POST-TRAINED MOE CAN SKIP HALF EXPERTS VIA SELF-DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture-of-Experts (MoE) scales language models efficiently through sparse expert activation, and its dynamic variant further reduces computation by adjusting the activated experts in an input-dependent manner. Existing dynamic MoE methods usually rely on pre-training from scratch or task-specific adaptation, leaving the practical conversion of fully trained MoE underexplored. Enabling such adaptation would directly alleviate the inference costs by allowing easy tokens to bypass unnecessary experts during serving. This paper introduces **Z**ero-**E**xpert Self-**D**istillation **A**daptation (**ZEDA**), a low-cost framework that transforms post-trained static MoE models into efficient dynamic ones. To stabilize this architectural conversion, ZEDA injects parameter-free zero-output experts into each MoE layer and adapts the augmented model through two-stage self-distillation, utilizing the original MoE as a frozen teacher and applying a group-level balancing loss. On Qwen3-30B-A3B and GLM-4.7-Flash across 11 benchmarks spanning math, code, and instruction following, ZEDA eliminates over 50% of expert FLOPs at marginal accuracy loss. It outperforms the strongest dynamic MoE baseline by 11.7 and 14.9 points on the two models, and delivers ~1.20× end-to-end inference speedup.

1 INTRODUCTION

MoE architectures have significantly advanced the scaling of large language models (LLMs) by increasing model capacity while keeping bounded per-token computation (Lepikhin et al., 2020; Fedus et al., 2022; Du et al., 2022; Dai et al., 2024; Jiang et al., 2024). Building upon this foundation, a variant we refer to as *dynamic MoE*, further introduces token-level dynamism that adjusts the number of activated experts, enabling an input-dependent allocation of computation budgets (Jin et al., 2024; Team et al., 2025; Wu et al., 2025; Guo et al., 2024; Chaudhari et al., 2026; Zeng et al., 2024). Prior studies show that easy tokens can be processed with substantially fewer experts without compromising output quality, making dynamic MoE a principled route to inference-time efficiency (Lu et al., 2024; Jin et al., 2024; Team et al., 2025; Zeng et al., 2024; Huang et al., 2024).

Most existing approaches to dynamic MoE concentrate on either pre-training dynamic MoE models from scratch (Jin et al., 2024; Team et al., 2025; Chaudhari et al., 2026) or adapting a pre-trained base model into a task-specific dynamic MoE (Zeng et al., 2024), leaving the migration of fully trained MoE models largely unexplored. Yet in practical deployment, MoE models have typically undergone an extensive training pipeline encompassing both pre-training and post-training such as supervised fine-tuning (SFT), reinforcement learning (RL), and on-policy distillation (OPD) (Team, 2026; Zeng et al., 2026; DeepSeek-AI, 2026). We refer to such models as *post-trained MoE* throughout this paper. If such a post-trained static MoE model could be converted into a more efficient dynamic counterpart with the architecture and primary training already finalized, the resulting inference savings would be of tremendous practical value given the ever-growing serving costs and demand.

However, directly applying existing dynamic MoE methods to such models risks disrupting the carefully calibrated routing and capability distributions established during the full training pipeline. In this paper, we focus on exploring *whether a post-trained MoE model can be cost-effectively migrated into a more efficient dynamic MoE without sacrificing its established capabilities*.

We introduce **Z**ero-**E**xpert Self-**D**istillation **A**daptation (**ZEDA**), transforming a post-trained MoE model into a dynamic one with faster inference at minimal adaptation cost. ZEDA injects parameterless *zero experts* (Jin et al., 2024; Team et al., 2025), whose outputs are identically zero, into

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

the existing expert pool of a post-trained MoE model. This expands the router candidate pool with zero-computation experts while the activation number remains unchanged, naturally reducing active normal experts. The augmented model is then adapted through a two-stage self-distillation process comprising SFT (Ouyang et al., 2022) and OPD (Gu et al., 2023; Agarwal et al., 2024; Lu & Lab, 2025), using the original MoE as a fixed teacher, to recover performance under the new dynamic routing regime. To make this architectural conversion stable, ZEDA further introduces a Group Auxiliary Loss $\mathcal{L}_{GA}$ that regulates the relative activation frequency between normal experts and zero experts while preserving the learned routing structures among normal experts.

Experiments on Qwen3-30B-A3B (Yang et al., 2025) and GLM-4.7-Flash (Zeng et al., 2025) across 11 benchmarks spanning math, code, and instruction following demonstrate the effectiveness of ZEDA. Our method successfully migrates post-trained MoE models into dynamic ones in less than 31 hours for Qwen and 62 hours for GLM on 8 NVIDIA H200 GPUs. This adaptation eliminates over half of the expert computation and achieves an inference speedup around 20%, while incurring only a marginal accuracy loss compared with the original model. These results demonstrate both the effectiveness and efficiency of the dynamic computation enabled by ZEDA. ZEDA outperforms the strongest baseline by an average of 11.7 points on Qwen and 14.9 points on GLM, and also achieves the best overall performance among our proposed variants. Through detailed illustrative visualizations and analysis, the dynamic characteristics of the zero expert activation and the operating mechanisms of ZEDA are clearly revealed. The following are several key takeaways:

Takeaways

1. ZEDA cost-effectively converts post-trained MoE models into dynamic ones across diverse domains, reducing over half of the MoE computation while maintaining performance (§ **3**).
2. Injecting zero experts and group-level balancing strategy minimize disruptions to original routing distribution, facilitating stable adaptation of post-trained MoE (§ **2**).
3. ZEDA enables adaptive computation allocation based on token-level difficulty signals, including teacher–student distribution gap, model uncertainty, and response patterns (§ **4.1**).

2 METHOD

We propose **Z**ero-**E**xpert Self-**D**istillation **A**daptation (**ZEDA**), a method that transforms a post-trained MoE model into a dynamic one with faster inference at minimal adaptation cost, by augmenting each MoE module with zero experts and adapting the expanded model through self-distillation. In the following, we present the overall adaptation framework in Section 2.1, and then introduce the Group Auxiliary Loss $\mathcal{L}_{GA}$ that regulates zero expert utilization in Section 2.2.

2.1 ADAPTATION FRAMEWORK

ZEDA first injects zero experts (Jin et al., 2024), whose outputs are identically zero, into a post-trained MoE, architecturally converting it into a dynamic one whose activated normal experts number varies across tokens. The augmented model is then adapted via two-stage self-distillation with the original MoE as a fixed teacher, yielding a more efficient dynamic MoE with negligible performance loss.

Zero-Expert Injection. Consider a post-trained MoE model where each MoE module contains N normal experts $\mathcal{E} = \{E_1, \ldots, E_N\}$ and activates K of them per token. For an input hidden state h, the router selects a top-K subset $\mathcal{S}(h) \subseteq \mathcal{E}$ and produces $y(h) = \sum_{i \in \mathcal{S}(h)} g_i(h)\, E_i(h)$, where $g_i(h)$ is the normalized routing weight for expert E_i.

ZEDA introduces N_Z additional experts $\mathcal{Z} = \{Z_1, \ldots, Z_{N_Z}\}$ that satisfy $Z_j(h) = 0$ for all j, referred to as *zero experts*. The augmented expert pool $\mathcal{E}' = \mathcal{E} \cup \mathcal{Z}$ expands the router from N to $N + N_Z$ candidates while the top-K budget remains unchanged. The dynamic MoE output becomes

$$\tilde{y}(h) = \sum_{i \in \tilde{\mathcal{S}}(h) \cap \mathcal{E}} \tilde{g}_i(h)\, E_i(h), \tag{1}$$

where $\tilde{\mathcal{S}}(h)$ denotes the top-K set selected from $\mathcal{E}'$ and $\tilde{g}_i(h)$ is the corresponding routing weight. Because zero experts contribute no computation, selecting them reduces the number of active normal

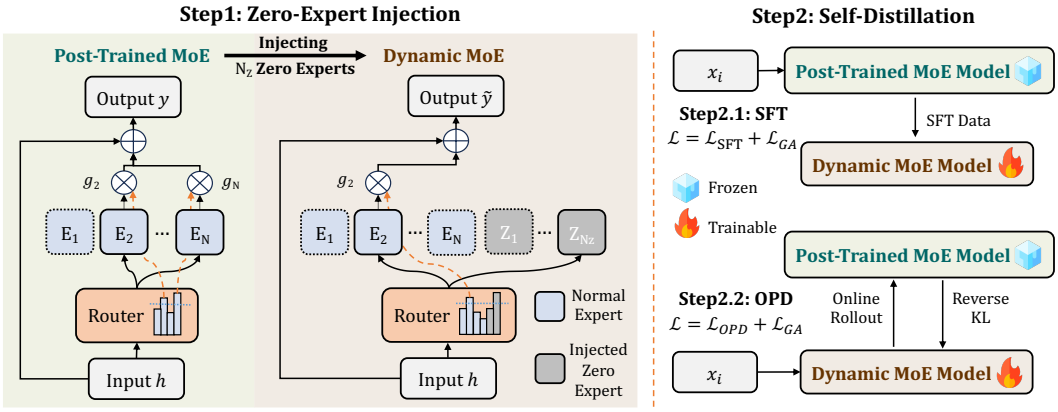

Figure 1: Illustration of ZEDA. ZEDA leverages the post-trained MoE to initialize the dynamic MoE (with zero-expert injection) and further utilizes it as a teacher model for distillation.

experts, yielding token-dependent computation. We also compare the zero expert with another zero-computation alternative, *copy expert*, which outputs its input, in Appendix B.1, showing that copy experts induce both scale and direction mismatches.

For router initialization, the original router parameters for the N normal experts are kept unchanged. The new router parameters corresponding to the N_Z zero experts are drawn from a Gaussian distribution matching the mean and variance of the original router parameters in the same module, preserving the post-trained scale of router logits while inserting new routing options.

