assume $\Delta_k(x) > 0$ and $1 \le k < N$.

Lemma B.5 (Stochastic routing at a Top-k boundary crossing)**.** *For every $\epsilon > 0$, there exists a gradient perturbation ξ such that*

$$\|\xi\|_F = \rho_{\mathrm{grad}}(x) + \epsilon$$

and

$$\mathcal{S}_k(W - \eta\xi, x) \neq \mathcal{S}_k(W, x).$$

Under this same perturbation,

$$D_{\mathrm{stoch}}(x) \le \frac{\Delta_k(x)}{2\sqrt{2}} + \frac{\eta\epsilon}{2}\|x\|_2.$$

Proof. We begin with the direction that reaches the nearest boundary in Section B.1, and extend the perturbation slightly in that direction. Set

$$\xi := \left(\frac{\Delta_k(x)}{2\eta\|x\|_2^2} + \frac{\epsilon}{\sqrt{2}\|x\|_2}\right)(e_a - e_b)x^\top.$$

Because $\|(e_a - e_b)x^\top\|_F = \sqrt{2}\|x\|_2$, its norm is

$$\begin{aligned}\|\xi\|_F &= \frac{\Delta_k(x)}{\sqrt{2}\eta\|x\|_2} + \epsilon \\ &= \rho_{\mathrm{grad}}(x) + \epsilon.\end{aligned}$$

We next show that the perturbation changes the selected set. The reduction in the logit gap between a and b is

$$\begin{aligned}\eta\langle \xi, (e_a - e_b)x^\top\rangle_F &= \Delta_k(x) + \sqrt{2}\eta\epsilon\|x\|_2 \\ &> \Delta_k(x).\end{aligned}$$

The sufficient condition in Lemma B.1 therefore applies:

$$\mathcal{S}_k(W - \eta\xi, x) \neq \mathcal{S}_k(W, x).$$

Finally, we use Lemma B.4 for the same ξ. By the expression for $\rho_{\mathrm{grad}}(x)$ in Lemma B.2,

$$\begin{aligned}D_{\mathrm{stoch}}(x) &\le \frac{\eta}{2}\|\xi\|_F\|x\|_2 \\ &= \frac{\eta}{2}\big(\rho_{\mathrm{grad}}(x) + \epsilon\big)\|x\|_2 \\ &= \frac{\Delta_k(x)}{2\sqrt{2}} + \frac{\eta\epsilon}{2}\|x\|_2.\end{aligned}$$

□

For fixed $x \neq 0$ and $\eta > 0$, the bound in the final lemma approaches zero as $\Delta_k(x)$ and ϵ approach zero. Nevertheless, every constructed perturbation with $\epsilon > 0$ changes the hard Top-k selected set. Thus the routing distribution can change by a vanishing amount under a perturbation that crosses a hard selection boundary.

B.3 Variational motivation for the training objective

Section 3.2 introduces latent expert assignments to connect stochastic routing with variational inference. For a training pair (x_m, y_m), the likelihood in Eq. 15 evaluates the prediction made using sampled assignments $z_{1:K}$. The posterior in Eq. 16 describes the assignments after observing y_m. Because this posterior is generally intractable, Eq. 17 uses a factorized routing distribution to approximate it.

We first derive the exact negative evidence lower bound (ELBO) associated with this approximation. We then examine what its prior term encourages. That examination motivates the aggregated-prior objective in Eq. 19 and explains its load-balancing term in Eq. 20.

For brevity, write

$$q_m(z_{1:K}) := q_\phi(z_{1:K} \mid x_m) = \prod_{j=1}^{K} q_\phi(z_j \mid x_m), \qquad p(z_{1:K}) = \prod_{j=1}^{K} p(z_j).$$

We assume that $p(z = i) > 0$ for every expert i, so the KL divergences below are finite.

Lemma B.6 (Exact variational objective)**.** *For fixed model parameters θ, minimizing the empirical posterior KL in Eq. 18 with respect to ϕ is equivalent, up to an additive term independent of ϕ, to minimizing*

$$\mathcal{L}_{\mathrm{ELBO}} := -\frac{1}{M}\sum_{m=1}^{M} \mathbb{E}_{z_{1:K}\sim q_m} \log p_\theta(y_m \mid x_m, z_{1:K}) + \frac{K}{M}\sum_{m=1}^{M} D_{\mathrm{KL}}\big(q_\phi(z \mid x_m)\|p(z)\big). \tag{21}$$

By Eq. 15, the likelihood in the first term can also be written as $p_\theta(y_m \mid \hat{f}_K(x_m; z_{1:K}))$.

The first term fits the prediction target under sampled expert assignments. The second compares each example's routing distribution with the prior. Its coefficient is K because the variational distribution contains K independently sampled assignments.

