LExI: Layer-Adaptive Active Experts for Efficient Mixture-of-Expert Model Inference

Krishna Teja Chitty-Venkata [*1] **Murali Emani** [1]

Abstract

Mixture-of-Experts (MoE) models scale efficiently by activating only a subset of experts per token, offering a computationally sparse alternative to dense architectures. While prior post-training optimizations, such as inter- and intra-expert pruning, reduce memory usage but provide limited gains in inference-time compute efficiency. Moreover, existing MoE architectures typically activate a fixed number of experts uniformly across all layers, resulting suboptimal performance. In this work, we first demonstrate that MoE pruning improves only the memory footprint but does not significantly improve inference performance. To address this, we introduce **LExI**, a data-free optimization technique that determines the optimal number of active experts per layer in a pretrained MoE. LExI leverages only the model's weights to estimate the relative importance of each layer and adaptively assigns the number of active experts per layer. Experiments on several MoEs demonstrate that LExI significantly outperforms traditional MoE pruning approaches in terms of inference efficiency with negligible accuracy loss. For example, using LExI, Qwen1.5-MoE achieves the same throughput on Nvidia H100 GPU with 10% better accuracy than traditional expert pruning.

1. Introduction

Large Language Models (LLMs) and Vision Language Models (VLMs) have achieved remarkable performance through model scaling, but require tremendous compute resources. Mixture-of-Experts (MoE) models (Cai et al., 2025) have emerged as a promising approach to increase model capacity without a proportional rise in inference cost. In an MoE, multiple expert subnetworks are trained, and a sparse gating or router network activates only a small subset of experts (k experts) per input token. This sparse computation allows MoEs to outperform dense models with the same number of active model parameters. Prominent examples include Mixtral (Jiang et al., 2024), Qwen1.5-MoE-A2.7B (Team, 2024), OLMoE (Muennighoff et al., 2024), MolmoE (Deitke et al., 2024) and DeepSeek-VL2 (Wu et al., 2024).

One commonly used post-training optimization for MoEs is expert pruning, which removes redundant experts. Recent methods such as NAEE (Lu et al., 2024), MoE-Pruner (Xie et al., 2024) EEP (Liu et al., 2024b) and MoE-I^2 (Yang et al., 2024) introduce various pruning strategies. NAEE removes experts from an MoE, while MoE-I^2 prunes the FFN dimension within each expert. Although these methods reduce the memory, our performance evaluation across several MoEs using the vLLM inference framework reveals a critical limitation: *pruning does not consistently translate into faster inference*, and in some cases, it even degrades performance. This degradation is primarily due to the sparse structure of MoEs. In expert pruning, the input token still needs to be routed to the same number of top-k experts, as determined by the router. While some experts are pruned, the remaining ones must process a disproportionately larger number of tokens, increasing their computational load. In batched inference, this load imbalance can lead to longer processing time per expert, thereby increasing overall latency. While aggressive expert pruning levels can yield noticeable speedups, they typically result in significant accuracy degradation, making them impractical. Moreover, these expert pruning approaches usually rely on training data for pruning experts.

The current MoE models use fixed top-k routing, where the same number of experts is activated for each token. This static design is suboptimal as different layers may require different numbers of active experts depending on token (Guo et al., 2024). Beyond computation, inter-GPU communication overhead also becomes a significant performance bottleneck in MoEs. Increasing the number of active experts

[*]Work done while at Argonne National Laboratory. Now at Red Hat AI. [1]Argonne National Laboratory, Lemont, IL, USA. Correspondence to: Krishna Teja Chitty-Venkata <schittyvenkata@anl.gov>, Murali Emani <memani@anl.gov>.

AdaptFM: Resource-Adaptive Foundation Model Inference Workshop in the 43^{rd} International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026.

