

ROUTE EXPERTS BY SEQUENCE, NOT BY TOKEN

Tiansheng Wen^{*,1,2}, Yifei Wang^{*,3}, Aosong Feng⁴, Long Ma², Xinyang Liu⁵,
Yifan Wang¹, Lixuan Guo², Bo Chen^{2,†}, Stefanie Jegelka^{6,3}, Chenyu You^{1,†}

¹Stony Brook University, ²Xidian University, ³MIT

⁴Yale University, ⁵The University of Texas at Austin, ⁶TUM

ABSTRACT

Mixture-of-Experts (MoE) architectures scale large language models (LLMs) by activating only a subset of experts per token, but the standard *TopK* routing assigns the same fixed number of experts to all tokens, ignoring their varying complexity. Prior adaptive routing methods introduce additional modules and hyperparameters, often requiring costly retraining from scratch. We propose **Sequence-level TopK (SeqTopK)**, a minimal modification that shifts the expert budget from the token level to the sequence level. By selecting the top $T \cdot K$ experts across all T tokens, SeqTopK enables end-to-end learned dynamic allocation – assigning more experts to difficult tokens and fewer to easy ones – while preserving the same overall budget. SeqTopK requires only a few lines of code, adds less than 1% overhead, and remains fully compatible with pretrained MoE models. Experiments across math, coding, law, and writing show consistent improvements over TopK and prior parameter-free adaptive methods, with gains that become substantially larger under higher sparsity (up to 16.9%). These results highlight SeqTopK as a simple, efficient, and scalable routing strategy, particularly well-suited for the extreme sparsity regimes of next-generation LLMs. Code is available at [here](#).

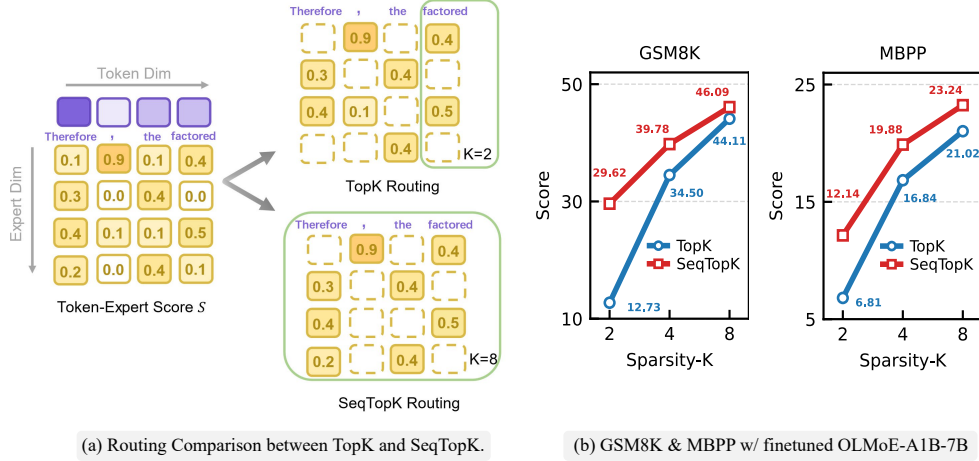


Figure 1: **Overview of our proposed method.** (a) Illustrative comparison between standard TopK Routing and SeqTopK Routing in MoE models. Under the same expert budget for a sequence ($T \cdot K = 3 \cdot 2$ in this case), SeqTopK routes experts via comparing expert scores across all tokens in a sequence, enabling dynamic and context-aware allocation of experts (e.g., more experts for hard tokens) via end-to-end training. (b) Performance of finetuned OLMoE-A1B-7B on GSM8K and MBPP datasets. SeqTopK consistently outperforms TopK under different expert budgets ($K = 2, 4, 8$), and the gain is much larger under sparser MoEs.

*Equal contribution.

†Correspondence: bchen@mail.xidian.edu.cn, chenyu.you@stonybrook.edu

1 INTRODUCTION

Mixture-of-Experts (MoE) architectures have emerged as a central paradigm for scaling large language models (LLMs), offering massive capacity without a proportional increase in computation (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022). By activating only a small subset of experts per token, MoEs distribute knowledge across specialized modules while maintaining efficiency. However, the dominant routing strategy – *TopK selection* – treats all tokens uniformly, assigning each token the same fixed number of experts (K) regardless of its complexity or information content. This rigidity overlooks the heterogeneity of language (Smith & Levy, 2013): some tokens are trivial and require little capacity (e.g., “the”), while others (e.g., legal terms) are far more informative and benefit from broader expert consultation. As a result, computation is wasted on easy tokens, while complex tokens receive insufficient expert allocation under TopK routing.

To address this limitation, a growing body of work has explored adaptive routing strategies (Huang et al., 2024; Lu et al., 2024; Wang et al., 2024c; Guo et al., 2024; Wu et al., 2025; Jin et al., 2024a). While promising, these approaches often introduce additional modules and hyperparameters, and typically require pretraining the MoE model from scratch with the new routing. As a result, they come with substantial cost, architectural complexity, and limited large-scale validation. This raises a natural question: *do we truly need entirely new routing mechanisms to enable adaptive allocation?*

In this paper, we instead revisit standard TopK routing and propose a simple but effective relaxation: shifting the budget from the *token level* to the *context level*. Instead of enforcing K experts per token, we allocate the top $T \cdot K$ experts across all T tokens in a sequence. This *Sequence-level TopK* (SeqTopK) routing enables the model, through end-to-end likelihood maximization, to assign more experts to challenging tokens and fewer to trivial ones, thereby achieving dynamic allocation without introducing new architectural components. Crucially, SeqTopK preserves the same overall budget as TopK, requires only a few lines of code to implement (Figure 1), and introduces no additional parameters or hyperparameters. It is fully compatible with existing MoE models and can be directly fine-tuned from pretrained TopK checkpoints. SeqTopK also integrates naturally with autoregressive decoding via an on-the-fly updated list of expert scores, which we term the *Expert Cache* (analogous to the KV cache in attention). Overall, SeqTopK incurs less than 1% overhead in computation and memory, while seamlessly extending the capabilities of existing MoE implementations.

Despite its simplicity, SeqTopK consistently outperforms standard TopK and prior adaptive methods. The gains are especially pronounced under higher sparsity (up to 16.89%), highlighting its promise as state-of-the-art MoE models, such as GPT-OSS (Agarwal et al., 2025) and Qwen-Next (Yang et al., 2025), continue to scale with extreme sparsity. We validate SeqTopK across diverse domains, including math, coding, law, and writing, and demonstrate its data-driven adaptive behavior that allocates more experts to harder tokens.

Our contributions are summarized as follows:

- We propose **SeqTopK**, a routing mechanism that maintains the same compute budget as the TopK routing while enabling context-level dynamic expert allocation.
- For efficient autoregressive inference, we introduce **online SeqTopK with Expert Cache**, which maintains an updated list of expert scores to support online routing that closely mirrors training-time SeqTopK.
- We provide extensive validation across multiple downstream tasks, showing that SeqTopK consistently outperforms the standard TopK and delivers larger gains as sparsity increases, along with qualitative evidence of its context-aware dynamic routing behavior.