Proof. We start with one training pair. Substituting Bayes' rule from Eq. 16 into its posterior KL gives

$$\begin{aligned} &D_{\mathrm{KL}}\big(q_m(z_{1:K})\|p_\theta(z_{1:K} \mid x_m, y_m)\big) \\ &\quad= \mathbb{E}_{q_m} \log \frac{q_m(z_{1:K})}{p_\theta(z_{1:K} \mid x_m, y_m)} \\ &\quad= -\mathbb{E}_{q_m} \log p_\theta(y_m \mid x_m, z_{1:K}) \\ &\qquad + D_{\mathrm{KL}}\big(q_m(z_{1:K})\|p(z_{1:K})\big) + \log p_\theta(y_m \mid x_m). \end{aligned}$$

The final term does not depend on ϕ. Hence the first two terms determine the minimization with respect to the variational parameters.

We next use the factorization in Eq. 17 and the factorized prior. Since each sampled assignment has the same routing distribution,

$$\begin{aligned} D_{\mathrm{KL}}\big(q_m(z_{1:K})\|p(z_{1:K})\big) &= \sum_{j=1}^{K} D_{\mathrm{KL}}\big(q_\phi(z_j \mid x_m)\|p(z_j)\big) \\ &= K D_{\mathrm{KL}}\big(q_\phi(z \mid x_m)\|p(z)\big). \end{aligned}$$

Averaging over the M training pairs proves Eq. 21. □

We now examine the effect of the token-level prior penalty in Eq. 21. Define the aggregated routing distribution, as in Proposition 3.2, by

$$\bar{q}_\phi(z) := \frac{1}{M}\sum_{m=1}^{M} q_\phi(z \mid x_m).$$

For each m, split the logarithm in the token-level KL:

$$\log \frac{q_\phi(z \mid x_m)}{p(z)} = \log \frac{q_\phi(z \mid x_m)}{\bar{q}_\phi(z)} + \log \frac{\bar{q}_\phi(z)}{p(z)}.$$

Taking the expectation under $q_\phi(z \mid x_m)$ and averaging over m yields

$$\frac{1}{M}\sum_{m=1}^{M} D_{\mathrm{KL}}\big(q_\phi(z \mid x_m)\|p(z)\big)$$

$$= \frac{1}{M}\sum_{m=1}^{M} D_{\mathrm{KL}}\big(q_\phi(z \mid x_m)\|\bar{q}_\phi(z)\big) + \frac{1}{M}\sum_{m=1}^{M}\sum_{z} q_\phi(z \mid x_m)\log\frac{\bar{q}_\phi(z)}{p(z)}. \tag{22}$$

Using $M^{-1}\sum_m q_\phi(z \mid x_m) = \bar{q}_\phi(z)$ in the last term gives

$$\frac{1}{M}\sum_{m=1}^{M} D_{\mathrm{KL}}\big(q_\phi(z \mid x_m)\|p(z)\big) = D_{\mathrm{KL}}\big(\bar{q}_\phi(z)\|p(z)\big) + \frac{1}{M}\sum_{m=1}^{M} D_{\mathrm{KL}}\big(q_\phi(z \mid x_m)\|\bar{q}_\phi(z)\big). \tag{23}$$

The second term in Eq. 23 has a useful interpretation. Let T be a uniformly sampled index from $\{1, \ldots, M\}$, and then sample an expert assignment according to $Z \mid T = m \sim q_\phi(z \mid x_m)$. The marginal distribution of Z is $\bar{q}_\phi(z)$. By the definition of mutual information,

$$I_\phi(T; Z) = \frac{1}{M}\sum_{m=1}^{M} D_{\mathrm{KL}}\big(q_\phi(z \mid x_m)\|\bar{q}_\phi(z)\big). \tag{24}$$

Combining Eqs. 21, 23, and 24, we can express the exact negative ELBO as

$$\mathcal{L}_{\mathrm{ELBO}} = \mathcal{L}_{\mathrm{LM}} + K D_{\mathrm{KL}}\big(\bar{q}_\phi(z)\|p(z)\big) + K I_\phi(T; Z), \tag{25}$$

where $\mathcal{L}_{\mathrm{LM}}$ is the expected negative log-likelihood defined in Eq. 19.

Equation 25 separates two effects of the exact prior penalty. The aggregated KL encourages the average routing distribution to match the prior. The mutual-information term measures how much observing an expert assignment reveals about the training example that produced it. When routing distributions differ strongly across examples, this term is large. Penalizing it therefore discourages input-dependent routing preferences.

This matters for expert specialization. The prediction loss can favor different experts for different examples, but the mutual-information penalty pulls their routing distributions toward the same aggregate. For instance, under a uniform prior, assigning every example an approximately uniform routing distribution makes the mutual-information term small, even though those distributions express little example-specific preference. The exact ELBO must balance this pressure against the prediction loss.

