MoEs Are Stronger than You Think: Hyper-Parallel Inference Scaling with RoEs

Soheil Zibakhsh[1,2]* **Mohammad Samragh**[1]† **Kumari Nishu**[1]
Lauren Hannah[1] **Arnav Kundu**[1] **Minsik Cho**[1]
[1]Apple [2]University of California San Diego

Abstract

The generation quality of large language models (LLMs) is often improved by utilizing inference-time scaling methods such as Chain-of-Thought (sequential scaling) or Self-Consistency (parallel scaling). We introduce **Roster of Experts (RoE)**, a training-free inference algorithm for Mixture-of-Experts (MoE) models that defines a third, orthogonal scaling direction. Rather than generating longer outputs or aggregating multiple sequences, RoE improves the next token prediction distribution by treating a single MoE as a dynamic ensemble of internal subnetworks. It injects controlled stochasticity into the expert routing mechanism via Gumbel-Top-k sampling, enabling the model to aggregate multiple diverse subnetworks into a single, higher-quality token prediction. Crucially, because RoE improves the model's output distribution rather than sampling from it, it is the only approach in this family that is applicable to open-ended generation tasks where sequence-level aggregation is infeasible. Furthermore, RoE is orthogonal to existing scaling methods; combining RoE with Self-Consistency yields greater accuracy than either method alone. To overcome the computational cost of multiple forward passes at inference, we employ an efficient batching strategy that exploits hardware parallelism to minimize latency, requiring no fine-tuning of model parameters.

1 Introduction

Large language models (LLMs) derive their power from the vast number of parameters they are trained on, yet at inference time, not all of this capacity is necessarily utilized. This is particularly evident in Mixture-of-Experts (MoE) models, where only a small fraction of parameters are activated for any given token. The routing mechanism that governs this selection is inherently soft and probabilistic: multiple experts may have nearly identical router scores, yet standard top-k routing collapses this rich distribution to a single deterministic choice, discarding viable computational paths. We argue that this represents an underutilized opportunity. A single forward pass through a MoE model is just one sample from a distribution over possible subnetworks, and the particular subnetwork selected by deterministic routing is not necessarily the one that produces the best prediction. This observation motivates a new question: can we improve the quality of a model's output distribution by better utilizing the internal computational capacity that already exists within the model?

A growing body of work addresses a related but distinct goal: improving model outputs by investing additional computation at inference time. These *test-time scaling* methods broadly fall into two categories, illustrated in Figure 1. *Sequential scaling* methods, such as Chain-of-Thought prompting (Wei et al., 2022), improve reasoning quality by prompting the model to produce longer, more structured intermediate outputs. *Parallel scaling* methods, such as Self-Consistency (Wang et al., 2022), generate multiple independent output sequences and aggregate them, for example via majority vote, to arrive at a more reliable final answer. Despite their differences, both paradigms operate at the level of the generated sequence: they either extend or multiply the model's outputs. This shared characteristic imposes two fundamental limitations. First, neither approach can improve the quality of the model's output distribution itself; the distribution from which tokens are drawn remains unchanged. Second, sequence-level aggregation is infeasible for open-ended generation tasks where no single extractable answer exists, such as code generation or summarization.

In this paper, we propose **Roster of Experts (RoE)**, a training-free inference algorithm for MoE

*Work done during an internship at Apple
† Correspondence to: szibakhsh@ucsd.edu and m_samraghrazlighi@apple.com

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 45797–45814
October 24-29, 2026 ©2026 Association for Computational Linguistics

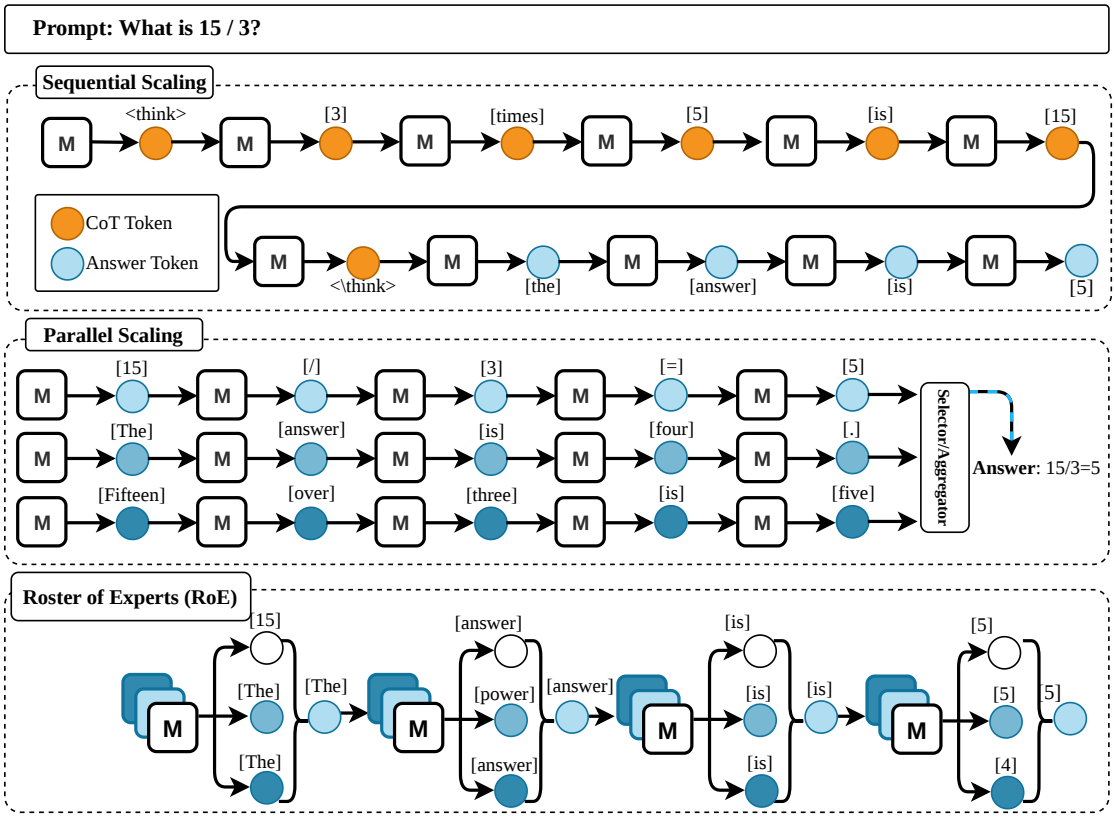


Figure 1: A categorization of inference-time scaling strategies. **(I) Sequential Scaling:** Enhancing performance by generating longer, structured outputs like a chain of thought (Wei et al., 2022). **(II) Parallel Scaling:** Generating multiple token sequences and aggregating them, as in Self-Consistency (Wang et al., 2022). **(III) Roster of Experts (RoE) scaling:** Aggregating results from diverse internal computation paths on a per-token basis. Unlike sequential and parallel scaling, RoE operates at the token level and can be composed with both.

models that addresses both limitations by operating at an entirely different level: the model's internal routing decisions. Rather than generating more tokens or aggregating more sequences, RoE improves the quality of the output distribution itself by better utilizing the computational capacity already present in the model. At each generation step, RoE injects controlled stochasticity into the expert routing mechanism via Gumbel-Top-k sampling (Kool et al., 2019), runs multiple stochastic forward passes through diverse expert subnetworks, and aggregates the resulting logits into a single, higher-quality token prediction. Crucially, this is not sampling from the model's output distribution: it is sampling over which subnetwork *the model is* for that token, thereby raising the quality of the output distribution before any token is ever drawn from it. Any subsequent inference strategy, including Self-Consistency or temperature sampling, then operates on this improved distribution. This positions RoE as a third, orthogonal scaling direction, illustrated as paradigm (III) in Figure 1.