2 BACKGROUND

Mixture of Experts (MoEs). We first introduce a generic MoE architecture commonly used in Transformer-based language models (LMs) (Vaswani et al., 2017). A standard decoder-only transformer (Radford et al., 2019) comprises L layers, where each block can be represented as follows:

$$\mathbf{x}_t^l = \text{Self-Attn}(\mathbf{h}_{1:t}^{l-1}) + \mathbf{h}_{1:t}^{l-1}, \quad (1)$$

$$\mathbf{h}_t^l = \text{FFN}(\mathbf{x}_t^l) + \mathbf{x}_t^l, \quad t \in [T], \quad (2)$$

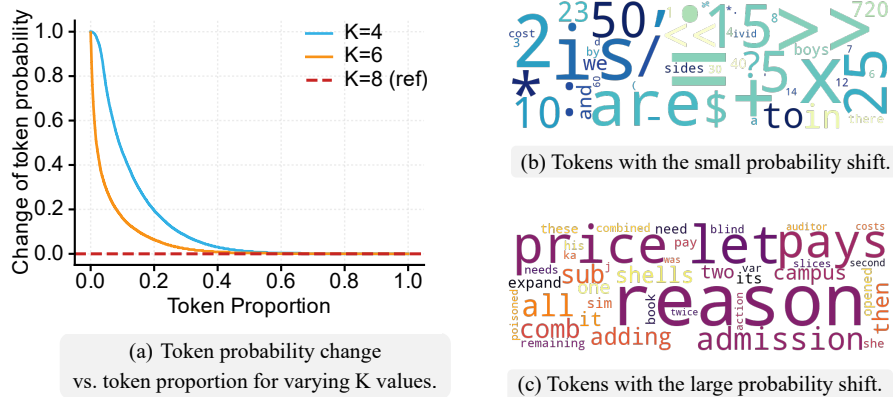


Figure 2: **Change of token probability $P(x_t|x_{<t})$ under varying active experts (K). (a) Token probability change vs. token proportion.** Given the same prefix, we sample over 10k tokens across different K values, computing probability differences relative to $K = 8$ (the original OLMoE setting). About 60% of tokens show little probability change when K is reduced from 8 to 4 or 6, while 10% change dramatically, indicating that different tokens can require quite different numbers of activated experts to predict. **(b) & (c) Word clouds of the top 50 tokens with small (< 0.01) and large (> 0.5) token probability shifts.** Tokens with the larger probability differences are content words that influence semantic direction or topic shifts, whereas tokens with the smaller probability differences are numbers or function words that maintain structure.

where T denotes the length of the sequence $x_{1:T}$, Self-Attn($\cdot$) and FFN($\cdot$) are self-attention and the Feed-Forward Network. Here, $x_t^l \in \mathbb{R}^D$ is the hidden state of the t -th token and $h_t^l \in \mathbb{R}^D$ the output of the l -th transformer block. From now on, we drop the layer index l for better clarity.

A typical approach for constructing MoE language models involves substituting FFNs in transformers with MoE layers. These MoE layers partition the FFN into N smaller FFNs, $E_1, \dots, E_N$, referred to as *experts*, where N denotes the total number of experts. Subsequently, each token x_t is assigned to a few experts as determined by a router module $R(\cdot)$. The output is computed as:

$$\mathbf{h}_t = \sum_{i=1}^N R_i(\mathbf{x}_t) E_i(\mathbf{x}_t) + \mathbf{x}_t. \quad (3)$$

Standard TopK Routing in MoEs. TopK routing (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022) is a widely used, de facto strategy for routing experts in modern LLMs. For each input token x_t , the TopK router identifies the K most relevant experts and sets the relevance scores of all other experts to zero, i.e.,

$$\text{(scores)} \quad s_t = \sigma(\mathbf{W}x_t), \mathbf{W} \in \mathbb{R}^{N \times D}; \quad (4)$$

$$(\text{TopK routing}) \quad R_i(\mathbf{x}_t) = \begin{cases} 1, & \text{if } i \in \arg \text{TopK}_{i \in [N]} \mathbf{s}_{t,i} \\ 0, & \text{otherwise,} \end{cases} \quad (5)$$

where $\sigma(\cdot)$ denotes the *softmax* function that normalizes the scores over the N experts, and $\arg \text{TopK}(\cdot)$ returns the index set of the k largest elements along the chosen dimension¹.

Due to the limited space, we defer a detailed discussion on other related works to Appendix A.

3 METHOD

3.1 SEQTOPK: FROM TOKEN-LEVEL TO SEQUENCE-LEVEL ROUTING

¹Since the primary distinction between TopK and SeqTopK lies in expert allocation, we omit the final routing-weight selection step in Eq. 5 (*i.e.*, selecting weights from s_i based on the chosen indices) for simplicity.

Token Heterogeneity in Standard TopK.

While standard TopK routing adaptively selects experts for each token, it enforces a *fixed budget of K experts per token*, regardless of predictive difficulty. This assumption ignores the heterogeneity among tokens in a sequence. As shown in Figure 2, when reducing the routing budget from $K = 8$ to $K = 4$, over 60% of tokens exhibit *little or no degradation* in their predicted likelihood $P(x_t | x_{<t})$, whereas approximately 15% of tokens suffer a *drop of more than 0.5*. This discrepancy highlights substantial variation in token difficulty and the computational resources required. This suggests that standard TopK over-serves easy tokens and under-serves difficult ones, leading to inefficiency and degraded accuracy in sparse regimes.

Limitation of Token-level Routing. Several recent works attempt adaptive routing by replacing TopK with learnable thresholding functions (e.g., ReLU gating in ReMoE or identity experts (Jin et al., 2024a; Team et al., 2025b)), thereby allowing different tokens to activate different numbers of experts. However, such methods show limited performance gains because they remain fundamentally at the local *token level*: routing decisions depend only on each token’s own scores, without considering the surrounding context. In contrast, the power of self-attention in Transformers arises precisely from *contextualization*: predictions are informed by comparing a token with its neighbors. MoE layers that treat tokens independently lack this context-aware adaptivity.

Sequence-level Routing with SeqTopK. Motivated by these observations, we propose **SeqTopK**, as shown in Figure 3, a context-aware routing strategy that performs expert selection at the sequence level. Let

$$(\text{SeqTopK scores}) \quad \mathbf{S} = [s_1, \dots, s_T]^\top \in \mathbb{R}^{T \times N}, s_t = \sigma(\mathbf{W}\mathbf{x}_t) \in \mathbb{R}^N, \quad (6)$$

where $\mathbf{S}$ collects the expert scores for all tokens in a sequence of length T with N experts. SeqTopK selects the top K_{seq} entries across all $T \cdot N$ elements in the score matrix $\mathbf{S}$:

$$(\text{SeqTopK routing}) \quad R_i(\mathbf{x}_t) = \begin{cases} 1, & \text{if } (t, i) \in \arg \text{TopK}_{t \in [T], i \in [N]} \mathbf{S}_{t,i}, \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

Here $K_{\text{seq}} = T \cdot K$, where K is the original budget of TopK and T is the number of tokens in a sequence, which ensures the total budget of SeqTopK exactly matches that of TopK. Thus, with equal overall compute, tokens can now receive different numbers of experts. Empirically, SeqTopK allocates more experts to difficult tokens and fewer to trivial ones, improving performance without increasing cost. Importantly, the allocation depends not only on a token’s absolute score but also on its *relative importance within the sequence*, thereby enabling context-aware routing in MoEs.

Balanced Routing with Token-level Bounds. To prevent degenerate allocations (e.g., assigning no experts to some tokens), we enforce per-token bounds for the number of experts. Specifically, each token is guaranteed at least one expert, and capped at $K_{\text{tok}} + 2$ experts, where K_{tok} is the original token-level sparsity. This ensures all tokens contribute to MoE training while avoiding domination by a small subset of tokens. According to ablation studies in Section C, we find that while SeqTopK’s performance exhibits slight fluctuations depending on the specific upper bound chosen, the bounded variant consistently yields robust improvements across most of the evaluated settings.

Compatibility with TopK MoE models. While many sophisticated designs for context-aware routing could be envisioned, we deliberately designed SeqTopK to remain closely aligned with standard TopK. At the implementation level, it essentially alters only the indexing dimensions of the TopK operator, *requiring just a few lines of code to modify* (Figure 1). This simplicity yields several practical advantages. First, no additional hyperparameters or auxiliary modules are introduced. Second,

```
# scores: [B,T,N], routed_score: [B,T,N]
def TopK_Router(scores, k):
    val, idx = torch.topk(scores, k, dim=-1)
    routed_score = torch.zeros_like(scores)
    routed_score.scatter_(-1, idx, val)
    return routed_score

-----
# scores: [B,T,N], routed_score: [B,T,N]
def Seq_TopK_Router(scores, k):
    flat = scores.flatten(start_dim=-2)
    val, idx = torch.topk(flat, k=T*k, dim=-1)
    routed_score = torch.zeros_like(flat)
    routed_score.scatter_(-1, idx, val)
    return routed_score.view(*scores)

(b) SeqTopK Routing (ours)
```

Figure 3: **(a) & (b)** Simple PyTorch implementations of TopK and SeqTopK routing. With a minimal modification of TopK routing (highlighted), SeqTopK enables dynamic and context-aware expert allocation by comparing expert scores across all tokens in a sequence.

pretrained TopK MoE checkpoints can be readily adapted to SeqTopK with only a few hundred fine-tuning steps (Section 4.1). Third, the computational and memory overhead introduced is negligible, typically less than 1% in our profiling. Taken together, these properties make SeqTopK a practical, lightweight drop-in replacement for standard TopK routing, enabling MoEs to perform context-aware adaptive routing and deliver improved performance across diverse downstream tasks.