Therefore, the objective in Proposition 3.2 retains the aggregated prior-matching term but does not explicitly penalize $I_\phi(T; Z)$. Different examples can therefore develop different, potentially sharp expert preferences while their average routing distribution remains close to the prior. The prediction loss still determines which assignments are useful for each example. Additionally, we replaces the ELBO's fixed coefficient K on aggregate prior matching with a tunable coefficient α:

$$\mathcal{L}_{\mathrm{train}} = \mathcal{L}_{\mathrm{LM}} + \alpha D_{\mathrm{KL}}\big(\bar{q}_\phi(z)\|p(z)\big).$$

Finally, we specialize the prior to the uniform distribution used in Section 3.2.

Lemma B.7 (Uniform-prior matching)**.** *Let $p(z = i) = 1/N$ for $i = 1, \ldots, N$, and define*

$$\bar{\pi}_{\phi,i} := \frac{1}{M}\sum_{m=1}^{M} \pi_{\phi,i}(x_m) = \bar{q}_\phi(z = i).$$

Then

$$D_{\mathrm{KL}}\big(\bar{q}_\phi(z)\|p(z)\big) = \sum_{i=1}^{N} \bar{\pi}_{\phi,i}\log\frac{\bar{\pi}_{\phi,i}}{1/N}.$$

This divergence is minimized exactly when $\bar{\pi}_{\phi,i} = 1/N$ for every i.

The lemma concerns the average routing probabilities. Since an expert's expected sampling frequency equals its routing probability, balancing these averages balances expected expert assignments across the batch.

Proof. Substituting the uniform prior and $\bar{q}_\phi(z = i) = \bar{\pi}_{\phi,i}$ into the definition of KL divergence gives

$$D_{\mathrm{KL}}\big(\bar{q}_\phi(z)\|p(z)\big) = \sum_{i=1}^{N} \bar{q}_\phi(z = i) \log \frac{\bar{q}_\phi(z = i)}{p(z = i)}$$
$$= \sum_{i=1}^{N} \bar{\pi}_{\phi,i} \log \frac{\bar{\pi}_{\phi,i}}{1/N}.$$

This proves Eq. 20. KL divergence is nonnegative, with equality exactly when its two distributions agree. For the uniform prior, equality holds precisely when $\bar{\pi}_{\phi,i} = 1/N$ for every expert. □

B.4 Straight-Through Gumbel-Softmax

A key challenge in our sampling-based router is that sampling from a categorical distribution is not differentiable. To enable gradient-based optimization, we use the straight-through Gumbel-Softmax estimator Jang et al. (2017). Given sampling distribution $\pi_\phi(x) = (\pi_{\phi,1}(x), \ldots, \pi_{\phi,N}(x))^\top$, we sample independent Gumbel noise variables $g_i \sim \mathrm{Gumbel}(0, 1)$ and compute the relaxed sample

$$\tilde{z}_i = \frac{\exp((\log \pi_{\phi,i}(x) + g_i)/\tau)}{\sum_{j=1}^{N} \exp((\log \pi_{\phi,j}(x) + g_j)/\tau)}, \qquad i = 1, \ldots, N, \tag{26}$$

where $\tau > 0$ is the temperature. During the forward pass, we discretize $\tilde{z}$ into a one-hot vector using an $\arg\max$ operation, so that only one expert is selected for each sample. During the backward pass, we use the continuous relaxation $\tilde{z}$ to compute gradients. This straight-through estimator allows the router to perform discrete expert sampling while remaining trainable by backpropagation.

Temperature annealing. The temperature τ controls the sharpness of the relaxed categorical sample. A larger τ produces smoother samples and encourages exploration among experts, while a smaller τ makes the relaxed sample closer to a one-hot categorical draw. In practice, we anneal τ during training. Using a larger temperature in the early stage stabilizes optimization and prevents premature router collapse, whereas gradually reducing τ encourages more discrete expert selection and reduces the mismatch between training and inference. Experimental choice of τ is available in Appendix G.5.

C Perturbing Types

To systematically evaluate the robustness of MoE routing, we design three complementary perturbation settings that operate at different levels of the model pipeline. Following prior work, the first setting directly corrupts the input data by replacing tokens with synonyms which largely preserving the contextual meaning Wang et al. (2023); Singh et al. (2025). The second setting adds Gaussian noise to the input embeddings to introduce representation-level perturbations Zoph et al. (2022). The third setting injects random noise into the router parameters to simulate instability caused by noisy or out-of-distribution data Agarwal et al. (2022); Wu et al. (2022). Together, these settings allow us to examine whether routing decisions remain stable under different types of perturbations.

