BATCH-AWARE HIERARCHICAL ROUTING FOR MIXTURE-OF-EXPERTS WITH REDUCED MEMORY TRAFFIC AND THEORETICAL GUARANTEES

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture-of-Experts (MoE) architectures reduce computation by activating a small subset of experts per token. However, under batch-based execution, memory access is determined by the union of experts selected across the batch, so memory movement remains a key bottleneck. We propose Batch-Aware Hierarchical MoE Routing (BAR), which augments token-level routing with a batch-level router that first selects a small subset of experts per batch, followed by token-level refinement. This design explicitly reduces memory traffic while improving efficiency and expert specialization. We theoretically show that BAR reduces memory access, inference cost, and training computation, established through the training dynamics analysis of two-layer ReLU experts. Across evaluation on large-scale MoE language models, BAR substantially reduces HBM traffic and improves inference throughput while closely preserving base-model performance. For example, on DeepSeek-MoE-16B, BAR reduces HBM traffic by 60% and improves throughput by $2.5\times$, while maintaining perplexity within 2% of the base model.

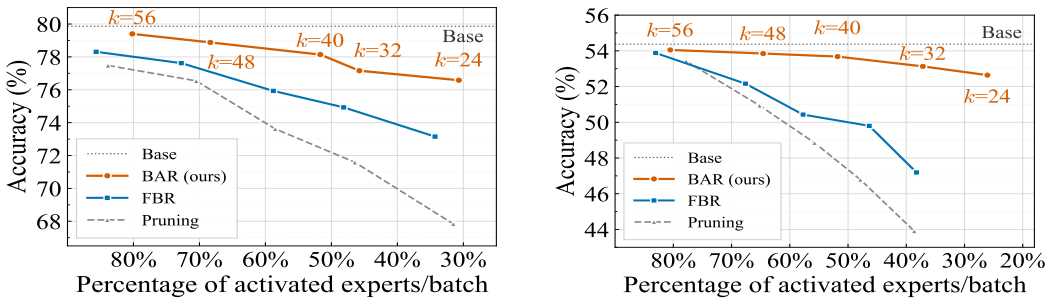
1 INTRODUCTION

The sparse Mixture-of-Experts (MoE) architecture has emerged as a powerful paradigm for scaling large models, particularly large language models, while maintaining manageable computational cost. An MoE layer consists of a collection of sub-networks, referred to as experts, together with a routing mechanism that activates only a small subset of experts for each input (Jacobs et al., 1991; Jordan & Jacobs, 1993; Tresp, 2000). This sparse activation enables models with billions or even trillions of parameters to be trained and deployed with computation comparable to much smaller dense models (Bengio et al., 2013; Eigen et al., 2014; Shazeer et al., 2017). In Transformer-based architectures, routing is performed at the token level, including token-choice (each token selects a few experts) (Shazeer et al., 2017; Fedus et al., 2022; Wang et al., 2025) and expert-choice (each expert selects a fixed number of tokens) (Zhou et al., 2022; Riquelme et al., 2021; Puigcerver et al., 2024).

However, model parameters are stored in high-bandwidth memory (HBM) and must be fetched into the compute hierarchy when their corresponding experts are activated. In practice, tokens are processed in batches, so the expert-weight traffic for a batch is determined by the union of experts selected across all tokens in a batch. While existing sparse routing mechanisms reduce arithmetic computation through sparse token–expert interactions, they do not necessarily reduce this union proportionally. Consequently, a large fraction of expert weights may still need to be accessed and moved for each batch, resulting in substantial HBM traffic. As model sizes grow, the dominant bottleneck shifts from arithmetic computation to data movement. Even accessing on-package memory can be one to two orders of magnitude more expensive than arithmetic operations, while off-chip memory access can be 100–1000$\times$ more costly (Horowitz, 2014).

Several recent works propose static expert pruning, which permanently removes unimportant experts to reduce computation and memory footprint (Zhou et al., 2022; Chowdhury et al., 2024; Lu et al., 2024; Zhou et al., 2025; Yang et al., 2025), and can consequently reduce memory traffic during inference. However, substantial reductions in memory traffic typically require aggressive expert removal, which often leads to significant accuracy degradation. In contrast, our goal is to directly limit the number of experts accessed for each batch, while retaining the full expert pool across batches.

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



(a) DeepSeek-MoE-16B on HellaSwag

(b) Qwen1.5-MoE-A2.7B (14.3B) on MMLU

Figure 1: Zero-shot accuracy on (a) HellaSwag using DeepSeek-MoE-16B and (b) MMLU using Qwen1.5-MoE-A2.7B (14.3B total parameters) under static pruning, frequency-based routing (FBR), and batch-aware routing (BAR). Benchmark evaluations compare performance across varying active expert percentages, referenced against the "token-choice only" baseline model.

Moreover, despite the empirical success of MoE models, the theoretical understanding of their generalization remains limited. Only a few recent works provide theoretical training dynamics and generalization analyses. Chen et al. (2022) prove theoretically that a sample-choice MoE outperforms convolutional neural networks (CNN) for binary classification on linearly separable data. Chowdhury et al. (2023) analyze the training dynamics and the resulting generalization of expert-choice routing, and Chowdhury et al. (2024) analyze the generalization of static expert pruning.

In this paper, we propose a Batch-Aware hierarchical MoE Routing, designed to reduce not only expert computation but also batch-wise expert weight movement. The key idea is to introduce a batch-level router in addition to token-level routing. Specifically, the batch-level router first selects a small subset of relevant experts for each batch, reducing the candidate expert set. Then, an expert-choice router selects the most informative tokens for each selected expert. By directly reducing the number of activated experts at the batch level, our method lowers memory traffic directly. Meanwhile, jointly filtering both experts and tokens yields substantial computational savings, enhances expert specialization, and improves generalization. Our major contributions are summarized as follows:

1. **A batch-aware routing framework with theoretical guarantees.** We propose a novel routing mechanism that integrates batch-level expert selection with token-level expert-choice routing. This joint design introduces new challenges in analyzing both training dynamics and generalization. We establish convergence guarantees for the proposed routing scheme under stochastic gradient descent when each expert is a two-layer ReLU network. Our analysis reveals a synergistic effect: batch-level routing reduces the candidate expert set, while token-level routing refines token assignments, together accelerating expert specialization and improving generalization performance.

2. **Reduced computation and memory traffic with theoretical characterization.** Our batch-level routing directly reduces the number of activated experts per batch, thereby reducing both memory traffic and inference computation. Moreover, our analysis of the training dynamics shows that this computational saving does not come at the cost of increased training iterations. Together with the reduced per-iteration cost, this leads to lower overall training computation.

3. **Empirical advantage on large-scale language tasks.** We validate BAR on large-scale MoE language models, including DeepSeek-MoE-16B, Qwen1.5-MoE-A2.7B, and OLMoE-1B-7B. On DeepSeek-MoE-16B, activating only 24 out of 64 experts per batch reduces HBM traffic by approximately 60% and improves inference throughput by $2.5\times$, while maintaining perplexity within 2% of the base model. At matched expert budgets, BAR consistently achieves better task performance than static pruning and frequency-based routing; as illustrated in Figure 1, it remains within 3.25 percentage points of the base-model accuracy on HellaSwag while activating only 37.5% of the experts. BAR also improves training time by up to $2.11\times$ on OLMoE.

1.1 Related Works

Model pruning and compression. Pruning reduces the model size and thus inference cost in various neural network models where unimportant weights or neurons are removed to induce sparsity while

preserving accuracy. Pruning has been exploited to compress large neural models (including large language models) (Xie et al., 2024; Lee et al., 2025; Sun et al., 2024; Ma et al., 2023; Frantar & Alistarh, 2023b). In parallel, several approaches focus on compressing MoE models via inter-expert compression and intra-expert factorization (Zhang et al., 2025; Yang et al., 2024), as well as applying post-training quantization (PTQ) (Kim et al., 2022; 2023; Frantar & Alistarh, 2023a; Yi et al., 2025) and expert-aware mixed-precision schemes (Li et al., 2024d; Huang et al., 2025) to reduce memory/bandwidth and latency while maintaining accuracy. All these compression methods are orthogonal to the proposed joint routing method in this paper and can be combined with it.

Optimization and generalization analysis of neural networks. Earlier works have established optimization and generalization guarantees for neural networks using neural tangent kernel (NTK)–based analyses (Jacot et al., 2018; Lee et al., 2019; Du et al., 2019; Allen-Zhu et al., 2019b; Li et al., 2022). These approaches rely on the overparameterized regime in which network weights remain close to initialization, and therefore cannot capture realistic training dynamics involving substantial representation learning. To overcome this limitation, recent studies focus on *feature learning* dynamics, aiming to characterize how neural networks adapt their internal representations during training and to derive generalization guarantees that better align with practical behavior. Such analysis has been exploited in Transformer-type of architectures (Jelassi et al., 2022; Li et al., 2023d; Ildiz et al., 2024; Nichani et al., 2024; Chen et al., 2024; Li et al., 2023a; 2024c; Luo et al., 2024) and MoEs (Chowdhury et al., 2023; 2024). Such analyses are typically limited to shallow networks, as increasing depth leads to nonlinear compositions and concatenations of weights across layers, making the training problem highly nonconvex and difficult to analyze (Karp et al., 2021; Brutzkus & Globerson, 2021; Li et al., 2023b; Zhang et al., 2023). Our work follows this line of feature-learning analysis.

2 Our Proposed Hierarchical MoE Routing Method

2.1 Existing MoE Routing Methods

Before introducing our proposed hierarchical batch-aware routing method, we first briefly discuss the existing token-level MoE routing methods, including token-choice and expert-choice methods. Consider an MoE layer that includes N experts. Let $f_s(\cdot)$ represent the function by expert s, $s \in [N]$. Consider a batch $x = [x^{(1)}, x^{(2)}, \ldots, x^{(n)}]$ in $\mathbb{R}^{d\times n}$ of n tokens/patches[1], where $x^{(p)} \in \mathbb{R}^d$.

Token-choice routing: A token-choice router is parameterized by a set of N learnable router weights $w_{\text{T}}^{(s)} \in \mathbb{R}^d$ for every expert s. Each token is processed by v selected experts with $v \ll N$. Specifically, given a token $x^{(j)}$, the raw routing score $\bar{h}_j^{(s)}$ for expert s is computed by $\bar{h}_j^{(s)} = \langle w_{\text{T}}^{(s)}, x^{(j)}\rangle$. Let $J_{\text{T}}(x^{(j)}) \subset [N]$ denote the index set of v selected experts with the top raw routing values $\bar{h}_j^{(s)}$, and $|J_{\text{T}}(x^{(j)})| = v$. The normalized gating values for selected experts are computed by

$$\bar{H}_j^{(s)} = \exp(\bar{h}_j^{(s)})/\big(\sum_{s'\in J_{\text{T}}(x^{(j)})} \exp(\bar{h}_j^{(s')})\big), \forall s \in J_{\text{T}}(x^{(j)}) \tag{1}$$

The output of the MoE layer by token-choice routing is

$$f_{\text{T}}(x^j) = \sum_{s\in J_{\text{T}}(x^{(j)})} \bar{H}_j^{(s)} \cdot f_s(x^{(j)}). \tag{2}$$

Token-choice routing suffers from a load-balancing issue, where some experts become overloaded while others are underutilized (Wang et al., 2025). A regularizer can be included in the training objective to encourage load balancing (Fedus et al., 2022); however, this increases training complexity, introduces additional hyperparameter tuning, and still does not fully guarantee that expert overflow will be avoided (Wang et al., 2025).

Expert-choice routing: Expert-choice routing mitigates this issue by reversing the assignment: each expert selects v tokens. Specifically, let $w_{\text{P}}^{(s)} \in \mathbb{R}^d$ denote the expert-choice router weight for expert s. The raw routing score by expert s for any patch $x^{(j)}$ is computed as $h_j^{(s)} = \langle w_{\text{P}}^{(s)}, x^{(j)}\rangle$. For each

[1]We use tokens and patches interchangeably in this paper, because our proposed routing method applies to both language and vision models.

expert s, the indices of the top v patches based on these scores are stored in the set $J_{\mathrm{P}}^{(s)}(x)$ with $|J_{\mathrm{P}}^{(s)}(x)| = v$. The normalized routing score for each token is then calculated by

$$H_j^{(s)}(x) = \exp(h_j^{(s)}) / \big(\textstyle\sum_{i \in J_{\mathrm{P}}^{(s)}(x)} \exp(h_i^{(s)}) \big), \quad \forall j \in J_{\mathrm{P}}^{(s)}(x). \tag{3}$$

The final output is the sum of all expert outputs, each of which is a weighted sum of selected tokens:

$$f_{\mathrm{P}}(x) = \textstyle\sum_{s=1}^{N} \sum_{j \in J_{\mathrm{P}}^{(s)}(x)} H_j^{(s)}(x) \cdot f_s(x^{(j)}), \tag{4}$$

Expert-choice routing assigns the same workload to every expert. However, even if an expert contributes little to the downstream task, it still has to process the full workload, which is wasteful.

2.2 Our Proposed Batch-aware Hierarchical Routing Method

Our **batch-aware routing** augments the standard token-level router (either token-choice or expert-choice) with an additional **batch-level router**. While the batch-level router is defined analogously to the token-level router, the key difference is that it selects a shared subset of k experts for the entire batch x, rather than selecting experts independently for each token $x^{(j)}$.

Batch-level routing: Consider a set of N learnable router weights $w_{\mathrm{S}}^{(s)} \in \mathbb{R}^d$ for every expert s. The raw batch-level routing score $g^{(s)}(x)$ for expert s and the whole batch x is computed by

$$g^{(s)}(x) = \textstyle\sum_{j=1}^{n} \langle w_{\mathrm{S}}^{(s)}, x^{(j)} \rangle. \tag{5}$$

Let $J_{\mathrm{S}}(x) \subset [N]$ denote the index set of k selected experts with the top raw routing values, and $|J_{\mathrm{S}}(x)| = k$. The normalized batch-level gating values for selected experts are computed by

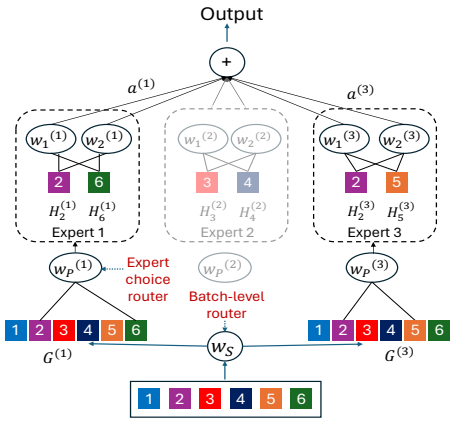


Figure 2: Illustration of batch-aware hierarchical routing. $n = 6$ and $N = 3$. The batch-level router w_{S} selects $k = 2$ experts. The expert-choice router w_{P} selects $v = 2$ out of n tokens for each selected expert.

$$G^{(s)}(x) = \exp(g^{(s)}(x)) / \big(\textstyle\sum_{s' \in J_{\mathrm{S}}(x)} \exp(g^{(s')}(x)) \big), \forall s \in J_{\mathrm{S}}(x) \tag{6}$$

Hierarchical Routing method: When combined with a token-level router (either token-choice or expert-choice), the batch-level router first selects a subset of k experts $J_{\mathrm{S}}(x)$ for the entire batch. When combined with token-choice routing, each token further selects its top-v experts $J_{\mathrm{T}}(x^j)$ and computes token-level routing values $\bar{H}_j^{(s)}(x^{(j)})$ in (1). The final output aggregates only over experts in the intersection $J_{\mathrm{S}}(x) \cap J_{\mathrm{T}}(x^{(j)})$, with contributions weighted by the product of batch-level and token-choice gating values $G^{(s)}(x)$ and $\bar{H}_j^{(s)}(x^{(j)})$, i.e.,

$$f(x) = \textstyle\sum_{j=1}^{n} \sum_{s \in J_{\mathrm{S}}(x) \cap J_{\mathrm{T}}(x^{(j)})} G^{(s)}(x) \cdot \bar{H}_j^{(s)} \cdot f_s(x^{(j)}). \tag{7}$$

In the expert-choice setting, each selected expert s in $J_{\mathrm{S}}(x)$ processes only its top-v tokens in $J_{\mathrm{P}}^{(s)}(x)$ and computes expert-choice routing values $H_j^{(s)}(x)$ in (3). The final output is a weighted sum over selected experts and their assigned tokens, again combining batch-level and expert-choice routing weights $G^{(s)}(x)$ and $H_j^{(s)}(x)$ multiplicatively, i.e.,

$$f(x) = \textstyle\sum_{s \in J_{\mathrm{S}}(x)} G^{(s)}(x) \cdot \sum_{j \in J_{\mathrm{P}}^{(s)}(x)} H_j^{(s)}(x) \cdot f_s(x^{(j)}). \tag{8}$$

Figure 2 shows our batch-aware routing method in the expert-choice setting, where each batch has $n = 6$ tokens, and the network has $N = 3$ experts. Each expert is a two-layer neural network.

Our proposed hierarchical routing structure reduces unnecessary computation by filtering experts at the batch level while preserving the fine-grained adaptivity of token-level routing. Our batch-level routing can be combined with both token-choice and expert-choice routing. We focus on the expert-choice setting (8) in theoretical analysis to facilitate tractability and enable direct comparison

with Chowdhury et al. (2023; 2024), which analyze the training dynamics and generalization under expert-choice routing. However, both combinations (7) and (8) are evaluated empirically[2].

Reduced inference computation by batch-aware hierarchical routing. Compared with using expert-choice routing only, which requires evaluating all N experts, our method activates only k experts for each batch. This reduces the inference-time computation by a factor of k/N.

Connection and difference to static expert pruning. To reduce the inference computation of MoE, some recent work (Chowdhury et al., 2024; Lu et al., 2024; Zhou et al., 2025; Yang et al., 2025) prune experts permanently after training. For example, Chowdhury et al. (2024) first trains an expert-choice router and prunes all the experts that contribute less to the output. The pruned experts are the same for all samples. If only k out of N experts remain, the inference computation can also be reduced by a factor of k/N. Our routing method can be viewed as a dynamic pruning approach that only k experts are activated to select tokens for each batch, but the active experts differ across batches. Moreover, we jointly train both the batch-level and expert-choice routers, whereas the prior method trains a standard expert-choice router and applies pruning only after training.

The MoE model with our proposed routing method in (7,8) can be trained using standard loss functions and optimization methods, such as stochastic gradient descent. As our proposed method combines both batch-level and token-level routing, its analysis is substantially more involved. The central technical challenge is to show that, despite the increased complexity induced by jointly training two routers, the learning dynamics converge with lower computational cost than training with batch-level or token-level routing alone, and yield a model with provable generalization guarantees. We establish these results theoretically in Section 3.

3 Theoretical Training Convergence and Generalization Guarantees

3.1 Key Theoretical Insights

Before formally presenting our theoretical result, we first summarize the major theoretical insights here. We consider a binary classification task in which each batch contains both discriminative and non-discriminative tokens. Two discriminative features o_1 and o_2 determine the class label, while the remaining non-discriminative features do not affect labels. Our major theoretical takeaways include:

(I1). The batch-level and token-level routers form a win-win specialization mechanism. After a small number of iterations, the batch-level router learns to route positive and negative batches to different experts that are specialized in learning o_1 and o_2 patterns respectively (Lemma 1), while the expert-choice router of each such expert further selects o_1 or o_2 tokens with high gating values (Lemma 2). This coordinated routing suppresses interference from non-discriminative tokens and non-specialized experts, enabling faster and more efficient expert specialization.

(I2). Batch-aware hierarchical routing reduces training computation while maintaining generalization (Theorem 1). Resulting from **(I1)**, hierarchical routing achieves convergence in a comparable, or even less, number of iterations required to train expert-choice router alone. Because it activates only k out of N experts per iteration, instead of all experts by expert-choice-only routing, the expert computation during training can be reduced by up to a factor of N/k.

(I3). The Batch-aware hierarchical routing reduces inference computation and expert-weight memory movement compared to existing routing method (Theorem 2). During inference, only a fraction $\frac{k}{N}$ of experts are used per batch while other experts are inactivated. It reduces both inference computation and memory access compared with expert-choice routing only.

3.2 Problem Setup

We first specify the setup for our formal theoretical analysis.

[2]Prior work has shown that token-choice and expert-choice routing exhibit similar behavior at the token level (Chowdhury et al., 2023; 2024). All publicly available PyTorch implementations of MoE models adopt token-choice routing. Fig. 5 in Appendix A.4 compares token-choice and expert-choice routing, while all other experiments build upon existing token-choice implementations.

(a) Network architecture: Here each expert s is a two-layer nonlinear neural network. Specifically, $f_s(x^{(j)})$ in (8) is

$$f_s(x^{(j)}) = W_2^{(s)}\sigma(W_1^{(s)\top}x^{(j)}) \tag{9}$$

The hidden layer weight matrix of the expert s is $W_1^{(s)} \in \mathbb{R}^{d\times m}$, and m is the hidden dimension. $w_r^{(s)} \in \mathbb{R}^d$ denotes the rth column of $W_1^{(s)}$. $W_2^{(s)} \in \mathbb{R}^{1\times m}$ represents the output-layer weights. We use the ReLU activation $\sigma(z) = \mathrm{ReLU}(z) := \max(z, 0)$ for $z \in \mathbb{R}$. σ is applied element-wise in (9).

Although a two-layer network is a simplification of practical deep neural networks, it remains the state-of-the-art model for theoretical training dynamics and generalization analysis in mixture-of-experts frameworks (Chowdhury et al., 2023; 2024) and, more broadly, in general neural networks (Li et al., 2023c; 2024b). Due to the nonlinear activation, the learning problem of this two-layer model is inherently non-convex and requires substantial efforts for analyzing the training dynamics and the resulting generalization performance.

(b) Data model: We consider a simplified batch-level binary classification model, where each batch contains n tokens but the label is determined by one decisive token. While language and vision tasks involve more complex structures, model predictions are often driven by a sparse set of salient tokens or features. Our formulation captures this key property and isolates the routing-relevant structure in a tractable setting. Similar binary classification abstractions are adopted in prior theoretical analyses of generalization in MoE models (Chowdhury et al., 2024; Li et al., 2024a; Xue et al., 2023; Lin et al., 2024). and other neural networks (Li et al., 2023b; Shandirasegaran et al., 2026; Li et al., 2025).