Because RoE operates at the token level rather than the sequence level, it enjoys two properties that neither sequential nor parallel scaling can claim. First, it is applicable to open-ended generation tasks such as code generation, where methods like Self-Consistency cannot aggregate outputs across sequences. Second, it is strictly orthogonal to existing scaling methods and can be composed with them: applying RoE together with Self-Consistency yields greater accuracy than either method alone, as we demonstrate in Section 3. Furthermore, a naive implementation of RoE performing n independent forward passes would incur an n-fold increase in computation. We address this by exploiting structural overlap across forward passes, merging them into a single batched call

that leverages hardware-level parallelism, where latency grows sub-linearly with batch size. These gains require no fine-tuning of model parameters.

In short, the contributions of this work are as follows:

- We propose **Roster of Experts (RoE)**, a training-free inference algorithm for MoE models that improves the quality of the model's output distribution at the token level by sampling diverse internal computational pathways via Gumbel-Top-*k* routing and aggregating their outputs.
- We show through efficiency analysis that the computational overhead of RoE is manageable: our batched implementation keeps memory and latency increases modest even at large sample counts.
- In our design of RoE we think of ease of adoption; RoE is simple enough to be easily integrated to modern language models, incurs no extra training cost, and is efficient during inference.

2 RoE: Hyper Parallel Scaling of Mixture of Experts

Roster of Experts (RoE) enhances a pre-trained MoE model's performance by treating it as a dynamic ensemble. RoE is motivated by the observation that MoE routing decisions are inherently soft and probabilistic. The top-k selection discards experts with nearly identical router scores. RoE samples from this implicit distribution, treating the trained router as defining a probability distribution over expert subnetworks rather than a deterministic selector.

Given the aforementioned insight, we provide a high-level illustration of RoE in Figure 2. At each generation step, the MoE model generates multiple candidate output logits for a single input by sampling diverse expert selections from the model. These outputs are then aggregated to produce a single, more accurate prediction. This process relies on two key components: a stochastic routing mechanism to create diverse paths, and an efficient inference strategy to ensure practicality.

2.1 Gumbel-Top-*k* Routing for Path Diversity

Standard MoE models use deterministic top-k routing, where each token is routed to the k experts with the highest router logits. To generate diverse computational paths, we introduce controlled stochasticity into this selection process using Gumbel-Top-*k* routing. Given the router logits $R \in \mathbb{R}^E$ for a token over E experts, we perturb them with Gumbel noise before selecting the top-k experts. The indices of the selected experts are given by:

$$\text{Indices} = \text{Top-}k(\mathbf{R} + \tau \cdot \mathbf{G}, k) \tag{1}$$

where G is a vector of i.i.d. samples from the Gumbel(0, 1) distribution, and τ is a temperature parameter that controls the degree of stochasticity. When $\tau = 0$, this reduces to standard deterministic top-k routing. As τ increases, the selection becomes more random.

This method is a principled way to sample from the distribution implicitly defined by the router logits. The Gumbel-Max trick (Kool et al., 2019) establishes that adding Gumbel noise to logits before an $argmax$ operation is equivalent to sampling from the categorical distribution produced by applying a softmax to the logits. By extension, applying a Top-*k* operation to Gumbel-perturbed logits corresponds to sampling k elements without replacement from the distribution. This ensures that experts with higher router logits remain more likely to be selected even after adding Gumbel noise, preventing the selection from drifting too far from the trained router's predictions. Figure 2 illustrates this mechanism. Unlike temperature scaling or nucleus sampling, which alter how a token is drawn from a fixed output distribution, Gumbel-Top-*k* routing changes which experts are active before any output distribution is formed.

2.2 Choosing the Gumbel Temperature

The Gumbel temperature, τ, controls the degree of stochasticity in expert routing. We treat it as a layer-specific hyperparameter, defining a temperature vector $\boldsymbol{\tau} = \{\tau_i\}_{i \in \mathcal{L}_{MoE}}$, where $\mathcal{L}_{MoE}$ is the set of MoE layers. Setting a small value of τ keeps router selections nearly unchanged, reducing expert diversity per sample and making the next-token prediction closely match the underlying MoE. In contrast, an excessively large τ introduces too much randomness in expert selection, degrading prediction quality. Appendix D illustrates how task performance varies as the temperature increases. In Appendix A we conduct a study for guidelines for choosing the correct temperature hyperparameter.

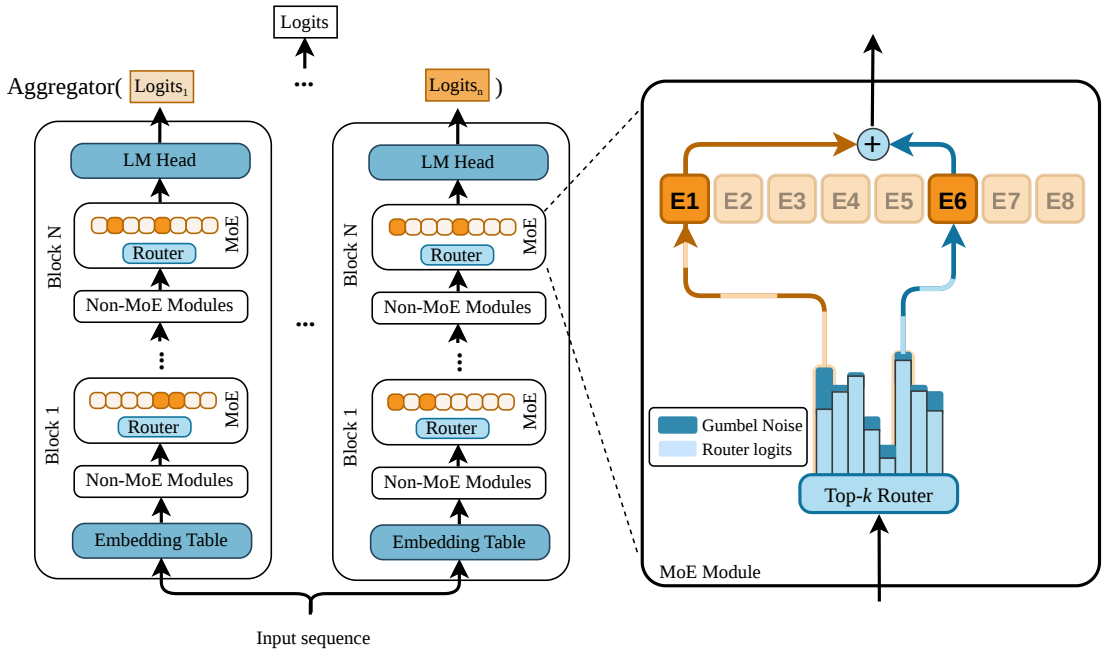


Figure 2: An illustration of the Roster of Experts (RoE) method. **Left:** For a single input, n distinct expert configurations are sampled by adding stochasticity to the expert routing at each MoE layer, and the resulting output logits are aggregated to form the final prediction. **Right:** A closer view of a single MoE layer shows $k = 2$ active experts (dark orange), where Gumbel noise (dark blue) is added to the router logits, and the top-k experts are selected based on these modified logits.

2.3 Efficient RoE Inference

A naive implementation of RoE, which performs n independent forward passes, would incur a prohibitive n-fold increase in computation. To make RoE practical, we leverage two critical optimization techniques: batched parallel processing and Key-Value (KV) caching.