Two-Stage Self-Distillation. ZEDA then adapts the augmented model via self-distillation, using the original post-trained MoE as a fixed teacher. The adaptation proceeds in two stages, supervised fine-tuning (SFT) followed by on-policy distillation (OPD). Let π_T denote the teacher (original MoE) distribution, π_θ the student (augmented model) distribution, and $\mathcal{D}$ the prompt set used for adaptation.

- The SFT stage trains π_θ on responses sampled from the teacher π_T. The training loss is:

$$\mathcal{L} = \mathcal{L}_{\text{SFT}} + \mathcal{L}_{\text{GA}} = -\mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_T(\cdot|x)} \left[\sum_{t=1}^{|y|} \log \pi_\theta(y_t \mid x, y_{<t}) \right] + \mathcal{L}_{\text{GA}}, \tag{2}$$

  where x is a prompt from $\mathcal{D}$, $y = (y_1, \ldots, y_{|y|})$ is a teacher-sampled response, and $\mathcal{L}_{\text{GA}}$ is the group auxiliary loss introduced in Section 2.2.
- The subsequent OPD stage (Gu et al., 2023; Agarwal et al., 2024) shifts to on-policy learning, where responses are sampled from the current student π_θ and the teacher evaluates the same trajectories to supply token-level targets. Following Thinking Machines (Lu & Lab, 2025), we cast the sampled-token reverse KL objective as a stop-gradient reward signal and optimize it within the policy optimization framework, yielding the training loss:

$$\mathcal{L} = \mathcal{L}_{\text{OPD}} + \mathcal{L}_{\text{GA}} = -\mathbb{E}_{\substack{x\sim\mathcal{D} \\ y\sim\pi_{\bar{\theta}}(\cdot|x)}} \left[\sum_{t=1}^{|y|} \frac{\pi_\theta(y_t \mid x, y_{<t})}{\pi_{\bar{\theta}}(y_t \mid x, y_{<t})} \, \text{sg}\left[\log \left(\frac{\pi_T(y_t \mid x, y_{<t})}{\pi_{\bar{\theta}}(y_t \mid x, y_{<t})} \right) \right] \right] + \mathcal{L}_{\text{GA}}, \tag{3}$$

  where $\pi_{\bar{\theta}}$ denotes the rollout policy and $\text{sg}[\cdot]$ denotes the stop-gradient operator.

The SFT stage stabilizes the initial transition from a static to a dynamic MoE, and the OPD stage further aligns the student with the teacher under the student's own rollout distribution.

2.2 Group Auxiliary Loss

ZEDA incorporates the Group Auxiliary Loss $\mathcal{L}_{GA}$ to regulate the relative activation frequency between normal experts and zero experts, thereby controlling the zero expert activation ratio r_{ZE}.

Auxiliary Loss. $\mathcal{L}_{GA}$ is derived from the vanilla auxiliary load balancing loss $\mathcal{L}_A$ (Lepikhin et al., 2020; Fedus et al., 2022), which encourages uniform routing across all experts. $\mathcal{L}_A$ is defined through the batch $\mathcal{B}$:

$$\mathcal{L}_A = \alpha \cdot \frac{N + N_Z}{K} \cdot \sum_{i=1}^{N+N_Z} f_i \cdot P_i, \quad \text{where} \quad f_i = \frac{1}{|\mathcal{B}|} \sum_{h \in \mathcal{B}} \mathbb{1}\Big\{ i \in \tilde{\mathcal{S}}(h) \Big\}, P_i = \frac{1}{|\mathcal{B}|} \sum_{h \in \mathcal{B}} \tilde{g}_i(h). \quad (4)$$

α is a scalar loss coefficient. However, applying $\mathcal{L}_A$ directly in ZEDA is problematic. A post-trained MoE model exhibits non-uniform, input-dependent routing patterns over normal experts, and enforcing expert-level uniformity would disrupt these learned distributions, degrading model performance. Appendix B.2 presents a dedicated experiment comparing $\mathcal{L}_A$ and $\mathcal{L}_{GA}$.

Group Load Balancing Loss. The objective of ZEDA is to regulate zero expert utilization while preserving the relative routing structure among normal experts. This motivates a group-level balancing strategy in which the N normal experts form a group $\mathcal{E}$ and the N_Z zero experts form a group $\mathcal{Z}$, with balancing applied only between the two groups. The Group Auxiliary Loss is defined as

$$\mathcal{L}_{GA} = \alpha \cdot \frac{N + N_Z \cdot w}{K} \cdot \left(\frac{f_{\mathcal{E}} \cdot P_{\mathcal{E}}}{N} + \frac{f_{\mathcal{Z}} \cdot P_{\mathcal{Z}}}{N_Z \cdot w} \right), \quad (5)$$

$$\text{where} \quad f_{\mathcal{E}} = \sum_{i \in \mathcal{E}} f_i, \quad P_{\mathcal{E}} = \sum_{i \in \mathcal{E}} P_i, \quad f_{\mathcal{Z}} = \sum_{i \in \mathcal{Z}} f_i, \quad P_{\mathcal{Z}} = \sum_{i \in \mathcal{Z}} P_i. \quad (6)$$

$w > 0$ is the relative weight of the zero-expert group, and a larger w encourages higher r_{ZE}.

Analogously to $\mathcal{L}_A$, minimizing $\mathcal{L}_{GA}$ drives the two groups toward an equilibrium in which the expected number of activated normal experts $K_{\mathcal{E}}$ and zero experts $K_{\mathcal{Z}}$ satisfy $K_{\mathcal{E}} : K_{\mathcal{Z}} = N : N_Z \cdot w$, yielding a target $r_{ZE} = (N_Z \cdot w)/(N + N_Z \cdot w)$. Since the constraint is imposed only at the group level, it does not explicitly flatten the routing distribution within the normal-expert group, which makes it better aligned with post-trained MoE adaptation. $\mathcal{L}_{GA}$ drives r_{ZE} toward the target value, while the other loss component ($\mathcal{L}_{\text{SFT}}$ or $\mathcal{L}_{\text{OPD}}$) optimizes performance. Under the joint influence, the model reaches a trade-off, causing r_{ZE} to converge to an appropriate value.

3 Experiments

3.1 Experimental Setup

Models. To evaluate the generalizability of ZEDA across different backbone architectures, two post-trained MoE models are selected: Qwen3-30B-A3B (Yang et al., 2025) and GLM-4.7-Flash (Zeng et al., 2025). Qwen3-30B-A3B is in Thinking mode throughout all experiments. Two models differ in scale and expert configuration. Qwen3-30B-A3B contains N=128 normal experts with K=8 activated per token, while GLM-4.7-Flash has N=64 and K=4. Following LongCat (Team et al., 2025), the number of injected zero experts N_Z is set to 64 and 32 for Qwen and GLM, respectively.

Evaluation Setup. To comprehensively assess performance, ZEDA is evaluated on 11 benchmarks spanning 3 categories. For math reasoning, the benchmarks include AIME 24, AIME 25, AIME 26 (Li et al., 2024), GSM8K (Cobbe et al., 2021), and MATH-500 (Lightman et al., 2023). For code generation, the benchmarks include LiveCodeBench v5 (LCB v5), LiveCodeBench v6 (LCB v6) (Jain et al., 2024), HumanEval+ (Liu et al., 2023), and MBPP+ (Liu et al., 2023). For instruction following, the benchmarks include IFEval (Zhou et al., 2023) and IFBench (Pyatkin et al., 2025). We report avg@32 for AIME 24, AIME 25, and AIME 26 to reduce variance on these small-scale competition benchmarks, avg@8 for the 4 coding benchmarks, and avg@1 for all remaining benchmarks.

Implementation Details. For the inference efficiency of the adapted dynamic MoE, w in $\mathcal{L}_{GA}$ (Eq. (5)) is set to 2, which drives the target r_{ZE} toward 50%, and the loss coefficient α is set to 0.1. Ablation studies on w and α are presented in Section 4.3 and Section B.7. The self-distillation data contains 60k prompts in total. It consists of 17k math prompts and 15k coding prompts randomly sampled from NVIDIA AceReason-1.1-SFT (Liu et al., 2025), together with 28k chat prompts randomly sampled from NVIDIA Llama-Nemotron-Post-Training-Dataset (Bercovich et al., 2025).

Baselines. AdaMoE (Zeng et al., 2024) and Dynamic Skipping (Lu et al., 2024) serve as dynamic-routing baselines. Both are reproduced using the same 60k-prompt corpus as ZEDA. AdaMoE uses the same training hyperparameters as ZEDA, whereas Dynamic Skipping follows its original adaptation procedure. We further propose three variants to evaluate the efficacy of ZEDA's components. ZEDA_{SFT}, which applies only the SFT stage of ZEDA, is included to isolate the contribution of OPD. To validate the dynamic expert selection mechanism, we propose Naive Expert Truncation (NET), a straightforward variant of ZEDA that directly halves the number of activated experts in the original MoE model. NET is combined with SFT alone or SFT followed by OPD, yielding NET_{SFT} and $\text{NET}_{\text{SFT}\to\text{OPD}}$, respectively. More experimental setup details are reported in Appendix C.

3.2 Main Results

Performance. Table 1 summarizes the performance of all methods on 11 benchmarks spanning mathematical reasoning, code generation, and instruction following. Compared with the original post-trained MoE, ZEDA incurs only a marginal average accuracy loss while eliminating over half of the expert computation, and even surpasses the original model on several individual benchmarks such as IFBench, demonstrating the practical utility of the dynamic MoE models produced by ZEDA. Among all baselines, ZEDA achieves the highest average evaluation scores on both Qwen3-30B-A3B and GLM-4.7-Flash, indicating its effectiveness and robustness across architectures. Furthermore, ZEDA achieves superior overall performance over all three variants, ZEDA_{SFT}, NET_{SFT} and $\text{NET}_{\text{SFT}\to\text{OPD}}$, demonstrating the contributions of OPD and the dynamic expert selection mechanism. Moreover, the dynamic routing baselines exhibit severe capability imbalances, where AdaMoE collapses on hard reasoning like AIME 24 and Dynamic Skipping fails on code generation. ZEDA is the only method preserving competitive performance uniformly across all domains. Finally, ZEDA achieves average r_{ZE} values of 51.2% on Qwen and 53.0% on GLM, exceeding or matching the baselines, indicating that ZEDA attains better performance with comparable or lower computation.