Specifically, let $\{(x_i, y_i)\}_{i=1}^{N_{\text{sample}}}$ denote i.i.d. training batches from an unknown distribution $\mathcal{D}$, where $y_i \in \{+1, -1\}$. There are two *class-discriminative patterns*, o_1 and o_2 in $\mathbb{R}^d$, and a set of *class-irrelevant patterns* Q with $|Q| \le d - 2$. o_1, o_2, and all patterns in Q are unit norm and mutually orthogonal. All n tokens in $x \in \mathbb{R}^{d\times n}$ are sampled from these patterns. Moreover, every positive batch x_i ($y_i = +1$) contains exactly one token corresponding to o_1, and every negative batch x_i ($y_i = -1$) contains exactly one token corresponding to o_2. o_1 and o_2 do not appear in the same x.

(c) Method: We consider the hinge loss $\hat{\ell}(f(x), y) = \max\big(1 - yf(x), 0\big)$. Following Chowdhury et al. (2024); Li & Liang (2018); Allen-Zhu et al. (2019a), we fix $W_2^{(s)} = a^{(s)}\mathbf{1}^\top$ during training, where $\mathbf{1} \in \mathbb{R}^m$ is the all-ones vector and $a^{(s)}$ is sampled from $\{+1, -1\}$ with equal probability. Let $S_1 \subset [N]$ denote the set of experts s with $a^{(s)} = 1$ and S_2 denote the set of experts with $a^{(s)} = -1$. The output of experts in S_1 contributes positively to the final output value while the experts in S_2 contribute negatively.

We employ standard stochastic gradient descent with batch size B and step sizes η_{S}, η_{P}, and η_{e} for the batch-level router weights $w_{\mathrm{S}}^{(s)}$, the expert-choice router weights $w_{\mathrm{P}}^{(s)}$ and the hidden layer weights $W_1^{(s)}$, respectively. We employ the same trick as in Zhang et al. (2023); Chen et al. (2024) of using the untruncated loss $\ell(f(x), y) = 1 - yf(x)$ to compute gradients. To facilitate analysis, we introduce a short warm-up phase where the batch-level router weights $w_{\mathrm{S}}^{(s)}$ are fixed at their initial values, while only the expert-choice router weights $w_{\mathrm{P}}^{(s)}$ and expert network weights $W_1^{(s)}$ are trained for T_1 iterations. After this phase, all weights are trained jointly until T iterations, see Algorithm 1 in Appendix. The warm-up phase ensures that experts initially receive batches in an approximately unbiased manner, because a poorly learned batch-level router can lead to highly unbalanced allocation across experts in the early stage and hinder effective learning. The length of this phase T_1 is order-wise smaller than T and thus negligible in the overall training process. In practice, we train all weights jointly from the start without a warm-up phase and observe similar empirical performance.

(d) Router property: We follow the same definition of router proficiency for expert-choice router as in Chowdhury et al. (2024). Specifically, let $p_1^{(s,t)}$ and $p_2^{(s,t)}$ denote the *proficiency* of the expert-choice router for expert s at iteration t for selecting class-discriminative patterns o_1 and o_2, respectively, with gating value at least $1/v$. In particular,

$$p_1^{(s,t)} := \mathbb{P}\Big[(x, +1) \sim \mathcal{D} : \exists\, j \in J_{\mathrm{P}}^{(s,t)} \quad \text{s.t. } x^{(j)} = o_1 \text{ and } H_j^{(s,t)} \ge \tfrac{1}{v}\Big]. \tag{10}$$

The definition of $p_2^{(s,t)}$ follows analogously for o_2 patterns in negative batches. Larger values of $p_1^{(s,t)}$ and $p_2^{(s,t)}$ indicate that expert s is more likely to select class-discriminative patterns.

For batch-level routers, let $G^{(s,t)}$ denote the normalized gating value assigned to expert s by the batch-level router at iteration t. We assume that at initialization, $G^{(s,0)} = \Omega(1/k)$, for all k selected experts s, i.e., $G^{(s,0)} \geq \frac{C}{k}$ for some constant C. Since $G^{(s,0)}$ are softmax-normalized scores over the top-k experts, this assumption ensures that the router is not excessively biased toward a small subset of experts at initialization and can be easily met in practice. We also scale the initial weights such that $\max_{s\in[N],i\in[n]} \|w_{\mathrm{S}}^{(s,i,0)}\|_2$, $\max_{s\in[N]} \|w_{\mathrm{P}}^{(s,0)}\|_2$, $\max_{s\in[N],r\in[m]} \|w_r^{(s,0)}\|_2$ are all constants.

3.3 Formal Theoretical Learning Dynamics and Generalization Results

We first present two important lemmas that illustrate how the batch-level router (Lemma 1), the expert-choice router, and the experts (Lemma 2) themselves gradually learn their functionalities during training and then present the final generalization result of the learned model in Theorem 1. The learning rate and the training iterations satisfy the following condition:

Condition 1. $\eta_P = O(\eta_e/(dk^2v^2))$, $\eta_S = O(\eta_e)$, $T_1 = \Omega(Nv/\eta_e)$, $T_2 = \Omega\big(v\sqrt{kN\log k/(\eta_e\eta_S)}\big)$, $T_3 = \Omega\big((v^2/\eta_e)\sqrt{d\log v}\big)$, $T = T_1 + T_2 + T_3$.

Lemma 1 (Batch-level router learns to select important experts)**.** *Condition 1 holds. Then for any $t \geq T_1 + T_2$, it holds that*

(i) there exists an expert $s \in S_1$ with $p_1^{(s,0)} = \Omega(1)$ s.t. $G^{(s,t)}(x) \geq \frac{1}{2}$ for each positive batch x;

(ii) there exists an expert $s \in S_2$ with $p_2^{(s,0)} = \Omega(1)$ s.t. $G^{(s,t)}(x) \geq \frac{1}{2}$ for each negative batch x.

Lemma 1 shows that after T_1+T_2 training iterations, the batch-level router learns to route batches, with high gating values $G^{(s,t)}$, to experts s that have higher proficiency in extracting class-discriminative patterns o_1 and o_2. In particular, batches are preferentially assigned to experts that have higher *initial* expert-router proficiency.

Lemma 2 (Important Experts become more specialized)**.** *Condition 1 holds. At time T, some experts in S_1 specialize in learning o_1, i.e., there exists $s \in S_1$ with $p_1^{(s,0)} = \Omega(1)$, s.t.*

(i) Its expert-choice router becomes fully specialized in selecting o_1:

$$p_1^{(s,T)} = 1, \quad \textit{and} \quad H_j^{(s,T)}(x) \geq \tfrac{1}{2} \ \textit{ if } x^{(j)} = o_1. \tag{11}$$

(ii) A constant fraction of its neurons r are strongly aligned with o_1:

$$\langle w_r^{(s,T)}, o_1\rangle = \Omega\big(vC_2\sqrt{d\log v}\big). \tag{12}$$

Moreover, no experts in S_2 are specialized in learning o_1. Specifically, for any $s \in S_2$,

(iii) Its expert-choice router does not specialize in selecting o_1, i.e., $p_1^{(s,T)} \leq p_1^{(s,0)}$.

(iv) The alignment of any neuron r with o_1 does not grow, i.e., $\langle w_r^{(s,T)}, o_1\rangle \leq O(1)$.

(v) The alignment of any neuron r with any irrelevant patch $q \in Q$ does not grow either:

$$\langle w_r^{(s,T)}, q\rangle \leq O(1) + O\big(v^2\sqrt{\log(v)/d}\big). \tag{13}$$

The counterpart results hold for experts in S_2 specializing in learning o_2 patterns.

Lemma 2 shows that some experts in S_1, which contribute positively to the output, will specialize in selecting the o_1 token with high gating values (see (i)). A constant fraction of neurons in these experts will develop large projections along o_1 (see (ii)). In contrast, experts in S_2, which contribute negatively, do not specialize on o_1 (see (iii)). Their neurons' projections along o_1 and irrelevant directions remain bounded (see (iv)-(v)).

Thus, for batches with o_1 patterns, the positive contributions from S_1 dominate the output, while negative experts contribute comparatively little, yielding an overall positive output. Symmetrically, for o_2 patterns, experts in S_2 dominate, producing a negative output. This mechanism leads to accurate generalization, formalized below.

Theorem 1 (Generalization Guarantee of Learned Model with Reduced Training Computation)**.** *Condition 1 holds. Then the sign of $f^{(T+1)}(x)$ by the returned model at time $T+1$ accurately predicts the label y, i.e.,*

$$\mathbb{P}\left[\forall(x,y)\sim\mathcal{D}: y\cdot f^{(T+1)}(x)>0\right]=1. \tag{14}$$

Moreover, batch-level hierarchical routing reduces the total expert computation during training by up to a factor of N/k.

(14) suggests that, the MoE model can predict the class of a sample accurately, after $T=T_1+T_2+T_3$ iterations. From Condition 1, and because k and N are both order-wise smaller than the token dimension d, T_3 is order-wise larger than T_1 and T_2 and therefore dominates the total number of training iterations T. The term T_3 corresponds to the specialization of the experts and the expert-choice router, and its order matches that of expert-choice-only training in Chowdhury et al. (2024) (see Eq. (9) in Lemma 4.1). Moreover, since batches are routed only to a subset of suitable experts rather than to all experts, the number of iterations in T_3, while of the same order as in Chowdhury et al. (2024), can involve a smaller constant factor. We do not further tighten this constant, as doing so would unnecessarily complicate the analysis without affecting the order-level conclusions. Overall, the total number of iterations is comparable to, or potentially smaller than, that of expert-choice-only approaches. Furthermore, because only k out of N experts are activated in each forward and backward pass, the per-iteration computational cost is reduced. Consequently, the overall training computation can be reduced by up to a factor of N/k compared to training with expert-choice routing alone.

Because only k out of N experts are activated for each batch of tokens, it is straightforward to see that the proposed method reduces both expert-weight memory traffic and inference computation.

Theorem 2 (Reduced Memory Traffic and Inference Computation)**.** *The batch-aware hierarchical routing method can reduce both the amount of expert weights accessed and the inference computation by up to a factor of N/k, compared with expert-choice routing.*

4 Empirical Experiments

We present experimental results on language tasks, with additional results on both language and vision tasks provided in Appendix A. We consider three pretrained MoE backbones: OLMoE-1B-7B, Qwen1.5-MoE-A2.7B, and DeepSeek-MoE-16B. Starting from their pretrained token-choice routing models, we augment each MoE layer with our batch-level router and jointly fine-tune the LoRA weights and the batch-level router. For the zero-shot experiments, we keep the instruction-tuned backbone fixed and train only the batch-level router.

Batch-aware routing yields substantial system-level savings while maintaining accuracy. We compare BAR with two baselines under the same expert budget k: *static pruning*, which permanently retains the same k experts for all inputs, and *frequency-based routing* (FBR), which selects the top-k experts for each batch based on the token-level routing frequencies. Both baselines activate the same number of experts as BAR and thus provide comparisons under similar system costs.

We measure HBM traffic, L2 cache misses, latency, and throughput on all three backbones. Table 1 reports the results on ELI5 dataset using DeepSeek-MoE-16B model. Reducing the expert budget from $k=64$ to $k=24$ reduces HBM traffic and L2 misses to about 40% of the base model and improves throughput by $2.5\times$. BAR, pruning, and FBR have comparable system costs at the same expert budget, showing that the additional batch-level routing overhead is small. Moreover, BAR maintains perplexity within 2% of the base model on ELI5 even at $k=24$, whereas FBR and static pruning incur substantially larger degradation. The same trend is observed on DialogSum with OLMoE-1B-7B and ASQA with Qwen1.5-MoE-A2.7B (Appendix A.6 Tables 9 and 8), where BAR consistently achieves the best task performance among the three methods at matched expert budgets.

Out-of-domain zero-shot performance. We further examine zero-shot generalization when the instruction-tuned backbone is fixed and the batch-level router is trained for 1,000 steps on WikiText-103. We then evaluate zero-shot accuracy on MMLU, ARC, and HellaSwag. Figure 1 shows representative results on HellaSwag and MMLU. BAR remains within 3.25 percentage points of the base-model accuracy on HellaSwag while activating only 37.5% of the experts per batch, and outperforms FBR and static pruning by 3.85 and 9.43 points at the same expert budget, respectively.

Table 1: Full system-level efficiency and generalization quality on ELI5 using DeepSeek-MoE-16B (32 generated tokens, $B = 32$).

Budget (k)	Method	HBM Traffic/Step GB (% of Base)	L2 Misses/Layer Millions (% of Base)	Throughput (tokens/s)	Latency/Step (ms)	Perplexity
Base	Token-choice only	29.124 (100.0%)	31.553 (100.0%)	125.94	254.08	10.83
$k = 48$	**BAR (ours)**	21.649 (74.3%)	23.475 (74.4%)	186.86	171.26	**10.87**
	FBR	21.010 (72.1%)	22.718 (72.0%)	188.17	170.05	11.96
	Pruning	22.333 (76.7%)	24.138 (76.5%)	187.52	170.66	12.99
$k = 32$	**BAR (ours)**	14.885 (51.1%)	16.155 (51.2%)	251.68	127.14	**10.89**
	FBR	14.048 (48.2%)	15.240 (48.3%)	241.84	132.32	13.18
	Pruning	15.171 (52.1%)	16.407 (52.0%)	264.67	120.90	20.11
$k = 24$	**BAR (ours)**	11.542 (39.6%)	12.495 (39.6%)	317.58	100.77	**11.04**
	FBR	10.570 (36.3%)	11.485 (36.4%)	307.53	104.06	14.40
	Pruning	11.609 (39.9%)	12.558 (39.8%)	330.18	96.93	34.76

Results on all three benchmarks and backbones are reported in Appendix A.7. BAR can substantially reduce per-batch expert access while largely preserving out-of-domain zero-shot performance.

Robustness to batch composition. Since BAR makes an expert-selection decision at the batch level, we examine whether its performance is sensitive to how inputs are grouped. We first randomly reshuffle sequences across batches. We further construct three types of contaminated batches: *heterogeneous* batches by replacing sequences with samples from a different dataset, *boilerplate* batches with repetitive low-information text, and *adversarial* batches with sequences whose routed experts differ maximally from those of the original batch. Across 20 random reshufflings, downstream performance varies by less than 0.5% in both DBpedia-14 and DialogSum (Tables 5 and 6 in A.5). Table 2 further shows that BAR remains robust under all three types of contamination. Even with 50% adversarial contamination at $k = 32$, the model achieves 0.979 accuracy, while the selected expert subset retains a Jaccard overlap of 0.901. Similar results on DialogSum are reported in Appendix A.5.2. These results indicate that BAR is robust to incidental changes in batch composition.

Table 2: Expert-subset stability on DBpedia-14 (Switch-Base-128, N=128, B=16) at the 50% setting. Cells report accuracy with Jaccard overlap in parentheses.

k	Hetero.	Boiler.	Adv.
96	0.991 (0.962)	0.989 (0.961)	0.987 (0.950)
64	0.984 (0.945)	0.987 (0.942)	0.985 (0.925)
32	0.982 (0.911)	0.981 (0.915)	0.979 (0.901)

Reduced training computation. We finally evaluate the training cost of BAR using OLMoE-1B-7B on DialogSum. As shown in Table 3, BAR reduces total training/finetuning FLOPs to 74%–88% of the base model and achieves 1.26–$2.11\times$ wall-clock speedup, while maintaining comparable ROUGE-1 scores. These results validate the training-efficiency characterization (I2) in Section 3.1.

Table 3: Training cost for OLMoE-1B-7B on DialogSum. total FLOPs (% of base), wall-clock time [speedup], and ROUGE-1 (R-1). Base = token-choice routing only.

Cond.	FLOPs ($\times 10^{16}$)	Time (s) [Speedup]	R-1
Base	3.909 (100%)	3274.95 [1.00×]	0.470
k=48	3.446 (88.1%)	2594.48 [1.26×]	0.466
k=32	3.144 (80.4%)	2096.62 [1.56×]	0.450
k=16	2.904 (74.3%)	1553.98 [2.11×]	0.430

BAR preferentially drops low-information tokens. As k decreases, tokens routed to experts outside the batch-level expert set are dropped. Table 4 in Appendix A.3 shows that low-information tokens, such as articles and prepositions, tend to be dropped first, followed by auxiliary verbs and conjunctions, while content-bearing tokens are retained under moderate expert budgets. This observation suggests that BAR can reduce computation while preserving task-relevant information.

5 Conclusion

We proposed Batch-Aware Hierarchical MoE Routing, which integrates batch-level and token-level routing to reduce memory movement and computation while preserving model performance. We provide theoretical guarantees characterizing its training dynamics and generalization, together with reductions in training and inference computation. Empirically, BAR reduces memory traffic, inference latency, and training cost while maintaining both in-domain and out-of-domain performance, and consistently outperforms static pruning and frequency-based routing at matched expert budgets.

AI USE STATEMENT

In preparing this work, we did not use generative AI tools to generate research ideas, to design the proposed batch-aware hierarchical routing methodology, to derive any of the theoretical results (assumptions, lemmas, theorems, or proofs), or to design and execute the experiments reported in the paper; none of the experimental data or results were generated by AI tools. Generative AI tools were used only in a limited, editorial capacity to help polish the writing of the manuscript (e.g., grammar and clarity suggestions). All AI-assisted text was reviewed and verified by the authors, and we take full responsibility for the final content of this work, including all text, claims, and artifacts.

ETHICS STATEMENT

This work studies the computational and memory efficiency of Mixture-of-Experts (MoE) architectures via a batch-aware routing mechanism. All experiments use standard, publicly available benchmark datasets and existing open-source model architectures (e.g., OLMoE, Qwen1.5-MoE, Switch Transformer, V-MoE, T5-Flan); we do not collect, use, or release any private, personally identifiable, or human-subjects data, and we do not conduct crowdsourcing or human-subject studies. The contribution is algorithmic (a routing mechanism for existing models) rather than a new dataset or a new high-risk model, so no additional release safeguards are required. We are not aware of any direct negative societal impacts, fairness or privacy concerns, or conflicts of interest raised by this research.

REPRODUCIBILITY STATEMENT

We have taken several steps to support reproducibility of both the theoretical and empirical results in this paper. For the theoretical results in Section 3, the full set of assumptions, formal statements, and complete proofs are provided in the appendix: preliminaries and notation, the supporting lemmas (Appendix C), and the proofs of the main lemmas and theorem (Appendix D), with pointers from the main text to the relevant appendix results. For the empirical results in Section 4, we specify the MoE architectures and routing baselines compared, the datasets and tasks evaluated, and the key routing hyperparameters (e.g., the number of batch- and token-level experts selected); Appendix A provides additional experiments on sentiment classification, dialogue summarization, and vision tasks together with the corresponding training and evaluation hyperparameters; Appendix A.5 reports robustness checks under random batch reshuffling and degenerate batchmates, Appendix A.6 reports system-level efficiency and training-cost measurements, and Appendix A.7 reports zero-shot benchmark evaluation, each specifying the backbones, budgets, and evaluation protocol used and Appendix A.8 reports the compute resources (hardware, memory, and runtime) used to obtain these results. All datasets and pretrained models used in our experiments are publicly available, and we cite their original sources throughout. We plan to publicly release our implementation upon publication.

REFERENCES

Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. *Learning and generalization in overparameterized neural networks, going beyond two layers*. Curran Associates Inc., Red Hook, NY, USA, 2019a.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019b.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013. URL https://arxiv.org/abs/1308.3432.

Alon Brutzkus and Amir Globerson. An optimization and generalization analysis for max-pooling networks. In Cassio de Campos and Marloes H. Maathuis (eds.), *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pp. 1650–1660. PMLR, 27–30 Jul 2021. URL https://proceedings.mlr.press/v161/brutzkus21a.html.

Siyu Chen, Heejune Sheen, Tianhao Wang, and Zhuoran Yang. Unveiling induction heads: Provable training dynamics and feature learning in transformers. *Advances in Neural Information Processing Systems*, 37:66479–66567, 2024.

Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. DialogSum: A real-life scenario dialogue summarization dataset. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 5062–5074, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.449. URL https://aclanthology.org/2021.findings-acl.449.

Zixiang Chen, Yihe Deng, Yue Wu, Quanquan Gu, and Yuanzhi Li. Towards understanding the mixture-of-experts layer in deep learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 23049–23062. Curran Associates, Inc., 2022.

Mohammed Nowaz Rabbani Chowdhury, Shuai Zhang, Meng Wang, Sijia Liu, and Pin-Yu Chen. Patch-level routing in mixture-of-experts is provably sample-efficient for convolutional neural networks. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 6074–6114, 2023.

Mohammed Nowaz Rabbani Chowdhury, Meng Wang, Kaoutar El Maghraoui, Naigang Wang, Pin-Yu Chen, and Christopher Carothers. A provably effective method for pruning experts in fine-tuned sparse mixture-of-experts. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 8815–8847, 2024.

Simon S. Du, Jason D. Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts, 2014. URL https://arxiv.org/abs/1312.4314.

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.

Elias Frantar and Dan Alistarh. Qmoe: Practical sub-1-bit compression of trillion-parameter models, 2023a. URL https://arxiv.org/abs/2310.16795.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10323–10337. PMLR, 23–29 Jul 2023b. URL https://proceedings.mlr.press/v202/frantar23a.html.

Mark Horowitz. Computing's energy problem (and what we can do about it). In *IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 10–14. IEEE, 2014.

Wei Huang, Yue Liao, Yukang Chen, Jianhui Liu, Haoru Tan, Si Liu, Shiming Zhang, Shuicheng Yan, and Xiaojuan Qi. Mixture compressor for mixture-of-experts llms gains more. In *International Conference on Learning Representations (ICLR)*, 2025.

Muhammed Emrullah Ildiz, Yixiao Huang, Yingcong Li, Ankit Singh Rawat, and Samet Oymak. From self-attention to markov models: Unveiling the dynamics of generative transformers. In *International Conference on Machine Learning*, pp. 20955–20982. PMLR, 2024.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991. URL https://api.semanticscholar.org/CorpusID:572361.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Samy Jelassi, Michael Sander, and Yuanzhi Li. Vision transformers provably learn spatial structure. *Advances in Neural Information Processing Systems*, 35:37822–37836, 2022.

M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the em algorithm. In *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, volume 2, pp. 1339–1344 vol.2, 1993. doi: 10.1109/IJCNN.1993.716791.

Stefani Karp, Ezra Winston, Yuanzhi Li, and Aarti Singh. Local signal adaptivity: Provable feature learning in neural networks beyond kernels. *Advances in Neural Information Processing Systems*, 34:24883–24897, 2021.

Young Jin Kim, Rawn Henry, Raffy Fahim, and Hany Hassan Awadalla. Who says elephants can't run: Bringing large scale moe models into cloud scale production. *SustaiNLP 2022*, pp. 36, 2022.

Young Jin Kim, Raffy Fahim, and Hany Hassan Awadalla. Mixture of quantized experts (moqe): Complementary effect of low-bit quantization and robustness. *CoRR*, abs/2310.02410, 2023. URL https://doi.org/10.48550/arXiv.2310.02410.

Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Jaeseong Lee, Seung-won Hwang, Aurick Qiao, Daniel F Campos, Zhewei Yao, and Yuxiong He. STUN: Structured-then-unstructured pruning for scalable MoE pruning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13660–13676, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.671. URL https://aclanthology.org/2025.acl-long.671/.

Binghui Li, Zhixuan Pan, Kaifeng Lyu, and Jian Li. Feature averaging: An implicit bias of gradient descent leading to non-robustness in neural networks. In *The Thirteenth International Conference on Learning Representations*, 2024a.

Hongkang Li, Meng Wang, Sijia Liu, Pin-Yu Chen, and Jinjun Xiong. Generalization guarantee of training graph convolutional networks with graph topology sampling. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 13014–13051. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/li22u.html.

Hongkang Li, Meng Wang, Songtao Lu, Hui Wan, Xiaodong Cui, and Pin-Yu Chen. Transformers as multi-task feature selectors: Generalization analysis of in-context learning. In *NeurIPS 2023 Workshop on Mathematics of Modern Machine Learning*, 2023a. URL https://openreview.net/forum?id=BMQ4i2RVbE.

Hongkang Li, Meng Weng, Sijia Liu, and Pin-Yu Chen. A theoretical understanding of shallow vision transformers: Learning, generalization, and sample complexity. In *International Conference on Learning Representations*, 2023b.

Hongkang Li, Shuai Zhang, Meng Wang, Yihua Zhang, Pin-Yu Chen, and Sijia Liu. Theoretical characterization of neural network generalization with group imbalance, 2023c. URL https://openreview.net/forum?id=ig4E0Y11pX.

Hongkang Li, Meng Wang, Tengfei Ma, Sijia Liu, Zaixi Zhang, and Pin-Yu Chen. What improves the generalization of graph transformers? a theoretical dive into the self-attention and positional encoding. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024b.

Hongkang Li, Meng Wang, Tengfei Ma, Sijia Liu, Zaixi Zhang, and Pin-Yu Chen. What improves the generalization of graph transformers? a theoretical dive into the self-attention and positional encoding. In *International Conference on Machine Learning*, pp. 28784–28829. PMLR, 2024c.

Hongkang Li, Yihua Zhang, Shuai Zhang, Pin-Yu Chen, Sijia Liu, and Meng Wang. When is task vector provably effective for model editing? a generalization analysis of nonlinear transformers. In *International Conference on Learning Representations*, 2025.

Pingzhi Li, Xiaolong Jin, Yu Cheng, and Tianlong Chen. Examining post-training quantization for mixture-of-experts: A benchmark. *CoRR*, abs/2406.08155, 2024d. URL https://doi.org/10.48550/arXiv.2406.08155.

Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 8168–8177, Red Hook, NY, USA, 2018. Curran Associates Inc.

Yuchen Li, Yuanzhi Li, and Andrej Risteski. How do transformers learn topic structure: Towards a mechanistic understanding. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19689–19729. PMLR, 23–29 Jul 2023d. URL https://proceedings.mlr.press/v202/li23p.html.

Yong Lin, Lu Tan, Yifan Hao, Ho Nam Wong, Hanze Dong, Weizhong Zhang, Yujiu Yang, and Tong Zhang. Spurious feature diversification improves out-of-distribution generalization. In *12th International Conference on Learning Representations, ICLR 2024*, 2024.

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 Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6159–6172, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.334. URL https://aclanthology.org/2024.acl-long.334/.

Yuankai Luo, Hongkang Li, Lei Shi, and Xiao-Ming Wu. Enhancing graph transformers with hierarchical distance structural encoding. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 57150–57182. Curran Associates, Inc., 2024. doi: 10.52202/079017-1821.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 21702–21720. Curran Associates, Inc., 2023.

Eshaan Nichani, Alex Damian, and Jason D Lee. How transformers learn causal structure with gradient descent. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 38018–38070, 2024.

Joan Puigcerver, Carlos Riquelme, Basil Mustafa, and Neil Houlsby. From sparse to soft mixtures of experts. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=jxpsAj7ltE.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595, 2021.

Mugunthan Shandirasegaran, Hongkang Li, Songyang Zhang, Meng Wang, and Shuai Zhang. A theoretical analysis of mamba's training dynamics: Filtering relevant features for generalization in state space models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=hvpKqEYJjj.

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. In *International Conference on Learning Representations*, 2017.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=PxoFut3dWW.

Volker Tresp. Mixtures of gaussian processes. In *Neural Information Processing Systems*, 2000. URL https://api.semanticscholar.org/CorpusID:10098704.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts, 2025. URL https://openreview.net/forum?id=y1iU5czYpE.

Yanyue Xie, Zhi Zhang, Ding Zhou, Cong Xie, Ziang Song, Xin Liu, Yanzhi Wang, Xue Lin, and An Xu. Moe-pruner: Pruning mixture-of-experts large language model using the hints from its router, 2024. URL https://openreview.net/forum?id=hB6jYbvypa.

Yihao Xue, Siddharth Joshi, Eric Gan, Pin-Yu Chen, and Baharan Mirzasoleiman. Which features are learnt by contrastive learning? on the role of simplicity bias in class collapse and feature suppression. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 38938–38970, 2023.

Cheng Yang, Yang Sui, Jinqi Xiao, Lingyi Huang, Yu Gong, Yuanlin Duan, Wenqi Jia, Miao Yin, Yu Cheng, and Bo Yuan. Moe-i2: Compressing mixture of experts models through inter-expert pruning and intra-expert low-rank decomposition. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 10456–10466, 2024.

Xican Yang, Yuanhe Tian, and Yan Song. Moe pathfinder: Trajectory-driven expert pruning. 2025.

Rongjie Yi, Liwei Guo, Shiyun Wei, Ao Zhou, Shangguang Wang, and Mengwei Xu. EdgeMoE: Empowering Sparse Large Language Models on Mobile Devices . *IEEE Transactions on Mobile Computing*, 24(08):7059–7073, August 2025. ISSN 1558-0660. doi: 10.1109/TMC.2025.3546466. URL https://doi.ieeecomputersociety.org/10.1109/TMC.2025.3546466.

Jinhao Zhang, Yunquan Zhang, Boyang Zhang, Zeyu Liu, and Daning Cheng. Moqe: Improve quantization model performance via mixture of quantization experts. 2025.

Shuai Zhang, Meng Weng, Pin-Yu Chen, Sijia Liu, Songtao Lu, and Miao Liu. Joint edge-model sparse learning is provably efficient for graph neural networks. In *International Conference on Learning Representations*, 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.

Yixiao Zhou, Ziyu Zhao, Dongzhou Cheng, Zhiliang Wu, Jie Gui, Yi Yang, Fei Wu, Yu Cheng, and Hehe Fan. Dropping experts, recombining neurons: Retraining-free pruning for sparse mixture-of-experts LLMs. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 15169–15186, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.820. URL https://aclanthology.org/2025.findings-emnlp.820/.

A Additional Experiment

A.1 Experiments on Sentiment Classification tasks

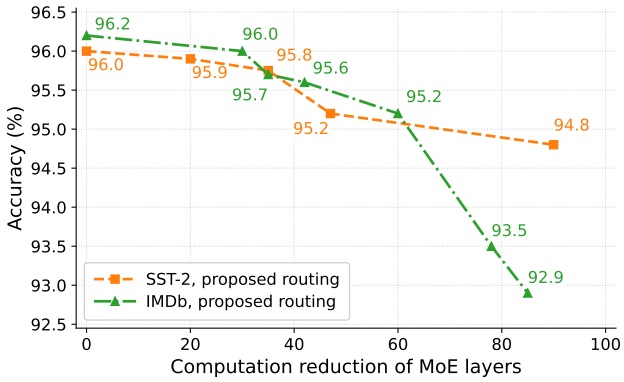


Figure 3: Computation and accuracy tradeoff for sentiment classification tasks

We further evaluate the proposed routing method on language MoE models, on sentiment classification task, using the Switch-Base-16 architecture (Fedus et al., 2022). Switch-Base-16 is a Switch Transformer with $N = 16$ experts in each sparse MoE layer and approximately 1.1B parameters. For these experiments, we add a batch-level router to each MoE layer of the pretrained model and jointly fine-tune the batch-level router, token-level expert router, and expert parameters. We evaluate the model on two sentiment classification benchmarks, SST-2 and IMDb, where the number of tokens used for the batch level decision is n=16. The number of experts selected by the batch-aware router is controlled by k. When $k = 16$, all experts remain available, and the model reduces to the standard token-choice routing baseline.

We report the inference-time computation reduction in the MoE layers as

$$\text{Computation Reduction} = \frac{N_{\text{baseline}} - N_{\text{executed}}}{N_{\text{baseline}}} \times 100,$$

where N_{baseline} denotes the number of token–expert interactions executed by the token-choice MoE baseline and N_{executed} denotes the number of token–expert interactions executed after applying the batch-level constraint. Thus, the reported reduction reflects the percentage of token–expert computations skipped in the MoE layers relative to standard token-choice routing.

Figure 3 shows the trade-off between MoE-layer computation reduction and classification accuracy on SST-2 and IMDb. Overall, the proposed routing method preserves strong classification performance even when a large fraction of MoE-layer computation is removed. On SST-2, accuracy changes only slightly as the computation reduction increases, decreasing from 96.0% with no reduction to 95.8% at 37.5% reduction, 95.2% at 50.0% reduction, and 94.8% at 87.5% reduction. IMDb shows a similar trend under moderate computation reduction: accuracy decreases from 96.2% with no reduction to 96.0% at 25.0% reduction and 95.6% at 37.5% reduction. The accuracy remains 95.2% at 62.5% reduction, but drops more noticeably when the available expert set becomes very small, reaching 93.5% and 92.9% at higher reduction levels.

These results suggest that the batch-aware routing constraint can substantially reduce MoE-layer computation during inference while maintaining competitive generalization performance on sentiment classification. The trade-off is especially favorable at moderate values of k, where the model removes a significant portion of expert computation with only minor accuracy degradation. The larger degradation on IMDb at the most aggressive reduction settings indicates that longer and more diverse reviews may require access to a broader expert set than shorter sentiment classification inputs.

Training details: For IMDb sentiment classification, we fine-tune `google/switch-base-16` using the Hugging Face IMDb dataset. We split the original training set into 90% training and 10% validation examples. Each review is formatted using the prompt template "Review: {text}\nSentiment:" and the target label is represented as the verbalized class token, either "positive" or "negative". Inputs

are truncated to a maximum length of 512 tokens using a head–tail strategy that keeps the first 384 tokens and the last 128 tokens of each review. We train using AdamW with learning rate 3×10^{-5}, weight decay 0.1, linear learning-rate decay with a warmup ratio of 0.06, gradient clipping at 1.0. The effective training batch size was 128.

A.2 Experiments on Vision tasks

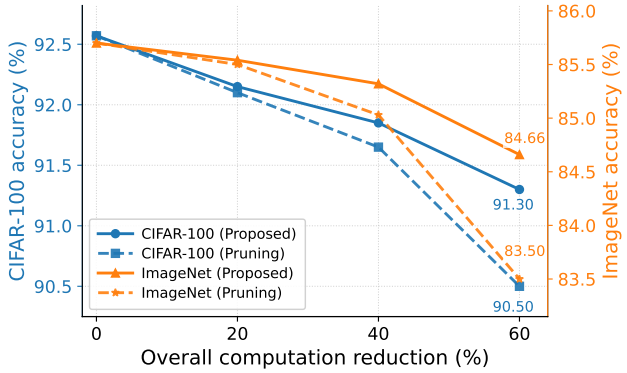


Figure 4: Computation and accuracy tradeoff for vision tasks.

We evaluated the proposed batch-aware routing method on vision classification tasks. We adopt the Vision Mixture-of-Experts (V-MoE) architecture of Riquelme et al. (2021), which extends the Vision Transformer by replacing selected feed-forward layers with sparse MoE layers. In our experiments, we use a V-MoE model based on a 12-block ViT encoder, where each odd-indexed transformer block is replaced by an MoE block with $N = 8$ experts.

The pretrained V-MoE model uses token-level routing, where each image patch is independently routed to experts. We insert a batch-level router before each MoE block. The batch-level router selects k experts out of the total $N = 8$ experts for each image, and token-level routing is then restricted to this selected expert subset. Any token in that batch whose token choice router's decision does not match with the batch-level decision is dropped. The added batch-level router is randomly initialized and jointly fine-tuned with the token-level router and expert parameters. We fine-tune the resulting model on CIFAR-100 and ImageNet (ILSVRC2012).

We report the overall inference computation reduction as

$$\text{Computation Reduction} = \frac{C_{\text{baseline}} - C_{\text{method}}}{C_{\text{baseline}}} \times 100,$$

where C_{baseline} denotes the total inference computation of the original V-MoE model and C_{method} denotes the total inference computation after applying the batch-level constraint. Since only the MoE layers are modified, this metric accounts for both the reduced expert computation inside MoE blocks and the unchanged computation in the attention, embedding, normalization, and classification layers. At the MoE-layer level, selecting k out of N experts reduces the available expert computation by a factor of k/N.

We compare our method with static expert pruning (Chowdhury et al., 2024). Unlike our method, which selects a batch-dependent subset of experts at inference time, static pruning permanently removes a fixed subset of experts after training and then applies additional fine-tuning. For a fair comparison, we match the number of active experts in both methods so that they operate under comparable computation budgets.

Joint routing outperforms static expert pruning. Figure 4 reports the trade-off between overall computation reduction and top-1 accuracy on CIFAR-100 and ImageNet. Across both datasets, the proposed routing method preserves accuracy better than static expert pruning under the same computation budget. At 40% overall computation reduction, the proposed method achieves 91.86% accuracy on CIFAR-100 and 85.33% accuracy on ImageNet, compared with 91.65% and 85.04% for static pruning. At the more aggressive 60% computation reduction setting, our method obtains

91.30% accuracy on CIFAR-100 and 84.66% accuracy on ImageNet, while static pruning drops to 90.50% and 83.50%, respectively.

These results show that retaining batch-dependent expert selection is more effective than permanently removing experts. Static pruning applies the same reduced expert set to all inputs, which can limit the model's ability to adapt to different images. In contrast, our method keeps all experts available globally but activates only a small batch-specific subset for each input. This allows the model to reduce inference computation while preserving more of the conditional computation capability of the original V-MoE model. The advantage becomes more pronounced at higher computation reduction, where removing experts permanently leads to larger accuracy degradation.

V-MoE training details: For the vision experiments, we use the public V-MoE implementation released by Riquelme et al. (2021). The repository provides code for training and fine-tuning sparse MoE vision models, and the released V-MoE B/16 checkpoint uses 8 experts on every odd transformer block. We follow this setup and use a ViT-B/16 backbone, where an input image is divided into non-overlapping 16×16 patches. For the 224×224 input resolution used in our experiments, this gives $14 \times 14 = 196$ patch tokens per image, with an additional classification token used for prediction. Each image is treated as one sample or batch of tokens.

We fine-tune the V-MoE model on CIFAR-100 and ImageNet (ILSVRC2012) using the training pipeline and data-processing utilities from the public V-MoE codebase. Starting from the released V-MoE checkpoint, we insert the proposed batch-level router before each MoE block and jointly fine-tune the batch-level router, token-level router, and expert parameters. The total number of experts in each MoE layer is $N = 8$, and we vary the number of selected experts per sample, $k \in \{2, 4, 6, 8\}$. The case $k = 8$ keeps all experts active and corresponds to the original token-level routing baseline. For $k < 8$, token-level routing is restricted to the expert subset selected by the batch-level router, reducing the MoE-layer expert computation while leaving the attention, embedding, normalization, and classification layers unchanged.

We report top-1 classification accuracy and overall computation reduction. Since the modification is applied only to MoE blocks, the reported reduction accounts for both the reduced expert computation in MoE layers and the unchanged computation in all non-MoE layers.

A.3 Visualization of Dropped Tokens under Batch-Aware Routing

We evaluate the proposed batch-aware routing mechanism using the Switch Transformer base-16 model on the IMDb sentiment classification task. The model contains 16 experts in each sparse MoE layer and uses token-choice routing as the baseline. In the baseline setting, each token is independently routed to its selected expert, and the corresponding token–expert computation is executed whenever capacity allows. In our method, the token-level routing decision is additionally constrained by a batch-level routing decision: for each batch group, the batch-level router selects a subset of experts, and a token is processed only if its token-level selected expert belongs to this batch-level expert subset.

Under this mechanism, a token is dropped from the MoE computation when its token-level selected expert does not match any of the experts selected by the batch-level router for the corresponding batch group. We compute the reduction in MoE computation as

$$\text{Computation Reduction} = \frac{N_{\text{baseline}} - N_{\text{executed}}}{N_{\text{baseline}}} \times 100,$$

where N_{baseline} denotes the number of token–expert interactions executed by the token-choice MoE baseline and N_{executed} denotes the number of token–expert interactions executed after applying the batch-level constraint. Thus, the reported reduction reflects the percentage of token–expert computations skipped in the MoE layers relative to standard token-choice routing.

Table 4 provides a qualitative visualization of this mechanism on an IMDb input example. Struck-through words indicate tokens whose token-level expert assignments are rejected because the corresponding expert is not selected by the batch-level router. The visualization shows how batch-aware routing selectively removes token–expert computations while retaining only those assignments that are consistent with both the token-level and batch-level routing decisions.

Table 4: Qualitative visualization of token dropping on an IMDb sentiment classification example using Switch Transformer base-16. Struck-through words indicate tokens whose token–expert assignments are skipped because their token-level selected experts are not included in the corresponding batch-level expert subset.

20% computation reduction at MoE layers (Articles & Prepositions Rejected)	**40% computation reduction at MoE layers (Auxiliary Verbs & Conjunctions Rejected)**
~~This~~ Academy Award winning short film can rank among ~~the~~ greatest ~~of~~ ~~the~~ genre. Told completely without dialogue, ~~it~~ ~~is~~ ~~a~~ visual treat about ~~a~~ young boy who buys ~~a~~ gold fish, lovingly places him ~~in~~ ~~a~~ bowl then goes off ~~to~~ school, leaving ~~the~~ gold fish unprotected and ~~a~~ window carelessly open. After ~~a~~ while, ~~a~~ neighboring orange tabby comes poking around, comes ~~in~~ through ~~the~~ window and heads slowly for ~~the~~ bowl. ~~The~~ fish apparently knows something ~~is~~ going on and becomes very excited. As ~~the~~ cat comes very near ~~to~~ ~~the~~ bowl, ~~the~~ fish jumps out. ~~The~~ cat catches ~~the~~ fish, drops him back ~~in~~ ~~the~~ bowl and exits through ~~the~~ window he came ~~in~~ just as ~~the~~ boy, not knowing what has happened, gets back. ~~This~~ was amazingly filmed with real animals; how Cousteau got these animals ~~to~~ behave ~~in~~ ~~this~~ manner ~~is~~ remarkable. I only wish ~~this~~ film were available now for people ~~to~~ see; I only saw ~~it~~ once, ~~in~~ 1959 when ~~it~~ was originally released, but ~~it~~ has remained unforgettable.	~~This~~ Academy Award winning short film ~~can~~ rank ~~among~~ ~~the~~ greatest ~~of~~ ~~the~~ genre. Told completely without dialogue, ~~it~~ ~~is~~ ~~a~~ visual treat ~~about~~ ~~a~~ young boy ~~who~~ buys ~~a~~ gold fish, lovingly places ~~him~~ ~~in~~ ~~a~~ bowl ~~then~~ goes off ~~to~~ school, leaving ~~the~~ gold fish unprotected ~~and~~ ~~a~~ window carelessly open. After ~~a~~ while, ~~a~~ neighboring orange tabby comes poking around, comes ~~in~~ through ~~the~~ window ~~and~~ heads slowly ~~for~~ ~~the~~ bowl. ~~The~~ fish apparently knows something ~~is~~ going on ~~and~~ becomes very excited. ~~As~~ ~~the~~ cat comes very near ~~to~~ ~~the~~ bowl, ~~the~~ fish jumps out. ~~The~~ cat catches ~~the~~ fish, drops ~~him~~ back ~~in~~ ~~the~~ bowl ~~and~~ exits through ~~the~~ window ~~he~~ came ~~in~~ just ~~as~~ ~~the~~ boy, ~~not~~ knowing ~~what~~ ~~has~~ happened, gets back. ~~This~~ ~~was~~ amazingly filmed ~~with~~ real animals; how Cousteau got ~~these~~ animals ~~to~~ behave ~~in~~ ~~this~~ manner ~~is~~ remarkable. ~~I~~ ~~only~~ wish ~~this~~ film ~~were~~ available now ~~for~~ people ~~to~~ see; ~~I~~ ~~only~~ saw ~~it~~ once, ~~in~~ 1959 ~~when~~ ~~it~~ ~~was~~ originally released, ~~but~~ ~~it~~ ~~has~~ remained unforgettable.
60% computation reduction at MoE layers (High-value Keywords Retained)	**80% computation reduction at MoE layers (Core Subject/Object/Action Retained)**
~~This~~ Academy Award winning short film ~~can~~ ~~rank~~ ~~among~~ ~~the~~ greatest ~~of~~ ~~the~~ genre. Told ~~completely~~ without dialogue, ~~it~~ ~~is~~ ~~a~~ visual treat ~~about~~ ~~a~~ young boy ~~who~~ buys ~~a~~ gold fish, ~~lovingly~~ places ~~him~~ ~~in~~ ~~a~~ bowl ~~then~~ ~~goes~~ ~~off~~ ~~to~~ school, leaving ~~the~~ gold fish unprotected ~~and~~ ~~a~~ window carelessly open. ~~After~~ ~~a~~ ~~while,~~ ~~a~~ neighboring orange tabby comes ~~poking~~ ~~around,~~ ~~comes~~ ~~in~~ ~~through~~ ~~the~~ window ~~and~~ heads ~~slowly~~ ~~for~~ ~~the~~ bowl. ~~The~~ fish ~~apparently~~ knows ~~something~~ ~~is~~ ~~going~~ ~~on~~ ~~and~~ becomes ~~very~~ excited. ~~As~~ ~~the~~ cat comes ~~very~~ near ~~to~~ ~~the~~ bowl, ~~the~~ fish jumps out. ~~The~~ cat catches ~~the~~ fish, drops ~~him~~ back ~~in~~ ~~the~~ bowl ~~and~~ exits ~~through~~ ~~the~~ window ~~he~~ ~~came~~ ~~in~~ ~~just~~ ~~as~~ ~~the~~ boy, ~~not~~ ~~knowing~~ ~~what~~ ~~has~~ happened, gets back. ~~This~~ ~~was~~ amazingly filmed ~~with~~ real animals; ~~how~~ Cousteau got ~~these~~ animals ~~to~~ behave ~~in~~ ~~this~~ manner ~~is~~ remarkable. ~~I~~ ~~only~~ wish ~~this~~ film ~~were~~ available ~~now~~ ~~for~~ people ~~to~~ see; ~~I~~ ~~only~~ saw ~~it~~ once, ~~in~~ 1959 ~~when~~ ~~it~~ ~~was~~ originally released, ~~but~~ ~~it~~ ~~has~~ remained unforgettable.	~~This~~ Academy Award ~~winning~~ ~~short~~ film ~~can~~ ~~rank~~ ~~among~~ ~~the~~ greatest ~~of~~ ~~the~~ genre. ~~Told~~ ~~completely~~ ~~without~~ dialogue, ~~it~~ ~~is~~ ~~a~~ ~~visual~~ ~~treat~~ ~~about~~ ~~a~~ ~~young~~ boy ~~who~~ buys ~~a~~ gold fish, ~~lovingly~~ ~~places~~ ~~him~~ ~~in~~ ~~a~~ bowl ~~then~~ ~~goes~~ ~~off~~ ~~to~~ school, ~~leaving~~ ~~the~~ ~~gold~~ ~~fish~~ ~~unprotected~~ ~~and~~ ~~a~~ window ~~carelessly~~ open. ~~After~~ ~~a~~ ~~while,~~ ~~a~~ ~~neighboring~~ orange tabby ~~comes~~ ~~poking~~ ~~around,~~ ~~comes~~ ~~in~~ ~~through~~ ~~the~~ ~~window~~ ~~and~~ ~~heads~~ ~~slowly~~ ~~for~~ ~~the~~ bowl. ~~The~~ fish ~~apparently~~ knows ~~something~~ ~~is~~ ~~going~~ ~~on~~ ~~and~~ ~~becomes~~ ~~very~~ excited. ~~As~~ ~~the~~ cat ~~comes~~ ~~very~~ near ~~to~~ ~~the~~ bowl, ~~the~~ fish jumps ~~out.~~ ~~The~~ cat catches ~~the~~ fish, drops ~~him~~ ~~back~~ ~~in~~ ~~the~~ bowl ~~and~~ exits ~~through~~ ~~the~~ window ~~he~~ ~~came~~ ~~in~~ ~~just~~ ~~as~~ ~~the~~ boy, ~~not~~ ~~knowing~~ ~~what~~ ~~has~~ ~~happened,~~ ~~gets~~ ~~back.~~ ~~This~~ ~~was~~ amazingly filmed ~~with~~ real animals; ~~how~~ Cousteau ~~got~~ ~~these~~ animals ~~to~~ behave ~~in~~ ~~this~~ ~~manner~~ ~~is~~ remarkable. ~~I~~ ~~only~~ ~~wish~~ ~~this~~ ~~film~~ ~~were~~ ~~available~~ ~~now~~ ~~for~~ ~~people~~ ~~to~~ ~~see;~~ ~~I~~ ~~only~~ ~~saw~~ ~~it~~ ~~once,~~ ~~in~~ 1959 ~~when~~ ~~it~~ ~~was~~ ~~originally~~ ~~released,~~ ~~but~~ ~~it~~ ~~has~~ ~~remained~~ unforgettable.