First, we utilize the parallel processing capabilities of modern accelerators. By processing the n samples for a single token generation step as a unified batch, we exploit hardware-level parallelization where latency grows sub-linearly with batch size. This significantly reduces wall-clock time compared to sequential execution. Second, we rely on standard KV-caching to avoid redundant computations of past hidden states. As we demonstrate in Section 3.3, the use of KV-caching is indispensable.

While these standard optimizations solve latency constraints, the memory footprint of the KV-cache naturally scales with the sample size n. To address this in strictly memory-constrained settings, we conceptualize an optimization scheme that reduces this footprint by sharing the cache across the batch. The intuition is that sufficient diversity can be maintained by applying stochastic routing only to the current token, allowing the reuse of a single, deterministic "clean" history. We include a preliminary proof-of-concept for this approach in our experiments with the OLMoE model, showing that it is possible to keep the memory footprint identical to that of a single sample.

3 Experiments

3.1 Experimental Setup

Models and Benchmarks. We evaluate RoE on three instruction-tuned MoE models: OLMoE-1B-7B-Instruct (Muennighoff et al., 2025), Mixtral-8x7B-Instruct-v0.1 (Jiang et al., 2024), and GPT-OSS-20B (OpenAI, 2025b). Our evaluation spans three domains: mathematical reasoning (GSM8K (Cobbe et al., 2021), SVAMP (Patel et al., 2021), AddSub (Hosseini et al., 2014), SingleEq (Koncel-Kedziorski et al., 2015), MultiArith (Roy and Roth, 2015)), commonsense reasoning (ARC-Easy, ARC-Challenge (Clark et al., 2018), OpenBookQA (Mihaylov et al., 2018), Social-I-QA (Sap et al., 2019), Hellaswag (Zellers et al., 2019)), and code gen-

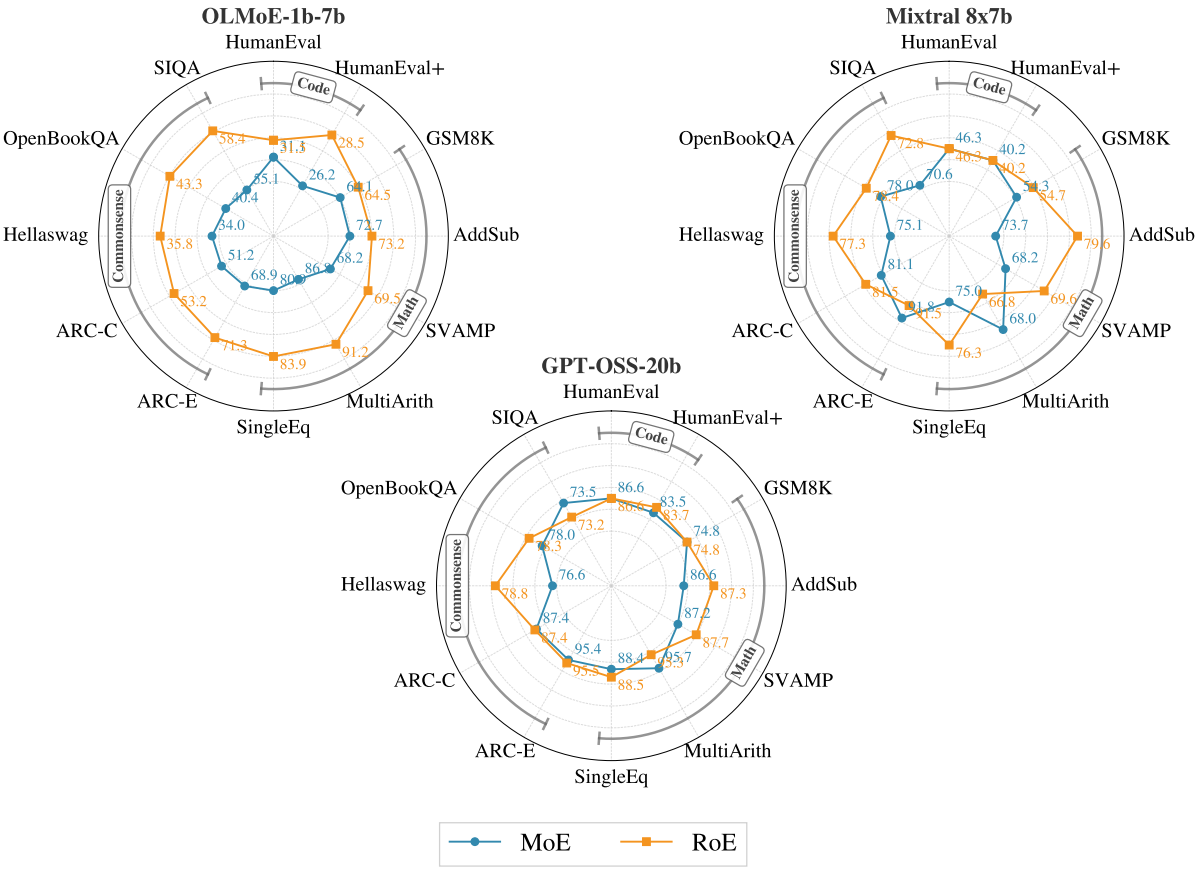


Figure 3: Performance comparison of base MoE models and RoE on five mathematical, five commonsense, and two code benchmarks. Accuracy is measured by exact match (except for HE and HE+, which use pass@1). Results are averaged over five random seeds. Axes are normalized to $(\min - 1\%, \max + 1\%)$ for visualization.

eration (HumanEval (Chen et al., 2021), HumanEvalPlus (Liu et al., 2023)).

Baseline and Evaluation Strategy. Our primary goal is to isolate the performance gains directly attributable to RoE's modification of the model's internal computational pathways. In our evaluations, greedy decoding is used precisely because RoE's contribution is to the output distribution itself, and greedy decoding isolates that improvement cleanly. Stochastic baselines would conflate improvements to the distribution with variance from sampling it.

RoE Configuration and Tuning. We tune the per-layer noise temperatures for RoE on each task's validation set using the Tree-structured Parzen Estimator (TPE) (Bergstra et al., 2011) implemented in Optuna (Akiba et al., 2019). All reported results are evaluated on the **test sets** after tuning, ensuring no data contamination from the **validation sets**. For computational efficiency, we optimize validation perplexity for math tasks and validation accuracy for commonsense and code tasks. The tuning budget and key RoE hyperparameters are summarized in the appendix (Table 1), while the complete per-layer temperature profiles are visualized in the appendix (Figure 10).

Implementation Details. Experiments use NVIDIA A100 80GB GPUs. RoE predictions are aggregated by probability averaging. For commonsense tasks, we follow Hu et al. (2023) and select the answer with the highest log-probability. Results are averaged over 5 seeds. Inference latency is measured as the wall-clock time for generating 128 tokens. We report exact-match accuracy for all tasks and pass@1 additionally for code tasks.

3.2 Main Results

Figure 3 presents the performance of RoE applied to the three MoE models across our suite of benchmarks. As is evident, RoE consistently improves performance across nearly all tasks and model scales, demonstrating its broad effectiveness as a post-hoc enhancement strategy.