Table 1: Performance of ZEDA and baselines on Qwen3-30B-A3B and GLM-4.7-Flash.

Method	Avg Acc	Avg r_{ZE}	Math					Code				IF	
			AIME 24	AIME 25	AIME 26	GSM8K	MATH-500	LCB v5	LCB v6	HumanEval+	MBPP+	IFBench	IFEval
Qwen3-30B-A3B	74.9	0.0	80.9	71.0	72.3	95.4	94.4	61.5	57.1	85.6	79.2	39.7	86.3
AdaMoE	54.8	51.9	25.0	24.8	36.7	92.4	79.8	36.1	34.3	80.3	72.8	38.7	82.4
Dynamic Skipping	62.5	49.0	74.4	58.2	63.8	91.6	92.8	54.5	46.0	51.2	63.6	32.0	59.5
NET_{SFT}	72.3	50.0	76.8	65.7	72.1	94.7	94.0	56.5	50.9	86.7	78.2	37.7	82.4
$\text{NET}_{\text{SFT}\to\text{OPD}}$	73.0	50.0	**79.5**	67.6	70.6	95.4	94.6	57.0	52.9	87.4	77.6	38.7	81.7
ZEDA_{SFT}	73.3	51.5	78.1	66.3	71.3	94.8	94.4	**58.2**	52.8	86.8	**78.6**	39.7	**85.2**
ZEDA	**74.2**	51.2	79.0	**69.1**	**72.5**	**95.5**	**95.2**	**58.2**	**53.2**	**88.5**	78.2	**42.3**	84.3
GLM-4.7-Flash	74.6	0.0	84.2	76.5	74.0	95.2	96.4	48.1	44.4	89.0	75.7	53.3	83.5
AdaMoE	59.0	47.0	44.1	42.4	47.3	93.9	86.4	26.4	28.6	82.7	69.5	47.7	80.0
Dynamic Skipping	57.2	46.0	64.0	61.2	65.4	71.4	78.4	39.3	36.6	66.3	47.0	42.7	56.6
NET_{SFT}	72.4	50.0	78.2	71.4	68.3	94.1	95.0	50.6	44.7	86.8	74.2	51.3	81.9
$\text{NET}_{\text{SFT}\to\text{OPD}}$	72.8	50.0	78.7	71.8	72.9	94.2	**95.6**	49.5	43.8	87.1	**74.3**	51.7	80.8
ZEDA_{SFT}	72.9	52.8	78.1	71.4	71.9	**95.2**	95.0	49.9	45.0	**88.1**	72.3	52.7	82.4
ZEDA	**73.9**	53.0	**79.8**	**73.1**	**74.4**	94.4	95.2	**51.6**	**45.6**	86.3	73.5	**53.7**	**85.4**

We conduct more experiments to further substantiate ZEDA's effectiveness. Training-stage ablations (Appendix B.3) support the necessity of OPD and the rationale for the SFT→OPD pipeline. Appendix B.4 shows no average accuracy gain from identical self-distillation on the unmodified model, suggesting that ZEDA's capability recovery is not merely due to extra training or data. Repeated runs (Appendix B.5) support the statistical reliability of ZEDA's gains.

Adaptation Time. Table 2 reports the training time of the ZEDA pipeline. ZEDA requires less than 31 hours for Qwen3-30B-A3B and 62 hours for GLM-4.7-Flash on 8 H200 GPUs, which is negligible compared with prior MoE pre-training and post-training costs, demonstrating its cost-effectiveness.

Table 2: Adaptation time (hours) of Qwen3-30B-A3B and GLM-4.7-Flash on 8 NVIDIA H200 GPUs

Qwen3-30B-A3B	All	SFT Data Rollout	SFT	OPD	GLM-4.7-Flash	All	SFT Data Rollout	SFT	OPD
	30.12	8.16	1.97	19.99		61.37	14.56	4.51	42.30

3.3 Inference Efficiency

ZEDA yields average zero-expert activation ratios (r_{ZE}) of 51.2% and 53.0% on Qwen and GLM respectively, effectively halving expert-level computation. We further demonstrate the practical inference speedups achieved by the resulting dynamic MoE. Inference efficiency is evaluated by comparing the original model with its ZEDA-adapted counterpart at 8192 sequence length, using SGLang (Zheng et al., 2024) as the inference framework. This controlled comparison is representative because ZEDA distills from the original MoE and produces similar response lengths in practice. We measure both prefill and decode efficiency in single-GPU and four-GPU expert-parallel settings, with the latter using DeepEP (Zhao et al., 2025) and an expert-parallel degree of 4 (EP=4). We also provide the theoretical analysis of inference efficiency in Appendix D.

As shown in Figure 2, ZEDA delivers consistent inference gains across both backbone models and deployment settings. On a single GPU, it achieves prefill speedups of 18% and 26% on Qwen and GLM, respectively, and approximately 19% decode speedup on both models. With DeepEP EP=4, the prefill gains remain substantial at 21% and 24%, while decode throughput improves by 16% and 18%, respectively. These results show that ZEDA's efficiency gains carry over from single-GPU inference to multi-GPU expert-parallel deployments.

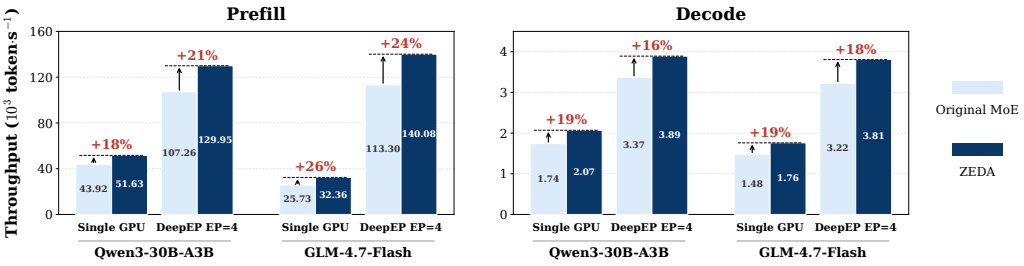

Figure 2: Prefill and decode throughput of the original MoE and ZEDA at 8192 sequence length, on a single GPU and with four-GPU expert parallelism (DeepEP EP=4). Speedup is defined relative to the original MoE under the same deployment setting.

4 Analysis

We provide a detailed analysis of the dynamic characteristics of zero expert activation (§ 4.1), the effects of different adaptation durations (§ 4.2) and target r_{ZE} values (§ 4.3), and ZEDA's training dynamics (§ 4.4). Additional analyses focusing on the impact of router probability renormalization and an ablation study of α are presented in Appendix B.6, B.7, respectively.

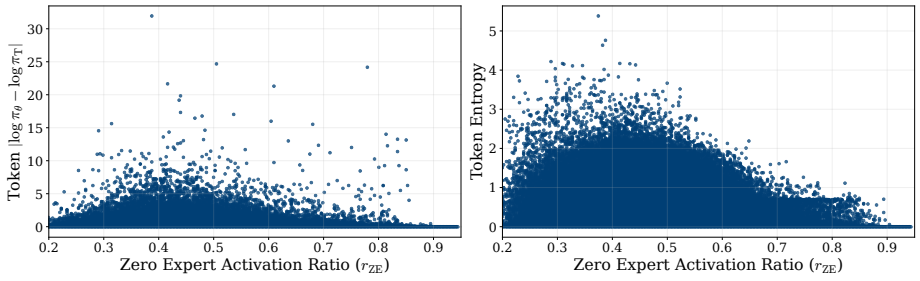

Figure 3: Distribution of token-level r_{ZE} versus teacher-student logp-diff (left) and entropy (right) for all generated tokens across 110 rollouts. Each point represents a single token.

4.1 Zero Expert Activation Dynamics

ZEDA transforms a static MoE model into a dynamic one in which different tokens exhibit different r_{ZE} values, corresponding to varying computation amounts. This section provides a deeper investiga-

tion into this token-level dynamism, using Qwen3-30B-A3B. The analysis examines how r_{ZE} relates to distillation signals, response patterns, layer-wise behavior, and task difficulty, aiming to establish connections between computation allocation in the dynamic MoE and other interpretable metrics.

Teacher-Student Logp-Diff and Entropy. To analyze factors affecting token-level r_{ZE}, 110 prompts (10 per benchmark) are sampled and decoded with the ZEDA-adapted model. For each generated token, we record the student log probability $\log \pi_\theta(y_t \mid x, y_{<t})$ and entropy, and compute the teacher log probability on the same token to obtain the teacher-student logp-diff Δ_{logp}. Figure 3 visualizes all tokens from the 110 prompts. Tokens with larger Δ_{logp} or higher entropy tend to have lower r_{ZE}, clustering in the upper-left. The dynamic MoE intrinsically allocates more computation, i.e., activates fewer zero experts, when the teacher-student distributional gap or model uncertainty is larger, both of which indicate greater token-level difficulty (Ko et al., 2026; Wang et al., 2025).

Response Pattern. Aligning per-token r_{ZE} of the 110 sampled responses with the decoded text reveals a clear relationship between r_{ZE} and the response pattern. Figure 4 presents 3 representative examples. Compared with natural text, code fragments and mathematical expression exhibit notably higher r_{ZE}, indicating that the model intrinsically assigns less computation to these structured segments. Since math and code rollouts often contain many such segments after the thinking process, their average r_{ZE} tends to increase toward the response end, while instruction-following responses show a more uniform r_{ZE} distribution, as illustrated in Figure 5.