3.2 ONLINE SEQTOPK: EFFICIENT DECODING WITH CACHED EXPERT SCORES

While SeqTopK seamlessly integrates into the training stage as a drop-in replacement for TopK, applying it directly at inference is problematic. Large language models generate tokens autoregressively from $t = 1$ to T , yet vanilla SeqTopK requires access to the routing scores of all tokens $\mathbf{x}_{1:T}$. In other words, vanilla SeqTopK is inherently *non-causal*², and thus makes it incompatible with autoregressive decoding where $\mathbf{x}_{m+1:T}$ is unavailable at step m .

Online SeqTopK. To address this limitation, we propose **Online SeqTopK**, which restricts expert selection at step m to the m tokens generated so far:

$$\text{(online SeqTopK scores)} \quad \mathbf{S}_m = [\mathbf{s}_1, \dots, \mathbf{s}_m]^\top \in \mathbb{R}^{m \times N}, \quad \mathbf{s}_t = \sigma(\mathbf{W}\mathbf{x}_t) \in \mathbb{R}^N, \quad (8)$$

$$\text{(online SeqTopK routing)} \quad R_i(\mathbf{x}_m) = \begin{cases} 1, & \text{if } (t, i) \in \arg \text{TopK}_{t \in [m], i \in [N]} \mathbf{S}_{t,i}, \\ 0, & \text{otherwise.} \end{cases} \quad (9)$$

Here we set the budget to $K_{\text{seq}} = m \cdot K$, so that at most $m \cdot K$ experts are activated across the first m tokens. When $m = T$, Online SeqTopK recovers the vanilla formulation in Eq. 7. Unlike training-time SeqTopK, however, the online variant cannot anticipate future tokens: once an expert assignment is made for earlier tokens, it is fixed and unaffected by later routing scores. Nevertheless, Online SeqTopK guarantees that for any $1 \leq m \leq T$, the cumulative number of activated experts never exceeds $m \cdot K$, and is thus always bounded by the $T \cdot K$ experts of the vanilla formulation. This property establishes a natural upper bound on computation for efficient autoregressive decoding, while in expectation yielding the same total number of activated experts as the offline variant.

Expert Cache. To avoid recomputing expert scores at each step, we introduce an **Expert Cache** that stores all previously computed scores and is updated only with the scores at the latest step:

$$\text{(Expert Cache)} \quad \mathbf{S}_m = \begin{pmatrix} \mathbf{S}_{m-1}^\top \\ \mathbf{s}_m^\top \end{pmatrix}, \quad \mathbf{s}_m = \sigma(\mathbf{W}\mathbf{x}_m) \in \mathbb{R}^N. \quad (10)$$

This design is analogous to the KV cache in self-attention: whereas the KV cache stores intermediate keys and values, the expert cache maintains routing scores for reuse. In practice, the expert cache can be co-located with the KV cache and managed with identical read/write mechanisms. Its memory footprint is also modest: while the KV cache has size $B \times T \times H$ (with hidden dimension $H \approx 4096$), the expert cache is only $B \times T \times N$ (with $N \in [8, 128]$ in typical MoE models). Consequently, Online SeqTopK introduces negligible memory and compute overhead.

3.3 COMPARISON WITH BATCHTOPK

A closely related variant of SeqTopK is BatchTopK (Bussmann et al., 2024), which selects experts across all tokens in a batch rather than within a single sequence. While effective in MoE-based diffusion image modeling, where batches are regular and homogeneous (Yuan et al., 2025; Shi et al., 2025), BatchTopK is ill-suited for language modeling. This is because text sequences in a batch vary in length and domain, making cross-sequence competition unfair and unstable. In deployment, batch sizes are flexible and unpredictable, creating mismatches between training and inference when routing depends on global batch composition. Moreover, selecting experts across independent user inputs risks information leakage, raising privacy and safety concerns in multi-tenant systems. In contrast, SeqTopK restricts competition to tokens within the same sequence, ensuring contextual consistency, robustness to batch size, and data privacy, making it better suited for adaptive routing in language modeling. We will show soon that it yields consistent gains over BatchTopK in practice.

²Although non-causal, SeqTopK does not actually leak differentiable information about future tokens, since the $\arg \text{TopK}$ operator in Eq. 7 is non-differentiable.

Table 1: **Benchmark results of models fine-tuned from the OLMoE-A1B-7B model (64 routing experts) with different routing methods.** SeqTopK outperforms baseline methods across all settings, with gains becoming much larger (up to 7.55%) as sparsity increases. Best in **bold**.

<i>OLMoE-A1B-7B</i>			GSM8k	MBPP	HumanEval	Summary	Law	Avg
Routing Methods	K	Sparsity Ratio	0-shot EM	3-shot Pass@1	0-shot Pass@1	0-shot Score	0-shot Score	- Score
Base	8	1/8	15.58	19.80	10.97	7.49	5.70	11.91
TopK	8	1/8	44.11	21.04	13.41	45.31	24.89	29.74
MRL-TopK			43.93	21.78	12.21	44.25	21.08	28.65
BatchTopK			44.80	22.63	14.02	42.89	22.89	29.45
SeqTopK			46.09	23.21	15.24	46.40	26.52	31.49
TopK	4	1/16	36.50	16.84	10.98	41.42	17.77	24.70
MRL-TopK			27.44	17.82	9.76	39.89	12.21	21.42
BatchTopK			35.93	18.62	12.82	39.79	17.82	24.99
SeqTopK			39.78	19.88	13.41	43.00	20.20	27.25
TopK	2	1/32	12.73	6.81	6.12	37.89	8.91	14.49
MRL-TopK			2.2	4.2	4.8	19.6	3.4	6.84
BatchTopK			18.49	6.60	6.71	37.18	11.89	16.17
SeqTopK			29.62	12.14	12.22	42.60	14.10	22.04

Table 2: **Benchmark results of models fine-tuned from the Qwen1.5-MoE-A2.7B model with different routing methods.** This model by default, uses 4 shared experts and then selects 4 experts out of 60 routing experts. SeqTopK consistently outperforms the others. Best in **bold**.

<i>Qwen1.5-MoE-A2.7B</i>			GSM8k	MBPP	HumanEval	Summary	Law	Avg
Methods	K	Sparsity-Ratio	0-shot EM	3-shot Pass@1	0-shot Pass@1	0-shot Score	0-shot Score	- Score
Base	4	1/8	38.69	38.84	32.31	28.29	18.20	31.27
TopK	4	1/8	55.16	35.02	36.89	39.21	42.32	41.72
MRL-TopK			54.88	35.76	36.13	38.74	40.25	41.15
BatchTopK			53.92	35.24	35.89	37.10	42.39	40.91
SeqTopK			55.87	36.41	37.20	41.31	45.29	43.19

4 BENCHMARK RESULTS AND ANALYSIS

In this section, we evaluate our proposed SeqTopK routing strategy from both pre-training and fine-tuning perspectives. Section 4.1 presents fine-tuning results for modern language MoE models under varying sparsity levels across tasks, including mathematical reasoning (Cobbe et al., 2021), code generation (Austin et al., 2021; Chen et al., 2021), summarization, and legal assistance (Wang et al., 2024b). Section 4.2 reports GPT-2-level MoE pre-training results and zero-shot performance on benchmark datasets. A detailed efficiency analysis is presented in Section 4.3, examining the additional training and inference costs of SeqTopK.