Input Data corruption. Our goal is to examine whether minor token-level noise can lead to significant fluctuations in router selection, even when the contextual meaning of the input does not change substantially. To validate this, we first sample $5M$ tokens from the model's pretraining dataset as the clean reference set, and then construct its corrupted counterpart by substituting selected tokens with synonym ones. This allows us to compare routing decisions before and after corruption and compute the routing change rate. During the experiment, we gradually increase the number of exchanged synonyms to approximately $1,000$ tokens, which are sampled uniformly from the validation context. This setting introduces direct token-level corruption without causing a significant semantic shift or distribution shift, allowing us to study router behavior under mild input noise.

Input embedding perturbation. Following prior work Zoph et al. (2022), we add Gaussian noise to the input embeddings and vary the noise variance within $[0, 1]$. This setting directly tests whether

the model can maintain stable routing and prediction performance when the input representation is slightly corrupted.

Router parameter perturbation. This setting evaluates the stability of the routing mechanism under distributional shifts. Prior work Agarwal et al. (2022) suggests that out-of-distribution (OOD) data can increase gradient variance, potentially pushing model updates toward suboptimal directions and degrading performance. To simulate this effect, we vary random noise with variance in $[0, 1]$ and inject the noise into the router parameters, thereby perturbing the router's update direction. By directly disturbing the router, we examine whether the learned routing strategy remains robust under OOD-induced routing instability.

D Evaluation Metrics

Expert Usage Entropy. We measure the diversity of expert utilization using the entropy of the empirical expert assignment distribution. Let n_i denote the number of tokens assigned to expert i within a batch or evaluation window, and let $p_i = n_i / \sum_{j=1}^{N} n_j$ be the corresponding usage frequency, where N is the number of experts. The expert usage entropy is defined as

$$H_{\text{usage}} = -\sum_{i=1}^{N} p_i \log p_i. \tag{27}$$

A higher entropy indicates that tokens are distributed more evenly across experts, suggesting better load balancing and less expert collapse. In contrast, a lower entropy means that routing decisions are concentrated on a small subset of experts, which may indicate under-utilization of model capacity or early specialization collapse. For comparison across models with different numbers of experts, we also report the normalized entropy $H_{\text{usage}} / \log N$, which lies in $[0, 1]$.

Top-1 Expert Fraction. We further quantify routing concentration using the top-1 expert fraction, defined as the fraction of tokens assigned to the most frequently selected expert. Using the same empirical assignment distribution p_i, this metric is given by

$$F_{\text{top1}} = \max_{i \in \{1,\dots,N\}} p_i. \tag{28}$$

A larger Top-1 expert fraction indicates that a dominant expert handles a large portion of tokens, reflecting imbalanced routing and possible expert collapse. A smaller value suggests that the router distributes tokens more evenly across experts. In the ideal balanced case, F_{top1} approaches $1/N$, while values close to 1 indicate that almost all tokens are routed to a single expert.

E Baselines

GShard. GShard Lepikhin et al. (2020) combines sparse expert routing with automatic sharding to support large-scale MoE training. Its router computes routing probabilities from token representations and selects a small subset of experts to process each token. The original formulation uses Top-2 routing, with expert capacity constraints limiting the number of tokens processed by each expert. An auxiliary load-balancing loss combines expert load with average routing probabilities to encourage more balanced expert utilization. GShard provides the conventional sparse routing baseline in our comparison.

Switch Transformer. Switch Transformer Fedus et al. (2022) simplifies sparse routing by selecting a single expert for each token in its original formulation, reducing routing and communication costs. To improve numerical stability, it performs local router computations in float32 while retaining lower precision elsewhere in the model. This selective precision strategy stabilizes routing probability computation without requiring the entire model to use higher precision. It also employs an auxiliary load-balancing loss to encourage balanced expert utilization during training.

StableMoE. StableMoE Dai et al. (2022) addresses routing fluctuations through a two-stage training strategy. In the first stage, it learns a balanced routing strategy while distilling token-to-expert assignments into a lightweight router that is separate from the backbone model. In the second stage, it freezes the distilled router and uses it to determine expert selection throughout the remaining training. This separation prevents updates to the backbone representations from changing expert selection, allowing experts to receive consistent token-to-expert assignments during the second stage.

DeepSeekMoE. DeepSeekMoE Dai et al. (2024) promotes expert specialization through fine-grained expert segmentation and shared expert isolation. It divides each expert into smaller experts and activates more of them per token, increasing the flexibility of expert combinations while maintaining the expert computation budget. It also designates a subset of experts as shared experts that process every token, while the remaining experts are selected through sparse routing. The shared experts capture common knowledge across tokens, reducing the need for routed experts to learn redundant information and allowing them to focus on more specialized knowledge.