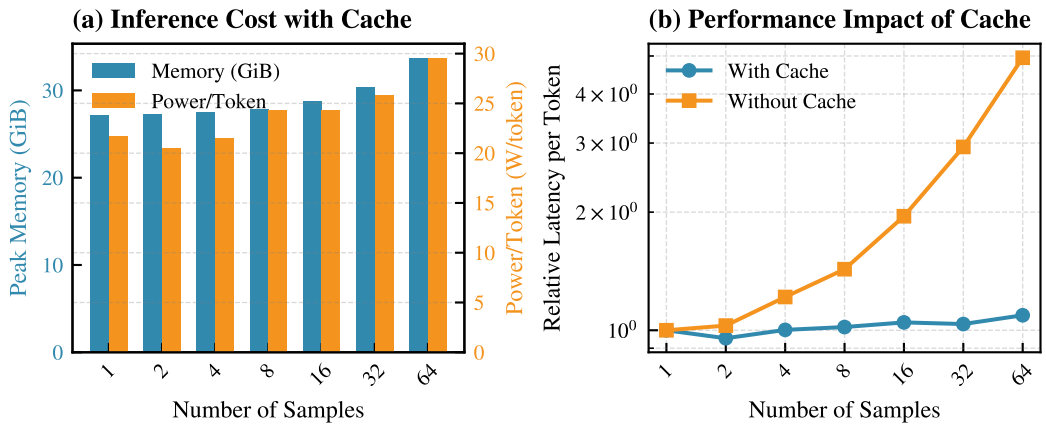


Figure 4: Impact of caching on RoE performance. **(a)** Resource usage with caching enabled. Peak memory (blue, left axis) and power per token (orange, right axis) show a modest increase as the sample count grows. **(b)** Latency comparison with and without caching. Without caching, latency per token rises sharply, highlighting the necessity of caching for scalable RoE inference.

The performance gains are most pronounced for OLMoE, the smallest and weakest base model, where RoE substantially lifts accuracy across all benchmark categories. We hypothesize that by diversifying computational paths, RoE unlocks latent capabilities inaccessible to a single, static routing configuration. This suggests the diversification is most impactful for models with more initial headroom for improvement.

However, the improvements are not uniform. For math benchmarks, we optimized RoE's hyperparameters for validation set perplexity. While RoE consistently lowered perplexity (see Figures 7-9 in the appendix), this did not always translate to higher generative accuracy. For instance, on MultiArith, RoE did not improve the accuracy of Mixtral or GPT-OSS. This result underscores the known disconnect between perplexity and generative reasoning performance, suggesting that tuning RoE directly on task-specific metrics could yield further gains where computationally feasible.

Finally, we observe a ceiling effect: as a model's baseline performance on a benchmark approaches saturation, the potential for improvement diminishes. For instance, the GPT-OSS model already achieves 95.7% on MultiArith, leaving minimal room for any method to provide substantial gains.

A key advantage of RoE is its applicability to open-ended generation, where parallel scaling methods, like majority voting over complete outputs, are infeasible. By aggregating logits at each token-generation step, RoE enhances the performance of open-ended tasks such as code generation, as evident in Figure 3.

3.3 Computational Overhead

We evaluate the computational overhead of RoE in terms of GPU memory footprint, power usage, and throughput (tokens/second). Our benchmark uses the OLMoE model to generate solutions for the first 100 problems from the GSM8K benchmark (Cobbe et al., 2021) on a single NVIDIA A100 GPU. To isolate the overhead of our method and ensure a fair comparison, we set the generation temperature and τ to 0, which produces identical output sequences across all runs. In all experiments, an off-the-shelf KV cache is utilized.

Figure 4(a) illustrates the relative increase in inference cost for RoE with our caching mechanism. The resource requirements grow modestly with the number of samples. For instance, using 64 samples increases the memory footprint by only 12% and power consumption by 20%. These results show that the additional computational cost is manageable and not prohibitive.

Figure 4(b) underscores the necessity of our caching scheme. By comparing RoE with and without caching, we observe that our approach drastically mitigates the increase in power consumption and latency. Without caching, the cost of sampling multiple paths would be impractical. These findings confirm that RoE offers a practical trade-off, enhancing model quality for a moderate and controllable increase in computational demand.

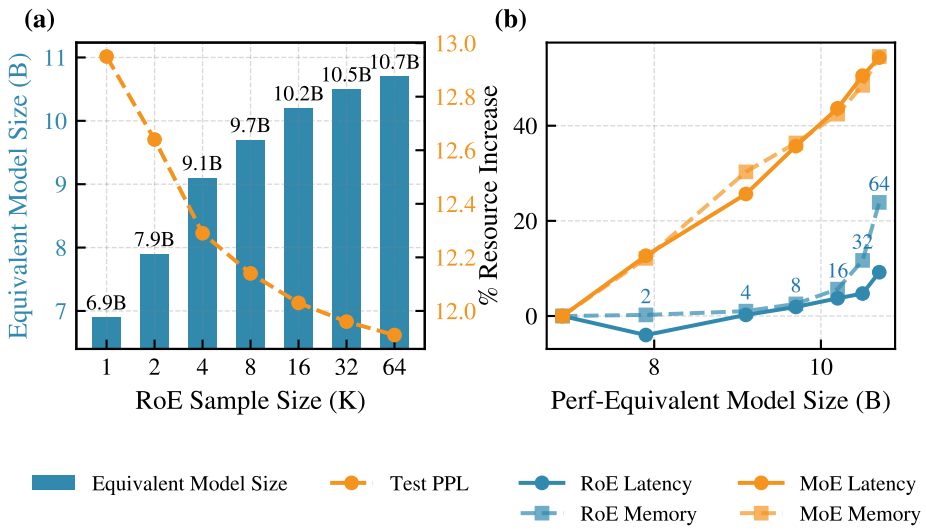


Figure 5: Performance and efficiency analysis of RoE. **(a)** The performance of RoE applied to OLMoE-1B-7B, evaluated via test perplexity on WikiText-103. Equivalent model size is estimated using scaling laws to approximate the efficiency-performance trade-off. **(b)** Comparison of the relative increase in latency and memory for RoE (blue) versus scaling up to an equivalently performing MoE model (orange). The numbers on the blue curve indicate the RoE sample size K.

3.4 Efficiency Analysis: RoE vs. Model Scaling

RoE enhances a model's performance, making it comparable to a larger version of the base model. To quantify this improvement, we measure the *equivalent model size* that a standard MoE would need to match the performance of RoE for a given number of samples K[1]. This allows us to directly compare the efficiency of RoE against the cost of simply using a larger MoE. We estimate this equivalent size by applying the scaling laws from Kaplan et al. (2020), using test perplexity on the WikiText-103 dataset (Merity et al., 2017) as the performance metric. While we acknowledge the imperfect correlation between perplexity and downstream performance, this established framework provides the most practical resource estimation benchmark in the absence of larger trained OLMoE variants. For each sample size K, we tune the RoE routing temperatures across all layers by optimizing perplexity on the WikiText-103 validation set over 50 trials with a TPE optimizer. We then measure the test perplexity of the best-performing temperature setting on the validation set. As shown in Figure 5(a), the effective model size grows monotonically with K, although the gains diminish as the sample size increases and the set of viable expert pathways saturates.

Figure 5(b) compares the computational cost of RoE (blue curve) with the cost of using a larger, equivalently performing standard MoE model (orange curve). The results show that RoE is substantially more efficient in terms of both latency and memory. For instance, applying RoE with $K = 32$ to the OLMoE-7B model yields performance comparable to that of a 10.5B OLMoE model. Notably, this configuration reduces memory overhead by 25% and per-token latency by 30% relative to the larger model, underscoring the efficiency of RoE as an alternative to conventional model scaling.