4.1 SCALING MODEL SPARSITY AT FINE-TUNING STAGE

Experiment Setup. We select MRL-TopK (Bussmann et al., 2025), BatchTopK (Bussmann et al., 2024), and standard TopK as baselines because they **enable MoE model fine-tuning while only varying in their routing mechanism** and requiring **no additional parameters**. We fine-tuned two models (Muennighoff et al., 2024; Team, 2024) and evaluated them on five downstream datasets using the framework of Feng et al. (2023). We then scaled the models to extreme sparsity (e.g., reducing the number of activated experts per token from 8 to 2) and reported performance across different sparsity levels. Zero-shot evaluations were conducted on GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021), Summary (Wang et al., 2024b), and Law (Wang et al., 2024b),

Table 3: **Zero-shot performance of 182M-sized MoE models pretrained with different routing methods**, where ReMoE uses ReLU as an adaptive routing function. SeqTopK shows best performance at each sparsity level; and comparing the two levels, SeqTopK excels $K = 4$ while others excel at $K = 8$, indicating that SeqTopK benefits more from higher MoE sparsity. Best in **bold**.

Method	Experts	K	Sparsity-Ratio	LAMBDA	RACE	ARC-E	ARC-C	Avg.
TopK	128	4	1/32	59.30	27.33	46.66	20.73	38.51
ReMoE				59.94	28.01	47.71	21.07	39.18
SeqTopK				60.82	28.70	48.07	23.41	40.25
TopK	64	8	1/8	60.15	27.61	47.19	20.06	38.75
ReMoE				61.73	28.15	49.12	19.06	39.52
SeqTopK				60.67	28.55	48.42	22.36	40.00

Table 4: **Efficiency Analysis for SeqTopK**. (a) We report the GPU hours for pre-training a 182M-size model over 30B tokens using various routing strategies. (b) We benchmark the OLMoE-A1B-7B model’s decoding efficiency for online SeqTopK and the overhead introduced by Expert-Cache, using *huggingface.gen()* for fair comparison.

(a) Pre-training Time.		(b) Inference throughput and peak memory use.		
Method	Wall-clock Time (GPU hours)	Method	Tokens/s	Peak Memory [GB]
TopK	244.2	TopK	141.23	18.21
ReMoE	245.0	SeqTopK	139.41 (-1%)	18.35 (+1%)
SeqTopK	246.4 (+1%)			

while 3-shot evaluations were performed on MBPP (Austin et al., 2021). Note that BatchTopK is sensitive to the evaluation batch size (see Section 4.4). Therefore, we report the best performance in Table 1 and Table 2. Detailed settings are provided in Section D.2.

Analysis of default sparsity results. Table 1 and Table 2 compare OLMoE-A1B-7B (Muennighoff et al., 2024) and Qwen1.5-MoE-A2.7B (Team, 2024) across five downstream tasks under different routing strategies. Base denotes the original model without fine-tuning, while *TopK* and *SeqTopK* represent models fine-tuned with standard TopK routing and the proposed SeqTopK routing, respectively. Under default sparsity level (*i.e.*, activate 8 out of 128 experts for OLMoE and 4 out of 60 experts for Qwen1.5), SeqTopK delivers average performance gains of **5.9%** for OLMoE and **3.6%** for Qwen1.5, demonstrating the effectiveness of its intrinsic context-aware routing. Notably, these improvements are obtained with only minimal code changes, yet SeqTopK consistently surpasses standard routing strategies across tasks. The smaller gain for Qwen1.5 can be attributed to its architecture, in which four of the eight experts are shared across tokens, limiting allocation flexibility compared to OLMoE, which has no shared experts. Unlike prior methods that require pre-training from scratch (Jin et al., 2024a; Wang et al., 2024c; Guo et al., 2024), SeqTopK plugs into pre-trained model fine-tuning, delivering low-overhead performance gains for real-world deployment. In Section B, we study applying SeqTopK decoding on top of TopK-fine-tuned models, showing that our approach works well with modern MoE frameworks.

Analysis of scaling sparsity results. As shown in Table 1, we compare routing strategies across varying sparsity levels. MRL-TopK exhibits marked degradation as sparsity increases and underperforms even under the default sparsity setting, suggesting inefficiencies in dynamic expert allocation. By contrast, BatchTopK can match the performance of standard TopK across sparsity levels, but it is highly sensitive to the evaluation batch size, which limits its practical deployment (see Section 4.4). Our proposed SeqTopK consistently outperforms baselines under all settings. As sparsity increases (*i.e.*, K decreases from 8 to 2), the performance gap widens. This trend suggests that SeqTopK’s context-aware expert allocation is particularly advantageous in high-sparsity regimes, where effective utilization of limited expert capacity is critical. Under the extreme regime ($K = 2$), SeqTopK achieves nearly **twice** the performance of the standard TopK strategy, further underscoring its strong ability to handle ultra-sparse settings via intrinsic dynamic routing. This is a fair comparison, as none of the evaluated models are pre-trained under such extreme sparsity conditions. The substan-

tial performance lead of SeqTopK suggests that it is particularly well-suited for ultra-sparse models in both pre-training (see Section 4.2) and fine-tuning stages.

4.2 GPT2-LEVEL PRE-TRAINING EVALUATION

Experiment Setup. We select ReMoE (Wang et al., 2024c) (ReLU-routed) and standard TopK serve as our baselines, distinguished **only by their routing mechanism**. Other related work, such as MoE++ (Jin et al., 2024a) and DynMoE (Guo et al., 2024), that incorporate additional zero or gate experts are excluded. For a detailed discussion about these works, see Section A. Following Wang et al. (2024c), we train the models on The Pile (Gao et al., 2020) for 60k steps (~30B tokens), which exceeds the compute-optimal dataset size predicted by Krajewski et al. (2024) and is sufficient for convergence. We evaluated their zero-shot performance across three downstream tasks: LAMBADA (Paperno et al., 2016); RACE (Lai et al., 2017); ARC (Clark et al., 2018). A more detailed experiment setup is provided in Section D.1.

Analysis. Table 3 reports the zero-shot accuracy of different routing methods on downstream tasks across sparsity levels. SeqTopK consistently outperforms the baselines, achieving a notable gain of **3.2%** and **4.5%** over TopK across sparsity regimes with only 1% additional overhead. As sparsity increases, the performance margin widens, indicating that SeqTopK better handles high-sparsity settings through its sequence-level routing. Compared with ReMoE (Wang et al., 2024c), which employs a three-stage training pipeline and is sensitive to sparsity-controlling hyperparameters, SeqTopK integrates seamlessly with existing MoE models, enabling on-the-fly dynamic routing with only a few lines of code.

4.3 EFFICIENCY ANALYSIS

We report the GPU hours for pre-training a 182M-size model over 30B tokens using various routing strategies in Table 4a. SeqTopK introduces only a marginal pre-training overhead compared to vanilla TopK, *i.e.*, 0.9%, improving efficiency–performance trade-off. We benchmark the OLMoE model’s decoding efficiency for online SeqTopK and the overhead introduced by Expert-Cache, using `huggingface.gen()` for fair comparison, with KV-Cache enabled for both methods. As shown in Table 4b, online SeqTopK achieves nearly identical throughput to standard TopK. The additional memory overhead of Expert-Cache is negligible, increasing peak GPU memory usage by only 0.8%. As discussed in Section 3.2, its theoretical memory cost is significantly lower than KV-Cache and it can be integrated with minor modifications.

4.4 EMPIRICAL UNDERSTANDINGS

In this section, we further provide in-depth understanding of SeqTopK regarding its adaptive behaviors, the distribution of activated experts, and sensitivity w.r.t. batch size compared to BatchTopK.

SeqTopK Enables More Balanced Expert Utilization. We first analyze routing behavior and expert load to compare SeqTopK with standard TopK. Following Lepikhin et al. (2020); Wu et al. (2024a), we compute the normalized routing entropy over the GSM8K dataset. For a MoE layer, let $p_{i,e}$ be the probability of token i being assigned to expert e , averaged over the dataset. The overall routing distribution is $p_e = \frac{1}{N} \sum_{i=1}^N p_{i,e}$, where N is the total number of tokens. The routing entropy is then defined as $H = -\sum_{e=1}^E p_e \log p_e$, and we normalize it by the maximum possible entropy $\log E$. A *higher* entropy means a more *balanced* expert utilization. As shown in Figure 5, SeqTopK generates softer, more distributed assignment probabilities over experts, and achieves higher entropy across layers. By adaptively allocating experts based on token importance, it balances expert utilization across sequences, resulting in a more robust and fine-grained capacity distribution compared to TopK.