At 20% computation reduction, the removed tokens are primarily articles and prepositions. In this regime, the narrative flow remains largely intact, and sentence structure is preserved with only minor omissions. As the computation reduction increases to moderate levels (40%), auxiliary verbs and conjunctions are increasingly removed, resulting in more compact sentence constructions. While syntactic smoothness degrades, the underlying meaning of the text remains clear, indicating that the model retains essential semantic relationships despite reduced linguistic redundancy.

At higher computation reduction levels (60%–80%), the resulting text is reduced to its core semantic content, leading to a noticeable drop in fluency. Nevertheless, key entities, actions, and outcomes

remain identifiable, suggesting that the routing mechanism consistently prioritizes semantically informative tokens even under aggressive sparsification. This qualitative behavior helps explain the modest performance degradation observed in our quantitative experiments at high sparsity levels.

A.4 Comparison with expert-choice routing setup

We start from a pretrained dense encoder–decoder model, Flan-T5-Small (77M parameters), and construct an MoE model by replacing each decoder feed-forward block with a set of eight experts initialized from the pretrained dense weights, while keeping the encoder unchanged. We compare three routing strategies: token-choice routing, expert-choice routing, and our proposed batch-aware hierarchical routing. Here, the number of tokens per batch is $n = 16$. All routers are randomly initialized, and all parameters are fine-tuned end-to-end.

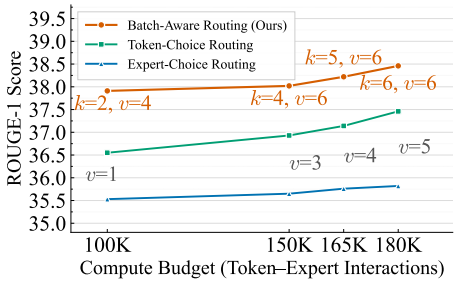


Figure 5: Comparison of our batch-aware routing with token-choice and expert-choice routing on DialogSum.

Fig. 5 compares learned models on the dialog summarization task on DialogSum dataset (Chen et al., 2021), which contains 13,460 dialogue–summary pairs. A higher ROUGE-1 score indicates better generalization. Under both token-level routing methods, the number of activated experts per batch is typically often close to all experts, whereas our method explicitly controls the number of activated experts via k. With the same inference computation, measured by the number of token–expert interactions, our method consistently achieves better generalization than both token-choice and expert-choice routing, or equivalently, requires significantly less inference computation to reach the same level of generalization, validating the theoretical insight (I3) in Section 3.1.

A.5 Robustness to Batch Composition

This section evaluates whether the batch-level routing decision is sensitive to how sequences happen to be grouped into a batch, or to the presence of degenerate batchmates (heterogeneous, repetitive boilerplate, or adversarially constructed sequences). Robustness here matters because, unlike token-choice routing, our batch-level router conditions its decision on the composition of the whole batch; if that decision were highly sensitive to incidental batch composition, the resulting computation savings could come at the cost of unpredictable accuracy swings in deployment.

A.5.1 Robustness to Random Batch Reshuffling

We first test whether simply re-partitioning the same evaluation set into different batches changes downstream performance. For a fixed batch size $B = 16$, we randomly reshuffle the test set into new batches $n_{\text{shuffle}} = 20$ times to obtain a mean and standard deviation for each value of k.

Table 5: Downstream accuracy under randomly re-batched groupings on DBpedia-14 using Switch-Base-128 ($B = 16$, 20 random reshuffles). Accuracy is reported as mean $\pm$ standard deviation across the 20 shuffles.

k	k/N	**Accuracy (mean $\pm$ std)**
128 (=N)	100%	0.996 ± 0.001
96	75%	0.993 ± 0.001
64	50%	0.991 ± 0.003
32	25%	0.987 ± 0.005

Across a classification task on DBpedia-14 (Switch-Base-128, Table 5) and a generation task on DialogSum (OLMoE-1B-7B, Table 6), the standard deviation across reshuffles is small relative to the gap between successive values of k: at $k/N = 50\%$, DBpedia-14 accuracy varies by only ± 0.003 around a mean of 0.991, and DialogSum ROUGE-1 varies by only ± 0.0012 around a mean of 0.4503.

Table 6: Downstream accuracy under randomly re-batched groupings on DialogSum using OLMoE-1B-7B ($B = 16$, 20 random reshuffles), reported as ROUGE-1 mean $\pm$ standard deviation.

k	k/N	**ROUGE-1 (mean $\pm$ std)**
64 (=N)	100%	0.4702 ± 0.0005
48	75%	0.4660 ± 0.0009
32	50%	0.4503 ± 0.0012
16	25%	0.4301 ± 0.0016

The particular way the evaluation set happens to be split into batches therefore has a negligible effect on the reported numbers, on both a classification task and a generation task.

A.5.2 Robustness to Heterogeneous, Boilerplate, and Adversarial Batchmates

A more demanding setting is when a batch contains degenerate or adversarial sequences alongside legitimate inputs. If the batch-level router were easily distracted by such sequences, an input's selected expert subset could change simply because of who else happens to share its batch. We construct three types of degenerate batchmates: *heterogeneous* (sequences drawn from a topically distinct dataset), *boilerplate* (a fixed, highly repetitive template sequence duplicated across the batch), and *adversarial* sequences (inputs whose routed experts diverge maximally from those selected for the batch's original sequences). We replace 25% or 50% of a $B = 16$ batch with one of these three sequence types and measure two quantities: task performance on the batch's original sequences, and the Jaccard overlap between the batch-level expert selection before and after contamination. An overlap of 1.0 means the injected batchmates left the batch-level decision entirely unchanged. The DBpedia-14/Switch-Base-128 results for this setting are reported in the main text as Table 2 (Section 4); Table 7 below reports the DialogSum/OLMoE-1B-7B analog.

Table 7: Expert-subset stability under heterogeneous, boilerplate, and adversarial batch composition on DialogSum using OLMoE-1B-7B ($N = 64$ total experts). Each cell reports the model's ROUGE-1 score on the batch's original sequences, with the Jaccard overlap between the batch-level expert selection before and after contamination given in parentheses. Evaluated on 256 samples with a batch size of 16.

Budget (k)	**Heterogeneous**		**Boilerplate**		**Adversarial**	
	25%	**50%**	**25%**	**50%**	**25%**	**50%**
$k = 48$	0.4637 (0.978)	0.4591 (0.974)	0.4617 (0.977)	0.4587 (0.971)	0.4602 (0.966)	0.4572 (0.959)
$k = 32$	0.4496 (0.972)	0.4488 (0.959)	0.4463 (0.965)	0.4448 (0.952)	0.4437 (0.940)	0.4418 (0.933)
$k = 16$	0.4294 (0.960)	0.4286 (0.952)	0.4279 (0.954)	0.4261 (0.944)	0.4230 (0.921)	0.4204 (0.913)

Tables 2 and 7 exhibit the same qualitative pattern on both a classification task and a generation task, and at both a large ($N = 128$) and a moderate ($N = 64$) expert count. Adversarial batchmates are the most disruptive of the three, as expected, since they are explicitly constructed to pull the batch-level decision away from the batch's own preference, with boilerplate and heterogeneous sequences following in that order. Even so, the Jaccard overlap remains above 0.91 at 25% contamination and above 0.90 at 50%, and the corresponding task performance degrades gracefully: accuracy on DBpedia-14 drops by at most 0.008 and ROUGE-1 on DialogSum by at most 0.010 across every perturbation type and severity we tested, relative to the unperturbed setting at the same budget. This robustness holds across the full range of k rather than only near $k = N$, indicating that reducing the number of batch-level experts does not make the router more brittle to batch composition. Finally, batches containing sequences of differing lengths require no separate treatment, since padded tokens are masked out before the batch-level routing score is computed and therefore contribute nothing to the batch-level decision by construction.

A.6 System-Level Efficiency Measurements

We next measure system-level efficiency directly, rather than through the proxy of token–expert interaction counts used elsewhere in the paper. We compare three ways of reducing the number of active experts at inference time, all starting from the same pretrained MoE checkpoint. For the two

baselines, we first LoRA fine-tune the model to convergence under standard token-choice routing, and refer to the resulting model as the *base* throughout; **Pruning**, a static method that permanently keeps the top-k experts per layer ranked by router-weight magnitude and discards the rest, and **FBR** (frequency-based routing), a dynamic method that for each batch keeps the top-k experts chosen by majority vote over the batch's token-level routing decisions, are then applied to that fine-tuned base. Our proposed **BAR** instead starts from the same pretrained checkpoint and LoRA fine-tunes the experts and the batch-level router jointly to convergence, so that the batch-level router, which restricts each token's choice to k of the N experts, is learned together with the experts it selects among rather than imposed after the fact. Both paths use the same early-stopping criterion, and BAR in fact reaches convergence in fewer total FLOPs and less wall-clock time than the base model (Table 3), so its advantage does not come from a larger training budget. All three methods activate exactly k experts per layer and are therefore directly comparable in memory traffic. The DeepSeek-MoE-16B results for this setting, together with perplexity, are reported in the main text as Table 1 (Section 4); we report the OLMoE-1B-7B and Qwen1.5-MoE-A2.7B analogs below.

Table 8: System-level efficiency and generation quality on ASQA using Qwen1.5-MoE-A2.7B ($N = 60$, 32 generated tokens, $B = 32$). All three methods activate k experts per layer and are therefore matched in memory traffic. Percentages are relative to the base "token-choice routing only" model. Bold marks the best value among the three methods at each budget.

Budget (k)	Method	HBM Traffic/Step GB (% of Base)	L2 Misses/Layer Millions (% of Base)	Throughput (tok/s)	Latency/Step (ms)	ROUGE-1
$N = 60$	Base	20.993 (100.0%)	25.848 (100.0%)	128.21	249.6	0.398
$k = 48$	**BAR (ours)**	17.829 (84.9%)	21.971 (85.0%)	212.86	150.34	**0.396**
	FBR	16.527 (78.7%)	20.342 (78.7%)	213.01	150.24	0.389
	Pruning	17.550 (83.6%)	21.609 (83.6%)	225.37	141.98	0.372
$k = 32$	**BAR (ours)**	12.021 (57.3%)	14.837 (57.4%)	295.09	108.45	**0.386**
	FBR	12.001 (57.2%)	14.785 (57.2%)	287.25	111.39	0.375
	Pruning	11.077 (52.8%)	13.674 (52.9%)	304.76	104.99	0.326
$k = 24$	**BAR (ours)**	7.108 (33.9%)	8.969 (34.7%)	358.32	89.31	**0.371**
	FBR	9.395 (44.8%)	11.632 (45.0%)	341.23	93.79	0.344
	Pruning	9.479 (45.2%)	11.761 (45.5%)	369.33	86.66	0.296

Table 9: System-level efficiency and generation quality on DialogSum using OLMoE-1B-7B ($N = 64$, 32 generated tokens, $B = 32$). All three methods activate k experts per layer and are therefore matched in memory traffic. Percentages are relative to the base "token-choice routing only" model. Bold marks the best value among the three methods at each budget.

Budget (k)	Method	HBM Traffic/Step GB (% of Base)	L2 Misses/Layer Millions (% of Base)	Throughput (tok/s)	Latency/Step (ms)	ROUGE-1
$N = 64$	Base	13.57 (100.0%)	24.621 (100.0%)	113.81	281.18	0.470
$k = 48$	**BAR (ours)**	10.40 (76.6%)	18.869 (76.5%)	155.92	205.24	**0.466**
	FBR	10.42 (76.8%)	18.891 (76.7%)	156.79	204.09	0.440
	Pruning	10.52 (77.5%)	19.055 (77.4%)	160.12	199.85	0.428
$k = 32$	**BAR (ours)**	7.06 (52.0%)	12.816 (52.1%)	204.72	156.31	**0.450**
	FBR	7.08 (52.2%)	12.822 (52.1%)	205.76	155.52	0.412
	Pruning	6.85 (50.5%)	11.705 (47.5%)	216.54	147.78	0.143
$k = 16$	**BAR (ours)**	3.59 (26.5%)	6.524 (26.5%)	292.69	109.33	**0.430**
	FBR	3.64 (26.8%)	6.612 (26.9%)	299.82	106.73	0.394
	Pruning	3.17 (23.4%)	5.510 (22.4%)	391.01	81.84	0.106

Table 1 in Section 4, together with Tables 9 and 8 here, report the same efficiency measurements on three backbones (DeepSeek-MoE-16B, OLMoE-1B-7B, and Qwen1.5-MoE-A2.7B), each on its own downstream task. In every case, BAR reduces HBM traffic and per-step latency by roughly the nominal k/N ratio and raises throughput by up to 2.5–2.8$\times$ over the base model, with the realized throughput gain tracking the memory reduction rather than falling short of it, which indicates that

evaluating the batch-level routing decision adds negligible overhead. No single method dominates on raw system cost. Static pruning attains the best throughput and latency at most budgets, since applying a fixed mask is the cheapest decision to execute, whereas the lowest memory traffic alternates between the three: FBR is lowest on DeepSeek-MoE-16B at every budget, while BAR is lowest on Qwen1.5-MoE at $k = 24$ (33.9% of base against pruning's 45.2%) and on OLMoE-1B-7B at $k = 48$. The three methods therefore occupy a narrow band in system cost, hence establishing themselves as comparable methods under similar memory traffic budget. As a result, the meaningful differences lie in what each preserves, in terms of performance, within that band.

On quality the ordering is unambiguous and identical across all three backbones and every budget: BAR first, FBR second, static pruning well behind. FBR is the stronger baseline and degrades gracefully, since its per-batch selection retains some of the adaptivity that BAR learns, but it trails BAR at every setting we tested, by 0.036 ROUGE-1 at $k = 16$ on DialogSum and by 0.027 at $k = 24$ on ASQA. Static pruning deteriorates far more sharply as k shrinks, because a single fixed expert subset cannot adapt to what a given batch requires: its ROUGE-1 falls to 0.106 at $k = 16$ on DialogSum against BAR's 0.430, to 0.296 at $k = 24$ on ASQA against BAR's 0.371, and its perplexity on ELI5 more than triples at $k = 24$ (34.76 against a base of 10.83) while BAR stays within 2% of the base model. Pruning's marginal advantage in throughput is therefore largest precisely where its quality loss is most severe, so the trade it offers grows worse, not better, as the budget tightens.

A.7 Zero-Shot Benchmark Evaluation

Finally, we evaluate whether the reduced expert budget selected by BAR preserves general-purpose capability, as measured by standard zero-shot benchmarks rather than the fine-tuning task the router was trained on. We take the instruction-tuned checkpoint for each model (`OLMoE-1B-7B-0125-SFT`, `Qwen1.5-MoE-A2.7B-Chat`, and `deepseek-moe-16b-chat`) and evaluate it zero-shot on MMLU, ARC, and HellaSwag. Because BAR leaves the backbone untouched here, the underlying checkpoint sets the achievable ceiling, so every number is reported relative to that same checkpoint. Pruning and FBR are applied directly to it. For BAR, since the batch-level router does not exist in the checkpoint, we train only the additional router for 1000 steps on WikiText-103, a generic language-modeling corpus unrelated to any of the three benchmarks, and then evaluate zero-shot, so that no benchmark gain can be attributed to task-specific fine-tuning. We also report the average number of experts activated per layer under each method.

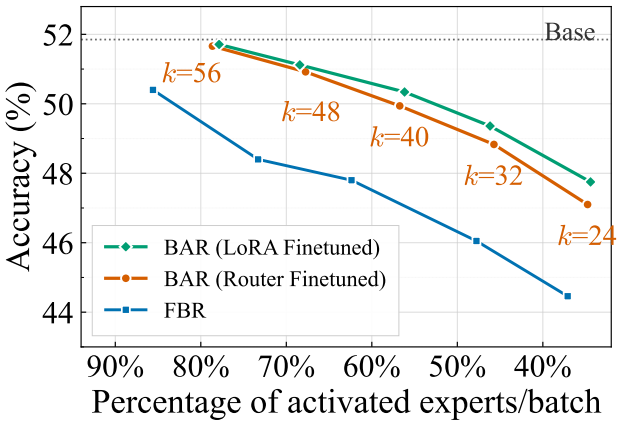


Figure 6: Zero-shot accuracy on MMLU across active expert budgets for OLMoE-1B-7B. We compare Frequency-Based Routing (FBR) against Batch-Aware Routing (BAR) under two training regimes: router-only fine-tuning and joint router with LoRA-adapted expert fine-tuning. Performance is evaluated against the unconstrained token-choice baseline.

Tables 10, 11, and 12 show that BAR preserves general-purpose zero-shot capability substantially better than Pruning and FBR at every expert budget and on all three backbones. BAR stays close to the instruction-tuned checkpoint throughout: at $k = 56$ it is within half a point on every benchmark

Table 10: Zero-shot benchmark accuracy across sparsity levels k on MMLU, ARC, and HellaSwag using OLMoE-1B-7B ($N = 64$, $B = 32$). "Avg activated expert" is the mean number of experts activated per layer. Bold marks the best accuracy and the lowest average activated experts among BAR, Pruning, and FBR at each k.

Budget (k)	Method	MMLU		ARC		HellaSwag	
		Accuracy	Avg activated expert	Accuracy	Avg activated expert	Accuracy	Avg activated expert
$N = 64$	Base	51.85	57.29	50.00	56.16	77.65	58.25
$k = 56$	**BAR (ours)**	**51.66**	**50.33**	**49.87**	**46.43**	**77.16**	**51.01**
	FBR	50.40	54.78	48.43	54.94	76.29	54.73
	Pruning	49.19	54.92	47.95	54.79	75.07	54.96
$k = 48$	**BAR (ours)**	**50.92**	**43.36**	**49.55**	**40.28**	**76.51**	**44.06**
	FBR	48.40	46.91	47.16	46.92	75.61	46.98
	Pruning	47.09	45.99	46.25	45.76	73.61	45.95
$k = 40$	**BAR (ours)**	**49.94**	**36.31**	**49.16**	**34.02**	**76.00**	**37.05**
	FBR	47.80	39.92	45.49	39.93	74.82	39.99
	Pruning	45.71	38.95	43.60	38.85	70.62	38.98
$k = 32$	**BAR (ours)**	**48.83**	**29.26**	**48.89**	**27.83**	**75.74**	**29.84**
	FBR	46.05	30.57	44.80	30.82	73.51	31.36
	Pruning	43.83	30.20	40.96	30.90	68.22	30.98
$k = 24$	**BAR (ours)**	**47.10**	**22.25**	**47.73**	**21.33**	**74.53**	**22.62**
	FBR	44.46	23.74	42.15	23.88	69.33	23.63
	Pruning	41.30	23.10	39.76	23.95	64.20	23.21

Table 11: Zero-shot benchmark accuracy across sparsity levels k on MMLU, ARC, and HellaSwag using Qwen1.5-MoE-A2.7B ($N = 60$, $B = 32$). "Avg activated expert" is the mean number of experts activated per layer. Bold marks the best accuracy and the lowest average activated experts among BAR, Pruning, and FBR at each k.