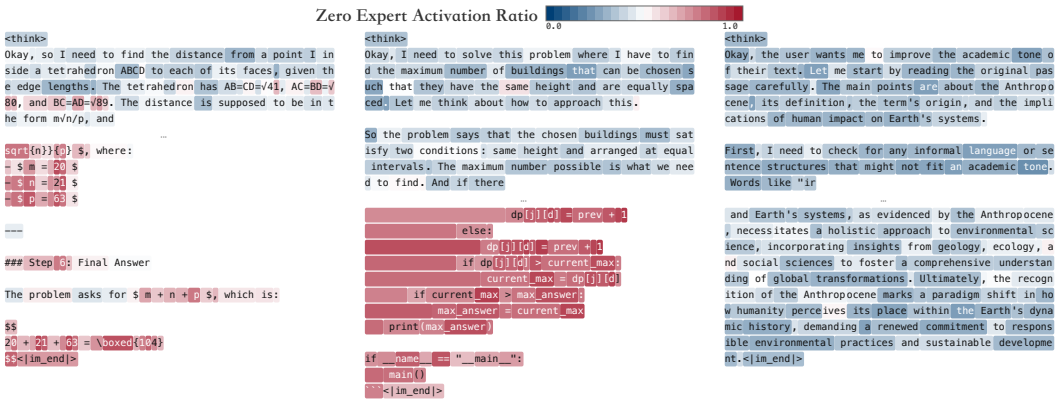

Figure 4: Visualization of per-token r_{ZE} for decoded text, showing one sampled response from AIME 24 (left), LiveCodeBench v5 (middle), and IFBench (right), respectively. Due to space constraints, only the first and last 80 tokens of each response are retained.

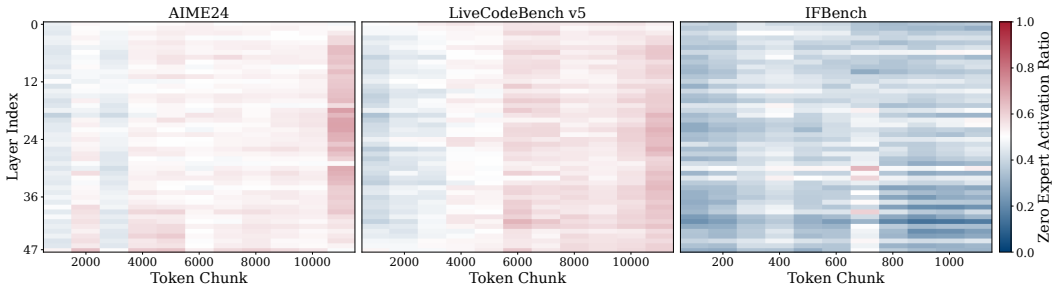

Figure 5: Visualization of r_{ZE} across layers and response positions, using the same data in Figure 4. Token-level r_{ZE} values are averaged over chunks of size 1000 (for AIME 24 and LiveCodeBench v5) and 100 (for IFBench), respectively. The last chunk is averaged over its actual token number.

Layer. For each of the 110 responses, r_{ZE} on the 48 MoE layers of the dynamic model is computed. Figure 5 presents the layer-wise r_{ZE} distributions for 3 representative cases. Although minor variations exist across layers, the differences are relatively small and exhibit no systematic pattern.

Human-Perceived Task-Level Difficulty. The relationship between r_{ZE} and task difficulty is further investigated. Table 3 reports the r_{ZE} and performance of ZEDA on MATH-500, which provides human-annotated difficulty levels, and on AIME 24, a task generally considered more challenging. ZEDA achieves comparable r_{ZE} across all five difficulty levels of MATH-500 and AIME 24, suggesting r_{ZE} is largely independent of task difficulty. Notably, ZEDA preserves the model's sensitivity to human-perceived task-level difficulty and its ability to adjust computational effort accordingly, as reflected in its response length. As shown in Table 3, the average number of response tokens increases with task difficulty.

Table 3: Performance, r_{ZE} and average number of response tokens of ZEDA across five difficulty level tasks of MATH-500 and AIME 24.

	MATH-500					**AIME 24**
	Level 1	**Level 2**	**Level 3**	**Level 4**	**Level 5**	
Performance (r_{ZE})	95.3 (51.1)	95.6 (51.8)	97.1 (52.2)	94.5 (52.5)	94.0 (52.5)	79.0 (52.1)
Avg. Response Length (tokens)	2720.1	3034.5	4123.6	5279.4	8337.7	15404.1

4.2 Effect of Adaptation Cost

To study the scaling trend of zero expert adaptation, we extend the SFT experiments to larger datasets and track the average benchmark score and r_{ZE} throughout the SFT stage. As illustrated in Figure 6 (left), both metrics exhibit similar evolution as the amount of SFT data and GPU hours increases, which is a rapid initial ascent followed by convergence at approximately 60k prompts. This trend indicates that the majority of useful adaptation happens relatively early, when the router is learning how to incorporate the injected zero experts while preserving the backbone's original capabilities. After this point, additional supervised adaptation mainly provides incremental refinements.

These empirical results justify our 60k self-distillation prompt setting, as this scale achieves a performance plateau and stable routing patterns. The observed "dual saturation" underscores the sample efficiency of ZEDA, demonstrating that the modified architecture can reach a stable, high-performance state with affordable post-training costs. The SFT plateau motivates switching to a different training stage, and Appendix B.3 further shows that OPD yields additional gains under a comparable budget, supporting the necessity of the subsequent OPD stage.

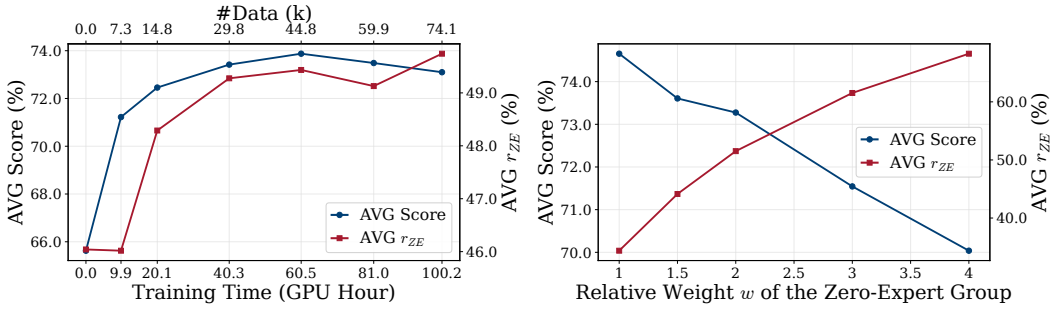

Figure 6: Scaling trend of average benchmark score and r_{ZE} over different amounts of SFT data and adaptation time (left). Effect of the zero-expert group weight on average score and r_{ZE} (right).

4.3 Effect of w and r_{ZE}

To investigate the impact of group-level balancing strength and the zero expert activation ratio, we vary the zero-expert group weight w and analyze its effect on model performance and routing. As shown in Figure 6 (right), increasing w monotonically elevates r_{ZE} but leads to a gradual decline in benchmark scores. This confirms that w serves as an effective control knob for the quality-efficiency trade-off in ZEDA. Empirical results indicate that w=2 offers the optimal balance, yielding significantly higher zero-expert utilization than $w\in\{1, 1.5\}$ while maintaining competitive performance. In contrast, pushing w further to 3 or 4 causes a more pronounced accuracy drop. Consequently, we select w=2 as the preferred operating point.

4.4 Training Dynamics

Figure 7 presents the training dynamics of Qwen3-30B-A3B during SFT. The average r_{ZE} rises toward the 50% target and then fluctuates slightly. Meanwhile, $\mathcal{L}_{SFT}$ decreases rapidly at first and more gradually thereafter, whereas the unweighted group-balancing term ($\mathcal{L}_{GA}/\alpha$) reaches its equilibrium early. These trends indicate that SFT first establishes a stable expert-computation budget and then continues optimizing the distillation objective without materially changing the average r_{ZE}. During the subsequent OPD stage, we observe that r_{ZE} exhibits only minor fluctuations, indicating that OPD further refines the model while preserving the routing equilibrium established by SFT.

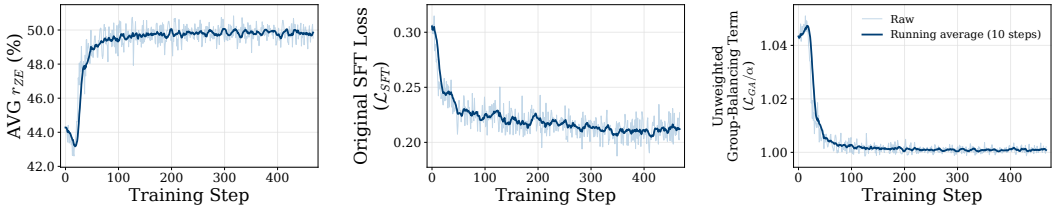

Figure 7: Training dynamics of Qwen3-30B-A3B during the SFT stage.

5 Related Work

Dynamic Expert Activation in MoE LLMs. Mixture of Experts (MoE) LLMs improve the capacity-efficiency trade-off through sparse expert activation (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022; Du et al., 2022; Zoph et al., 2022; Dai et al., 2024; Jiang et al., 2024), yet typically allocate a fixed top-k budget to every token. Prior work either reduces redundant computation through pruning, merging, compression, or skipping (Lu et al., 2024; Liu et al., 2024; Li et al., 2023; Chen et al., 2024; 2025; Zhang et al., 2025; Hao et al., 2026), or enables variable-size activation via null experts, learned token-dependent k, threshold routing, or dynamic expert sets (Zhou et al., 2022; Huang et al., 2024; Zeng et al., 2024; Yue et al., 2024; Guo et al., 2024; Sun et al., 2026; Jin et al., 2024). Most closely related, MoE++ (Jin et al., 2024) introduces zero-computation experts, a paradigm also validated at scale by LongCat Flash (Team et al., 2025). Unlike these pretraining-time approaches, ZEDA efficiently converts a well post-trained static MoE into a dynamic one, without repeating prohibitively expensive training or substantial router redesign.