3.5 RoE as an Orthogonal Scaling Direction

We investigate the interaction between RoE and sequence-level aggregation methods, specifically Self-Consistency (SC) (Wang et al., 2022). While RoE (Hyper-Parallel scaling) enhances the model's intrinsic generation capabilities by diversifying internal computational paths, SC (Parallel scaling) improves performance by selecting the most consistent output across multiple independent generations. However, a key limitation of SC is its reliance on tasks where a distinct final answer can be extracted and aggregated. In contrast, RoE operates directly at the token-level, making it applicable to

[1]We overload the symbol by case: lowercase k is the number of active experts per token (top-k routing), uppercase K is the number of RoE samples. Only K varies in our experiments; k stays at the model's trained value.

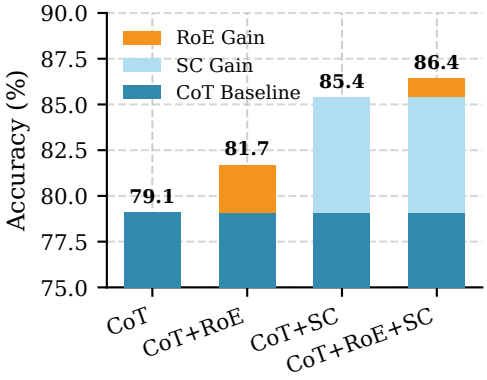


Figure 6: Ablation study showing the additive gains of RoE and Self-Consistency over the CoT baseline.

any generative task, including open-ended generation where answer aggregation is infeasible. Despite these functional differences, because they target distinct stages of the inference process, they can be complementary. To demonstrate this complementary relationship, we conducted an experiment on the SingleEq (Koncel-Kedziorski et al., 2015) benchmark comparing RoE ($K = 32$) against Self-Consistency ($N = 5$) and their combination. As illustrated in Figure 6, both methods individually outperform the greedy Chain-of-Thought (CoT) baseline (79.1%). RoE achieves 81.7% and SC achieves 85.4% in isolation. Crucially, the combined approach (**SC + RoE**) yields a substantial improvement, reaching an accuracy of **86.4%**. This result confirms that RoE enhances the model's fundamental token-prediction quality, which in turn provides a stronger probability distribution for sequence-level methods like SC to exploit.

4 Related Work

Our work, Roster of Experts (RoE), introduces a method for trading additional inference-time compute for improved model performance. Several strategies improve model performance by increasing computation at inference time. Inspired by Mirtaheri et al. (2025), we categorize them into three paradigms, as illustrated in Figure 1.

Sequential Scaling: Sequential scaling methods improve reasoning quality by prompting the model to generate longer, more structured intermediate steps. The seminal work in this area is Chain-of-Thought (CoT) prompting (Wei et al., 2022; Nye et al., 2021), which elicits step-by-step reasoning. In this category, we prompt the model to think, question itself, and do a step-by-step analysis (OpenAI, 2025a; Kojima et al., 2022). This approach has been extended by a large body of subsequent work that explores more complex reasoning structures, such as Tree-of-Thoughts (Yao et al., 2023), at the cost of increased sequential generation steps.

Parallel Scaling: Parallel scaling generates multiple independent outputs for the same prompt and aggregates them to produce a final, more robust answer (Brown et al., 2024). The simplest technique to mention for this category is beam search (Freitag and Al-Onaizan, 2017), where multiple sequences of tokens are evolved at the inference time and the best sequence is chosen as the response. Another prominent example is Self-Consistency (SC) (Wang et al., 2022), which samples diverse reasoning paths using stochastic decoding (e.g., non-zero temperature) and selects the most frequent answer via majority vote. While highly effective, the standard voting mechanism limits SC to tasks with easily verifiable answers, such as math or multiple-choice questions. Recent work aims to extend this paradigm to open-ended generation tasks by developing more sophisticated aggregation strategies that do not rely on a single, extractable answer and instead aggregate on a sequence level (Chen et al., 2023; Taubenfeld et al., 2025; Wang et al., 2024).

Roster of Experts: We introduce a distinct, third paradigm. Unlike sequential and parallel methods that scale the *output generation process*, RoE diversifies the *internal computation paths* within the model for a single token prediction by treating a single MoE model as an ensemble of subnetworks. RoE aggregates multiple internal predictions by sampling different sets of active experts for each forward pass, producing a single, higher-quality output distribution for the next token. This approach can be combined with both sequential and parallel scaling. This is also distinct from adjusting the global sparsity of a pretrained MoE. A model trained at a fixed top-k has an operating point tied to that sparsity level, and shifting it at inference requires post-hoc adaptation to remain well-behaved (Hannah et al., 2025). RoE holds k fixed at its trained value and varies expert identity across passes rather than expert count within one.

A related direction was explored by Geiping et al. (2026), who designed a special recurrent architecture allowing for variable-depth inference. However, their method requires a model to be specifically pretrained for this capability. In contrast, RoE is a training-free strategy that can be applied post-hoc to any pretrained MoE model, offering a prac-

tical way to unlock enhanced performance from existing artifacts dynamically, and on demand.

5 Conclusion

We introduce RoE, a training-free inference algorithm for MoE models that defines a third, orthogonal scaling direction. By treating a single MoE as a dynamic ensemble, RoE improves token-level prediction quality without modifying model parameters. Because RoE operates at the token level, it applies to open-ended generation tasks where sequence-level aggregation is infeasible. Experiments across reasoning and code benchmarks show consistent gains, and combining RoE with Self-Consistency yields strictly higher accuracy than either method alone, while incurring modest computational overhead.

Limitations

While RoE demonstrates consistent empirical gains, several limitations warrant acknowledgment.

First, our efficiency analysis comparing RoE against equivalent larger models relies on scaling law extrapolation rather than direct comparison against trained models of varying sizes. This is a practical constraint, as larger OLMoE variants are not publicly available, and it means the precise computational trade-offs reported (e.g., matching a 10.5B model at 7B cost) should be interpreted as estimates rather than exact measurements.

Second, selecting the Gumbel temperature remains a non-trivial hyperparameter optimization problem. Our current approach relies on Bayesian optimization over a validation set, which itself incurs additional compute. A more principled theoretical or empirical framework for setting temperatures, potentially without requiring a full search procedure, is an important direction for future work.

Third, the KV-cache memory footprint naturally grows with the number of RoE samples, which may become a meaningful constraint in long-context settings. One promising mitigation is to share a single deterministic history cache across the batch, applying stochasticity only to the current token. While this approach is conceptually well-motivated, its practical behavior across diverse tasks and model architectures is not yet well understood, and further research is needed to determine how reliably it can reduce memory overhead without sacrificing the diversity that drives RoE's gains.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

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. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. 2023. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

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

Markus Freitag and Yaser Al-Onaizan. 2017. Beam search strategies for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 56–60.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. 2026. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *Advances in Neural Information Processing Systems*, 38:41340–41391.

Lauren Hannah, Soheil Zibakhsh, Kumari Nishu, Arnav Kundu, Mohammad Samragh Razlighi, Mehrdad Farajtabar, and Minsik Cho. 2025. Moe-phds: One moe checkpoint for flexible runtime sparsity. *arXiv preprint arXiv:2509.23012*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve

arithmetic word problems with verb categorization. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 523–533.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 5254–5276.

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. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Wouter Kool, Herke Van Hoof, and Max Welling. 2019. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International conference on machine learning*, pages 3499–3508. PMLR.

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

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2381–2391.