SeqTopK Activates More Experts on Hard Tokens. Inspired by Wang et al. (2025), we study how token entropy correlates with the average number of activated experts. The token entropy for token t is calculated by: $H_t = -\sum_{j=1}^V p_{t,j} \log(p_{t,j})$, where $p_t \in \mathbb{R}^V$ is the corresponding probability distribution over the vocabulary. A token with higher entropy indicates greater uncertainty in the model’s prediction, so ideally, the model should allocate more resources to process it. As shown in Figure 4, we sample over 10K tokens generated by a SeqTopK fine-tuned Qwen1.5 and calculate the

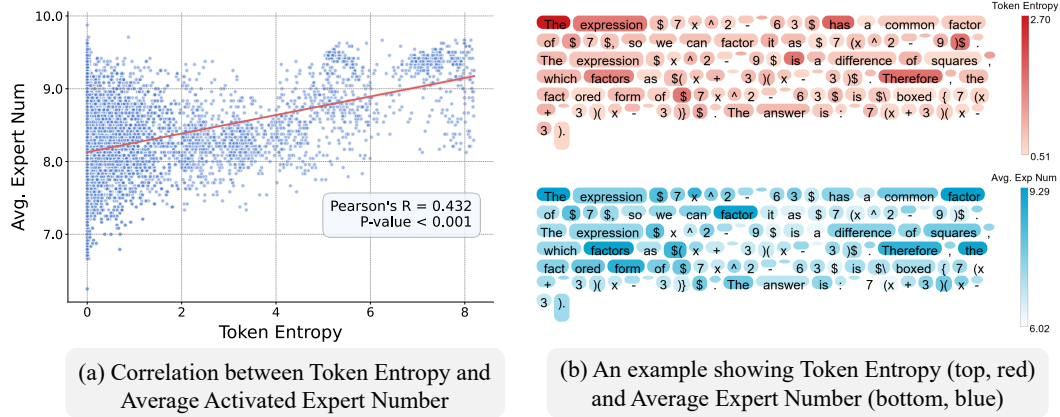


Figure 4: **Correlation between token entropy and expert activation.** (a) Analysis of 10K tokens generated by fine-tuned Qwen-1.5-A2.7B, showing that higher token entropy – defined as the entropy of the output distribution at each token, capturing prediction uncertainty and token hardness – correlates well with a larger number of average activated experts. (b) Illustration from a specific generated sequence, where SeqTopK often activates more experts on high-entropy tokens, such as the “\$” symbol marking the start of a math expression.

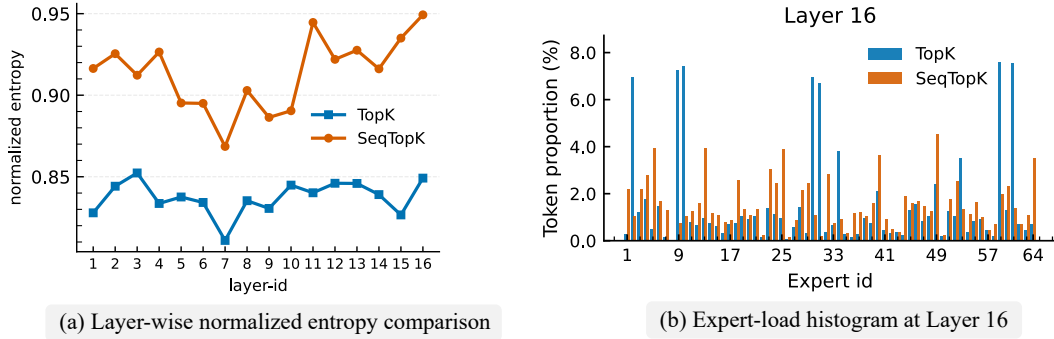


Figure 5: **Routing Dynamics of SeqTopK.** (a) Layer-wise normalized entropy comparison. *Higher* entropy means more *balanced* expert utilization. SeqTopK consistently exhibits higher entropy than TopK, suggesting that its sequence-level routing encourages more uniform (*i.e.*, balanced) expert utilization. (b) Expert-load histogram at layer 16. SeqTopK presents smoother and more balanced expert utilization compared to TopK.

correlation between token entropy and activated experts. With its context-aware design, SeqTopK distributes the expert budget according to token difficulty: high-entropy tokens, *often those that steer the direction of reasoning*, receive more experts, while low-entropy tokens, *typically carrying out routine steps*, are assigned fewer. Figure 4(b) presents an example of expert allocation. Turning words with high entropy, like “therefore,” receive more experts, while low-entropy symbols, like “+” get fewer. Beyond high-entropy tokens, SeqTopK also favors key content words (*e.g.*, factor) and the “\$” symbol, which marks the start of a math expression.

Normal-like Distribution of Activated Experts. To further understand the difference between SeqTopK and TopK, we visualize the activated pattern in SeqTopK in Figure 6(a). Given that MoE models consist of multiple MoE layers, we collect data on the number of activated experts per token per layer and show the expert activation patterns. Specifically, we analyze the expert activation distribution for 10K tokens (without token-level upper bound) across 16 layers of the OLMoE model on GSM8K dataset. Compared to a fixed TopK ($K = 8$) selection, SeqTopK dynamically allocates the token budget ranging from 1 to 18, following a normal-like distribution. This demonstrates an emergent adaptiveness, where the model decides the expert budget activation for each token at each layer, resulting in performance gain.

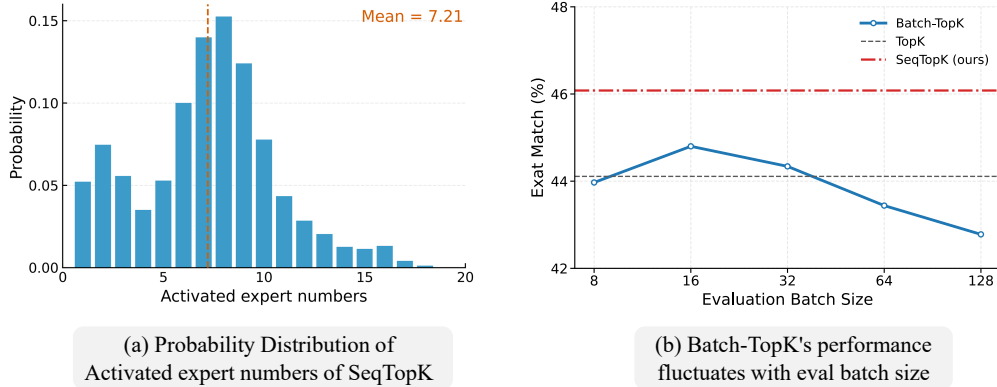


Figure 6: **(a) Expert Activation Pattern of SeqTopK (w/o token-level upper bound).** We present the distribution of OLMoE’s expert activation patterns on the GSM8K dataset with the SeqTopK routing strategy. SeqTopK learns to adaptively activate each token with a varying number of tokens from 1 (lower bound) to 18, following a normal-like distribution. **(b) Batch-TopK is highly influenced by the evaluation batch size**, while SeqTopK consistently outperforms TopK.

Training with Global SeqTopK and Online SeqTopK. We further investigate the effects of two variants of SeqTopK during training: *Global SeqTopK*, which routes experts based on all tokens in the input sequence, and *Online SeqTopK*, which mirrors the autoregressive decoding procedure used at inference time (Section 3.2). As shown in Table 5, while Online SeqTopK also outperforms the standard TopK in most cases, it consistently underperforms compared to Global SeqTopK. We attribute this gap to the fact that Global SeqTopK, having access to the full sequence, benefits from richer contextual information, whereas Online SeqTopK operates causally with limited context. Moreover, the TopK selection—being a non-differentiable, discrete operation—does not leak information from future tokens during gradient updates. Based on these observations, we adopt Global SeqTopK as the default choice for training.

Table 5: **Comparison between Global and Online SeqTopK at training time** on the OLMoE-A1B-7B model with 8 (out of 64) active experts.

Routing	GSM8k	MBPP	HumanEval
TopK	44.11	21.04	13.41
Online SeqTopK	45.80	22.41	12.86
Global SeqTopK	46.09	23.21	15.24

Sensitivity w.r.t. Batch Size. We further investigate the effect of evaluation batch size for different methods. Since the total expert budget of BatchTopK is influenced by the training batch size, it is unclear whether this method would stay robust during evaluation. As shown in Figure 6(b), BatchTopK achieves peak accuracy when the training and evaluation batch sizes match (16 per device in our setup) but degrades as the evaluation batch size increases. While effective in MoE-based diffusion models, its sensitivity to batch size limits practicality, where large evaluation batches are common, leading to unstable inference.