Budget (k)	Method	MMLU		ARC		HellaSwag	
		Accuracy	Avg activated expert	Accuracy	Avg activated expert	Accuracy	Avg activated expert
$N = 60$	Base	54.37	52.28	69.41	49.47	79.86	54.68
$k = 56$	**BAR (ours)**	**54.05**	48.31	**69.15**	45.62	**79.40**	**51.31**
	FBR	53.88	49.82	68.35	44.90	78.31	54.82
	Pruning	53.42	**46.67**	67.22	**43.84**	77.48	53.67
$k = 48$	**BAR (ours)**	**53.85**	**38.74**	**68.77**	39.75	**78.88**	**43.74**
	FBR	52.16	40.56	67.10	42.41	77.62	46.56
	Pruning	50.94	39.12	65.49	**37.50**	76.54	45.12
$k = 40$	**BAR (ours)**	**53.68**	**31.08**	**68.34**	**33.18**	**78.14**	**33.08**
	FBR	50.43	34.63	65.46	36.94	75.93	37.63
	Pruning	48.85	33.45	63.81	35.05	73.62	37.45
$k = 32$	**BAR (ours)**	**53.13**	**22.29**	**67.25**	**25.82**	**77.16**	**29.29**
	FBR	49.80	27.81	63.76	28.76	74.93	30.81
	Pruning	46.79	28.73	60.14	27.41	71.59	29.73
$k = 24$	**BAR (ours)**	**52.64**	**15.64**	**66.16**	**18.19**	**76.58**	**19.64**
	FBR	47.19	22.95	61.30	19.37	73.15	21.95
	Pruning	43.91	23.18	57.38	19.08	67.84	20.18

and every backbone, and even at $k = 24$ it remains within 1.7–4.8 points. The baselines separate immediately and the gap widens as k shrinks. By $k = 24$, BAR leads static pruning by 5.8–8.7 points on MMLU, by 8.0–8.8 points on ARC, and by 8.4–10.3 points on HellaSwag, where pruning falls as far as 13.5 points below the checkpoint on OLMoE-1B-7B against BAR's 3.1. FBR is the stronger

Table 12: Zero-shot benchmark accuracy across sparsity levels k on MMLU, ARC, and HellaSwag using DeepSeek-MoE-16B ($N = 64$, $B = 32$). "Avg activated expert" is the mean number of experts activated per layer. Bold marks the best accuracy and the lowest average activated experts among BAR, Pruning, and FBR at each k.

Budget (k)	Method	MMLU		ARC		HellaSwag	
		Accuracy	Avg activated expert	Accuracy	Avg activated expert	Accuracy	Avg activated expert
$N = 64$	Base	51.33	54.63	57.29	50.74	79.52	59.48
$k = 56$	**BAR (ours)**	**51.09**	**49.92**	**56.93**	**44.15**	**79.14**	**53.48**
	FBR	50.81	52.53	55.98	46.82	78.47	55.12
	Pruning	50.43	50.69	54.65	45.21	78.02	54.06
$k = 48$	**BAR (ours)**	**50.68**	**42.80**	**56.44**	**36.78**	**78.81**	**45.19**
	FBR	49.81	45.35	54.40	39.45	77.64	47.38
	Pruning	48.22	44.07	52.24	38.60	75.11	46.52
$k = 40$	**BAR (ours)**	**50.11**	**33.51**	**56.08**	**27.91**	**78.06**	**36.84**
	FBR	47.44	35.17	52.32	30.34	75.24	38.71
	Pruning	46.95	35.10	50.58	29.87	72.88	37.95
$k = 32$	**BAR (ours)**	**49.06**	**25.02**	**55.72**	**22.46**	**77.65**	**28.15**
	FBR	46.02	28.69	51.25	23.75	74.20	31.04
	Pruning	43.90	29.05	47.94	23.18	68.39	30.67
$k = 24$	**BAR (ours)**	**48.17**	**16.17**	**54.96**	**17.83**	**76.27**	**18.72**
	FBR	44.06	22.86	49.86	21.52	72.42	23.41
	Pruning	41.53	19.36	46.51	20.09	66.84	21.83

baseline, since its per-batch frequency vote captures some of the adaptivity that BAR learns, but it trails BAR by 2.6–5.5 points on MMLU and 4.9–5.6 on ARC at the same budget.

These margins are obtained while BAR activates the fewest experts of the three. The average activation counts are close at mild sparsity, but diverge as k tightens: at $k = 24$ on Qwen1.5-MoE-A2.7B, BAR averages 15.6 experts per layer against 23.2 for pruning and 23.0 for FBR, and DeepSeek-MoE-16B shows the same ordering (16.2 against 19.4 and 22.9). BAR therefore attains higher accuracy at strictly lower memory cost rather than trading one against the other. The explanation is the same one that accounts for the fine-tuning results in Section 4: because the batch-level router conditions its selection on the input batch, it retains access to whichever experts a given batch actually requires and activates no more than that, whereas static pruning commits to one fixed subset regardless of input. Broad-coverage benchmarks are the setting where that commitment is most costly, since consecutive batches draw on markedly different knowledge, and it is exactly where BAR's advantage is largest.

Moreover, Figure 6 separates the two training regimes on MMLU for OLMoE-1B-7B, plotting accuracy against the fraction of experts actually activated per batch. Training only the batch-level router and leaving the backbone frozen tracks the jointly trained router with LoRA-adapted experts closely at every budget: the two curves are indistinguishable at $k = 56$ (0.09 points apart) and separate only gradually as the budget tightens, reaching 0.65 points at $k = 24$. The accuracy BAR retains is therefore attributable to the routing decision itself rather than to adaptation of the expert parameters, and the numbers reported in Tables 10–12 are those of the router-only regime, the more conservative of the two. From the figure, we can observe that FBR lies well below both variants across the whole range.

A.8 Compute Resources

We report the compute resources used in our experiments to clarify the hardware requirements for reproducing the results. All experiments, including LoRA fine-tuning, batch-aware router training, throughput benchmarking, and hardware-counter profiling, were conducted on a local multi-GPU machine with 8 NVIDIA RTX A5000 GPUs, each with 24GB of memory, shared with other users on the cluster. No experiment required the full 8-GPU machine or a distributed training framework.

Our target models are OLMoE-1B-7B (6.9B total and 1.3B active parameters, 64 routed experts, 16 sparse decoder layers), Qwen1.5-MoE-A2.7B-Chat (14.3B total and 2.7B active parameters, 60 routed experts, 24 sparse decoder layers), and DeepSeek-MoE-16B-Chat (16.4B total parameters, 64 routed experts, 27 sparse and 1 dense decoder layer). The latter two exceed a single 24GB GPU in bfloat16, requiring approximately 28–33GB for weights alone, but fit comfortably across two GPUs, while OLMoE-1B-7B fits on a single device. Because all fine-tuning was parameter-efficient, with LoRA adapters together with the batch-aware router's own lightweight parameters, the trainable parameter count was on the order of 7–11M per model, and the resulting optimizer-state memory did not require parameter sharding. Each of the larger models was therefore loaded once with automatic device mapping across an NVLink-paired GPU pair under an explicit per-GPU memory cap, which pipeline-splits the frozen base model's layers across the two devices while the adapter and router parameters and their optimizer states fit alongside. This allowed several independent experiments to run concurrently on different GPU pairs of the same machine rather than requiring exclusive use of all 8 GPUs.

Beyond training, a substantial share of the compute went to system-level profiling rather than model fitting. Hardware counters were collected directly from the GPU, covering HBM read and L2 cache behaviour, across a sweep of expert-restriction levels for all three models. Profiling used a two-process design in which an unprofiled forward pass first captures realistic decode-step activations and a separate instrumented pass then collects counters, so that profiling overhead remains isolated from model loading and generation.

B PRELIMINARIES

Gradient Calculations: The gradients of the loss with respect to the model's weights for an input-output pair (x, y) are as follows.

Expert Neuron Gradient:

$$\frac{\partial l^{(t)}(x,y)}{\partial w_r^{(s,t)}} = -ya^{(s)}G^{(s)} \sum_{j \in J_{\mathrm{P}}^{(s,t)}(x)} H_j^{(s,t)} x^{(j)} \cdot \mathbf{1}_{\langle w_r^{(s,t)}, x^{(j)} \rangle \geq 0}$$

Expert-choice Router Gradient:

$$\frac{\partial l^{(t)}(x,y)}{\partial w_{\mathrm{P}}^{(s,t)}} = -ya^{(s)}G^{(s)} \sum_{j \in J_{\mathrm{P}}^{(s,t)}(x)} H_j^{(s)} \sigma_j^{(s)} \sum_{i \in J_{\mathrm{P}}^{(s,t)} \setminus \{j\}} H_i^{(s)} (x^{(j)} - x^{(i)})$$

Batch-level Router Gradient:

$$\begin{aligned}\frac{\partial l^{(t)}(x,y)}{\partial w_{\mathrm{S}}^{(s,t)}} = &- ya^{(s)}G^{(s)}(1-G^{(s)}) \sum_{i \in [n]} x^{(i)} \sum_{j \in J_{\mathrm{P}}^{(s,t)}(x)} H_j^{(s,t)} \sigma_j^{(s,t)} \\ &+ y \sum_{ii \in J_{\mathrm{S}}^{(t)}(x) \setminus \{s\}} a^{(ii)} G^{(s)} G^{(ii)} \sum_{i \in [n]} x^{(i)} \sum_{j \in J_{\mathrm{P}}^{(s,t)}(x)} H_j^{(ii,t)} \sigma_j^{(ii,t)}\end{aligned}$$

Batch Gradients: Using Stochastic Gradient Descent (SGD), the batch gradient for a batch B_t of size B is the average of the gradients for each sample in the batch. For the expert weights, expert-choice router, and batch-level router, respectively, these are:

$$\frac{\partial l^{(t)}}{\partial w_r^{(s,t)}} = \frac{1}{B} \sum_{(x,y) \in B_t} \frac{\partial l^{(t)}(x,y)}{\partial w_r^{(s,t)}}$$

$$\frac{\partial l^{(t)}}{\partial w_{\mathrm{P}}^{(s,t)}} = \frac{1}{B} \sum_{(x,y) \in B_t} \frac{\partial l^{(t)}(x,y)}{\partial w_{\mathrm{P}}^{(s,t)}}$$

$$\frac{\partial l^{(t)}}{\partial w_{\mathrm{S}}^{(s,t)}} = \frac{1}{B} \sum_{(x,y) \in B_t} \frac{\partial l^{(t)}(x,y)}{\partial w_{\mathrm{S}}^{(s,t)}}$$

Algorithm: The training process is divided into two distinct stages: an initial *Exploration Stage* where the expert and expert-choice router weights are trained while the batch-level router remains fixed, followed by a *Learning Stage* where all weights are updated. The procedure is detailed in Algorithm 1.

Algorithm 1 The Training Algorithm for Theoretical Analysis

Require: Training data $\{(x_i, y_i)\}_{i=1}^{N_{\text{sample}}}$; learning rates $\eta_{\text{S}}, \eta_{\text{P}}, \eta_{\text{e}}$; time steps T_1, T; batch size B.

Stage 1: Exploration Stage

for $t = 0, 1, \ldots, T_1 - 1$ **do**

  $w_r^{(s,t+1)} \leftarrow w_r^{(s,t)} - \eta_{\text{e}} \widetilde{\nabla}_{w_r}^{(s,t)} \quad \forall r \in [m], s \in J_{\text{S}}^{(t)}(x)$

  $w_{\text{P}}^{(s,t+1)} \leftarrow w_{\text{P}}^{(s,t)} - \eta_{\text{P}} \widetilde{\nabla}_{w_{\text{P}}}^{(s,t)} \quad \forall s \in J_{\text{S}}^{(t)}(x)$

  $w_{\text{S}}^{(s,t+1)} \leftarrow w_{\text{S}}^{(s,t)}$ ▷ batch-level router is frozen

end for

Stage 2: Learning Stage

for $t = T_1, \ldots, T - 1$ **do**

  $w_r^{(s,t+1)} \leftarrow w_r^{(s,t)} - \eta_{\text{e}} \widetilde{\nabla}_{w_r}^{(s,t)} \quad \forall r \in [m], s \in J_{\text{S}}^{(t)}(x)$

  $w_{\text{P}}^{(s,t+1)} \leftarrow w_{\text{P}}^{(s,t)} - \eta_{\text{P}} \widetilde{\nabla}_{w_{\text{P}}}^{(s,t)} \quad \forall s \in J_{\text{S}}^{(t)}(x)$

  $w_{\text{S}}^{(s,t+1)} \leftarrow w_{\text{S}}^{(s,t)} - \eta_{\text{S}} \widetilde{\nabla}_{w_{\text{S}}}^{(s,t)} \quad \forall s \in J_{\text{S}}^{(t)}(x)$

end for

NOTATIONS AND DEFINITIONS

Asymptotic Notation. We use $\widetilde{O}(\cdot)$ and $\widetilde{\Omega}(\cdot)$ to hide poly-logarithmic factors, i.e., $\text{poly}(\log(d))$. "With high probability" (w.h.p.) refers to a probability of at least $1 - 1/\text{poly}(d)$.

Model Components. We define the following values for a given input x and expert s at time t:

- $G^{(s,t)}(x)$: The normalized routing score from the batch-level router for expert s.
- $H_j^{(s,t)}(x)$: The normalized routing score from the expert-choice router of expert s for the jth patch. For simplicity, we denote $H_1^{(s,t)}$ and $H_2^{(s,t)}$ for task-specific patches o_1 and o_2. Similarly, we denote $H_q^{(s,t)}$ for any task-irrelevant patch $q \in Q$.
- $l_q^{(s,t)}(x)$: The number of copies of a task-irrelevant pattern q among the top-v patches selected by the expert-choice router.

Expert Sets. We restate the definitions of the sets of positively and negatively contributing experts:

$$S_1 := \{s \in [N] : a^{(s)} = +1\}$$
$$S_2 := \{s \in [N] : a^{(s)} = -1\}$$

Probability Measures. We define the following conditional probabilities for observing specific patches among the top-v selected patches:

$$p_1^{(s,t)} := \mathbb{P}\left[\exists j \in J_{\text{P}}(x) \text{ s.t. } x^{(j)} = o_1 \text{ and } H_1^{(s,0)} \geq \frac{1}{v} \mid y = 1\right]$$
$$p_2^{(s,t)} := \mathbb{P}\left[\exists j \in J_{\text{P}}(x) \text{ s.t. } x^{(j)} = o_2 \text{ and } H_2^{(s,0)} \geq \frac{1}{v} \mid y = -1\right]$$
$$p_q^{(s,t)} := \mathbb{P}\left[\exists j \in J_{\text{P}}(x) \text{ s.t. } x^{(j)} = q \text{ and } H_q^{(s,0)} \geq \frac{1}{v}\right] \quad \forall q \in Q$$

Expert Outputs. The output of an expert's hidden neurons for a specific patch is denoted by σ. In expert s, the output for the jth patch is denoted by $\sigma_j^{(s,t)}(x)$ or $\sigma_j^{(s,t)}$. For simplicity, we denote $\sigma_1^{(s,t)}$ and $\sigma_2^{(s,t)}$ for task-specific patches o_1 and o_2. Similarly, we denote $\sigma_q^{(s,t)}$ for any task-irrelevant patch

$q \in Q$. Which means,

$$\sigma_j^{(s,t)}(x) := \sum_{r=1}^{m} \text{ReLU}(\langle w_r^{(s,t)}, x^{(j)} \rangle)$$
$$\sigma_1^{(s,t)} := \sum_{r=1}^{m} \text{ReLU}(\langle w_r^{(s,t)}, o_1 \rangle) \quad \text{and} \quad \sigma_2^{(s,t)} := \sum_{r=1}^{m} \text{ReLU}(\langle w_r^{(s,t)}, o_2 \rangle)$$
$$\sigma_q^{(s,t)} := \sum_{r=1}^{m} \text{ReLU}(\langle w_r^{(s,t)}, q \rangle) \quad \text{for any pattern } q \in Q$$

From these, we define the difference δ between task-relevant and task-irrelevant pattern outputs:

$$\delta_{1,q}^{(s,t)} := \sigma_1^{(s,t)} - \sigma_q^{(s,t)}$$
$$\delta_{2,q}^{(s,t)} := \sigma_2^{(s,t)} - \sigma_q^{(s,t)}$$
$$\delta_{q,q'}^{(s,t)} := \sigma_q^{(s,t)} - \sigma_{q'}^{(s,t)}$$

Initial Weight Norms. We define constants representing the maximum L_2 norms of the initial weights:

$$C_0 := \max_{s \in [N]} \|w_{\text{S}}^{(s,0)}\|_2$$
$$C_1 := \max_{s \in [N]} \|w_{\text{P}}^{(s,0)}\|_2$$
$$C_2 := \max_{s \in [N], r \in [m]} \|w_r^{(s,0)}\|_2$$

We analyze the case where $v \geq e^{2C_1}$, which means $||w_{\text{P}}^{s,0}|| \leq \frac{1}{2}\log(v)$; for $s \in [N]$. Also we consider, $||w_{\text{S}}^{(s,p,0)}|| \leq \frac{\log(k)}{2n}$ for $s \in [N]$ and $p \in [n]$

Definition 1 (Important and Unimportant Experts)**.** An expert $s \in [N]$ is **important** for the class-1 task-specific feature o_1 if $p_1^{(s,0)} = \Omega(1)$, and **unimportant** if $p_1^{(s,0)} = O(1/d)$. Similarly, an expert is **important** for the class-2 feature o_2 if $p_2^{(s,0)} = \Omega(1)$, and **unimportant** if $p_2^{(s,0)} = O(1/d)$.

Assumptions on Pre-trained Model. We assume that the pre-trained model contains at least one class-1 important expert in the set $S_1 = \{s \in S : a^{(s)} = +1\}$ and at least one class-2 important expert in $S_2 = \{s \in S : a^{(s)} = -1\}$. This is a minimal assumption, requiring only one functionally important expert per class.

For any class-1 important expert s, we assume its expert-choice router weights are minimally separated. Specifically, the component along o_1 (i.e., $\langle w_{\text{P}}^{(s,0)}, o_1 \rangle$) is separated from any irrelevant pattern $q \in Q$ by at least $\Omega(1/d)$, such that $|\langle w_{\text{P}}^{(s,0)}, o_1 - q \rangle| = C_{\text{P}}$, where $C_p = \Theta(1/d)$. Moreover, as $G^{(s,0)}$ is calculated by performing softmax operation over k values, we can assume, during the initial case, $G^{(s,0)} = \Theta(\frac{1}{k})$.

Furthermore, we assume that for any class-1 important expert s, a constant fraction of its neurons are activated by the feature o_1, meaning $|\{r \in [m] : \langle w_r^{(s,0)}, o_1 \rangle > 0\}| = \Omega(1)$. Similar assumptions are made for class-2 important experts. These minimal conditions are necessary for an expert to be considered important for a task.

COMPONENTS OF THE EXPERT'S GRADIENT

We decompose the gradient of a hidden neuron r in any selected expert s with respect to the task-specific patterns (o_1, o_2) and any task-irrelevant pattern $q \in Q$ as follows:

$$\left\langle \frac{\partial l(x,y)}{\partial w_r^{(s,t)}}, q \right\rangle = \begin{cases} 0 & \text{if } \langle w_r^{(s,t)}, q\rangle < 0 \\ 0 & \text{if } \langle w_r^{(s,t)}, q\rangle \geq 0 \text{ but } \nexists j \in J_{\text{P}}^{(s,t)} : x^{(j)} = q \\ -ya^{(s)} G^{(s,t)} l_q^{(s,t)} H_q^{(s,t)} & \text{if } \langle w_r^{(s,t)}, q\rangle \geq 0 \text{ and } \exists j \in J_{\text{P}}^{(s,t)} : x^{(j)} = q \end{cases}$$

$$\left\langle \frac{\partial l(x,y)}{\partial w_r^{(s,t)}}, o_1 \right\rangle = \begin{cases} 0 & \text{if } y = -1 \text{ or } \langle w_r^{(s,t)}, o_1\rangle < 0 \\ 0 & \text{if } y = +1, \langle w_r^{(s,t)}, o_1\rangle \geq 0 \text{ but } \nexists j \in J_{\text{P}}^{(s,t)} : x^{(j)} = o_1 \\ -a^{(s)} G^{(s,t)} H_1^{(s,t)}(x) & \text{if } y = +1, \langle w_r^{(s,t)}, o_1\rangle \geq 0 \text{ and } \exists j \in J_{\text{P}}^{(s,t)} : x^{(j)} = o_1 \end{cases}$$

$$\left\langle \frac{\partial l(x,y)}{\partial w_r^{(s,t)}}, o_2 \right\rangle = \begin{cases} 0 & \text{if } y = +1 \text{ or } \langle w_r^{(s,t)}, o_2\rangle < 0 \\ 0 & \text{if } y = -1, \langle w_r^{(s,t)}, o_2\rangle \geq 0 \text{ but } \nexists j \in J_{\text{P}}^{(s,t)} : x^{(j)} = o_2 \\ a^{(s)} G^{(s,t)} H_2^{(s,t)}(x) & \text{if } y = -1, \langle w_r^{(s,t)}, o_2\rangle \geq 0 \text{ and } \exists j \in J_{\text{P}}^{(s,t)} : x^{(j)} = o_2 \end{cases}$$

COMPONENTS OF THE EXPERT-CHOICE ROUTER'S GRADIENT

Similarly, we decompose the gradient of the expert-choice router for any selected expert s along the different patterns. For o_1, o_2 and any $q \in Q$:

$$\left\langle \frac{\partial l(x,y)}{\partial w_{\text{P}}^{(s,t)}}, q \right\rangle = \begin{cases} 0 & \text{if } \nexists j \in J_P^{(s,t)} : x^{(j)} = q \\ ya^{(s)} G^{(s,t)} l_q^{(s,t)} H_q^{(s,t)} \sum\limits_{j \in J_{\text{P}}^{(t)}(x) \setminus \{i | x^{(i)} = q\}} H_j^{(s,t)} (\sigma_j^{(s,t)} - \sigma_q^{(s,t)}) & \text{if } \exists j \in J_P^{(s,t)} : x^{(j)} = q \end{cases}$$

$$\left\langle \frac{\partial l(x,y)}{\partial w_{\text{P}}^{(s,t)}}, o_1 \right\rangle = \begin{cases} 0 & \text{if } y = -1 \text{ or } \nexists j \in J_P^{(s,t)} : x^{(j)} = o_1 \\ a^{(s)} G^{(s,t)} H_1^{(s,t)}(x) \sum\limits_{j \in J_{\text{P}}^{(t)}(x) \setminus \{i | x^{(i)} = o_1\}} H_j^{(s,t)} (\sigma_j^{(s,t)} - \sigma_1^{(s,t)}) & \text{if } y = +1 \text{ and } \exists j \in J_P^{(s,t)} : x^{(j)} = o_1 \end{cases}$$

$$\left\langle \frac{\partial l(x,y)}{\partial w_{\text{P}}^{(s,t)}}, o_2 \right\rangle = \begin{cases} 0 & \text{if } y = +1 \text{ or } \nexists j \in J_P^{(s,t)} : x^{(j)} = o_2 \\ -a^{(s)} G^{(s,t)} H_2^{(s,t)}(x) \sum\limits_{j \in J_{\text{P}}^{(t)}(x) \setminus \{i | x^{(i)} = o_2\}} H_j^{(s,t)} (\sigma_j^{(s,t)} - \sigma_2^{(s,t)}) & \text{if } y = -1 \text{ and } \exists j \in J_P^{(s,t)} : x^{(j)} = o_2 \end{cases}$$