Self-Distillation. Knowledge distillation has evolved from teacher–student transfer to sequence-level, autoregressive, rationale-augmented, and on-policy variants, with MiniLLM and GKD mitigating off-policy train–test mismatch using student-sampled trajectories (Hinton et al., 2015; Kim & Rush, 2016; Sanh et al., 2019; Hsieh et al., 2023; Gu et al., 2023; Agarwal et al., 2024; Lu & Lab, 2025). In parallel, self-distillation is effective without external teachers but has mainly targeted reasoning and continual learning (Furlanello et al., 2018; Zhang et al., 2019; Zhao et al., 2026; Shenfeld et al., 2026; Hübotter et al., 2026). Its use for efficient architecture adaptation remains limited to static transformations (Hoshino et al., 2025; Chen et al., 2026; Kim et al., 2024). In contrast, we use on-policy self-distillation to adapt a well post-trained MoE into a dynamic one and reduce redundant expert activation during adaptation, filling a critical gap in current studies on efficient MoE adaptation. More detailed related work is deferred to Appendix A.

6 Conclusion

This study presents ZEDA, a lightweight and effective framework for migrating post-trained static MoE models to dynamic ones through zero-expert injection and two-stage self-distillation. With the group auxiliary loss, ZEDA regulates computation allocation while preserving the delicate routing distributions of the original MoE. Empirical evaluations across multiple architectures and benchmarks demonstrate that ZEDA eliminates over half the expert computation and provides significant inference speedups with negligible impact on model performance. These findings validate that post-trained MoE models can be adapted to efficient dynamic ones via self-distillation, offering a practical solution for enhancing the deployment efficiency of large-scale MoE systems across diverse domains.

AI USE STATEMENT

In this work, we used generative AI tools for polishing the paper text. We have not used generative AI tools for the conceptualization of the research topic, design of methodology, planning and execution of experiments, analysis of results, etc., and the remaining required-disclosure tasks are not applicable to this work. We have reviewed all AI-assisted work. We cross-checked all AI-assisted text against the experimental records and cited sources. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This study trains and evaluates language models on publicly available reasoning data and benchmarks. No human subjects or personal data are used. We think our work doesn't introduce any ethical concerns.

REPRODUCIBILITY STATEMENT

We provide the implementation code and scripts of ZEDA and main baselines in the Supplementary Material for reproduction. And we also specify the experimental configurations and hyperparameters in detail in Section 3.1 and Appendix C. All experiments use publicly available base models and data.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The twelfth international conference on learning representations*, 2024.

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, 2023.

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, et al. Llama-nemotron: Efficient reasoning models. *arXiv preprint arXiv:2505.00949*, 2025.

Marmik Chaudhari, Idhant Gulati, Nishkal Hundia, Pranav Karra, and Shivam Raval. Moe lens–an expert is all you need. *arXiv preprint arXiv:2603.05806*, 2026.

I Chen, Hsu-Shen Liu, Wei-Fang Sun, Chen-Hao Chao, Yen-Chang Hsu, Chun-Yi Lee, et al. Retraining-free merging of sparse moe via hierarchical clustering. *arXiv preprint arXiv:2410.08589*, 2024.

Yingfa Chen, Zhen Leng Thai, Zihan Zhou, Zhu Zhang, Xingyu Shen, Shuo Wang, Chaojun Xiao, Xu Han, and Zhiyuan Liu. Hybrid linear attention done right: Efficient distillation and effective architectures for extremely long contexts. *arXiv preprint arXiv:2601.22156*, 2026.

Yuanteng Chen, Yuantian Shao, Peisong Wang, and Jian Cheng. Eac-moe: Expert-selection aware compressor for mixture-of-experts large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12942–12963, 2025.

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

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1280–1297, 2024.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

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

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

Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *International conference on machine learning*, pp. 1607–1616. PMLR, 2018.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*, 2023.

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.

Jiawei Hao, Zhiwei Hao, Jianyuan Guo, Li Shen, Yong Luo, Han Hu, and Dan Zeng. Lightmoe: Reducing mixture-of-experts redundancy through expert replacing. *arXiv preprint arXiv:2603.12645*, 2026.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Yuichiro Hoshino, Hideyuki Tachibana, Muneyoshi Inahara, and Hiroto Takegawa. Rad: Redundancy-aware distillation for hybrid models via self-speculative decoding. *arXiv preprint arXiv:2505.22135*, 2025.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 8003–8017, 2023.

Quzhe Huang, Zhenwei An, Nan Zhuang, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. Harder task needs more experts: Dynamic routing in moe models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12883–12895, 2024.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

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*, 2024.

Sungyoon Kim, Youngjun Kim, Kihyo Moon, and Minsung Jang. Ladimo: Layer-wise distillation inspired moefier. *arXiv preprint arXiv:2408.04278*, 2024.

Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 1317–1327, 2016.

Jongwoo Ko, Sara Abdali, Young Jin Kim, Tianyi Chen, and Pashmina Cameron. Scaling reasoning efficiently via relaxed on-policy distillation. *arXiv preprint arXiv:2603.11137*, 2026.

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

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9, 2024.

Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. Merge, then compress: Demystify efficient smoe with hints from its routing policy. *arXiv preprint arXiv:2310.01334*, 2023.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The twelfth international conference on learning representations*, 2023.

Enshu Liu, Junyi Zhu, Zinan Lin, Xuefei Ning, Matthew B Blaschko, Shengen Yan, Guohao Dai, Huazhong Yang, and Yu Wang. Efficient expert pruning for sparse mixture-of-experts language models: Enhancing performance and reducing inference costs. *arXiv preprint arXiv:2407.00945*, 2024.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 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, 2023.

Zihan Liu, Zhuolin Yang, Yang Chen, Chankyu Lee, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Acereason-nemotron 1.1: Advancing math and code reasoning through sft and rl synergy. *arXiv preprint arXiv:2506.13284*, 2025.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

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. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6159–6172, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. *arXiv preprint arXiv:2507.02833*, 2025.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

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

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Hanchi Sun, Yixin Liu, Yonghui Wu, and Lichao Sun. Expert threshold routing for autoregressive language modeling with dynamic computation allocation and load balancing. *arXiv preprint arXiv:2603.11535*, 2026.

Meituan LongCat Team, Bei Li, Bingye Lei, Bo Wang, Bolin Rong, Chao Wang, Chao Zhang, Chen Gao, Chen Zhang, Cheng Sun, et al. Longcat-flash technical report. *arXiv preprint arXiv:2509.01322*, 2025.

Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

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.

Haoyuan Wu, Haoxing Chen, Xiaodong Chen, Zhanchao Zhou, Tieyuan 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.

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.

Tongtian Yue, Longteng Guo, Jie Cheng, Xuange Gao, Hua Huang, and Jing Liu. Ada-k routing: Boosting the efficiency of moe-based llms. In *The Thirteenth International Conference on Learning Representations*, 2024.

Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, et al. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *arXiv preprint arXiv:2508.06471*, 2025.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

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

Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3713–3722, 2019.

Zeliang Zhang, Xiaodong Liu, Hao Cheng, Chenliang Xu, and Jianfeng Gao. Diversifying the expert knowledge for task-agnostic pruning in sparse mixture-of-experts. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 86–102, 2025.

Chenggang Zhao, Shangyan Zhou, Liyue Zhang, Chengqi Deng, Zhean Xu, Yuxuan Liu, Kuai Yu, Jiashi Li, and Liang Zhao. Deepep: an efficient expert-parallel communication library. https://github.com/deepseek-ai/DeepEP, 2025.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583, 2024.

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

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

Zilin Zhu, Chengxing Xie, Xin Lv, and slime Contributors. slime: An llm post-training framework for rl scaling. https://github.com/THUDM/slime, 2025. GitHub repository. Corresponding author: Xin Lv.

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

A Extended Related Work

A.1 Dynamic Expert Activation in Mixture-of-Experts LLMs

Mixture-of-Experts (MoE) has emerged as an effective architecture for scaling large language models by increasing model capacity while keeping bounded per-token computation (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022; Du et al., 2022). Subsequent studies (Zoph et al., 2022; Dai et al., 2024; Jiang et al., 2024) further improved sparse expert training and specialization, making MoE a practical design for large-scale language modeling. Nevertheless, in standard MoE architectures, routing is typically constrained by a fixed top-k policy, meaning that while expert activation is input-dependent, the computation budget remains largely static across tokens.

To address this limitation, prior work has mainly proceeded along two directions. One line of work improves efficiency by reducing expert redundancy or activated computation, and can be further categorized into expert pruning (Lu et al., 2024; Liu et al., 2024), merging (Li et al., 2023; Chen et al., 2024) and compression (Li et al., 2023; Chen et al., 2025; Zhang et al., 2025; Hao et al., 2026). The other direction replaces the static top-k routing policy with dynamic expert activation, enabling token-level input-dependent allocation of computation budgets (Lu et al., 2024; Jin et al., 2024; Zhou et al., 2022; Huang et al., 2024; Zeng et al., 2024; Yue et al., 2024; Guo et al., 2024; Sun et al., 2026). Early work relaxed the fixed-cardinality assumption by allowing the number of activated experts to vary across tokens, either through expert-selected token assignment (Zhou et al., 2022) or by allocating more experts to harder inputs (Huang et al., 2024). More recent work adapts these methods to modern autoregressive MoE language models: AdaMoE (Zeng et al., 2024) introduces null experts so that the number of real activated experts can vary with minimal changes to standard routing, Ada-K Routing (Yue et al., 2024) explicitly learns a token-dependent k for expert routing, DynMoE (Guo et al., 2024) jointly auto-tunes both the total number of experts and the per-token activation budget, and Expert Threshold Routing (Sun et al., 2026) replaces fixed top-k selection with threshold-based activation to obtain causal variable-size expert sets with improved load balancing, MoE++ (Jin et al., 2024) extends dynamic routing into dynamic computation-path selection by introducing zero-computation experts, which allow some tokens to bypass expensive FFN computation. Dynamic activation can also be introduced from a deployment perspective via inference-time expert skipping (Lu et al., 2024), where selected experts are conditionally bypassed at inference time without fundamentally changing the underlying router.