5 CONCLUSION & DISCUSSION

In this work, we introduced Sequence-level TopK (SeqTopK), a minimal yet powerful replacement for standard TopK routing. By comparing tokens within each sequence under the same overall budget as TopK, SeqTopK enables context-aware allocation of expert capacity: difficult, informative tokens receive more experts, while trivial tokens receive fewer. The method is entirely parameter-free, requires neither thresholds or auxiliary predictors, and can be implemented with only a few lines of code while adding less than 1% overhead.

Several studies have explored *expert-level specialization* in MoE models, employing techniques such as expert grouping (Guo et al., 2025), expert merging (Li et al., 2025b), and expert pruning (Lu et al., 2024) to identify the most useful experts for each token. In contrast, we focus on *token-level specialization*, noting that not all tokens warrant equal computational resources. SeqTopK enables the model to autonomously allocate computation under a fixed budget, making it orthogonal to the aforementioned methods and can be easily combined with them.

REFERENCES

- Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.
- Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyouon Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, et al. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. *arXiv preprint arXiv:2507.10524*, 2025.
- Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, et al. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.
- Bart Bussmann, Patrick Leask, and Neel Nanda. Batchtopk sparse autoencoders. *arXiv preprint arXiv:2412.06410*, 2024.
- Bart Bussmann, Noa Nabeshima, Adam Karvonen, and Neel Nanda. Learning multi-level features with matryoshka sparse autoencoders. *arXiv preprint arXiv:2503.17547*, 2025.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- 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 preprint arXiv:1803.05457*, 2018.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. Stable-moe: Stable routing strategy for mixture of experts. *arXiv preprint arXiv:2204.08396*, 2022.
- 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. *arXiv preprint arXiv:2401.06066*, 2024.
- Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.
- William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.
- Zhida Feng, Zhenyu Zhang, Xintong Yu, Yewei Fang, Lanxin Li, Xuyi Chen, Yuxiang Lu, Jiaying Liu, Weichong Yin, Shikun Feng, et al. Ernie-vilg 2.0: Improving text-to-image diffusion model with knowledge-enhanced mixture-of-denoising-experts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10135–10145, 2023.
- Tianyu Fu, Haofeng Huang, Xuefei Ning, Genghan Zhang, Boju Chen, Tianqi Wu, Hongyi Wang, Zixiao Huang, Shiyao Li, Shengen Yan, et al. Moa: Mixture of sparse attention for automatic large language model compression. *arXiv preprint arXiv:2406.14909*, 2024.
- Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- Yongxin Guo, Zhenglin Cheng, Xiaoying Tang, Zhaopeng Tu, and Tao Lin. Dynamic mixture of experts: An auto-tuning approach for efficient transformer models. *arXiv preprint arXiv:2405.14297*, 2024.
- Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33: 9782–9793, 2020.
- Quzhe Huang, Zhenwei An, Nan Zhuang, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. Harder tasks need more experts: Dynamic routing in moe models. *arXiv preprint arXiv:2403.07652*, 2024.
- Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.
- Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.
- Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.
- Peng Jin, Bo Zhu, Li Yuan, and Shuicheng Yan. Moe++: Accelerating mixture-of-experts methods with zero-computation experts. *arXiv preprint arXiv:2410.07348*, 2024a.
- Peng Jin, Bo Zhu, Li Yuan, and Shuicheng Yan. Moh: Multi-head attention as mixture-of-head attention. *arXiv preprint arXiv:2410.11842*, 2024b.
- Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.
- Jakub Krajewski, Jan Ludziejewski, Kamil Adamczewski, Maciej Pióro, Michał Krutul, Szymon Antoniuk, Kamil Ciebiera, Krystian Król, Tomasz Odrzygóźdź, Piotr Sankowski, et al. Scaling laws for fine-grained mixture of experts. *arXiv preprint arXiv:2402.07871*, 2024.
- Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. Matryoshka representation learning. *Advances in Neural Information Processing Systems*, 35:30233–30249, 2022.
- Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*, 2017.
- Dmitry Lepikhin, Hyoungho 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.
- 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*, 2025a.
- Dongxu Li, Yudong Liu, Haoning Wu, Yue Wang, Zhiqi Shen, Bowen Qu, Xinyao Niu, Guoyin Wang, Bei Chen, and Junnan Li. Aria: An open multimodal native mixture-of-experts model. *arXiv preprint arXiv:2410.05993*, 2024.
- Wei Li, Lujun Li, You-Liang Huang, Mark G Lee, Shengjie Sun, Wei Xue, and Yike Guo. Structured mixture-of-experts llms compression via singular value decomposition. *International Conference on Machine Learning*, 2025b.

- Yunxin Li, Shenyuan Jiang, Baotian Hu, Longyue Wang, Wanqi Zhong, Wenhan Luo, Lin Ma, and Min Zhang. Uni-moe: Scaling unified multimodal llms with mixture of experts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025c.
- Bin Lin, Zhenyu Tang, Yang Ye, Jiayi Cui, Bin Zhu, Peng Jin, Jinfa Huang, Junwu Zhang, Yatian Pang, Munan Ning, et al. Moe-llava: Mixture of experts for large vision-language models. *arXiv preprint arXiv:2401.15947*, 2024.
- Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.
- Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024b.
- Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. *arXiv preprint arXiv:2402.14800*, 2024.
- Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*, 2023.
- Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*, 2024.
- Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*, 2016.
- Piotr Piękos, Róbert Csordás, and Jürgen Schmidhuber. Mixture of sparse attention: Content-based learnable sparse attention via expert-choice routing. *arXiv preprint arXiv:2505.00315*, 2025.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.
- Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, et al. Hash layers for large sparse models. *advances in neural information processing systems*, 34:17555–17566, 2021.
- Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472, 2022.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.
- Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarczyk, 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.
- Yikang Shen, Zhen Guo, Tianle Cai, and Zengyi Qin. Jetmoe: Reaching llama2 performance with 0.1 m dollars. *arXiv preprint arXiv:2404.07413*, 2024.
- Minglei Shi, Ziyang Yuan, Haotian Yang, Xintao Wang, Mingwu Zheng, Xin Tao, Wenliang Zhao, Wenzhao Zheng, Jie Zhou, Jiwen Lu, et al. Diffmoe: Dynamic token selection for scalable diffusion transformers. *arXiv preprint arXiv:2503.14487*, 2025.