Auxiliary-FreeMoE. Auxiliary-FreeMoE Wang et al. (2024) balances expert load by adjusting expert selection without an auxiliary load-balancing loss. It adds an expert-specific bias to each routing score before Top-k selection and updates these biases according to recent expert load. Biases are increased for underutilized experts and decreased for overloaded experts, encouraging more balanced token-to-expert assignments. These biases affect expert selection only; the selected expert outputs are weighted using the original routing scores. This mechanism controls expert utilization without introducing an additional load-balancing term into the training objective.

F Implementation Details

All models are pretrained from scratch on 30B FineWeb-Edu tokens using the GPT-2 BPE tokenizer (50,257 tokens) and a context length of 2,048. We use 24-layer, decoder-only pre-norm Transformers with a hidden size of 1,024, 16 attention heads, RMSNorm, and tied input and output embeddings. Each feed-forward block is replaced by an MoE layer whose experts have an intermediate size of 4,096. Our 8-, 16-, and 32-expert configurations contain approximately 1.8B, 3.4B, and 6.6B parameters, respectively. We consider $K \in \{2, 4, 8\}$: Top-K methods activate the K highest-scoring experts, whereas our method draws K samples from the learned categorical routing distribution.

Our method uses stochastic expert sampling during both training and testing; we do not replace sampling with deterministic Top-K routing at inference. Each test forward pass therefore draws new expert assignments, with Gumbel noise sampled independently for every token, layer, sample, and forward pass. During training, we use a straight-through relaxation with discrete assignments in the forward pass and relaxed routing probabilities in the backward pass.

Training is conducted on four NVIDIA H100 GPUs using distributed data parallelism and `bfloat16` precision. We use 8 sequences per GPU and 8 gradient-accumulation steps, giving a global batch of 256 sequences (524,288 tokens) per update. Gradients are clipped to a norm of 1.0. Two-dimensional weight matrices are optimized with Muon using a learning rate of 2×10^{-2} and momentum 0.95; embeddings, normalization parameters, and other one-dimensional parameters use AdamW with a learning rate of 3×10^{-4}, $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$, and weight decay 0.1. We apply a 2% linear warmup followed by cosine decay to 10% of the peak learning rate.

The training objective combines next-token cross-entropy with the load-balancing objective associated with each routing method. We use the same load-balancing coefficient, $\alpha = 0.2$, for all methods. For every baseline, we use the authors' original released code and perform a grid search over its method-specific hyperparameters, selecting the configuration with the best validation performance. The architecture, data, token budget, context length, K, hardware, and shared optimization settings are held fixed across methods.

G Additional Experiments

G.1 Training Dynamics and Convergence Comparison

In Figure 6, we compare the training dynamics of different MoE routing methods in terms of NLL loss and perplexity. Overall, all methods exhibit a rapid decrease during the early training stage and gradually converge as training proceeds. Among them, our method consistently achieves the lowest NLL loss and perplexity throughout most of training, indicating more stable and efficient training. GShard and Switch Transformer show competitive performance but remain slightly worse than ours in the later stage. In contrast, Shared Expert(DeepseekMoE) and StableMoE converge to higher loss and perplexity values, suggesting less favorable training efficiency under the same setting. The zoomed-in regions further highlight the performance gap near convergence, where Gumbel-KLDiv maintains a clear advantage over other baselines. These results demonstrate that the proposed routing strategy not only improves final model quality but also provides an effective training paradigm.

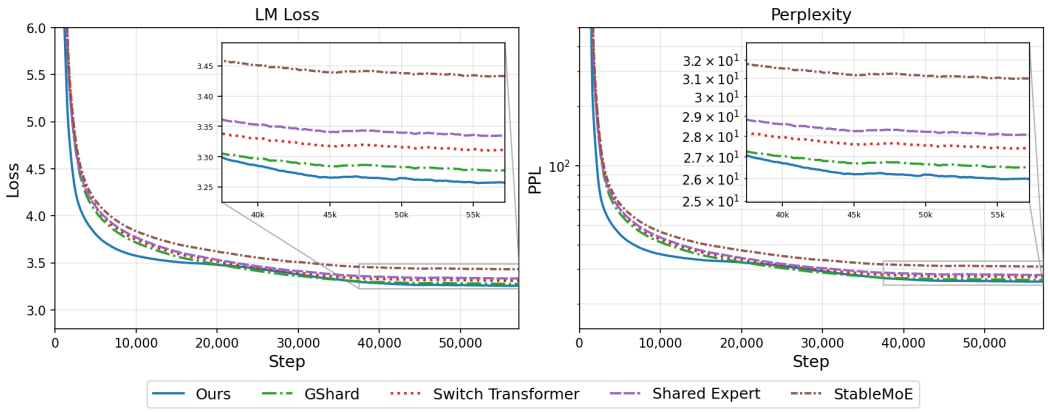


Figure 6: **Training dynamics across different MoE routing methods.** We compare NLL Loss and Perplexity with various baselines throughout training. Lower values indicate better performance. Our method consistently achieves lower NLL loss and perplexity, especially near convergence.