Parsa Mirtaheri, Ezra Edelman, Samy Jelassi, Eran Malach, and Enric Boix-Adsera. 2025. Let me think! a long chain of thought can be worth exponentially many short ones. *Advances in Neural Information Processing Systems*, pages 166795–166826.

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

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. 2021. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*.

OpenAI. 2025a. Gpt-5 [large language model]. https://openai.com/index/introducing-gpt-5/. Released August 7, 2025.

OpenAI. 2025b. gpt-oss-120b & gpt-oss-20b model card.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 2080–2094.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 4463–4473.

Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. 2025. Confidence improves self-consistency in llms. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20090–20111.

Han Wang, Archiki Prasad, Elias Stengel-Eskin, and Mohit Bansal. 2024. Soft self-consistency improves language models agents. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 2: Short papers)*, pages 287–301.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

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

A Optimizing RoE Temperature

Selecting the optimal τ presents a hyperparameter optimization problem, balancing the potential performance gain against the cost of the search. The primary challenge is the search space, which grows exponentially with the number of MoE layers, rendering an exhaustive search infeasible. A practical search strategy therefore requires an efficient optimization algorithm and a carefully chosen validation metric.

A.1 Optimization Metric

We consider two metrics for guiding the hyperparameter search: validation perplexity and task-specific accuracy.

Validation Perplexity (PPL) is computationally inexpensive, as it only requires a single forward pass over the validation set. However, PPL is an indirect measure of generative reasoning. A low PPL indicates that the model assigns a high probability to a ground-truth sequence, but does not guarantee that the model can generate a correct solution independently.

Validation Accuracy directly measures the model's ability to solve the task, making it a more faithful metric for generative performance. The main drawback is its high computational cost, as it requires generating a full solution for each validation example. This cost can make it prohibitive for large-scale hyperparameter searches.

A.2 Search Strategy

Given the cost of each evaluation, Bayesian optimization methods are well-suited for this task. In our experiments (Section 3), we employ the Tree-structured Parzen Estimator (TPE) (Bergstra et al., 2011) via the Optuna framework (Akiba et al., 2019). To make the search tractable for models with many MoE layers, we introduce two heuristics to prune the vast search space, based on empirical observations:

I. Apply RoE to middle layers only. We observe that the initial and final layers of a transformer are more sensitive to routing perturbations. The initial layers process raw token embeddings, while the final layers consolidate information for the output prediction. Our experiments show preference of the optimizer to reduce the stochasticity of initial and final layers. We therefore constrain the search by setting the temperature to zero ($\tau_i = 0$) for the first and last L MoE layers, applying RoE only to the intermediate ones.

II. Bound the temperature range. We empirically observe that temperature values above 0.5 introduce excessive routing noise, which consistently harms model performance. Consequently, we restrict the search space for each non-zero temperature to the range $[0, 0.5]$ which guides the Bayesian optimization to faster convergence.

B Experiment Setup Details

The design details of the experiments are summarized in Table 1.

C Tuning History and Results Heatmap

The optimization history is given in Figures 7, 8,and 9, and the resulting heatmaps are given in Figure 10.

D Impact of Routing Temperature

The routing temperature is a key hyperparameter in RoE, governing the diversity of sampled expert paths. To understand its effect, we conduct a sensitivity analysis where we apply a uniform temperature across all MoE layers and sweep its value in increments of 0.05.

As shown in Figure 11, performance consistently follows a concave trend: accuracy improves as the temperature increases from zero, peaks at an optimal value, and then declines. This decline occurs because excessively high temperatures introduce too much noise into the routing decisions, leading to the selection of less relevant experts and degrading the final prediction quality. Furthermore, we observe that sparsity levels influence hyperparameter sensitivity rather than total gain: highly sparse models (e.g., GPT-OSS) are more sensitive to noise than less sparse ones (e.g., Mixtral).

Table 1: Experiment setup for model tuning across different task categories. For each category, we specify the tuning configuration and model-specific hyperparameters. N denotes the number of RoE samples, T is the maximum temperature in tuning, and L refers to the number of initial and final skipped layers. Sample denotes the number of validation set samples, Trial is the number of optimization trials. PPL denotes if perplexity was used as the optimization objective.

Category	Task	Sample/Trial	PPL	OLMoE			Mixtral			GPT-OSS		
				N	T	L	N	T	L	N	T	L
Math	GSM8K AddSub SVAMP MultiArith SingleEq	100/50	✓	32	0.5	1	64	0.25	5	64	0.2	5
Common	SiQA OBQA Hellaswag ARC-E ARC-C	300/100	✗	32	0.5	3	64	0.3	3	64	0.2	5
Code	Humaneval Humaneval+	50/50	✗	32	0.5	1	64	0.25	5	64	0.2	5

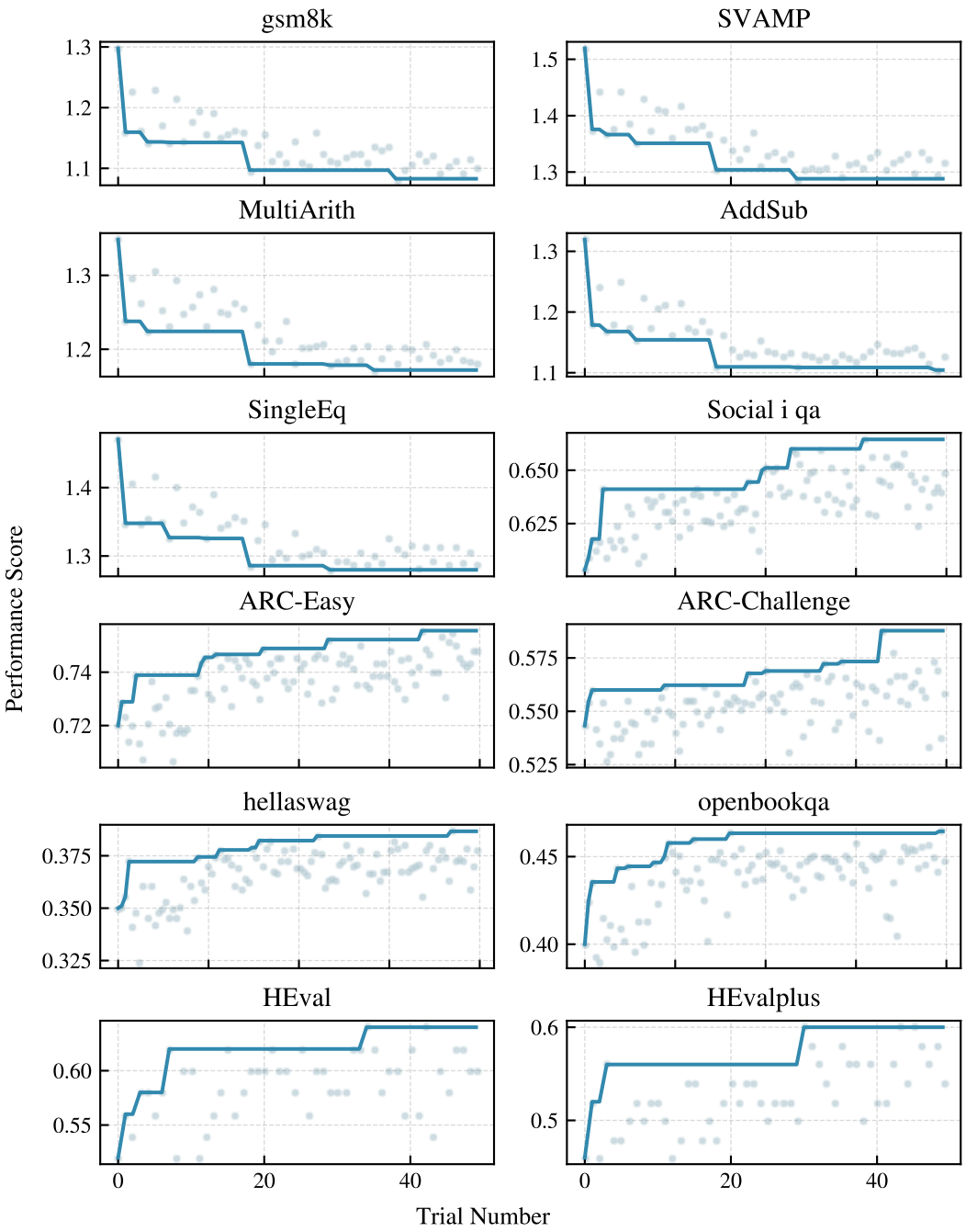


Figure 7: Optimization history of OLMoE on the benchmarks on the validation set. Test set evaluation results are provided in Figure 3