- Nathaniel J Smith and Roger Levy. The effect of word predictability on reading time is logarithmic. *Cognition*, 128(3):302–319, 2013.
- Haotian Sun, Tao Lei, Bowen Zhang, Yanghao Li, Haoshuo Huang, Ruoming Pang, Bo Dai, and Nan Du. Ec-dit: Scaling diffusion transformers with adaptive expert-choice routing. *arXiv preprint arXiv:2410.02098*, 2024.
- Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijie Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025a.
- Meituan LongCat Team, Bayan, Bei Li, Bingye Lei, Bo Wang, Bolin Rong, Chao Wang, Chao Zhang, Chen Gao, Chen Zhang, Cheng Sun, Chengcheng Han, Chenguang Xi, Chi Zhang, Chong Peng, Chuan Qin, Chuyu Zhang, Cong Chen, Congkui Wang, Dan Ma, Daoru Pan, Defei Bu, Dengchang Zhao, Deyang Kong, Dishan Liu, Feiye Huo, Fengcun Li, Fubao Zhang, Gan Dong, Gang Liu, Gang Xu, Ge Li, Guoqiang Tan, Guoyuan Lin, Haihang Jing, Haomin Fu, Haonan Yan, Haoxing Wen, Haozhe Zhao, Hong Liu, Hongmei Shi, Hongyan Hao, Hongyin Tang, Huantian Lv, Hui Su, Jiacheng Li, Jiahao Liu, Jiahuan Li, Jiajun Yang, Jiaming Wang, Jian Yang, Jianchao Tan, Jiaqi Sun, Jiaqi Zhang, Jiawei Fu, Jiawei Yang, Jiayi Hu, Jiayu Qin, Jingang Wang, Jiyuan He, Jun Kuang, Junhui Mei, Kai Liang, Ke He, Kefeng Zhang, Keheng Wang, Keqing He, Liang Gao, Liang Shi, Lianhui Ma, Lin Qiu, Lingbin Kong, Lingtong Si, Linkun Lyu, Linsen Guo, Liqi Yang, Lizhi Yan, Mai Xia, Man Gao, Manyuan Zhang, Meng Zhou, Mengxia Shen, Mingxiang Tuo, Mingyang Zhu, Peiguang Li, Peng Pei, Peng Zhao, Pengcheng Jia, Pingwei Sun, Qi Gu, Qianyun Li, Qingyuan Li, Qiong Huang, Qiyuan Duan, Ran Meng, Rongxiang Weng, Ruichen Shao, Rumei Li, Shizhe Wu, Shuai Liang, Shuo Wang, Suogui Dang, Tao Fang, Tao Li, Tefeng Chen, Tianhao Bai, Tianhao Zhou, Tingwen Xie, Wei He, Wei Huang, Wei Liu, Wei Shi, Wei Wang, Wei Wu, Weikang Zhao, Wen Zan, Wenjie Shi, Xi Nan, Xi Su, Xiang Li, Xiang Mei, Xiangyang Ji, Xiangyu Xi, Xiangzhou Huang, Xianpeng Li, Xiao Fu, Xiao Liu, Xiao Wei, Xiaodong Cai, Xiaolong Chen, Xiaoqing Liu, Xiaotong Li, Xiaowei Shi, Xiaoyu Li, Xili Wang, Xin Chen, Xing Hu, Xingyu Miao, Xinyan He, Xuemiao Zhang, Xueyuan Hao, Xuezhi Cao, Xunliang Cai, Xurui Yang, Yan Feng, Yang Bai, Yang Chen, Yang Yang, Yaqi Huo, Yerui Sun, Yifan Lu, Yifan Zhang, Yipeng Zang, Yitao Zhai, Yiyang Li, Yongjing Yin, Yongkang Lv, Yongwei Zhou, Yu Yang, Yuchen Xie, Yueqing Sun, Yuewen Zheng, Yuhua Wei, Yulei Qian, Yunfan Liang, Yunfang Tai, Yunke Zhao, Zeyang Yu, Zhao Zhang, Zhaohua Yang, Zhenchao Zhang, Zhikang Xia, Zhiye Zou, Zhizhao Zeng, Zhongda Su, Zhuofan Chen, Zijian Zhang, Ziwen Wang, Zixu Jiang, Zizhe Zhao, Zongyu Wang, and Zunhai Su. Longcat-flash technical report, 2025b. URL <https://arxiv.org/abs/2509.01322>.
- Qwen Team. Qwen1.5-moe: Matching 7b model performance with 1/3 activated parameters", February 2024. URL <https://qwenlm.github.io/blog/qwen-moe/>.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- 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*, 2024a.
- Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025.
- Zihan Wang, Deli Chen, Damai Dai, Runxin Xu, Zhuoshu Li, and Yu Wu. Let the expert stick to his last: Expert-specialized fine-tuning for sparse architectural large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024b.
- Ziteng Wang, Jun Zhu, and Jianfei Chen. Remoe: Fully differentiable mixture-of-experts with relu routing. *arXiv preprint arXiv:2412.14711*, 2024c.
- Tiansheng Wen, Yifei Wang, Zequn Zeng, Zhong Peng, Yudi Su, Xinyang Liu, Bo Chen, Hongwei Liu, Stefanie Jegelka, and Chenyu You. Beyond matryoshka: Revisiting sparse coding for adaptive representation. In *Forty-second International Conference on Machine Learning*, 2025.

- Haoyuan Wu, Haoxing Chen, Xiaodong Chen, Zhanchao Zhou, Tiejuan Chen, Yihong Zhuang, Guoshan Lu, Zenan Huang, Junbo Zhao, Lin Liu, et al. Grove moe: Towards efficient and superior moe llms with adjugate experts. *arXiv preprint arXiv:2508.07785*, 2025.
- Haoze Wu, Zihan Qiu, Zili Wang, Hang Zhao, and Jie Fu. Gw-moe: Resolving uncertainty in moe router with global workspace theory. *arXiv preprint arXiv:2406.12375*, 2024a.
- Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, Zhenda Xie, Yu Wu, Kai Hu, Jiawei Wang, Yaofeng Sun, Yukun Li, Yishi Piao, Kang Guan, Aixin Liu, Xin Xie, Yuxiang You, Kai Dong, Xingkai Yu, Haowei Zhang, Liang Zhao, Yisong Wang, and Chong Ruan. Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding. *arXiv preprint arXiv:2412.10302*, 2024b.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.
- Feng Yao, Junxia Cui, Ruohan Zhang, Liyuan Liu, Shibo Hao, Li Zhang, Chengyu Dong, Shuo-hang Wang, Yelong Shen, Jianfeng Gao, and Jingbo Shang. Densemixer: Improving moe post-training with precise router gradient, 2025. URL <https://fengyao.notion.site/moe-posttraining>.
- Chenyu You, Weicheng Dai, Yifei Min, Lawrence Staib, and James S Duncan. Implicit anatomical rendering for medical image segmentation with stochastic experts. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 561–571. Springer, 2023.
- Chenyu You, Haocheng Dai, Yifei Min, Jasjeet S Sekhon, Sarang Joshi, and James S Duncan. Uncovering memorization effect in the presence of spurious correlations. *Nature Communications*, 2025.
- Yike Yuan, Ziyu Wang, Zihao Huang, Defa Zhu, Xun Zhou, Jingyi Yu, and Qiyang Min. Expert race: A flexible routing strategy for scaling diffusion transformer with mixture of experts. *arXiv preprint arXiv:2503.16057*, 2025.

A RELATED WORK

Mixture-of-Experts. The concept of Mixture-of-Experts (MoE) was first introduced by Jacobs et al. (1991); Jordan & Jacobs (1994) as a way to model heterogeneous data with specialized modules. In this framework, only a subset of parameters, called *experts*, is activated for each input, allowing the model to scale capacity without proportionally increasing computation. Shazeer et al. (2017) extended this idea to large-scale language modeling, training LSTM-based MoE models that achieved performance competitive with dense counterparts. Early MoE works (Roller et al., 2021; Dai et al., 2022) adopt fixed routing strategies, in which each expert is typically assigned a specific role to ensure stable routing and training. Subsequently, GShard (Lepikhin et al., 2020) and Switch Transformer (Fedus et al., 2022) introduced learnable Top-K routing strategies, enabling MoE language models to scale to unprecedented sizes. However, these methods often suffer from load imbalance across experts, requiring auxiliary balancing losses (Wang et al., 2024a). Building on this line, Dai et al. (2024) proposed fine-grained experts that increase the total number of experts by decomposing large experts into smaller ones (e.g., splitting 8 experts into 64 experts with $0.25\times$ size each), thereby substantially improving combinatorial flexibility and enabling more precise specialization. Liu et al. (2024a;b) further introduce shared experts to capture and consolidate common knowledge across varying contexts and incorporate an expert-grouping strategy to enhance expert specialization. Beyond advances in model architectures and training strategies, MoE concepts have been broadly adopted in diverse domains, including large language models (Jiang et al., 2024; Muennighoff et al., 2024; Guo et al., 2025; Li et al., 2025a; Yang et al., 2025; Team et al., 2025a), multimodal LLMs (Li et al., 2024; Lin et al., 2024; Li et al., 2025c; Wu et al., 2024b), diffusion models (Balaji et al., 2022; Feng et al., 2023; Sun et al., 2024), and attention mechanisms (You et al., 2023; Shen et al., 2024; Jin et al., 2024b; Fu et al., 2024; Piękos et al., 2025).

Despite these advances, recent studies suggest that *ultra-sparse* MoE architectures can offer superior efficiency with competitive performance. For example, Qwen3-Next (Yang et al., 2025) introduces an ultra-sparse MoE with only 10 activated experts per token out of 512 total experts, achieving competitive performance with dense models while incurring less than one-tenth of the training cost of dense models. However, our experiments, as shown in Figure 1 (b), show that fixed Top-K routing struggles in such regimes, as the constant number of activated experts per token limits performance under an extreme sparsity regime.