Table 4: **Downstream performance of MoE models pretrained on Dolma.** We report accuracy across eight downstream benchmarks for different MoE routing methods with 8 experts and 1.8B parameters. Bold values denote the best performance, underlined values denote the second-best performance, and arrows show the difference relative to GShard.

Setting	Method	ARC-E	PIQA	SIQA	WINO	HELLA	MNLI	QNLI	SST-2	Avg.
8 Experts *1.8B params*	GShard	32.5	53.0	38.0	47.8	28.5	33.8	48.7	52.0	41.8
	Switch Transformer	31.0↓1.5	48.8↓4.2	37.3↓0.7	**49.7**↑1.9	27.0↓1.5	34.0↑0.2	**52.7**↑4.0	50.6↓1.4	41.4↓0.4
	StableMoE	33.1↑0.6	54.2↑1.2	38.5↑0.5	48.0↑0.2	29.0↑0.5	34.1↑0.3	49.2↑0.5	52.7↑0.7	42.4↑0.6
	DeepSeekMoE	<u>34.0</u>↑1.5	<u>55.1</u>↑2.1	<u>39.0</u>↑1.0	48.5↑0.7	<u>29.7</u>↑1.2	<u>34.5</u>↑0.7	49.8↑1.1	<u>53.4</u>↑1.4	<u>43.0</u>↑1.2
	Ours	**39.0**↑6.5	**60.8**↑7.8	**41.4**↑3.4	<u>48.9</u>↑1.1	**32.8**↑4.3	**35.0**↑1.2	<u>50.5</u>↑1.8	**58.6**↑6.6	**45.9**↑4.1

G.2 Performance across Pretraining Corpora

To evaluate whether the benefits of our method extend across pretraining corpora, we conduct additional experiments on Dolma Soldaini et al. (2024). We use the 8-expert, 1.8B-parameter configuration and retain the same training token budget and other training settings as in the FineWeb-Edu Penedo et al. (2024) experiments. All models are evaluated on the same eight downstream benchmarks. Compared with the educational content selected for FineWeb-Edu, Dolma covers a broader mixture of sources, providing an additional setting for examining the effectiveness of our method under a different pretraining distribution.

Table 4 shows that our method achieves the highest average accuracy of 45.9, outperforming DeepSeekMoE, the strongest baseline on average, by 2.9 percentage points and GShard by 4.1 points. Our method ranks first on six of the eight benchmarks and improves over GShard on all eight. The improvements span commonsense reasoning, natural language inference, and sentiment classification, with gains of 6.5 points on ARC-E, 7.8 points on PIQA, 4.3 points on HellaSwag, and 6.6 points on SST-2 relative to GShard. These results extend the performance advantage observed in the main experiments to a different pretraining corpus.

Dolma also provides a more challenging pretraining setting for these downstream benchmarks under our training budget. Compared with the corresponding FineWeb-Edu results in Table 1, the average accuracy of every baseline evaluated on both corpora decreases. For example, GShard drops from 45.1 to 41.8, while DeepSeekMoE drops from 45.3 to 43.0. In this setting, the average improvement of our method over GShard increases from 2.6 points on FineWeb-Edu to 4.1 points on Dolma. The advantage over the strongest baseline also widens, from 2.4 to 2.9 points. The larger gains are particularly evident on ARC-E, PIQA, and HellaSwag, where the improvements over GShard increase from 3.0, 2.2, and 1.2 points to 6.5, 7.8, and 4.3 points, respectively.

Our method also exhibits the smallest decrease in average accuracy among the methods evaluated on both corpora: 1.8 points, compared with decreases ranging from 2.3 to 3.8 points for the baselines. Thus, the advantage persists as the pretraining corpus changes and becomes more pronounced in a setting where the baselines achieve lower downstream accuracy. Together, these results support the

Table 5: **Large-scale pretraining results on FineWeb-Edu.** We report total and activated parameter counts, estimated training FLOPs, and language modeling performance for 6.6B MoE models with 32 experts. Bold values denote the lowest perplexity and NLL loss.

Method	Parameters (B)		Training cost	Language modeling		
	Total	Active	FLOPs ↓	Train PPL ↓	Test PPL ↓	NLL Loss ↓
GShard	6.6	1.70	1.09×10^{21}	18.74	18.97	2.94
Switch Transformer	6.6	1.09	7.54×10^{20}	19.71	19.88	2.98
StableMoE	6.6	1.09	7.54×10^{20}	20.86	20.41	3.02
DeepSeekMoE	6.6	1.70	1.09×10^{21}	19.32	19.54	2.97
Auxilary-FreeMoE	6.6	1.70	1.09×10^{21}	20.74	20.45	3.04
Ours	6.6	1.34	8.88×10^{20}	**17.60**	**17.89**	**2.87**

effectiveness of our method across pretraining corpora and show that its benefits remain substantial under the more challenging Dolma setting.