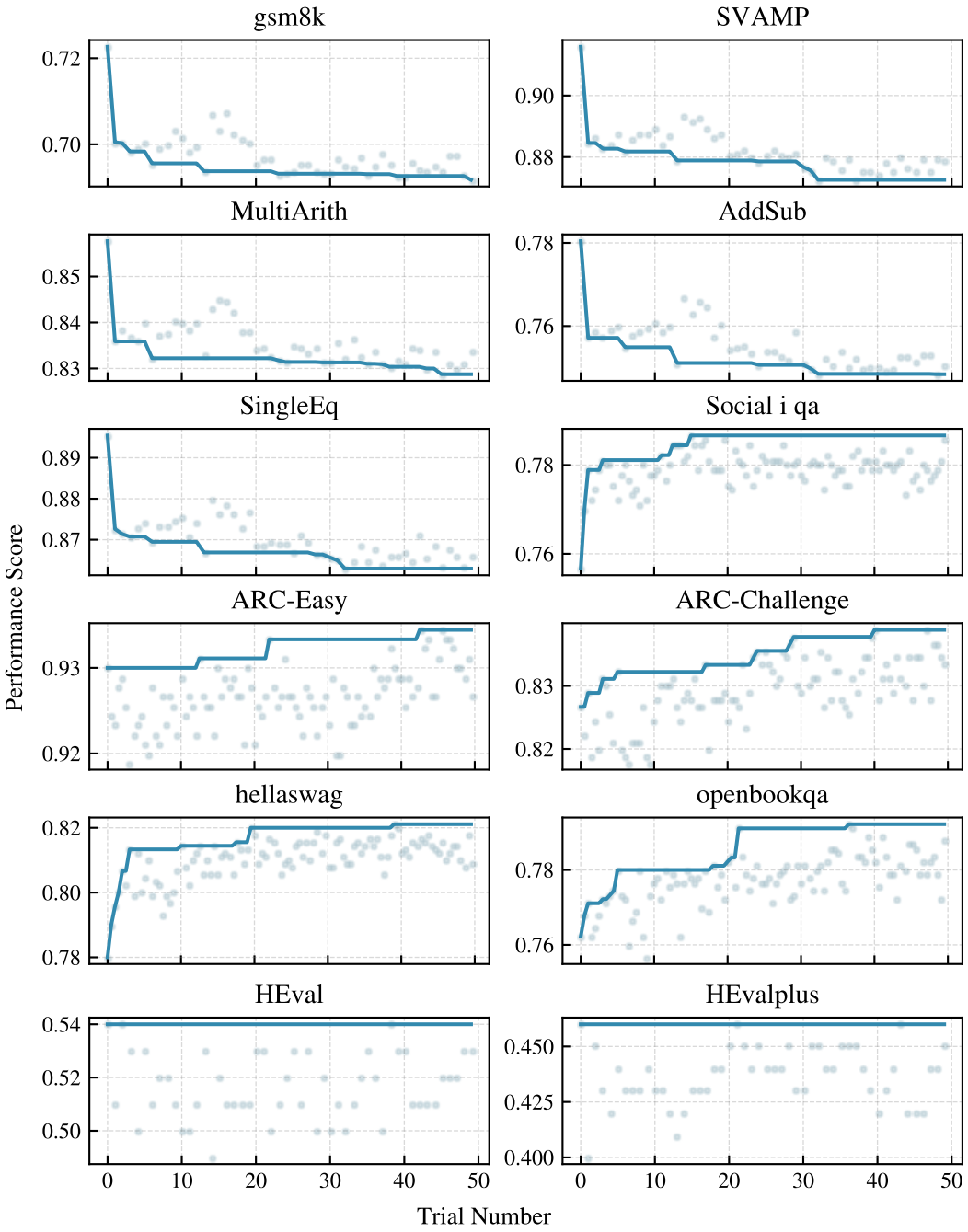


Figure 8: Optimization history of Mixtral on the benchmarks on the validation set. Test set evaluation results are provided in Figure 3