To address this limitation, we propose SeqTopK, which allocates each token’s expert budget via sequence-level comparison, assigning more experts to high-entropy tokens and fewer to low-entropy ones for adaptive routing in ultra-sparse regimes.

Adaptive Token Computation. A central idea in efficient model design is to allocate computation adaptively based on token importance. This principle has been applied from compact representations (Kusupati et al., 2022; Wen et al., 2025; You et al., 2025) to large language models (Hou et al., 2020; Schuster et al., 2022; Dehghani et al., 2018), consistently showing that selective computation can reduce cost while preserving, or even improving, generalization. Within the MoE framework, several works explore token-level dynamic routing. Huang et al. (2024); Lu et al. (2024) propose Top-P selection, where a token activates experts until their cumulative probability exceeds a threshold p . ReMoE (Wang et al., 2024c) replaces TopK and softmax with ReLU gating, coupled with L_1 regularization to encourage sparsity and balance. Other methods take a global perspective: Guo et al. (2024); Wu et al. (2025) add a learnable threshold or a judging expert to enforce cross-token adaptivity. Jin et al. (2024a); Team et al. (2025b) add zero-compute experts, allowing tokens routed to them to bypass to the next layer, enabling a dynamic per-token budget. Beyond expert routing, other works have treated adaptive token computation as a depth allocation problem. Mixture-of-Depths (Raposo et al., 2024) and Mixture-of-Recursions (Bae et al., 2025) route tokens across transformer layers to dynamically control computation depth.

Despite these advances, existing approaches face several critical limitations that hinder their widespread adoption. They typically rely on empirical thresholds that generalize poorly across diverse scenarios (Huang et al., 2024; Lu et al., 2024), introduce additional gating parameters that complicate integration with established frameworks (Guo et al., 2024; Wu et al., 2025), and employ multi-stage training procedures that substantially increase computational complexity while undermining training stability (Wang et al., 2024c). Overall, these methods realize dynamic budgeting via *expert reduction* (i.e., tokens can only activate less than or equal to K experts); in contrast, Seq-

TopK offers a *bidirectional* choice for each token, increasing or decreasing token budgets as needed. SeqTopK neither requires auxiliary parameters nor imposes additional training overhead, offering a streamlined solution that integrates seamlessly into existing architectures for efficient fine-tuning and continued pre-training.

B TRAINING-FREE SEQTOK

We investigate the effect of incorporating the online SeqTopK decoding strategy into a fine-tuned TopK model without additional training. As shown in Table 6, the training-free SeqTopK variant with Expert Cache achieves performance comparable to TopK, highlighting its seamless compatibility with modern MoE frameworks. To fully exploit SeqTopK’s potential, further fine-tuning (fewer than 100 steps) is required to enable the model to autonomously utilize context awareness for expert routing, as demonstrated by the results in Table 1.

Table 6: **Results of Training-Free SeqTopK on top of TopK fine-tuned model.**

<i>OLMoE-A1B-7B</i>			GSM8k	MBPP	HumanEval	Summary	Law	Avg
Routing Methods	K	Sparsity Ratio	0-shot EM	3-shot Pass@1	0-shot Pass@1	0-shot Score	0-shot Score	- Score
TopK	8	1/8	44.11	21.04	13.41	45.31	24.89	29.74
SeqTopK			44.04	20.58	13.03	44.87	24.17	29.34

C ABLATIONS OF UPPER BOUND SELECTION

Table 7 presents an ablation study on the selection of the upper bound for SeqTopK, evaluating configurations ranging from $K + 1$ to $K + 4$ and an unbounded setting across 5 benchmark datasets. We find that while SeqTopK’s performance exhibits slight fluctuations depending on the specific upper bound chosen. For example, $K + 1$ gains top performance on the Law dataset while $K + 3$ achieves the highest score on the summary dataset. Despite these fluctuations, SeqTopK consistently delivers robust performance improvement under most of the evaluated settings.

Table 7: **Ablations of upper bound selection of SeqTopK.** We present an ablation study on the selection of the upper bound for SeqTopK, ranging from $K + 1$ to $K + 4$, as well as configurations with no upper bound.

<i>OLMoE-A1B-7B</i>			GSM8k	MBPP	HumanEval	Summary	Law	Avg
Routing Methods	K	Upper bound	0-shot EM	3-shot Pass@1	0-shot Pass@1	0-shot Score	0-shot Score	- Score
Base TopK	8	K	15.58 44.11	19.80 21.04	10.97 13.41	7.49 45.31	5.70 24.89	11.91 29.74
SeqTopK	8	K+1	46.07	22.38	14.63	45.33	27.90	31.26
		K+2	46.09	23.21	15.24	46.40	26.52	31.49
		K+3	45.15	22.51	15.24	46.45	26.22	31.11
		K+4	44.54	22.00	15.01	44.77	25.18	30.30
		-	44.09	21.80	14.02	44.19	23.38	29.50

D DETAIL EXPERIMENT SETTING

D.1 PRE-TRAINING EXPERIMENT SETTING

We followed the training pipeline described in Wang et al. (2024c), using The Pile dataset (Gao et al., 2020) for 60,000 steps, which corresponds to approximately 30 billion tokens. This training

duration exceeds the compute-optimal dataset size suggested by Krajewski et al. (2024), ensuring the models reach convergence. Our pre-training experiments were conducted at two distinct sparsity levels, activating either 8 out of 64 experts or 4 out of 128 experts for each token. The specific configurations for each model are detailed in Table 8. We used a byte pair encoding (BPE) tokenizer (Sennrich et al., 2015) and maintained consistent training parameters across all models. We employed the AdamW optimizer (Loshchilov & Hutter, 2017) with ZeRO optimization (Loshchilov & Hutter, 2017), using default $\beta_1 = 0.9$ and $\beta_2 = 0.999$ values. The learning rate was set to $5e^{-4}$ with a cosine scheduler and the coefficient for the auxiliary loss (Wang et al., 2024a) is set to 0.01.

For evaluation, we evaluate the zero-shot performance of the trained models on the following downstream tasks: LAMBADA (Paperno et al., 2016); RACE (Lai et al., 2017); ARC (Clark et al., 2018). All models were trained on 8 NVIDIA A100 GPUs.

Table 8: Configurations for different models.

Methods	#Parameters	Total Experts	Activated Experts	hidden_size	num_layers	λ_0	α
TopK	182M	64/128	8/4	768	12	-	-
ReLU	182M	64/128	8/4	768	12	$1e^{-8}$	1.2
SeqTopK	182M	64/128	8/4	768	12	-	-

D.2 LLM FINE-TUNING EXPERIMENT SETTING

We select OLMoE-7B-A1B (Muennighoff et al., 2024) and Qwen1.5-14B-A2.7B (Team, 2024) as backbone models and use the codebase provided by Yao et al. (2025) for downstream fine-tuning. For the math domain, we train on GSM8K (Cobbe et al., 2021) and evaluate zero-shot performance. For the code domain, we train the models on the Python subset of the enormous CodeAlpaca dataset (Luo et al., 2023) to simulate a more targeted LLM customization scenario, and evaluate on HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). We also incorporate evaluation on the Specialized Tasks proposed by Wang et al. (2024b). All models are optimized within 100 steps under the same training setup, and TopK and SeqTopK use **identical hyperparameters** to ensure a fair comparison; the training hyperparameters for each model-task pair are listed in Table 9. All models were trained on 8 NVIDIA A100 GPUs.

We evaluate zero-shot performance on GSM8K (Exact Match) and HumanEval (Pass@1), 3-shot performance on MBPP (Pass@1), and use GPT-4o (Hurst et al., 2024) to assess the quality of answers on summarization and legal tasks following Wang et al. (2024b).

Table 9: Configurations for fine-tuning MoE models

Model	Dataset	Epoch	Batch_Size	Lr	Warmup_Ratio	Gradient_Ckecpointing
OLMOE	GSM8K	3	256	1e-6	0.03	False
	Codealpha	2	256	1e-6	0.03	False
	Summary	2	256	2e-5	0.03	False
	Law	4	256	2e-5	0.03	False
Qwen1.5	GSM8K	4	384	1e-6	0.1	True
	Codealpha	2	256	2e-6	0.1	True
	Summary	1	256	1e-6	0.1	True
	Law	4	128	1e-5	0.1	True