G.3 Performance at Larger Model Scales

We complement the upstream evaluation at the 1.8B scale reported in the main text with results for the 6.6B, 32-expert configuration. Table 5 reports language modeling performance and estimated training costs, allowing us to examine whether the advantages observed at the smaller scale persist at the larger scale.

As shown in Table 5, `StaVER` achieves the lowest training perplexity, test perplexity, and NLL loss among all compared methods. Relative to GShard, the strongest baseline in language modeling performance, `StaVER` reduces training perplexity from 18.74 to 17.60 and test perplexity from 18.97 to 17.89. The relative reduction in test perplexity increases from 3.0% at the 1.8B scale to 5.7% at the 6.6B scale. This larger improvement shows that `StaVER` retains its performance advantage as model size and the number of experts increase, providing evidence of its scalability within the evaluated range.

The estimated training costs also preserve the favorable performance–computation trade-off observed in the main experiments. Using the relative FLOPs from the 1.8B setting to estimate costs at the larger scale, `StaVER` requires approximately 18.5% fewer training FLOPs than GShard and DeepSeekMoE. Switch Transformer and StableMoE have lower estimated costs with their Top-1 routing designs, while `StaVER` achieves lower test perplexity by 1.99 and 2.52 points, respectively. Together, these results show that `StaVER` remains effective at the 6.6B scale, combining stronger language modeling performance with a favorable estimated training cost relative to the conventional Top-k baselines.

G.4 Training Efficiency of Sampling-Based Routing

Our proposed sampling-based routing improves training efficiency by activating fewer distinct experts per token. Unlike conventional Top-K routing, which selects K distinct experts, repeated samples in `StaVER` can select the same expert. These selections can share a single expert evaluation, reducing expert computation. Figures 7 and 8 illustrate this behavior across layers and its relationship with downstream performance for $K \in \{2, 3, 4\}$.

Fewer activated experts across layers. Figure 7 shows that `StaVER` frequently activates fewer than K distinct experts, with the activation distribution varying across tokens and layers. Earlier layers generally have a larger proportion of tokens activating multiple experts, whereas single-expert activation becomes more prevalent in deeper layers. This pattern persists as K increases: additional samples activate more distinct experts, but repeated selections remain common. Consequently, a larger sampling budget does not require a proportional increase in expert computation.

The average numbers of unique activated experts are 1.41, 1.69, and 1.91 for $K = 2$, 3, and 4, respectively. Compared with activating K distinct experts per token, these values represent reductions of approximately 29.5%, 43.7%, and 52.3% in unique expert activations. When repeated selections reuse the same expert output, these reductions directly lower the computation performed by the expert networks. In particular, increasing the sampling budget from two to four raises the average number of activated experts from 1.41 to only 1.91. `StaVER` therefore supports additional routing samples

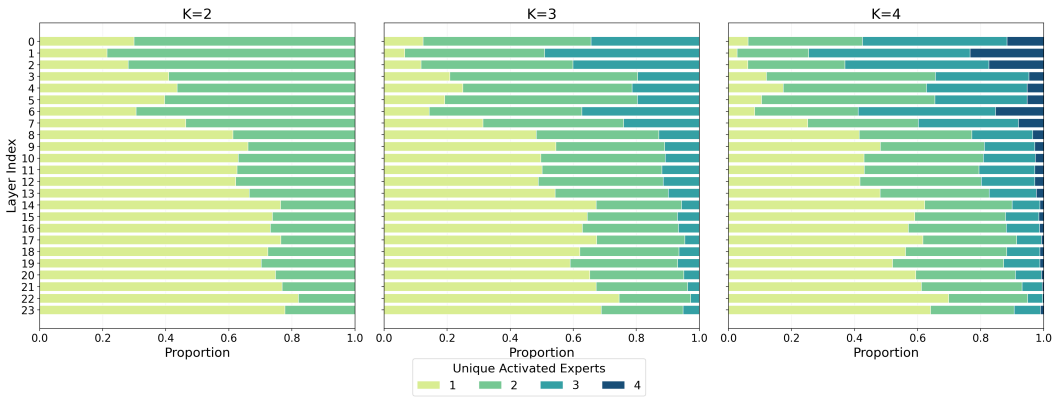


Figure 7: **Layer-wise expert activation under stochastic routing.** The bars show the proportion of tokens activating different numbers of unique experts in each layer for $K \in \{2, 3, 4\}$ samples. Increasing K activates more distinct experts, although repeated samples frequently select the same expert.

Table 6: **Ablation on Load Balancing coefficient α.** This table reports perplexity under different α values. `StaVER` performs best at $\alpha = 0.1$ and remains stable over a broad range.