In contrast to these approaches, we study a lower-cost form of dynamic expert activation for well-post-trained MoE models, without relying on expensive retraining or substantial router redesign. Our method operates entirely in the post-training regime and follows the zero-computation expert paradigm of MoE++ (Jin et al., 2024), which has also been validated at industrial scale in Meituan's LongCat-Flash (Team et al., 2025). This design avoids substantive architectural modifications to the underlying MoE model, making it particularly appealing for practical adaptation and deployment.

A.2 Self-Distillation

Knowledge distillation (KD) was originally introduced as a teacher–student framework in which a student network learns from the softened output distribution of a stronger teacher network (Hinton et al., 2015). This paradigm had already been broadly extended to language modeling, from sequence-level distillation in neural text generation (Kim & Rush, 2016), supervised distillation in autoregressive language models (Sanh et al., 2019) to more recent rationale-augmented distillation with large language models (Hsieh et al., 2023). More recently, on-policy distillation methods for language models argued that standard distillation is inherently off-policy, since the student is trained on teacher-generated trajectories but tested on its own generations. To reduce this mismatch, methods such as MiniLLM (Gu et al., 2023) and GKD (Agarwal et al., 2024) apply teacher supervision on student-sampled sequences, a perspective that has also been highlighted in recent practitioner discussions of language model post-training (Lu & Lab, 2025). In parallel, self-distillation has been shown to improve performance even without an external teacher (Furlanello et al., 2018; Zhang et al., 2019). More recent work has further explored on-policy self-distillation and demonstrated its potential in scenarios such as reasoning and continual learning (Zhao et al., 2026; Shenfeld et al., 2026; Hübotter et al., 2026).

Table 4: Comparison of two zero-compute-expert types, copy expert and zero expert, on mathematical reasoning benchmarks.

Method	Avg Acc	Avg r_{ZCE}	Math				
			AIME 24	AIME 25	AIME 26	GSM8K	MATH-500
Qwen3-30B-A3B	82.8	0.0	80.9	71.0	72.3	95.4	94.4
Copy Expert + $\mathcal{L}_{GA}$	20.7	53.2	1.0	2.9	0.8	58.8	40.0
Zero Expert + $\mathcal{L}_{GA}$	81.0	52.7	78.1	66.2	71.3	94.8	94.4

Beyond capability improvement and task-specific adaptation, recent work has also examined self-distillation in the context of architecture adaptation towards higher computational efficiency. RAD (Hoshino et al., 2025) and HALO (Chen et al., 2026) utilize self-distillation as a principled mechanism to transform standard full-attention layers into computationally efficient alternatives, thereby achieving substantial gains in inference efficiency while maintaining model performance. LaDiMo (Kim et al., 2024) employs layer-wise distillation to transform dense models into sparse MoE architectures, facilitating efficient sparse architecture adaptation. Nevertheless, existing efforts have primarily focused on static architecture conversion, with limited attention to using self-distillation for efficient dynamic MoE architectures. In particular, the introduction of on-policy self-distillation to reduce redundant expert activation in MoE models remains underexplored, representing a critical gap in the current literature.

B Extended Analysis and Ablation Studies

B.1 Zero Experts versus Copy Experts

An alternative to the zero expert is the *copy expert*, whose output equals its input, carrying negligible computational cost. In this section, we establish that zero expert is the preferable design for adapting a post-trained MoE model to a dynamic architecture.

Compared with zero experts, copy experts introduce stronger perturbations to the original model. Assuming the same router parameterization, the output of a dynamic MoE module with zero experts and that with copy experts are

$$\tilde{y}_{\text{zero}} = \sum_{i \in \tilde{\mathcal{S}} \cap \mathcal{E}} \tilde{g}_i(h)\, E_i(h), \quad \tilde{y}_{\text{copy}} = \tilde{y}^{\text{norm}}_{\text{copy}} + \tilde{y}^{\text{cp}}_{\text{copy}} = \sum_{i \in \tilde{\mathcal{S}} \cap \mathcal{E}} \tilde{g}_i(h)\, E_i(h) + \sum_{j \in \tilde{\mathcal{S}} \cap \mathcal{Z}} \tilde{g}_j(h)\, h, \quad (7)$$

where $\tilde{y}^{\text{norm}}_{\text{copy}}$ and $\tilde{y}^{\text{cp}}_{\text{copy}}$ are the normal expert component and copy component of the copy-expert model, respectively. Zero experts implement true expert omission, whereas copy experts incur an additional term $\sum_{j \in \tilde{\mathcal{S}}(h) \cap \mathcal{Z}} \tilde{g}_j(h)\, h$ rather than a no-op.

To study the effect of the zero-compute-expert type, we compare two post-training adaptation variants on Qwen3-30B-A3B that are identical in training data, SFT recipe, and routing regularization, differing only in whether the inserted zero-compute experts are instantiated as copy experts or zero experts. For both variants, the routing regularizer is the group auxiliary loss $\mathcal{L}_{GA}$ in Eq. (5) with coefficient α=0.1 and group weight w=2.0, matching the setting used in Section 3.1. We report the average accuracy, the average activation ratio of the inserted zero-compute experts r_{ZCE}, and the performance on five mathematical reasoning benchmarks. Table 4 shows that the copy-expert type performs substantially worse than the zero-expert type despite nearly identical activation ratios (53.2% vs. 52.7%). While the zero-expert type largely preserves the mathematical reasoning ability of the original model, the copy-expert type leads to severe performance degradation across all five benchmarks. The gap is particularly pronounced on the more challenging AIME tasks, where the copy-expert type achieves only 1.0, 2.9, and 0.8 on AIME 24, AIME 25, and AIME 26, respectively. These empirical results suggest that, for post-training adaptation of a post-trained MoE model toward dynamic computation, the zero-expert type is substantially more suitable than the copy-expert type.

We extract the hidden-state outputs of the MoE blocks and conduct the following experiments to further analyze why copy experts are harmful from two complementary perspectives:

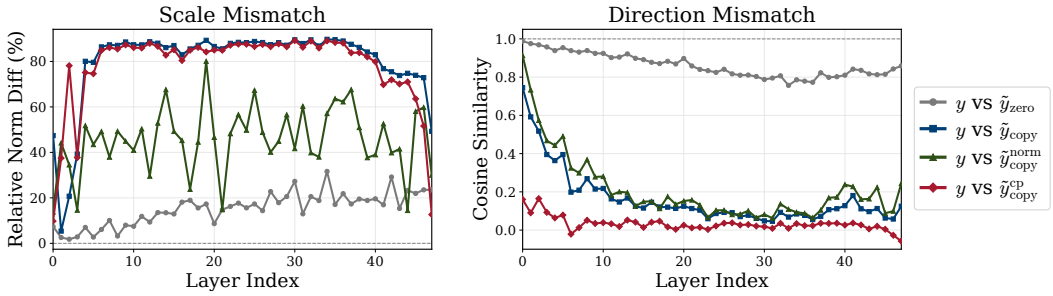

Figure 8: Layer-wise comparison between the original MoE output y and four variants: $\tilde{y}_{\text{zero}}$, $\tilde{y}_{\text{copy}}$, $\tilde{y}_{\text{copy}}^{\text{norm}}$ and $\tilde{y}_{\text{copy}}^{\text{cp}}$, including relative absolute L2-norm difference (left) and cosine similarity (right).

- **Scale Mismatch.** The left panel of Figure 8 shows that the full copy expert output has a much larger norm mismatch with the original output than the zero expert output does. Decomposition further shows that the normal expert component is less mismatched than the copy component, but still consistently more mismatched than the zero-expert output. This indicates that the copy component is the main source of scale mismatch, and also pulls the normal expert component away from the original scale.
- **Direction Mismatch.** The right panel of Figure 8 shows the same trend in direction space. The zero expert output remains well aligned with the original output, whereas the full copy expert output shows a clear directional mismatch. The copy component stays strongly misaligned across layers, while the mismatch of the normal expert component grows from shallow to deep layers. This suggests that the copy component is the primary cause of directional mismatch and progressively drags the normal expert component away from the original direction as depth increases.

B.2 Auxiliary-Loss Comparison

Experimental Setting. To isolate the effect of the routing regularizer in Section 2.2, we compare three zero-expert adaptation variants on Qwen3-30B-A3B that are identical in architecture and SFT adaptation recipe, differing only in the balancing objective: one uses the standard expert-level auxiliary loss $\mathcal{L}_A$ in Eq. (4), and the other two use the proposed group auxiliary loss $\mathcal{L}_{GA}$ in Eq. (5). The experimental setting is otherwise the same as the SFT implementation details in Section 3.1. In particular, the auxiliary-loss coefficient α is set to 0.1 for all variants. For the group auxiliary loss, the relative weight of the zero-expert group w is set to 1.0 and 2.0, respectively. We report the results on five mathematical reasoning benchmarks.

Table 5: Comparison of auxiliary loss and group auxiliary loss under different w values on mathematical reasoning benchmarks.