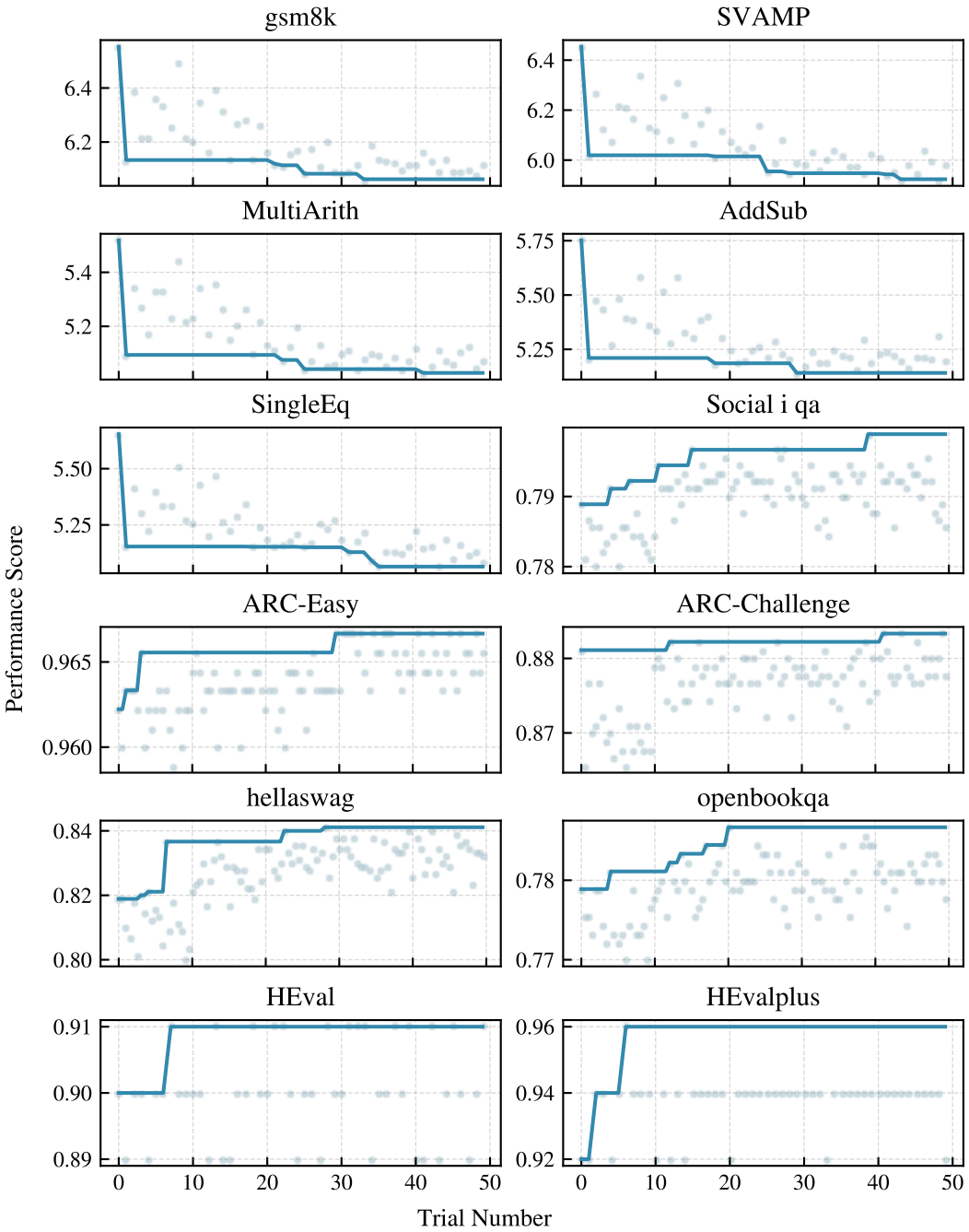
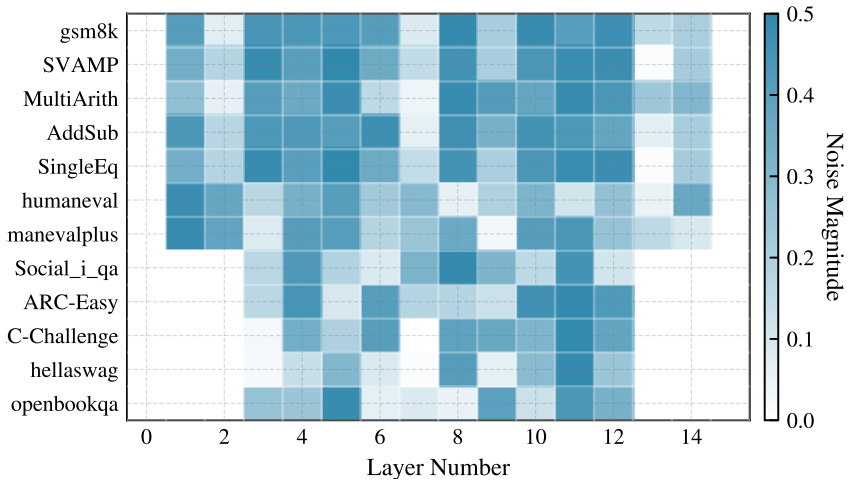
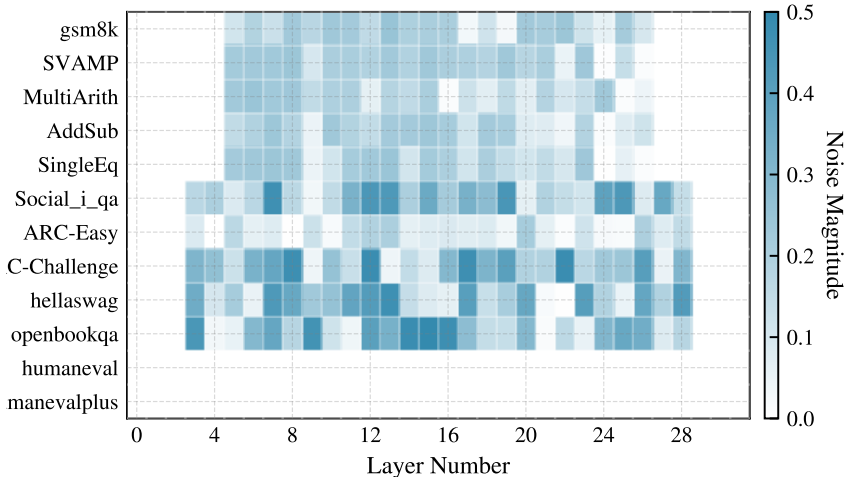


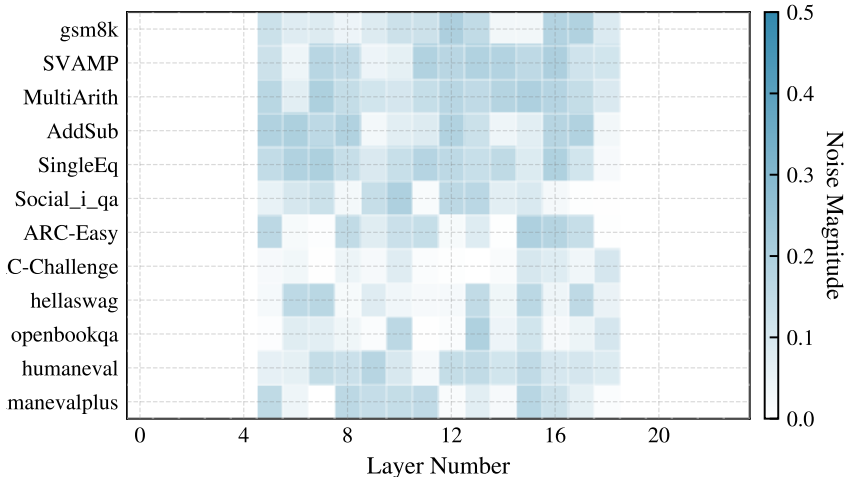
Figure 9: Optimization history of GPT-OSS on the benchmarks on the validation set. Test set evaluation results are provided in Figure 3



(a) OLMoE-1b-7b layer temperature heatmap.



(b) Mixtral-7x8 layer temperature heatmap.



(c) GPT-OSS-20B layer temperature heatmap.

Figure 10: Heatmap of per layer temperature of OLMoE, Mixtral, and GPT-OSS after hyper-parameter tuning on different layers.

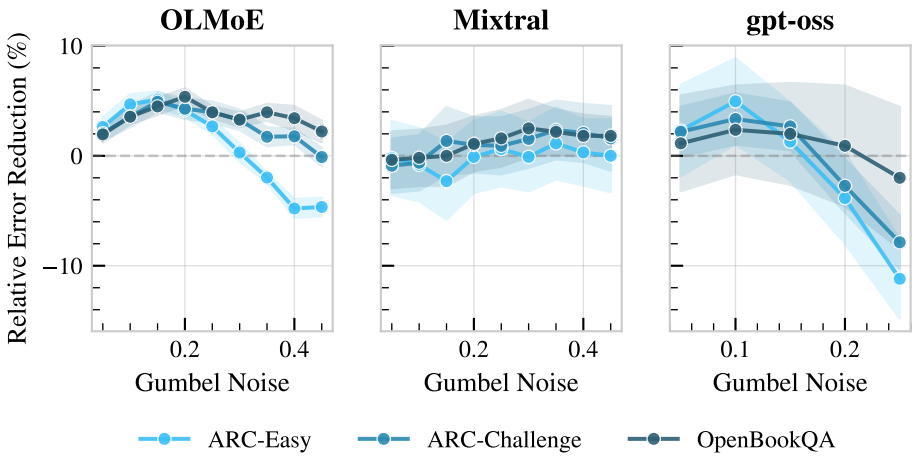


Figure 11: Impact of routing temperature on task performance. We apply a uniform temperature across all MoE layers and observe a concave relationship where performance peaks at a task-specific optimal value. Excessively high temperatures degrade performance by introducing noise into the expert selection process.