α	0	0.001	0.01	0.1	1	10
PPL↓	96.95	45.88	25.91	25.74	25.85	27.42

while keeping the number of distinct expert evaluations small. **Performance-computation trade-off.** Figure 8 shows that the accuracy benefit of activating more experts varies across tasks. Increasing K from 2 to 3 improves zero-shot accuracy on all four evaluated benchmarks. Increasing K further to 4 improves PIQA and QNLI, while MNLI and SST-2 achieve their highest observed accuracy at $K = 3$. On the latter two benchmarks, $K = 3$ therefore provides both higher accuracy and lower expert computation than $K = 4$. These results show that an intermediate sampling budget can offer a favorable performance–computation trade-off. Together, the two figures explain how `StaVER` reduces expert computation through repeated selections and how the sampling budget controls the balance between expert activation and downstream accuracy.

G.5 Hyperparameter Sensitivity

We evaluate the sensitivity of StaVER to two hyperparameters: the load-balancing coefficient α and the Gumbel-Softmax temperature τ. The coefficient α controls the strength of the prior matching term on the aggregated routing distribution, while τ controls the smoothness of the continuous relaxation used for gradient estimation in straight-through Gumbel-Softmax. Tables 6 and 7 report perplexity under different coefficient values and temperature settings.

`StaVER` performs well across a broad range of α. Table 6 shows that removing the load-balancing regularizer ($\alpha = 0$) results in a high perplexity of 96.95. Introducing a small coefficient of $\alpha = 0.001$ reduces perplexity to 45.88, and increasing it to 0.01 further improves perplexity to 25.91. Performance remains close to the best result for $\alpha \in \{0.01, 0.1, 1\}$, with perplexities of 25.91, 25.74, and 25.85, respectively. These settings span two orders of magnitude, yet differ by at most 0.17 perplexity. Increasing α to 10 raises perplexity to 27.42, suggesting that excessively strong regularization can compromise language modeling performance. Overall, these results support the importance of the load-balancing regularizer while showing that StaVER does not require a narrowly tuned coefficient within the effective range. The best observed result is obtained with $\alpha = 0.1$.

Temperature annealing consistently improves perplexity. Table 7 compares annealed and fixed Gumbel-Softmax temperatures. For each temperature evaluated under both settings, $\tau \in$

Table 7: **Ablation on Gumbel Temperature τ.** In this table, we compares annealed and fixed temperature settings for Gumbel-Softmax routing. Annealing τ consistently achieves lower perplexity than fixed temperatures.

τ Setting	Anneal To				Fixed			
	0.7	0.5	0.3	0.1	1.0	0.5	0.3	0.1
PPL↓	26.23	25.98	25.97	25.74	28.09	27.17	26.55	26.28

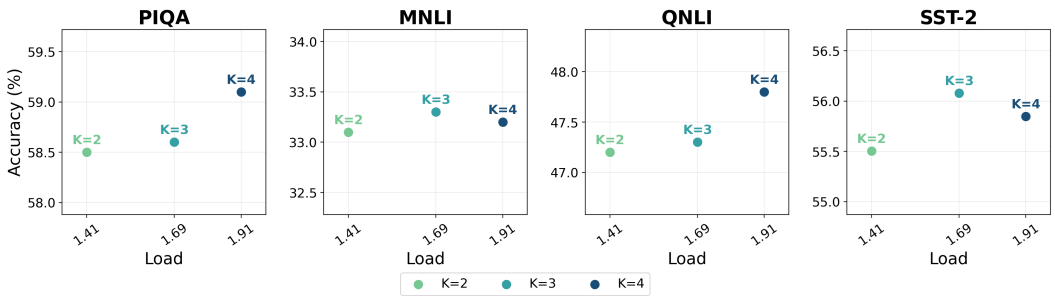


Figure 8: **Accuracy versus expert load.** Zero-shot accuracy is plotted against the average number of uniquely activated experts for $K \in \{2, 3, 4\}$. Higher load does not consistently improve performance, and the optimal K varies across benchmarks.

$\{0.5, 0.3, 0.1\}$, annealing achieves lower perplexity than using that temperature throughout training. Specifically, annealing reduces perplexity from 27.17 to 25.98 at $\tau = 0.5$, from 26.55 to 25.97 at $\tau = 0.3$, and from 26.28 to 25.74 at $\tau = 0.1$. Among the evaluated annealing settings, lowering the final temperature from 0.7 to 0.1 improves perplexity from 26.23 to 25.74.

These results favor gradually sharpening the continuous relaxation during training. A higher temperature produces softer relaxed samples, whereas a lower temperature brings them closer to the discrete selections used in the forward pass. Annealing allows this relaxation to change over training, and its advantage over matched fixed temperatures indicates that the temperature schedule matters beyond the final temperature alone. Within the evaluated settings, annealing to $\tau = 0.1$ achieves the lowest perplexity.