C SUPPORTING LEMMAS FOR THEOREM 1

Lemma 3 (Concentration of Batch Statistics)**.** *Let $\mathcal{D}_1$ be a subset of a domain $\mathcal{D}$, where the probability of an element being in this subset is $p = \mathbb{P}[(x,y) \sim \mathcal{D} : (x,y) \in \mathcal{D}_1]$. Then, with high probability over any randomly selected mini-batch $\mathcal{B}_k$ of size B from $\mathcal{D}$, the number of samples from $\mathcal{D}_1$ is close to its expectation:*

$$|\mathcal{B}_k \cap \mathcal{D}_1| - B \cdot p| = \tilde{O}(\sqrt{B})$$

Proof. Let us define a Bernoulli random variable Y for any sample $(x,y) \sim \mathcal{D}$:

$$Y := \begin{cases} 1 & \text{if } (x,y) \in \mathcal{D}_1 \\ 0 & \text{if } (x,y) \notin \mathcal{D}_1 \end{cases}$$

Thus, $Y \sim \text{Bernoulli}(p)$. For a mini-batch $\mathcal{B}_k$ of size B, we have B i.i.d. random variables, $Y_1, \ldots, Y_B$. The number of elements in the batch from $\mathcal{D}_1$ is the sum of these variables:

$$|\mathcal{B}_k \cap \mathcal{D}_1| = \sum_{j=1}^{B} Y_j$$

The expected value of this sum is $\mathbb{E}\left[|\mathcal{B}_k \cap \mathcal{D}_1|\right] = \sum_{j=1}^{B} \mathbb{E}[Y_j] = Bp$. By applying Hoeffding's inequality, the sum will be concentrated around its mean. Specifically, for a deviation of ϵ, the probability is bounded by $\mathbb{P}\left[|\sum Y_j - Bp| \geq \epsilon\right] \leq 2e^{-2\epsilon^2/B}$. Setting a deviation of $\tilde{O}(\sqrt{B})$ ensures this holds with high probability, completing the proof. □

Lemma 4 (Initial Gradient Bounds for Important Experts of Class 1)**.** *For any important expert $s \in \{i \in S_1 : p_1^{(i,0)} = \Omega(1)\}$, the gradients of the loss with respect to the initial weights are bounded with high probability over a randomly sampled batch:*

(i) $\left|\left\langle \frac{\partial l}{\partial w_P^{(s,0)}}, q \right\rangle\right| \leq O\left(\frac{mC_2}{d}\right) + \tilde{O}\left(\frac{mC_2}{\sqrt{B}}\right)$

(ii) $\left|\left\langle \frac{\partial l}{\partial w_P^{(s,0)}}, o_1 \right\rangle\right| \leq O(mC_2) + \tilde{O}\left(\frac{mC_2}{\sqrt{B}}\right)$

(iii) $\left\langle \frac{\partial l}{\partial w_r^{(s,0)}}, o_1 \right\rangle \leq -\frac{1}{2kv} + \tilde{O}\left(\frac{1}{kv\sqrt{B}}\right)$ *and* $\left\langle \frac{\partial l}{\partial w_r^{(s,0)}}, o_1 \right\rangle \geq -\frac{1}{2} - \tilde{O}\left(\frac{1}{\sqrt{B}}\right)$

(iv) $\left\langle \frac{\partial l}{\partial w_r^{(s,0)}}, o_2 \right\rangle \geq 0$ *and* $\left\langle \frac{\partial l}{\partial w_r^{(s,0)}}, o_2 \right\rangle \leq \frac{1}{2} + \tilde{O}\left(\frac{1}{\sqrt{B}}\right)$

(v) $\left|\left\langle \frac{\partial l}{\partial w_r^{(s,0)}}, q \right\rangle\right| \leq O\left(\frac{1}{d}\right) + \tilde{O}\left(\frac{1}{\sqrt{B}}\right)$

Proof. The proof proceeds by analyzing each statement individually. The proof for statement (i) is provided in detail, as its method forms the basis for the subsequent proofs.

First, we note that the difference between expert outputs for relevant and irrelevant patterns is bounded: $|\delta_{1,q}^{(s,0)}| = |\sum_{r=1}^{m} \text{ReLU}(\langle w_r^{(s,t)}, o_1\rangle) - \sum_{r=1}^{m} \text{ReLU}(\langle w_r^{(s,t)}, q\rangle)| = O(mC_2)$. Similarly, we can also show that $|\delta_{2,q}^{(s,0)}| = O(mC_2)$ and $|\delta_{q,q'}^{(s,0)}| = O(mC_2)$.

Proof of (i) The batch gradient can be expressed as an average over the initial batch B_0:

$$\left\langle \frac{\partial l}{\partial w_{\text{P}}^{(s,0)}}, q \right\rangle = \frac{1}{B} \sum_{(x,y)\in B_0} \left\langle \frac{\partial l(x,y)}{\partial w_{\text{P}}^{(s,0)}}, q \right\rangle$$

We define a set $S_q := \{(x,y) \sim D : \exists j \in J_{\text{P}}^{(s,0)} \text{ s.t. } x^{(j)} = q\}$, which contains samples relevant to the pattern $q \in Q$. The probability of a sample belonging to this set is $p_q^{(s,0)} = O(1/d)$.

We know that the gradient component is zero for any sample not in S_q. Therefore, the sum is only over the relevant subset of the batch:

$$\left\langle \frac{\partial l}{\partial w_{\text{P}}^{(s,0)}}, q \right\rangle = \frac{1}{B} \sum_{(x,y)\in B_0\cap S_q} \left\langle \frac{\partial l(x,y)}{\partial w_{\text{P}}^{(s,0)}}, q \right\rangle$$

The magnitude is thus bounded by the fraction of the batch in S_q times the output difference bound:

$$\left|\left\langle \frac{\partial l}{\partial w_{\text{P}}^{(s,0)}}, q \right\rangle\right| \leq \frac{|B_0 \cap S_q|}{B} O(mC_2)$$

From Lemma 3 , we know that with high probability, the batch fraction is close to the true probability: $\frac{|B_0\cap S_q|}{B} \leq p_q^{(s,0)} + \tilde{O}(\frac{1}{\sqrt{B}})$. Substituting $p_q^{(s,0)} = O(1/d)$ yields the final result.

Proof of (ii) Considering $p_1^{(s,0)} = \Omega(1)$, the proof follows the same procedure as in statement (i).

Proof of (iii) Considering $p_1^{(s,0)} = \Omega(1)$, the proof follows the same procedure as in statement (i).

Proof of (iv) As $p_2^{(s,0)} \geq 0$, using the same the procedure from statement (i), we can show that $\left\langle \frac{\partial l}{\partial w_r^{(s,0)}}, o_2 \right\rangle \geq 0$. Similarly, as $G^{(s,0)}(x) \leq 1$, the same procedure yields the upper bound $\left\langle \frac{\partial l}{\partial w_r^{(s,0)}}, o_2 \right\rangle \leq \frac{1}{2} + \tilde{O}(\frac{1}{\sqrt{B}})$.

Proof of (v) As we know that for any $(x, y) \sim D$, $l_q^{(s,t)}(x)G_q^{(s,t)}(x) \leq 1$, the proof is completed by following the same procedure as in statement (i). □

Lemma 5 (Initial Gradient Bounds for Important Experts of Class 2)**.** *For any important class-2 expert $s \in \{i \in S_2 : p_2^{(i,0)} = \Omega(1)\}$, the gradients of the loss with respect to the initial weights are bounded with high probability over a randomly sampled batch:*

$$\textit{(i)} \quad \left|\left\langle \frac{\partial l}{\partial w_P^{(s,0)}}, q \right\rangle\right| \leq O\left(\frac{mC_2}{d}\right) + \tilde{O}\left(\frac{mC_2}{\sqrt{B}}\right)$$

$$\textit{(ii)} \quad \left|\left\langle \frac{\partial l}{\partial w_P^{(s,0)}}, o_2 \right\rangle\right| \leq O(mC_2) + \tilde{O}\left(\frac{mC_2}{\sqrt{B}}\right)$$

$$\textit{(iii)} \quad \left\langle \frac{\partial l}{\partial w_r^{(s,0)}}, o_2 \right\rangle \leq -\frac{1}{2kv} + \tilde{O}\left(\frac{1}{v\sqrt{B}}\right) \quad \textit{and} \quad \left\langle \frac{\partial l}{\partial w_r^{(s,0)}}, o_2 \right\rangle \geq -\frac{1}{2} - \tilde{O}\left(\frac{1}{\sqrt{B}}\right)$$

$$\textit{(iv)} \quad \left\langle \frac{\partial l}{\partial w_r^{(s,0)}}, o_1 \right\rangle \geq 0 \quad \textit{and} \quad \left\langle \frac{\partial l}{\partial w_r^{(s,0)}}, o_1 \right\rangle \leq \frac{1}{2} + \tilde{O}\left(\frac{1}{\sqrt{B}}\right)$$

$$\textit{(v)} \quad \left|\left\langle \frac{\partial l}{\partial w_r^{(s,0)}}, q \right\rangle\right| \leq O\left(\frac{1}{d}\right) + \tilde{O}\left(\frac{1}{\sqrt{B}}\right)$$

Proof. The proof is analogous to the proof of Lemma 4. By applying the same techniques and conditioning on the expert being in the set $\{i \in S_2 : p_2^{(i,0)} = \Omega(1)\}$, we can establish the bounds from (i) to (v). □

Lemma 6. *At $t = T_1 = \Omega(\frac{NvC_2}{\eta_e})$ iterations and $\eta_P = O\Big(\frac{\eta_e C_P}{mC_2^2 k^2 v^2}\Big)$, for any important class-1 expert $s \in \{i \in S_1 : p^{(i,0)} = \Omega(1)\}$, the following hold:*

(i) The expert outputs have separated: $\delta_{1,q}^{(s,T_1)} = \Omega(mC_2)$, while other differences remain bounded: $\delta_{2,q}^{(s,T_1)} = O(mC_2)$ and $\delta_{q,q'}^{(s,T_1)} = O(mC_2)$.

(ii) For any pattern q with initial alignment $\langle w_P^{(s,0)}, o_1 - q\rangle > 0$, the alignment is preserved and bounded: $\Omega(C_P) < \langle w_P^{(s,T_1)}, o_1 - q\rangle < \log(v)$.

(iii) For any pattern q with initial alignment $\langle w_P^{(s,0)}, o_1 - q\rangle < 0$, the alignment is preserved and bounded: $\Omega(C_P) < \langle w_P^{(s,T_1)}, q - o_1\rangle < \log(v)$.

(iv) The probability of selecting the correct patch remains stable: $p_1^{(s,T_1)} = p_1^{(s,0)}$.

Proof. From Lemma 4 and considering a batch size of $B = \Omega(v^2 d^2)$, we get:

$$\left|\left\langle \frac{\partial l}{\partial w_{\mathrm{P}}^{(s,0)}}, o_1 \right\rangle\right| \leq O(mC_2), \quad \left|\left\langle \frac{\partial l}{\partial w_{\mathrm{P}}^{(s,0)}}, q \right\rangle\right| \leq O\Big(\frac{mC_2}{d}\Big).$$

As a result,

$$\begin{aligned}\left\langle w_{\mathrm{P}}^{(s,1)}, o_1 \right\rangle &= \left\langle w_{\mathrm{P}}^{(s,0)}, o_1 \right\rangle - \eta_{\mathrm{P}} \cdot \left\langle \frac{\partial l(x,y)}{\partial w_{\mathrm{P}}^{(s,0)}}, o_1 \right\rangle \\ &\geq \left\langle w_{\mathrm{P}}^{(s,0)}, o_1 \right\rangle - O(mC_2 \cdot \eta_{\mathrm{P}}).\end{aligned}$$

Similarly,

$$\begin{aligned}\left\langle w_{\mathrm{P}}^{(s,1)}, q \right\rangle &= \left\langle w_{\mathrm{P}}^{(s,0)}, q \right\rangle - \eta_{\mathrm{P}} \cdot \left\langle \frac{\partial l}{\partial w_{\mathrm{P}}(s,0)}, q \right\rangle \\ &\leq \left\langle w_{\mathrm{P}}^{(s,0)}, q \right\rangle + O\Big(\tfrac{mC_2 \cdot \eta_{\mathrm{P}}}{d}\Big).\end{aligned}$$

As a result, for $q \in Q$ for which $\left\langle w_{\text{P}}^{(s,0)}, o_1 - q \right\rangle > 0$:

$$\begin{aligned}\left\langle w_{\text{P}}^{(s,1)}, o_1 - q \right\rangle &\geq \left\langle w_{\text{P}}^{(s,0)}, o_1 - q \right\rangle - O(mC_2 \cdot \eta_{\text{P}}) \\ &\geq C_{\text{P}} - O(mC_2 \cdot \eta_{\text{P}}).\end{aligned}$$

Also, for $q \in Q$ for which $\left\langle w_{\text{P}}^{(s,0)}, o_1 - q \right\rangle < 0$:

$$\begin{aligned}\left\langle w_{\text{P}}^{(s,1)}, q - o_1 \right\rangle &\leq \left\langle w_{\text{P}}^{(s,0)}, q - o_1 \right\rangle + O(mC_2 \cdot \eta_{\text{P}}) \\ &\leq \frac{1}{\sqrt{2}} \cdot \log(v) + O(mC_2 \cdot \eta_{\text{P}}).\end{aligned}$$

Considering $n_{\text{P}} = O\Big(\frac{C_{\text{P}}}{mC_2}\Big)$, where $C_{\text{P}} = \Theta\big(\frac{1}{d}\big)$:

$$\left\langle w_{\text{P}}^{(s,1)}, o_1 - q \right\rangle \geq \Omega(C_{\text{P}}), \qquad \left\langle w_{\text{P}}^{(s,1)}, q - o_1 \right\rangle \leq \log(v),$$

which implies $p_1^{(s,1)} \geq p_1^{(s,0)}$.

From Lemma 4 we also know

$$\left\langle \frac{\partial l}{\partial w_{\text{r}}^{(s,0)}}, o_1 \right\rangle \geq -\frac{1}{2} - \tilde{O}\Big(\frac{1}{\sqrt{B}}\Big),$$

and similarly,

$$\left\langle \frac{\partial l}{\partial w_{\text{r}}^{(s,0)}}, o_1 \right\rangle \leq -\big(\tfrac{1}{kv}\big) + \tilde{O}\Big(\tfrac{1}{kv\sqrt{B}}\Big).$$

Also,

$$\left|\left\langle \frac{\partial l}{\partial w_{\text{r}}^{(s,0)}}, q \right\rangle\right| \leq O\big(\tfrac{1}{d}\big) + \tilde{O}\Big(\tfrac{1}{\sqrt{B}}\Big).$$

$$\text{From these, we get} \quad \left\langle w_r^{(s,1)}, o_1 \right\rangle \geq \left\langle w_r^{(s,0)}, o_1 \right\rangle + \Omega(\tfrac{n_{\text{e}}}{kv}) - \tilde{O}\Big(\tfrac{n_{\text{e}}}{kv\sqrt{B}}\Big)$$

Hence, for $B = \Omega(v^2 d^2)$

$$\sigma_1^{(s,1)} \geq \sigma_1^{(s,o)} + \Omega\big(\tfrac{m \cdot n_{\text{e}}}{kv}\big)$$

and similarly,

$$\sigma_1^{(s,1)} \leq \sigma_1^{(s,o)} + O(m \cdot n_{\text{e}})$$

Again,

$$\begin{aligned}&\left\langle w_r^{(s,1)}, q \right\rangle \leq \left\langle w_r^{(s,o)}, q \right\rangle + O\big(\tfrac{n_{\text{e}}}{d}\big), \\ \Rightarrow \quad &\sigma_q^{(s,1)} \leq \sigma_q^{(s,o)} + O\big(\tfrac{m \cdot n_{\text{e}}}{d}\big), \\ &\sigma_q^{(s,1)} \geq \sigma_q^{(s,0)} - O\big(\tfrac{m \cdot n_{\text{e}}}{d}\big).\end{aligned}$$

As a result,

$$\left|\sigma_q^{(s,1)} - \sigma_q^{(s,0)}\right| \leq O\big(\tfrac{m \cdot n_e}{d}\big).$$

From these we find

$$\delta_{1,q}^{(s,1)} \geq \delta_{1,q}^{(s,0)} + \Omega\big(\tfrac{m \cdot n_e}{kv}\big), \qquad \left|\delta_{q,q'}^{(s,1)} - \delta_{q,q'}^{(s,0)}\right| \leq O\big(\tfrac{m \cdot n_e}{d}\big).$$

Using the above equations, we can show that

$$\left|\left\langle \frac{\partial l}{\partial w_{\text{P}}^{(s,1)}}, q \right\rangle\right| \leq O\big(\tfrac{mC_2}{d}\big) + O\Big(\tfrac{mC_2}{\sqrt{B}}\Big) + O\big(\tfrac{m\cdot\eta_e}{d^2}\big)$$
$$\left|\left\langle \frac{\partial l}{\partial w_{\text{P}}^{(s,1)}}, o_1 \right\rangle\right| \leq O(mC_2) + O\Big(\tfrac{m\cdot C_2}{\sqrt{B}}\Big)$$

Now, for $t \leq T_1$, such that $p_1^{(s,0)} \leq p_1^{(s,t)} \leq 1$, by induction we can show:

$$\left\langle \frac{\partial l}{\partial w_{\text{P}}^{(s,t)}}, o_1 \right\rangle \leq O(mC_2) + O\Big(\tfrac{m\cdot C_2}{\sqrt{B}}\Big),$$

$$\left\langle \frac{\partial l}{\partial w_{\text{P}}^{(s,t)}}, q \right\rangle \geq -O\big(\tfrac{mC_2}{d}\big) - O\Big(\tfrac{mC_2}{\sqrt{B}}\Big)),$$

Therefore, for any $t = T_1$:

- for $q \in Q$ for which $\left\langle w_{\text{P}}^{(s,0)}, o_1 - q \right\rangle > 0$:

$$\left\langle w_{\text{P}}^{(s,T_1)}, o_1 - q \right\rangle \geq C_{\text{P}} - O(mC_2\eta_{\text{P}}\cdot T_1) - O\Big(\tfrac{m\cdot C_2\cdot\eta_{\text{P}}\cdot T_1}{\sqrt{B}}\Big) - O\Big(\tfrac{m\cdot\eta_{\text{e}}\cdot\eta_{\text{P}}\cdot T_1}{d^2}\Big)$$

- for $q \in Q$ for which $\left\langle w_{\text{P}}^{(s,0)}, o_1 - q \right\rangle < 0$:

$$\left\langle w_{\text{P}}^{(s,T_1)}, q - o_1 \right\rangle \leq \tfrac{1}{\sqrt{2}}\log(v) + O(mC_2\eta_{\text{P}}T_1) + O\Big(\tfrac{m\cdot C_2\cdot\eta_{\text{P}}\cdot T_1}{\sqrt{B}}\Big) + O\Big(\tfrac{m\cdot\eta_{\text{e}}\cdot\eta_{\text{P}}\cdot T_1}{d^2}\Big).$$

Moreover,

$$\delta_{1,q}^{(s,t)} \geq \delta_{1,q}^{(s,0)} + \Omega\big(\tfrac{m\cdot n_{\text{e}}\cdot t}{kv}\big), \qquad \left|\delta_{q,q'}^{(s,t)} - \delta_{q,q'}^{(s,0)}\right| \leq O\big(\tfrac{m\cdot n_{\text{e}}\cdot t}{d}\big).$$

Now, for $T' = \Omega\Big(\tfrac{kvC_2}{\eta_{\text{e}}}\Big)$, we can show Also,

$$\delta_{1,q}^{(s,t)} = \Omega(mC_2), \qquad \delta_{q,q'}^{(s,t)} = O(mC_2),$$

which also means

$$\delta_{2,q}^{(s,t)} = O(mC_2).$$

To ensure $p_1^{(s,0)} \leq p_1^{(s,t)} \leq 1$, we need

$$\eta_{\text{P}} = O\Big(\tfrac{C_{\text{P}}}{mC_2\cdot T_1}\Big), \quad \text{where } C_{\text{P}} = \Theta\big(\tfrac{1}{d}\big).$$

As a result, for $q \in Q$ for which $\left\langle w_{\text{P}}^{(s,0)}, o_1 - q \right\rangle > 0$:

$$\left\langle w_p^{(s,T_1)}, o_1 - q \right\rangle \geq C_{\text{P}} - O(mC_2\eta_{\text{P}}\cdot T_1) - O\Big(\tfrac{m\cdot C_2\cdot\eta_{\text{P}}\cdot T_1}{\sqrt{B}}\Big) - O\Big(\tfrac{m\cdot\eta_{\text{e}}\cdot\eta_{\text{P}}\cdot T_1}{d^2}\Big) \geq \Omega(C_{\text{P}})$$

Also, for $q \in Q$ for which $\left\langle w_{\text{P}}^{(s,0)}, o_1 - q \right\rangle < 0$:

$$\left\langle w_{\text{P}}^{(s,T_1)}, q - o_1 \right\rangle \leq \tfrac{1}{\sqrt{2}}\log(v) + O(mC_2\eta_{\text{P}}T_1) + O\Big(\tfrac{m\cdot C_2\cdot\eta_{\text{P}}\cdot T_1}{\sqrt{B}}\Big) + O\Big(\tfrac{m\cdot\eta_{\text{e}}\cdot\eta_{\text{P}}\cdot T_1}{d^2}\Big) \leq \log(v).$$

—

Now, from Lemma 4, we can also write

$$\left\langle w_{\text{P}}^{(s,1)}, o_1 \right\rangle \leq \left\langle w_{\text{P}}^{(s,0)}, o_1 \right\rangle + O(mC_2\cdot\eta_{\text{P}}), \qquad \left\langle w_{\text{P}}^{(s,1)}, q \right\rangle \geq \left\langle w_{\text{P}}^{(s,0)}, q \right\rangle - O\Big(\tfrac{mC_2\cdot\eta_{\text{P}}}{d}\Big).$$

As a result, for $\left\langle w_{\text{P}}^{(s,0)}, q - o_1 \right\rangle > 0$ and $\eta_{\text{P}} = O\Big(\frac{C_{\text{P}} \cdot \eta_{\text{e}}}{mC_2^2 \cdot kv}\Big)$, we have

$$\begin{aligned}\left\langle w_{\text{P}}^{(s,1)}, q - o_1 \right\rangle &\geq \left\langle w_{\text{P}}^{(s,0)}, q - o_1 \right\rangle - O(mC_2 \cdot \eta_{\text{P}}) \\ &\geq C_{\text{P}} - O(mC_2 \cdot \eta_{\text{P}}) \\ &\geq \Omega(C_{\text{P}}).\end{aligned}$$

This implies $p_1^{(s,1)} = p_1^{(s,0)}$.