Method	Avg Acc	Avg r_{ZE}	Math				
			AIME 24	AIME 25	AIME 26	GSM8K	MATH-500
Qwen3-30B-A3B	82.8	0.0	80.9	71.0	72.3	95.4	94.4
Zero Expert + $\mathcal{L}_A$	59.5	34.4	39.6	32.3	47.1	93.5	85.0
Zero Expert + $\mathcal{L}_{GA}$ (w=1.0)	82.2	35.0	79.4	69.7	71.9	95.2	94.8
Zero Expert + $\mathcal{L}_{GA}$ (w=2.0)	81.0	52.7	78.1	66.2	71.3	94.8	94.4

Conclusion. Compared with the original auxiliary loss, both group auxiliary loss variants deliver significant improvements, showing that the benefit of group-level balancing is robust across different w settings. In particular, replacing $\mathcal{L}_A$ with $\mathcal{L}_{GA}$ at w=1.0 improves average accuracy from 59.5 to 82.2, nearly recovering the original Qwen3-30B-A3B performance of 82.8, while maintaining a similar zero-expert activation ratio. Increasing the group weight to w=2.0 raises r_{ZE} further to 52.7 while preserving a strong average accuracy of 81.0, indicating a clear quality-efficiency trade-off.

These results are consistent with the design motivation in Section 2.2. A post-trained MoE model exhibits non-uniform, input-dependent routing patterns over normal experts, and enforcing expert-level uniformity disrupts these learned routing distributions, which can severely degrade model performance. By contrast, the group auxiliary loss regulates only the competition between the normal-expert group and the zero-expert group, thereby preserving the relative routing structure among normal experts while enabling controllable zero-expert utilization through w.

B.3 Effect of Training Stages

The full ZEDA pipeline employs a two-stage self-distillation strategy. To assess whether both stages are necessary, three variants are compared: (1) SFT only, (2) OPD only, and (3) the full SFT→OPD pipeline. To ensure a fair comparison, the SFT-only and OPD-only variants are trained with an increased number of steps so that their total computational cost matches or exceeds that of the full pipeline. As shown in Table 6, the full SFT→OPD pipeline consistently outperforms both single-stage alternatives, with OPD alone performing the worst. This may be because SFT first establishes stable zero-expert routing patterns, without which OPD must simultaneously learn routing decisions and generate coherent responses, compounding the adaptation difficulty. Once SFT has stabilized the router, OPD can focus on closing the remaining distribution gap under on-policy rollouts, yielding further gains that neither stage achieves in isolation. Under a comparable computational budget, the full SFT→OPD pipeline outperforms SFT alone, demonstrating the necessity of the OPD stage.

Table 6: Performance of ZEDA with different self-distillation strategies on Qwen3-30B-A3B.

	Avg Acc	Avg r_{ZE}	Math					Code				IF	
			AIME 24	AIME 25	AIME 26	GSM8K	MATH-500	LCB v5	LCB v6	HumanEval+	MBPP+	IFBench	IFEval
SFT	73.6	51.5	78.5	**69.3**	69.8	95.3	94.8	58.1	52.1	**88.6**	79.0	40.3	83.5
OPD	72.9	50.1	78.4	67.2	72.3	94.9	94.4	57.0	51.5	87.3	**79.6**	36.3	83.2
SFT → OPD	**74.2**	51.2	**79.0**	69.1	**72.5**	**95.5**	**95.2**	**58.2**	**53.2**	88.5	78.2	**42.3**	**84.3**

B.4 Self-Distillation without Zero Experts

To isolate the effect of the training procedure from zero-expert adaptation, we apply the same SFT→OPD setting as in the main experiments to the unmodified Qwen3-30B-A3B model without adding zero experts. As shown in Table 7, the reported average accuracy remains unchanged at 74.9. This control suggests that the same self-distillation procedure alone provides no extra accuracy gain for the unmodified model. ZEDA instead uses it to support adaptation to reduced expert computation while largely preserving accuracy.

Table 7: Performance of the unmodified Qwen3-30B-A3B model under the same SFT→OPD setting without zero-expert injection.

Model	Avg Acc	Avg r_{ZE}	Math					Code				IF	
			AIME 24	AIME 25	AIME 26	GSM8K	MATH-500	LCB v5	LCB v6	HumanEval+	MBPP+	IFBench	IFEval
Qwen3-30B-A3B	74.9	0.00	80.9	71.0	72.3	95.4	94.4	61.5	57.1	85.6	79.2	39.7	86.3
+ SFT → OPD	74.9	0.00	80.2	68.5	73.3	95.6	95.4	61.0	55.9	87.6	80.0	41.3	84.7

B.5 Repeated Experiments Results

We repeat the experiments of NET and ZEDA on Qwen3-30B-A3B in Table 1 two additional times, following the evaluation protocol in Section 3.1. Table 8 reports the mean and standard deviation over all three evaluation runs.

Table 8: Mean (standard deviation) over three runs on Qwen3-30B-A3B, in percent. For NET, r_{ZE} denotes the fixed expert-skipping ratio.

Method	Avg Acc	Avg r_{ZE}	Math					Code				IF	
			AIME 24	AIME 25	AIME 26	GSM8K	MATH-500	LCB v5	LCB v6	HumanEval+	MBPP+	IFBench	IFEval
NET_{SFT}	72.1(0.17)	50.0(0.00)	76.5(0.38)	65.8(0.29)	70.1(1.63)	94.7(0.16)	94.2(0.16)	56.1(0.31)	50.7(0.17)	86.7(0.05)	77.8(0.28)	38.2(0.75)	82.3(0.45)
$\text{NET}_{\text{SFT}\to\text{OPD}}$	72.9(0.14)	50.0(0.00)	78.5(0.78)	67.7(1.19)	70.6(0.53)	95.1(0.25)	94.4(0.43)	56.8(0.75)	52.5(0.26)	88.3(0.93)	77.7(0.12)	38.8(0.14)	81.5(1.15)
ZEDA_{SFT}	73.2(0.09)	51.6(0.05)	78.2(0.37)	66.7(0.33)	71.4(0.19)	95.1(0.22)	94.7(0.25)	57.7(0.38)	53.0(0.12)	86.7(0.25)	78.5(0.29)	39.3(0.29)	84.4(0.65)
ZEDA	74.0(0.17)	51.2(0.00)	78.8(0.28)	68.7(0.83)	72.0(0.85)	95.2(0.25)	94.7(0.52)	58.7(0.34)	53.4(0.31)	88.2(0.31)	78.6(0.33)	41.2(2.07)	84.9(0.76)

ZEDA improves average accuracy over NET by 1.1 percentage points under both SFT and SFT→OPD. This exceeds the observed standard deviations of average accuracy (0.09–0.17), supporting the robustness of the gains across repeated runs.

B.6 Impact of Router Probability Renormalization

As defined in Eq. (1), the dynamic MoE output after zero-expert injection is $\tilde{y}(h) = \sum_{i \in \tilde{\mathcal{S}}(h) \cap \mathcal{E}} \tilde{g}_i(h)\, E_i(h)$. In this formulation, the routing weights $\tilde{g}_i(h)$ of the remaining normal experts are not renormalized after zero experts are removed from the top-K selection. An alternative is to redistribute the probability among the active normal experts through renormalization. Concretely, the renormalized output becomes

$$\tilde{y}_{\text{renorm}}(h) = \sum_{i \in \tilde{\mathcal{S}}(h) \cap \mathcal{E}} \frac{\tilde{g}_i(h)}{\sum_{j \in \tilde{\mathcal{S}}(h) \cap \mathcal{E}} \tilde{g}_j(h)}\, E_i(h). \tag{8}$$

To evaluate this design choice, SFT is conducted on Qwen3-30B-A3B with and without renormalization under identical hyperparameters. As reported in Table 9, renormalization leads to a consistent accuracy drop compared with the default formulation. A likely reason is that in the original model, the sum of routing weights over the top-K experts is calibrated during pre-training to produce outputs at a certain magnitude. And renormalization artificially amplifies the routing weights of the active normal experts, inflating the effective scale of the MoE residual branch.

Table 9: Performance of ZEDA_{SFT} with and without renormalization on Qwen3-30B-A3B.

	Avg	Avg	Math					Code				IF	
	Acc	r_{ZE}	AIME 24	AIME 25	AIME 26	GSM8K	MATH-500	LCB v5	LCB v6	HumanEval+	MBPP+	IFBench	IFEval
w/ Renorm	71.6	51.0	76.6	64.8	67.9	94.8	94.2	55.0	48.6	**87.4**	77.7	38.3	82.1
w/o Renorm	**73.3**	51.5	**78.1**	**66.2**	**71.3**	94.8	**94.4**	**58.2**	**52.8**	86.8	**78.6**	**39.7**	**85.2**

B.7 $\mathcal{L}_{GA}$ Coefficient α Ablation

The loss coefficient α in Eq. (5) controls how strongly the Group Auxiliary Loss $\mathcal{L}_{GA}$ influences the overall training objective. To investigate its effect, the SFT stage of ZEDA is conducted on Qwen3-30B-A3B with α varied across $\{0.001, 0.01, 0.1, 1.0\}$, while the relative weight w is fixed at 2, corresponding to a target r_{ZE} of 50%. As shown in Table 10, increasing α from 0.001 to 0.1 drives r_{ZE} toward the target, whereas a further increase to 1.0 produces no material change in either r_{ZE} or average accuracy. We therefore set $\alpha = 0.1$, the smallest tested coefficient that effectively enforces the target utilization without unnecessarily strengthening the auxiliary regularization.

Table 10: ZEDA_{SFT} performance on Qwen3-30B-A3B with different α.