—

Now from the inequalities, we can write

$$\left\langle \frac{\partial l}{\partial w_r^{(s,0)}}, o_1 \right\rangle \geq -\frac{1}{2} - \tilde{O}\Big(\tfrac{1}{\sqrt{B}}\Big),$$

$$\text{Also, } \left\langle \frac{\partial l}{\partial w_r^{(s,0)}}, q \right\rangle \leq O(\frac{1}{d}) + \tilde{O}\Big(\tfrac{1}{\sqrt{B}}\Big)$$

which implies

$$\left\langle w_r^{(s,1)}, o_1 \right\rangle \leq \left\langle w_r^{(s,0)}, o_1 \right\rangle + O(\eta_{\text{e}}) + O(\frac{\eta_{\text{e}}}{\sqrt{B}}),$$

$$\sigma_1^{(s,1)} \leq \sigma_1^{(s,0)} + O(m\eta_{\text{e}}) + O(\frac{m\eta_{\text{e}}}{\sqrt{B}}).$$

Also,

$$\left\langle w_r^{(s,1)}, q \right\rangle \geq \left\langle w_r^{(s,0)}, q \right\rangle - O(\frac{\eta_{\text{e}}}{d}) - O(\frac{\eta_{\text{e}}}{\sqrt{B}}),$$

$$\sigma_q^{(s,1)} \geq \sigma_q^{(s,0)} - O(\frac{m\eta_{\text{e}}}{d}) - O(\frac{m\eta_{\text{e}}}{\sqrt{B}}).$$

As a result, for any q, we can say

$$\delta_{1,q}^{(s,1)} - \delta_{1,q}^{(s,0)} \leq O(m\eta_{\text{e}}) + O(\frac{m\eta_{\text{e}}}{\sqrt{B}}).$$

Since $p_1^{(s,1)} = p_1^{(s,0)}$, it follows that

$$\left\langle w_{\text{P}}^{(s,2)}, o_1 \right\rangle \leq \left\langle w_{\text{P}}^{(s,1)}, o_1 \right\rangle + O(mC_2 \cdot \eta_{\text{P}}) + O(m\eta_{\text{e}}\eta_{\text{P}}),$$

$$\left\langle w_{\text{P}}^{(s,2)}, q \right\rangle \geq \left\langle w_{\text{P}}^{(s,1)}, q \right\rangle - O\Big(\tfrac{mC_2 \cdot \eta_{\text{P}}}{d}\Big) - O\big(\tfrac{m\eta_{\text{e}}\eta_{\text{P}}}{d}\big).$$

As a result, for any q with $\left\langle w_{\text{P}}^{(s,0)}, o_1 - q \right\rangle \leq 0$:

$$\begin{aligned}\left\langle w_{\text{P}}^{(s,2)}, q - o_1 \right\rangle &\geq \left\langle w_{\text{P}}^{(s,1)}, q - o_1 \right\rangle - O(mC_2 \cdot \eta_{\text{P}}) - O(m\eta_{\text{e}}\eta_{\text{P}}) \\ &\quad - O\Big(\tfrac{mC_2 \cdot \eta_{\text{P}}}{d}\Big) - O\big(\tfrac{m\eta_{\text{e}}\eta_{\text{P}}}{d}\big) \\ &\geq \Omega(C_{\text{P}}).\end{aligned}$$

This implies $p_1^{(s,2)} = p_1^{(s,0)}$.

—

By induction, for $t < T_1$, selecting $\eta_p = O\Big(\frac{\eta_{\text{e}} C_{\text{P}}}{mC_2^2 k^2 v^2}\Big)$, we can show for any q with $\left\langle w_{\text{P}}^{(s,0)}, o_1 - q \right\rangle \leq 0$:

$$\left\langle w_{\text{P}}^{(s,t)}, q - o_1 \right\rangle \geq \Omega(C_{\text{P}}),$$

which means $p_1^{(s,t)} = p_1^{(s,0)}$.

Similarly, for $t < T_1$, we can show for any q with $\left\langle w_{\text{P}}^{(s,0)}, o_1 - q \right\rangle \geq 0$:

$$\left\langle w_{\text{P}}^{(s,t)}, o_1 - q \right\rangle \leq \log(v).$$

Here, the analysis considers t to be the number of iterations. However, due to the random selection of expert by the batch-level router, not all experts are selected in each iteration. As the probability of an expert being selected is $\frac{k}{N}$, the actual $T_1 = \Omega\Big(\frac{N\,v\,C_2}{\eta_e}\Big)$ □

Lemma 7 (Weight and Output Evolution for Class-2 Experts in Stage 1)**.** *At $t = T_1 = \Omega\left(\frac{N\,v\,C_2}{\eta_e}\right)$ iterations, for any important class-2 expert $s \in \{i \in S_2 : p_2^{(i,0)} = \Omega(1)\}$, the following hold:*

(i) The expert outputs have separated for the class-2 feature: $\delta_{2,q}^{(s,T_1)} = \Omega(mC_2)$, while other differences remain bounded.

(ii) For any pattern q with initial alignment $\langle w_P^{(s,0)}, o_2 - q\rangle > 0$, the alignment is preserved and bounded: $\Omega(C_P) < \langle w_P^{(s,T_1)}, o_2 - q\rangle < \log(v)$.

(iii) For any pattern q with initial alignment $\langle w_P^{(s,0)}, o_2 - q\rangle < 0$, the alignment is preserved and bounded: $\Omega(C_P) < \langle w_P^{(s,T_1)}, q - o_2\rangle < \log(v)$.

(iv) The probability of selecting the correct patch remains stable: $p_2^{(s,T_1)} = p_2^{(s,0)}$.

Proof. The proof is symmetric to that of Lemma 6. By applying the same techniques and focusing on the dynamics related to the class-2 feature o_2 for experts in $\{i \in S_2 : p_2^{(i,0)} = \Omega(1)\}$, we arrive at the stated conclusions. □

D Proof of Main Lemmas and Theorem

D.1 Proof of Lemma 2

For proving Lemma 2 we will divide it into two lemmas (8,9) and prove it separately.

Lemma 8. *With learning rates $\eta_P = O\Big(\frac{C_P \cdot \eta_e}{mC_2^2 \cdot k^2\, v^2}\Big)$ and $t \geq T = T_1 + T_2 + \Omega\left(\frac{v^2 C_2}{\eta_e}\sqrt{d\log v}\right)$, the following properties hold:*

There exists an important class-1 expert $s \in \{i \in S_1 : p_1^{(i,0)} = \Omega(1)\}$:

(i) Its expert-choice router becomes fully specialized in selecting o_1:

$$p_1^{(s,T)} = 1, \text{ and} \tag{15}$$

$$H_j^{(s,T)}(x) \geq \frac{1}{2}, \text{ if } x^{(j)} = o_1 \tag{16}$$

(ii) A constant fraction of its neurons r are strongly aligned with o_1:

$$\langle w_r^{(s,T)}, o_1\rangle = \Omega(vC_2\sqrt{d\log(v)}) \tag{17}$$

There exists an important class-2 expert $s \in \{i \in S_2 : p_2^{(i,0)} = \Omega(1)\}$:

(iii) Its expert-choice router becomes fully specialized in selecting o_2:

$$p_2^{(s,T)} = 1, \text{ and} \tag{18}$$

$$H_j^{(s,T)}(x) \geq \frac{1}{2}, \text{ if } x^{(j)} = o_2 \tag{19}$$

(iv) A constant fraction of its neurons r are strongly aligned with o_2:

$$\langle w_r^{(s,T)}, o_2\rangle = \Omega(vC_2\sqrt{d\log(v)}) \tag{20}$$

Proof. From Lemma 6, considering $\eta_e = O(\frac{1}{T_1+T_2})$ we can say:

$$\delta_{1,q}^{(s,T_1+T_2)} = \Omega(mC_2), \qquad \delta_{q,q'}^{(s,T_1+T_2)} = O(mC_2),$$

$$\left\langle \frac{\partial l}{\partial w_{\text{P}}^{(s,T_1+T_2)}}, o_1 \right\rangle \leq -\Omega\big(\tfrac{mC_2}{v}\big) - \tilde{O}\Big(\tfrac{mC_2}{v\sqrt{B}}\Big), \quad \left\langle \frac{\partial l}{\partial w_{\text{P}}^{(s,T_1+T_2)}}, q \right\rangle \geq -O\big(\tfrac{mC_2}{d}\big) - \tilde{O}\Big(\tfrac{mC_2}{\sqrt{B}}\Big).$$

As a result, for any q with $\left\langle w_{\text{P}}^{(s,T_1+T_2)}, o_1 - q \right\rangle \geq \Omega(C_p)$, we can say

$$\left\langle w_{\text{P}}^{(s,T_1+T_2+1)}, o_1 - q \right\rangle \geq \Omega(C_p) + \Omega\Big(\tfrac{mC_2\cdot\eta_{\text{P}}}{v}\Big).$$

This implies $p_1^{(s,T_1+T_2+1)} \geq p_1^{(s,T_1+T_2)} = p_1^{(s,0)}$.

Now, as $p_1^{(s,T_1+T_2)} = p_1^{(s,0)}$, we can write

$$\delta_{1,q}^{(s,T_1+T_2+1)} = \delta_{1,q}^{(s,T_1+T_2)} + \Omega\big(\tfrac{m\eta_{\text{e}}}{v}\big), \qquad \left|\delta_{q,q'}^{(s,T_1+T_2+1)} - \delta_{q,q'}^{(s,T_1+T_2)}\right| = O\big(\tfrac{m\eta_{\text{e}}}{d}\big).$$

As a result, using induction we can show that, for any q with $\left\langle w_{\text{P}}^{(s,t)}, o_1 - q \right\rangle > 0$ and $t > T_1 + T_2$, we have $p_1^{(s,t)} \geq p_1^{(s,0)}$.

As a result, we can say for all $q \in Q$,

$$\delta_{1,q}^{(s,t+1)} - \delta_{1,q}^{(s,t)} = \Omega\big(\tfrac{m\eta_e}{v}\big),$$
$$\left|\delta_{q,q'}^{(s,t)}\right| = O(mC_2).$$

Hence, for all $q \in Q$ we can write

$$\left\langle w_{\text{P}}^{(s,t+1)}, o_1 - q \right\rangle \geq \left\langle w_{\text{P}}^{(s,T_1+T_2)}, o_1 - q \right\rangle + \Omega\Big(\tfrac{mC_2\eta_{\text{P}}}{v}\Big)(t - T_1 - T_2) + \Omega\big(\tfrac{m\eta_e\eta_{\text{P}}}{v^2}\big)\frac{(t-T_1-T_2)^2}{2}.$$

We already know from Lemma 6 that $\left\langle w_{\text{P}}^{(s,T_1+T_2)}, o_1 - q \right\rangle < \log(v)$ .Let us assume, for time $t = T$, for all q, that $\left\langle w_{\text{P}}^{(s,T)}, o_1 - q \right\rangle < 2\log(v)$ holds. Now, if $\exists q'$ so that $q' \neq q$ and $\left\langle w_{\text{P}}^{(s,T)}, o_1 - q' \right\rangle < \log(v)$ then

$$\left\langle w_{\text{P}}^{(s,T)}, o_1 - q \right\rangle \leq \left\langle w_{\text{P}}^{(s,T_1+T_2)}, o_1 - q' \right\rangle + \left\langle w_{\text{P}}^{(s,T_1+T_2)}, q' - q \right\rangle \leq \left\langle w_{\text{P}}^{(s,T_1+T_2)}, q' - q \right\rangle + \log(v).$$

On the other hand, choosing

$$T = T_1 + T_2 + \Omega\left(\frac{v^2 C_2}{\eta_{\text{e}}}\sqrt{\frac{\log v}{C_{\text{P}}}}\right)$$

we get

$$\left\langle w_{\text{P}}^{(s,T+1)}, o_1 - q' \right\rangle > \log v.$$

Now we know, $\forall q$ and $\forall t$,

$$\delta_{1,q}^{(s,t)} - \delta_{1,q}^{(s,t-1)} = O(m\eta_e), \qquad \left|\delta_{q,q'}^{(s,t-1)} - \delta_{q,q'}^{(s,t)}\right| = O\big(\tfrac{m\eta_e}{d}\big).$$

Let $T'' = T_1 + T_2$. As a result,

$$\left\langle w_{\text{P}}^{(s,T''+t')}, q \right\rangle \geq \left\langle w_{\text{P}}^{(s,T'')}, q \right\rangle - O\Big(\tfrac{mC_2\eta_{\text{P}}}{d}\Big)t' - O\big(\tfrac{m\eta_{\text{e}}\eta_{\text{P}}}{d}\big)\frac{t'^2 - t'}{2}.$$

On the other hand,

$$\left\langle w_{\text{P}}^{(s,T''+t')}, q' \right\rangle \leq \left\langle w_{\text{P}}^{(s,T'')}, q' \right\rangle + O\Big(\tfrac{mC_2\eta_{\text{P}}}{d}\Big)t'.$$

Therefore,

$$\left\langle w_{\text{P}}^{(s,T''+t')}, q' - q \right\rangle \leq \left\langle w_{\text{P}}^{(s,T'')}, q' - q \right\rangle + O\Big(\tfrac{mC_2\eta_{\text{P}}}{d}\Big)t' + O\big(\tfrac{m\eta_e\eta_{\text{P}}}{d}\big)t'^2. \tag{21}$$

Now, for

$$t' = T_3 = O\Big(\tfrac{v^2 C_2}{\eta_{\text{e}}}\sqrt{\tfrac{\log v}{C_p}}\Big) \quad \text{and} \quad \left\langle w_{\text{P}}^{(0)}, q' - q \right\rangle \leq \tfrac{1}{2}\log v,$$

we get, for $T = T_1 + T_2 + T_3\Big(\frac{v^2 C_2}{\eta_e}\sqrt{\frac{\log v}{C_p}}\Big)$

$$\Big\langle w_{\mathrm{P}}^{(s,T)}, q' - q \Big\rangle \le \tfrac{1}{2}\log v + O\Big(\tfrac{v^2}{d}\log v\Big) \le \log v.$$

As a result, for

$$T = T_1 + T_2 + T_3, \quad \forall q \in Q, \quad \Big\langle w_{\mathrm{P}}^{(s,T)}, o_1 - q \Big\rangle \le 2\log v$$

Hence, our assumption holds.

Now, as $\forall q$, $\Big\langle w_{\mathrm{P}}^{(s,T+1)}, o_1 - q \Big\rangle > \log v$, we have $p_1^{(s,T+1)} = 1$ and $H_j^{(s,T)}(x) \ge \frac{1}{2}$, if $x^{(j)} = o_1$.

Next, since for any t, $p_1^{(s,t)} \ge p_1^{(s,0)}$, at least for a constant fraction of $r \in [m]$ of s,

$$\Big\langle w_r^{(s,T)}, o_1 \Big\rangle \ge \Big\langle w_r^{(s,0)}, o_1 \Big\rangle + \Omega(\frac{\eta_e}{v}T).$$

Therefore, for

$$T = T_1 + T_2 + \Omega\Big(\tfrac{v^2 C_2}{\eta_e}\sqrt{\tfrac{\log v}{C_p}}\Big),$$

we have

$$\Big\langle w_r^{(s,T)}, o_1 \Big\rangle = \Omega\Big(vC_2\sqrt{d\log v}\Big) \quad \text{for } \Omega(1) \text{ fraction of } r \in [m] \text{ of } s.$$

Hence (i) and (ii) is proved. Finally, using the same techniques as in (i) and (ii), we can also prove (iii) and (iv), respectively, for $s \in \{\, i \in S_2 : p_2^{(s,0)} = \Omega(1) \,\}$.

□

Lemma 9. *With learning rates $\eta_p = O\Big(\frac{C_p \cdot \eta_e}{mC_2^2 \cdot k^2\, v^2}\Big)$ and $t \ge T = T_1 + T_2 + \Omega\left(\frac{v^2 C_2}{\eta_e}\sqrt{d\log v}\right)$, the following properties hold:*
For any $s \in S_2$:

(i) Its expert-choice router does not specialize in selecting o_1:

$$p_1^{(s,T)} < p_1^{(s,0)} \tag{22}$$

(ii) The alignment of any neuron r with o_1 does not grow:

$$\langle w_r^{(s,T)}, o_1 \rangle \le O(1) \tag{23}$$

(iii) The alignment of any neuron r with any irrelevant patch $q \in Q$ does not grow either:

$$\langle w_r^{(s,T)}, q \rangle \le O(1) + O(v^2\sqrt{\frac{\log(v)}{d}}) \tag{24}$$

For any $s \in S_1$:

(iv) Its expert-choice router does not specialize in selecting o_2:

$$p_2^{(s,T)} < p_2^{(s,0)} \tag{25}$$

(v) The alignment of any neuron r with o_2 does not grow:

$$\langle w_r^{(s,T)}, o_2 \rangle \le O(1) \tag{26}$$

(vi) The alignment of any neuron r with any irrelevant patch $q \in Q$ does not grow either:

$$\langle w_r^{(s,T)}, q \rangle \le O(1) + O(v^2\sqrt{\frac{\log(v)}{d}}) \tag{27}$$

Proof. As $\left\langle \frac{\partial l}{\partial w_r^{(s,0)}}, o_1 \right\rangle \leq \frac{1}{2} + \tilde{O}(\frac{1}{\sqrt{B}})$; $\left\langle \frac{\partial l}{\partial w_r^{(s,0)}}, o_1 \right\rangle \geq 0$ and $\left\langle \frac{\partial l}{\partial w_r^{(s,0)}}, q \right\rangle \geq O(\frac{1}{d}) + \tilde{O}(\frac{1}{\sqrt{B}})$, we can write from lemma 6 and lemma 1 that for any $p_1^{(s,T_1+T_2)} \geq 0$:

$$\delta_{1,q}^{(s,T_1+T_2)} = O(mC_2), \qquad \delta_{q,q'}^{(s,T_1+T_2)} = O(mC_2),$$
$$\left\langle \frac{\partial l}{\partial w_{\text{P}}^{(s,T_1+T_2)}}, o_1 \right\rangle \geq 0, \qquad \left| \left\langle \frac{\partial l}{\partial w_{\text{P}}^{(s,T_1+T_2)}}, q \right\rangle \right| \leq O\big(\tfrac{k\,mC_2}{N\,d}\big) + \tilde{O}\Big(\tfrac{k\,mC_2}{N\sqrt{B}}\Big).$$

Hence we can write

$$\left\langle w_{\text{P}}^{(s,T_1+T_2+1)}, o_1 \right\rangle \leq \left\langle w_{\text{P}}^{(s,T_1+T_2)}, o_1 \right\rangle$$
$$\left\langle w_{\text{P}}^{(s,T_1+T_2+1)}, q \right\rangle \geq \left\langle w_{\text{P}}^{(s,T_1+T_2)}, q \right\rangle - O\Big(\tfrac{k\,mC_2\cdot\eta_{\text{P}}}{N\,d}\Big) - \tilde{O}\Big(\tfrac{k\,mC_2\cdot\eta_{\text{P}}}{N\sqrt{B}}\Big).$$

As a result, for any q such that $\left\langle w_{\text{P}}^{(s,T_1+T_2)}, q - o_1 \right\rangle > 0$,

$$\left\langle w_{\text{P}}^{(s,T_1+T_2+1)}, q - o_1 \right\rangle \geq \left\langle w_{\text{P}}^{(s,T_1+T_2)}, q - o_1 \right\rangle - O\Big(\tfrac{k\,mC_2\cdot\eta_{\text{P}}}{N\,d}\Big) - \tilde{O}\Big(\tfrac{k\,mC_2\cdot\eta_{\text{P}}}{N\sqrt{B}}\Big).$$

Considering $\eta_{\text{P}} = O\Big(\frac{\eta_{\text{e}} C_p}{mk^2v^2C_2^2}\Big)$, we get

$$\left\langle w_{\text{P}}^{(s,T_1+T_2+1)}, q - o_1 \right\rangle = \Omega(C_p),$$

which implies

$$p_1^{(s,T_1+T_2+1)} \leq p_1^{(s,T_1+T_2)}$$

Now, considering $B = \tilde{\Omega}(v^2d^2)$, we can write

$$\delta_{1,q}^{(s,T_1+T_2+1)} \leq \delta_{1,q}^{(s,T_1+T_2)}, \quad \left|\delta_{q,q'}^{(s,T_1+T_2+1)} - \delta_{q,q'}^{(s,T_1+T_2)}\right| \leq O\big(\tfrac{m\eta_{\text{e}}}{d}\big).$$

Hence, for any q such that $\left\langle w_{\text{P}}^{(s,T_1+T_2)}, q - o_1 \right\rangle > 0$:

$$\left\langle w_{\text{P}}^{(s,T_1+T_2+2)}, q - o_1 \right\rangle \geq \left\langle w_{\text{P}}^{(s,T_1+T_2+1)}, q - o_1 \right\rangle - O\Big(\tfrac{k\,mC_2\cdot\eta_{\text{P}}}{N\,d}\Big) - O\Big(\tfrac{k\,m\cdot\eta_e\cdot\eta_{\text{P}}}{N\,d}\Big) = \Omega(C_p).$$

This implies that

$$p_1^{(s,T_1+T_2+2)} \leq p_1^{(s,T_1+T_2)}.$$

Now, for any q such that $\left\langle w_{\text{P}}^{(s,T_1+T_2)}, q - o_1 \right\rangle > 0$, using induction for t such that $p_1^{(s,t)} \leq p_1^{(s,0)}$, we can write

$$\left\langle w_{\text{P}}^{(s,t+1)}, q - o_1 \right\rangle \geq \left\langle w_{\text{P}}^{(s,T_1+T_2)}, q - o_1 \right\rangle - O\Big(\tfrac{k\,mC_2\cdot\eta_{\text{P}}}{N\,d}\Big)(t - T_1 - T_2) - O\Big(\tfrac{k\,m\cdot\eta_e\cdot\eta_{\text{P}}}{N\,d}\Big)(t - T_1 - T_2)^2.$$

Considering

$$t = T = T_1 + T_2 + O\left(\frac{v^2C_2}{\eta_{\text{e}}} \cdot \sqrt{\frac{\log(v)}{C_{\text{P}}}}\right),$$

we can write

$$\left\langle w_{\text{P}}^{(s,T+1)}, q - o_1 \right\rangle = \Omega(C_{\text{P}}).$$

This also implies that for $t \geq T + 1$,

$$p_1^{(s,t)} \leq p_1^{(s,0)}.$$

Also, $\left\langle w_r^{(s,T)}, o_1 \right\rangle \leq \left\langle w_r^{(s,T_1+T_2)}, o_1 \right\rangle \leq O(1)$

and, $\left\langle w_r^{(s,T)}, q \right\rangle \leq \left\langle w_r^{(s,T_1+T_2)}, q \right\rangle + O(\frac{\eta_{\text{e}}}{d} \cdot (T - T_1 - T_2)) \leq O(1) + O(v^2C_2\sqrt{\frac{\log(v)}{d}})$

Hence, (i), (ii) and (iii) are proved.

Using similar technique for $s \in S_1$, we can also prove (iv), (v) and (vi).

□

Lemma 10. *With learning rates $\eta_p = O\Big(\frac{C_p \cdot \eta_e}{mC_2^2 \cdot k^2\, v^2}\Big)$ and $t \geq T = T_1 + T_2 + \Omega\left(\frac{v^2 C_2}{\eta_e}\sqrt{d \log v}\right)$, the following properties hold:*

(i) For any $s \in \{i \in S_1 : p_1^{(s,0)} = O(1/d)\}$: $p_1^{(s,T)} = O(1/d)$

(ii) For any $s \in \{i \in S_2 : p_2^{(s,0)} = O(1/d)\}$: $p_2^{(s,T)} = O(1/d)$

Proof. As $\left\langle \frac{\partial l}{\partial w_r^{(s,0)}}, o_1 \right\rangle \leq \frac{1}{2} + \tilde{O}(\frac{1}{\sqrt{B}})$; $\left\langle \frac{\partial l}{\partial w_r^{(s,0)}}, o_1 \right\rangle \geq 0$ and $\left\langle \frac{\partial l}{\partial w_r^{(s,0)}}, q \right\rangle \geq O(\frac{1}{d}) + \tilde{O}(\frac{1}{\sqrt{B}})$, we can write for any $p_1^{(s,0} = O(\frac{1}{d})$:

$$\delta_{1,q}^{(s,T_1+T_2)} = O(mC_2), \qquad \delta_{q,q'}^{(s,T_1+T_2)} = O(mC_2),$$

$$\left\langle \frac{\partial l}{\partial w_{\mathrm{P}}^{(s,T_1+T_2)}}, o_1 \right\rangle \leq O\big(\tfrac{k\, mC_2}{N\, d}\big) + \tilde{O}\Big(\tfrac{k\, mC_2}{N\sqrt{B}}\Big), \quad \left|\left\langle \frac{\partial l}{\partial w_{\mathrm{P}}^{(s,T_1+T_2)}}, q \right\rangle\right| \leq O\big(\tfrac{k\, mC_2}{N\, d}\big) + \tilde{O}\Big(\tfrac{k\, mC_2}{N\sqrt{B}}\Big).$$

Hence we can write

$$\left\langle w_{\mathrm{P}}^{(s,T_1+T_2+1)}, o_1 \right\rangle \leq \left\langle w_{\mathrm{P}}^{(s,T_1+T_2)}, o_1 \right\rangle + O\Big(\tfrac{k\, mC_2 \cdot \eta_{\mathrm{P}}}{N\, d}\Big) + \tilde{O}\Big(\tfrac{k\, mC_2 \cdot \eta_{\mathrm{P}}}{N\sqrt{B}}\Big),$$
$$\left\langle w_{\mathrm{P}}^{(s,T_1+T_2+1)}, q \right\rangle \geq \left\langle w_{\mathrm{P}}^{(s,T_1+T_2)}, q \right\rangle - O\Big(\tfrac{k\, mC_2 \cdot \eta_{\mathrm{P}}}{N\, d}\Big) - \tilde{O}\Big(\tfrac{k\, mC_2 \cdot \eta_{\mathrm{P}}}{N\sqrt{B}}\Big).$$

As a result, for any q such that $\left\langle w_{\mathrm{P}}^{(s,T_1+T_2)}, q - o_1 \right\rangle > 0$,

$$\left\langle w_{\mathrm{P}}^{(s,T_1+T_2+1)}, q - o_1 \right\rangle \geq \left\langle w_{\mathrm{P}}^{(s,T_1+T_2)}, q - o_1 \right\rangle - O\Big(\tfrac{k\, mC_2 \cdot \eta_{\mathrm{P}}}{N\, d}\Big) - \tilde{O}\Big(\tfrac{k\, mC_2 \cdot \eta_{\mathrm{P}}}{N\sqrt{B}}\Big).$$

Considering $\eta_{\mathrm{P}} = O\Big(\frac{\eta_e C_{\mathrm{P}}}{mk^2 v^2 C_2^2}\Big)$, we get

$$\left\langle w_{\mathrm{P}}^{(s,T_1+T_2+1)}, q - o_1 \right\rangle = \Omega(C_{\mathrm{P}}),$$

which implies

$$p_1^{(s,T_1+T_2+1)} \leq p_1^{(s,T_1+T_2)}$$

Now, considering $B = \tilde{\Omega}(v^2 d^2)$, we can write

$$\delta_{1,q}^{(s,T_1+T_2+1)} - \delta_{1,q}^{(s,T_1+T_2)} \leq O\big(\tfrac{m\eta_{\mathrm{e}}}{d}\big), \quad \left|\delta_{q,q'}^{(s,T_1+T_2+1)} - \delta_{q,q'}^{(s,T_1+T_2)}\right| \leq O\big(\tfrac{m\eta_{\mathrm{e}}}{d}\big).$$

Hence, for any q such that $\left\langle w_{\mathrm{P}}^{(s,T_1+T_2)}, q - o_1 \right\rangle > 0$:

$$\left\langle w_{\mathrm{P}}^{(s,T_1+T_2+2)}, q - o_1 \right\rangle \geq \left\langle w_{\mathrm{P}}^{(s,T_1+T_2+1)}, q - o_1 \right\rangle - O\Big(\tfrac{k\, mC_2 \cdot \eta_{\mathrm{P}}}{N\, d}\Big) - O\Big(\tfrac{k\, m \cdot \eta_{\mathrm{e}} \cdot \eta_{\mathrm{P}}}{N\, d}\Big) = \Omega(C_{\mathrm{P}}).$$

This implies that

$$p_1^{(s,T_1+T_2+2)} \leq p_1^{(s,T_1+T_2)}.$$

Now, for any q such that $\left\langle w_{\mathrm{P}}^{(s,T_1+T_2)}, q - o_1 \right\rangle > 0$, using induction for t such that $p_1^{(s,t)} \leq p_1^{(s,0)}$, we can write

$$\left\langle w_{\mathrm{P}}^{(s,t+1)}, q - o_1 \right\rangle \geq \left\langle w_{\mathrm{P}}^{(s,T_1+T_2)}, q - o_1 \right\rangle - O\Big(\tfrac{k\, mC_2 \cdot \eta_{\mathrm{P}}}{N\, d}\Big)\,(t - T_1 - T_2) - O\Big(\tfrac{k\, m \cdot \eta_{\mathrm{e}} \cdot \eta_{\mathrm{P}}}{N\, d}\Big)\,(t - T_1 - T_2)^2.$$

Considering

$$t = T = T_1 + T_2 + \Omega\left(\frac{v^2 C_2}{\eta_{\mathrm{e}}} \cdot \sqrt{\frac{\log(v)}{C_{\mathrm{P}}}}\right),$$

we can write

$$\left\langle w_{\mathrm{P}}^{(s,T+1)}, q - o_1 \right\rangle = \Omega(C_p).$$

This also implies that for $t \geq T+1$,

$$p_1^{(s,t)} \leq p_1^{(s,0)} = O(\frac{1}{d}).$$

Using similar technique for$s \in \{i \in S_2 : p_2^{(s,0)} = O(1/d)\}$, we can also prove (ii).

□

D.2 Proof of Lemma 1

Lemma 11. *With $\eta_p = O(\frac{\eta_e\, C_p}{mk^2v^2C_2^2})$, over a randomly sampled batch $B = \Omega(v^2d^2)$, we can say that for $t \geq T_1 + T_2$; where $T_2 = \Omega\Big(v\sqrt{\frac{Nk\log(k)}{m\eta_e\,\eta_S}}\Big)$:*

(i) $\exists s \in S_1$ with $p_1^{(s,0)} = \Omega(1)$ for which,

$$G^{(s,t)}(x) \geq \frac{1}{2}, \textit{for each positive batch } (x, +1) \tag{28}$$

(ii) $\exists s \in S_2$ with $p_2^{(s,0)} = \Omega(1)$ for which,

$$G^{(s,t)}(x) \geq \frac{1}{2}, \textit{for each negative batch } (x, -1) \tag{29}$$

Proof. (i) Note that, for each $i \in J_{\mathrm{S}}(x)$, the expert-choice (patch) router logit and normalized routing score:

$$z_i^{(t)} \;=\; \sum_{p\in[n]} \Big\langle w_{\mathrm{S}}^{(i,t)},\, x^{(p)} \Big\rangle, \qquad G^{(i,t)}(x) \;=\; \frac{\exp(z_i^{(t)})}{\sum_{j\in J_{\mathrm{S}}(x)} \exp(z_j^{(t)})}.$$

From Lemma 6 we know that at $t = T_1$, for,$(x, y) \sim D$ with $y = +1$, and $s \in \{i \in S_1 : p_1^{(i,0)} = \Omega(1)\}$:

$$\sigma_1^{(s,T_1)} \;\geq\; \Omega\big(mC_2\big), \qquad \sigma_q^{(s,T_1)} \;\leq\; O\big(mC_2\big).$$

As a result, we can write

$$\left\langle \frac{\partial l}{\partial w_{\mathrm{P}}^{(s,T_1)}},\, o_1 \right\rangle \;\leq\; -\,\Omega\!\left(\frac{m\,C_2}{k\,v}\right).$$

or equivalently, for any $q \in Q$,

$$\left\langle \frac{\partial l}{\partial w_{\mathrm{P}}^{(s,T_1)}},\, q \right\rangle \;\geq\; -\,O\!\left(\frac{m\,C_2}{d}\right).$$

Hence we can write,

$$\big\langle w_{\mathrm{P}}^{(s,T_1+1)}, o_1 \big\rangle \;\geq\; \big\langle w_{\mathrm{P}}^{(s,T_1)}, o_1 \big\rangle + \Omega\!\left(\frac{m\,C_2\,\eta_{\mathrm{P}}}{k\,v}\right). \tag{30}$$

$$\big\langle w_{\mathrm{P}}^{(s,T_1+1)}, q \big\rangle \;\leq\; \big\langle w_{\mathrm{P}}^{(s,T_1)}, q \big\rangle + O\!\left(\frac{m\,C_2\,\eta_{\mathrm{P}}}{d}\right). \tag{31}$$

As we consider $d >> v$, we can write,

$$\big\langle w_{\mathrm{P}}^{(s,T_1+1)}, o_1 - q \big\rangle \;\geq\; \big\langle w_{\mathrm{P}}^{(s,T_1)}, o_1 - q \big\rangle + \Omega\!\left(\frac{m\,C_2\,\eta_{\mathrm{P}}}{k\,v}\right)$$

As $\eta_{\mathrm{P}} = O(\frac{\eta_{\mathrm{e}}\, C_{\mathrm{P}}}{mk^2v^2C_2^2})$, this implies, $p_1^{(s,T_1+1)} \geq p_1^{(s,T_1)} = p_1^{(s,0)}$ Now, we know, $G^{(i,t)} \leq 1$ for $i \in [N]$ Hence, we can write,

$$\left\langle \frac{\partial l}{\partial w_{\mathrm{S}}^{(s,T_1)}},\, x^{(p)} \right\rangle \leq -\Omega\!\left(\frac{mC_2}{N\cdot v}\right)$$

As a result,

$$\Big\langle w_{\text{S}}^{(s,T_1+1)},\, x^{(p)} \Big\rangle \geq \Big\langle w_{\text{S}}^{(s,T_1)},\, x^{(p)} \Big\rangle + \Omega\Big(\frac{mC_2 \cdot \eta_{\text{S}}}{N \cdot v}\Big)$$

Similarly, for $i \notin \{s \in S : p_1^{(s,0)} = \Omega(1)\}$,

$$\Big\langle \frac{\partial l}{\partial w_{\text{S}}^{(i,T_1)}},\, x^{(p)} \Big\rangle \geq -O\Big(\frac{mC_2}{N \cdot d}\Big)$$

Hence,

$$\Big\langle w_{\text{S}}^{(i,T_1+1)},\, x^{(p)} \Big\rangle \leq \Big\langle w_{\text{S}}^{(i,T_1)},\, x^{(p)} \Big\rangle + O\Big(\frac{mC_2 \cdot \eta_{\text{S}}}{N \cdot d}\Big)$$

Summing over $p \in [n]$ and using $d \gg v$ gives a positive drift in the logit gaps:

$$\big(z_s^{(T_1+1)} - z_i^{(T_1+1)}\big) - \big(z_s^{(T_1)} - z_i^{(T_1)}\big) \;\geq\; \Omega\Big(\frac{mC_2\eta_{\text{S}}\, n}{Nv}\Big) \tag{32}$$

Now, for expert s, we know,

$$\Big\langle \frac{\partial l}{\partial w_r^{(s,T_1)}},\, o_1 \Big\rangle \leq -\Omega\Big(\frac{1}{k\, v}\Big)$$

$$\Big\langle \frac{\partial l}{\partial w_r^{(s,T_1)}},\, q \Big\rangle \geq -O\Big(\frac{1}{d}\Big)$$

Hence,

$$\sigma_1^{(s,T_1+1)} \geq \sigma_1^{(s,T_1)} + \Omega\Big(\frac{m\eta_{\text{e}}}{k\, v}\Big)$$

$$\sigma_q^{(s,T_1+1)} \leq \sigma_q^{(s,T_1)} + O\Big(\frac{m\eta_{\text{e}}}{d}\Big)$$

$$\delta_{1,q}^{(s,T_1+1)} \geq \Omega(mC_2) + \Omega\Big(\frac{m\eta_{\text{e}}}{k\, v}\Big)$$

$$\delta_{q,q'}^{(s,T_1+1)} \leq O(mC_2) + O\Big(\frac{m\eta_{\text{e}}}{d}\Big)$$

Note that, for $i \notin \{i \in S_1 : p_1^{(i,0)} = \Omega(1)\}$ and $y = 1$, we can say

$$\delta_{1,q}^{(i,T_1+1)} \leq O(mC_2) + O\Big(\frac{m\eta_e}{d}\Big)$$

$$\delta_{q,q'}^{(i,T_1+1)} \leq O(mC_2) + O\Big(\frac{m\eta_e}{d}\Big)$$

Using induction, for expert s, we can derive

$$\delta_{1,q}^{(s,T_1+t')} \geq \delta_{1,q}^{(s,T_1)} + \Omega\Big(\frac{m\eta_{\text{e}}}{k\, v} t'\Big)$$

$$\big|\, \delta_{q,q'}^{(s,T_1+t')} - \delta_{q,q'}^{(s,T_1)} \,\big| \leq O\Big(\frac{m\eta_{\text{e}}}{d} t'\Big)$$

Now using induction we can also write,

$$\Big\langle w_{\text{P}}^{(s,T_1+t')},\, o_1 - q \Big\rangle \geq \Big\langle w_{\text{P}}^{(s,T_1)},\, o_1 - q \Big\rangle + \Omega\Big(\frac{mC_2\eta_{\text{P}} t'}{kv}\Big) + \Omega\Big(\frac{m\eta_e\eta_{\text{p}} t'^2}{k^2 v^2}\Big)$$

which implies for $\eta_{\text{P}} = O(\frac{\eta_{\text{e}}\, C_{\text{P}}}{mk^2 v^2 C_2^2})$:

$$p_1^{(s,T_1+t')} \geq p_1^{(s,0)}$$

Similarly, for $t > T_1$: we can also show that

$$\left(z_s^{(T_1+t')} - z_i^{(T_1+t')}\right) \geq \left(z_s^{(T_1)} - z_i^{(T_1)}\right) + \Omega\left(\frac{mC_2\eta_{\text{S}}\, n}{Nv}t'\right) + \Omega(\frac{m\eta_{\text{e}}\eta_{\text{S}}}{kNv^2}t'^2) \tag{33}$$

We know, due to initialization, $\exists s \in \{s \in S_1 : p_1^{(s,0)} = \Omega(1)\}$ and $i \in [N]$ for which

$$\left|z_s^{(0)} - z_i^{(0)}\right| \leq \log(k)$$

As a result we can write, for $t = T_1 + T_2$, where $T_2 = \Omega(v\sqrt{\frac{kN\log(k)}{m\, n_{\text{e}} n_{\text{S}}}})$:$\exists s \in S_1$ with $p_1^{(s,0)} = \Omega(1)$ for which $G^{(s,t)} \geq \frac{1}{2}$,

(ii) Using the same method as of (i), considering Lemma 7, we can complete the proof.

□

D.3 Proof of Theorem 1

Proof. For any $s \in \{i \in S_1 : p_1^{(s,0)} = \Omega(1)\}$, we know that: $\left\langle w_r^{(s,T+1)}, o_1 \right\rangle \geq \Omega(v\, C_2\, \sqrt{d\, \log(v)})$

Also, for any $s \in S_2, \forall r$, $\left\langle \frac{\partial l}{\partial w_r^{(s,T)}}, o_1 \right\rangle \geq 0$. Hence, $\left\langle w_r^{(s,T''+1)}, o_1 \right\rangle \leq \left\langle w_r^{(s,T'')}, o_1 \right\rangle \leq O(C_2)$.
Moreover, for any $\forall s, \forall r$, $\left\langle \frac{\partial l}{\partial w_r^{(s,T)}}, q \right\rangle \geq -O(\frac{1}{d})$. Hence, $\left\langle w_r^{(s,T+1)}, q \right\rangle \leq \left\langle w_r^{(s,0)}, q \right\rangle + O(\frac{\eta_e \cdot T}{d}) \leq O(C_2) + O(\frac{v^2 C_2 \sqrt{d\log(v)}}{d})$ At first, the theorem is proved for samples with $y = +1$. After that similar technique is used for the case $y = -1$.

The final output is the sum of the outputs from the top-k selected experts, which are denoted by the set $J_{\text{S}}^{(t)}(x)$:

$$\begin{aligned} f^{(T+1)}(x) &= \sum_{s \in J_{\text{S}}^{(T+1)}(x)} f_s^{(T+1)}(x) \\ &= \sum_{s \in J_{\text{S}}^{(T+1)}(x)} a^{(s)} G^{(s,T+1)} \sum_{j \in J_{\text{P}}^{(T+1)}(x)} H_j^{(s,T+1)} \sum_{r \in [m]} \langle w_r^{(s,T+1)}, x^{(j)} \rangle \end{aligned}$$

We can split the sum over the sets of positive experts (S_1) and negative experts (S_2):

$$\begin{aligned} f^{(T+1)}(x) = &\sum_{s \in J_{\text{S}}^{(T+1)} \cap S_1} G^{(s,T+1)} \sum_{j \in J_{\text{P}}^{(T+1)}(x)} H_j^{(s,T+1)} \sum_{r \in [m]} \left\langle w_r^{(s,T+1)}, x^{(j)} \right\rangle \\ &- \sum_{s \in J_{\text{S}}^{(T+1)} \cap S_2} G^{(s,T+1)} \sum_{j \in J_{\text{P}}^{(T+1)}(x)} H_j^{(s,T+1)} \sum_{r \in [m]} \left\langle w_r^{(s,T+1)}, x^{(j)} \right\rangle \end{aligned}$$

To establish a lower bound, we consider the contribution only from the *important* positive experts ($s \in \{i \in S_1 : p_1^{(s,0)} = \Omega(1)\}$) and subtract the maximum possible negative contribution from all

experts in S_2 that are selected by batch-level router:

$$
\begin{aligned}
f^{(T+1)}(x) \geq & \sum_{s \in J_{\mathrm{S}}^{(T+1)} \cap \{i \in S_1 : p_1^{(s,0)} = \Omega(1)\}} G^{(s,T+1)} \sum_{j \in J_{\mathrm{P}}^{(T+1)}(x)} H_j^{(s,T+1)} \sum_{r \in [m]} \left\langle w_r^{(s,T+1)}, x^{(j)} \right\rangle \\
& - \sum_{s \in J_{\mathrm{S}}^{(T+1)} \cap S_2} G^{(s,T+1)} \sum_{j \in J_{\mathrm{P}}^{(T+1)}(x)} H_j^{(s,T+1)} \sum_{r \in [m]} \left\langle w_r^{(s,T+1)}, x^{(j)} \right\rangle \\
\geq & \sum_{s \in J_{\mathrm{S}}^{(T+1)} \cap \{i \in S_1 : p_1^{(s,0)} = \Omega(1)\}} G^{(s,T+1)} H_1^{(s,T+1)} \sum_{r \in [m]} \left\langle w_r^{(s,T+1)}, o_1 \right\rangle \\
& - \sum_{s \in J_{\mathrm{S}}^{(T+1)} \cap S_2} G^{(s,T+1)} \sum_{j \in J_{\mathrm{P}}^{(T+1)}(x)} H_j^{(s,T+1)} \sum_{r \in [m]} \left\langle w_r^{(s,T+1)}, q \right\rangle
\end{aligned}
$$

From Lemma 1 and Lemma 2, we can rewrite $f^{(T+1)}(x)$ as following:

$$
\begin{aligned}
f^{(T+1)}(x) \geq & \, G^{(s,T+1)} H_1^{(s,T+1)} \sum_{r \in [m]} \left\langle w_r^{(s,T+1)}, o_1 \right\rangle \\
& - \sum_{s \in J_{\mathrm{S}}^{(T+1)} \cap S_2} G^{(s,T+1)} \sum_{j \in J_{\mathrm{P}}^{(T+1)}(x)} H_j^{(s,T+1)} \sum_{r \in [m]} \left\langle w_r^{(s,T+1)}, q \right\rangle \\
& \geq \Omega(mC_2 v \sqrt{d \log(v)}) - O(kmC_2) - O(kmv^2 C_2 \sqrt{\frac{\log(v)}{d}})
\end{aligned}
$$

For $k = O(\sqrt{d})$, we can conclude $f^{(T+1)}(x) > 0$.
Hence, for batches with $y = +1$: we will find $y f^{(T+1)}(x) > 0$
Similarly, using the same technique we can show that,
for batches with $y = -1$: we will find $y f^{(T+1)}(x) > 0$
This concludes the proof.

□