α	Avg	Avg	Math					Code				IF	
	Acc	r_{ZE}	AIME 24	AIME 25	AIME 26	GSM8K	MATH-500	LCB v5	LCB v6	HumanEval+	MBPP+	IFBench	IFEval
0.001	74.3	31.8	79.2	68.5	73.1	95.2	94.8	59.7	55.6	88.3	79.0	38.0	86.3
0.01	73.6	44.6	78.3	67.8	71.7	95.2	94.4	59.0	52.5	87.1	78.4	41.7	84.1
0.1	73.3	51.5	78.1	66.2	71.3	94.8	94.4	58.2	52.8	86.8	78.6	39.7	85.2
1.0	73.3	50.9	77.4	67.3	69.8	94.9	94.8	59.0	54.4	87.0	78.7	39.7	83.6

C Experimental Setup Details

For evaluation, HumanEval+ and MBPP+ are two code generation benchmarks introduced by EvalPlus (Liu et al., 2023). All evaluations adopt a temperature of 0.6, a top-p value of 0.95, and a top-k value of 20, with a maximum generation length of 38k tokens following the Qwen3 setting (Yang et al., 2025). Following the conventions of Qwen3 (Yang et al., 2025) and IFBench (Pyatkin et al., 2025), results on IFEval and IFBench are reported as strict prompt accuracy and loose prompt accuracy, respectively.

For training, in the SFT stage, the learning rate is set to 2×10^{-5}. The subsequent stage employs Sampled-Token OPD with a learning rate of 5×10^{-6} for Qwen3-30B-A3B and 1×10^{-6} for

GLM-4.7-Flash, a batch size of 16 prompts $\times$ 2 sampled responses, a sampling temperature of 1.0, a maximum generation length of 32k tokens, and runs for 320 training steps. We cross-check the training and evaluation datasets to ensure no sample overlap. All experiments are conducted on the slime (Zhu et al., 2025), SGLang (Zheng et al., 2024), and Megatron (Shoeybi et al., 2019) codebases, and NVIDIA H200 and H20 GPUs.

For throughput evaluation, we use SGLang as the inference framework with the maximum concurrency set to 32. We randomly sample 256 examples from the training data to construct the test set. To ensure a fair comparison across models, for each target sequence length we control the total numbers of input and output tokens to be identical across compared models and to match the intended test sequence length. In addition, the input sequence content is kept exactly the same across models. We report throughput at sequence length 8192 for both the single-H200 setting and a four-GPU expert-parallel setting using DeepEP (Zhao et al., 2025) with EP=4. In each deployment setting, speedup is computed relative to the original MoE under the same configuration.

D Theoretical FLOPs Analysis

In this section, we analyze the theoretical FLOPs of both the prefill and decode stages for the original MoE model and the model adapted by ZEDA. We focus on dominant matrix multiplication terms and omit lower-order operations such as normalization, residual connections, activation functions, routing top-k selection, and softmax overhead. For a matrix multiplication $[m, n] \times [n, p]$, we count its cost as $2mnp$ FLOPs. All expressions are reported per Transformer layer. Multiplying by the number of layers does not change the ZEDA/original FLOP ratios when all layers share the same configuration.

The notation used throughout this section is summarized in Table 11.

Table 11: Notation used in the theoretical FLOP analysis.

Symbol	Description
l	Sequence length in the stage under analysis
H	Hidden size
H_{attn}	Attention intermediate size
g_{kv}	Ratio of KV heads to query heads in GQA
H_e	Expert intermediate size
N	Number of normal experts
N_Z	Number of zero-computation experts
K	Number of activated experts per token
r_{ZE}	Fraction of activated zero experts

D.1 Shared MoE Cost Decomposition

The MoE FFN and router costs have the same form in both stages; the only difference is the number of tokens processed in the current forward pass. Let n denote that token count. For the original MoE model, each token activates K normal experts, and each expert contains up, gate, and down projections. Hence, the expert FFN and router costs are

$$F_{\text{MoE,orig}}(n) = 6KnHH_e + 2NnH. \tag{9}$$

For the ZEDA model, only an $(1 - r_{ZE})$ fraction of the activated experts perform FFN computation, while the router scores both normal and zero-computation experts. Therefore,

$$F_{\text{MoE,ZEDA}}(n) = 6(1 - r_{ZE})KnHH_e + 2(N + N_Z)nH. \tag{10}$$

In the prefill stage, $n = l$. In the decode stage with KV cache, each forward pass processes one newly generated token, and the total decode cost is obtained by summing over all decode steps.

D.2 PREFILL STAGE

For grouped-query attention (GQA) (Ainslie et al., 2023), the prefill stage processes all l tokens in parallel. The attention cost therefore consists of five parts: the query projection, the key/value projections, the query-key score computation over all token pairs, the attention-value aggregation, and the output projection. These terms sum to

$$F_{\text{attn}}^{\text{pre}} = 4l^2 H_{\text{attn}} + 4(1 + g_{\text{kv}}) l H H_{\text{attn}}. \tag{11}$$

Substituting $n = l$ into Equations (9) and (10), and then adding the prefill attention term in Equation (11), yields the total prefill FLOPs of the original model and the ZEDA model. In both expressions, the first two terms come from attention, the third term is the MoE FFN cost, and the last term is the router cost:

$$F_{\text{orig}}^{\text{pre}} = 4l^2 H_{\text{attn}} + 4(1 + g_{\text{kv}}) l H H_{\text{attn}} + 6K l H H_e + 2N l H. \tag{12}$$

For the ZEDA model, the attention term remains unchanged, while the expert FFN cost is reduced by the factor $(1 - r_{ZE})$ and the router cost increases because the router now scores $N + N_Z$ experts:

$$F_{\text{ZEDA}}^{\text{pre}} = 4l^2 H_{\text{attn}} + 4(1 + g_{\text{kv}}) l H H_{\text{attn}} + 6(1 - r_{ZE}) K l H H_e + 2(N + N_Z) l H. \tag{13}$$

The corresponding FLOP ratio, obtained from Equations (12) and (13), is

$$\boxed{\frac{F_{\text{ZEDA}}^{\text{pre}}}{F_{\text{orig}}^{\text{pre}}} = \frac{2l H_{\text{attn}} + 2(1 + g_{\text{kv}}) H H_{\text{attn}} + 3(1 - r_{ZE}) K H H_e + (N + N_Z) H}{2l H_{\text{attn}} + 2(1 + g_{\text{kv}}) H H_{\text{attn}} + 3K H H_e + N H}} \tag{14}$$

D.3 DECODE STAGE

In the decode stage, we assume standard KV caching and analyze a decode-only process that generates l tokens. As in the prefill case, the attention cost consists of query projection, key/value projections, score computation, attention-value aggregation, and output projection. The difference is that at decode step t, only one new token is processed, and the score computation and attention-value aggregation each involve $t - 1$ cached tokens rather than all l tokens. Summing these per-step costs over all l decode steps gives

$$\begin{aligned} F_{\text{attn}}^{\text{dec}} &= \sum_{t=1}^{l} [4(t-1) H_{\text{attn}} + 4(1 + g_{\text{kv}}) H H_{\text{attn}}] \\ &= 2l(l-1) H_{\text{attn}} + 4(1 + g_{\text{kv}}) l H H_{\text{attn}}. \end{aligned} \tag{15}$$

Substituting the per-token MoE costs from Equations (9) and (10) across all l decode steps, and adding the accumulated attention cost in Equation (15), gives the total decode FLOPs. As in the prefill case, the first two terms correspond to attention, the third term is the MoE FFN cost, and the last term is the router cost:

$$F_{\text{orig}}^{\text{dec}} = 2l(l-1) H_{\text{attn}} + 4(1 + g_{\text{kv}}) l H H_{\text{attn}} + 6K l H H_e + 2N l H. \tag{16}$$

For the ZEDA model, the decode attention term is again identical to that of the original model, whereas the MoE branch differs in exactly the same way as in prefill: only an $(1 - r_{ZE})$ fraction of activated experts incur FFN cost, and the router expands from N to $N + N_Z$ outputs:

$$F_{\text{ZEDA}}^{\text{dec}} = 2l(l-1) H_{\text{attn}} + 4(1 + g_{\text{kv}}) l H H_{\text{attn}} + 6(1 - r_{ZE}) K l H H_e + 2(N + N_Z) l H. \tag{17}$$

Therefore, the decode-stage FLOP ratio, obtained from Equations (16) and (17), is

$$\boxed{\frac{F_{\text{ZEDA}}^{\text{dec}}}{F_{\text{orig}}^{\text{dec}}} = \frac{(l-1) H_{\text{attn}} + 2(1 + g_{\text{kv}}) H H_{\text{attn}} + 3(1 - r_{ZE}) K H H_e + (N + N_Z) H}{(l-1) H_{\text{attn}} + 2(1 + g_{\text{kv}}) H H_{\text{attn}} + 3K H H_e + N H}} \tag{18}$$

Table 12: Architectural parameters of Qwen3-30B-A3B used in the FLOP analysis.

Symbol	H	H_{attn}	g_{kv}	H_e	N	N_Z	K
Value	2048	4096	$1/8$	768	128	64	8

Table 13: Comparison between theoretical speedups derived from the FLOP analysis and measured empirical speedups on Qwen3-30B-A3B at sequence length 8192.

Length	Prefill Speedup		Decode Speedup	
	Theoretical	Empirical	Theoretical	Empirical
8192	1.178×	1.175×	1.261×	1.185×

D.4 Numerical Results

We instantiate the prefill and decode ratios in Equations (14) and (18) using the Qwen3-30B-A3B configuration (Yang et al., 2025) in Table 12.

To facilitate direct comparison with empirical measurements, we convert the FLOP ratios in Equations (14) and (18) into theoretical speedups by taking their reciprocals. Table 13 reports the resulting prefill and decode speedups for $l = 8192$ and $r_{ZE} = 0.5$, together with the corresponding empirically measured results.

At $l = 8192$, the empirical prefill speedup of $1.175\times$ closely matches the FLOP-based theoretical prediction of $1.178\times$, as shown in Table 13. We attribute the larger gap between theoretical and empirical speedups in decode primarily to its more memory-bound nature: reducing FLOPs does not proportionally reduce latency when memory accesses remain a bottleneck, which the FLOP-only analysis does not capture. Prefill is more compute-bound.