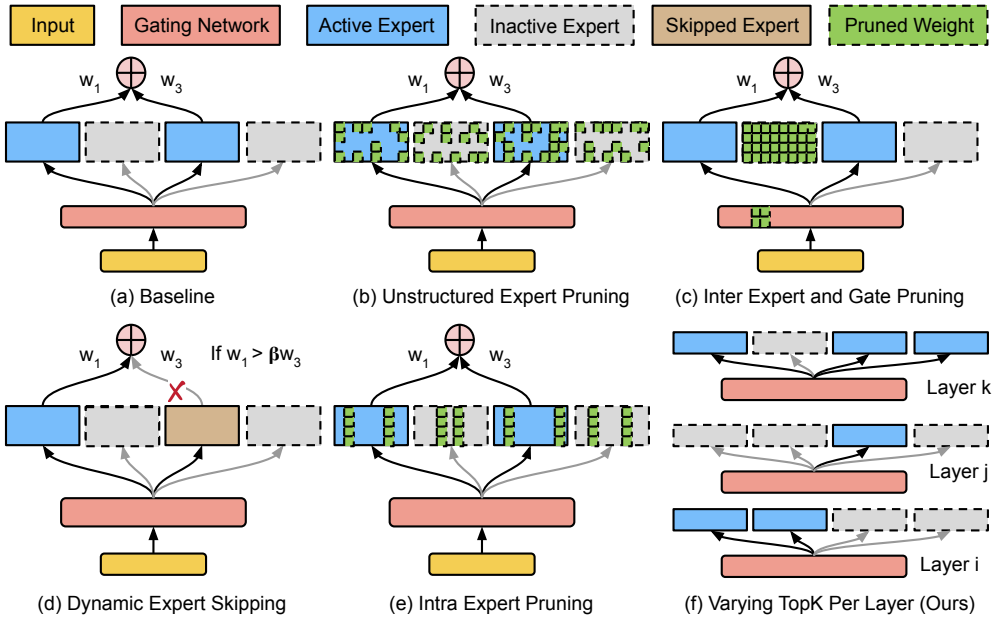


Figure 1. Overview of MoE Optimization Methods. (a) Baseline Trained Model (b) Unstructured Expert Pruning (SparseGPT (Frantar & Alistarh, 2023), Wanda (Sun et al., 2023), MoE-Pruner (Xie et al., 2024)) (c) Inter Expert Pruning (NAEE (Lu et al., 2024)) (d) Dynamic Expert Skipping (NAEE (Lu et al., 2024)) (e) Intra Expert Pruning (MoE-I^2 (Yang et al., 2024)) (f) **LExI**: Static Varying Active Experts (Topk) Per Layer (Ours)

per token increases the volume of communication operations such as all-reduce and broadcast, further adding to the inference cost. To address some of these limitations, NAEE (Lu et al., 2024) proposed a token-aware dynamic expert skipping strategy, which selectively skips an expert during inference. However, this strategy is highly tailored to the dataset and cannot work beyond top-k=2. In summary, existing MoE optimization strategies often rely on calibration sets for pruning or routing adjustments, making them unsuitable in deployment settings where access to data or retraining is infeasible. In addition, the dataset-driven solutions will make pruned models more optimized to that calibration dataset, potentially degrading the performance in unseen settings.

We propose ***LExI***, a novel post-training layer-adaptive active expert allocation mechanism that determines the optimal number of active experts per layer without depending on any dataset. Our method relies on the key observation that not all layers contribute equally to the final model performance and that expert redundancy varies significantly across depth. This raises a fundamental question: *Can we reduce the number of active experts per layer irrespective of the input token without sacrificing accuracy?* In particular, is it possible to statically assign different top-k values to each layer, so that every layer uses just enough experts to retain its contribution, while improving overall inference efficiency? By making layer-adaptive expert allocation decisions, LExI reduces computational overhead across the model irrespective of the input token, offering a more efficient alternative to the traditional fixed top-K routing. Our experiments show that LExI outperforms existing expert pruning techniques in both task performance and runtime efficiency. By reducing the average number of activated experts per layer, LExI reduces latency and memory bandwidth usage while maintaining competitive task accuracy across diverse tasks.

Contributions. Our contributions are as follows:

- We introduce **LExI**, a novel dataset-free optimization technique for static active expert assignment in pretrained MoE models. LExI is simple to implement and serves as an efficient, plug-and-play solution for inference across frameworks.
- We propose a data-free strategy to estimate the sensitivity of each expert using only expert weights. LExI combines this one-time profiling with evolutionary search to determine the optimal active experts layer in a computationally efficient manner.
- Unlike prior methods that demonstrate improvements on a narrow set of MoE models (Mixtral-8x7B), our approach generalizes across multiple state-of-the-art MoE architectures in both language and vision domains.
- We empirically show that expert pruning in MoE models does not significantly improve inference performance, and in some cases, can degrade it due to architectural sparsity and load imbalance. Our method provides a viable alternative to pruning, improving both accuracy and

hardware efficiency without requiring retraining or access to calibration data.

2. Background and Related Work

Mixture of Experts (MoEs) MoEs improve the scalability and efficiency of LLMs by introducing subnetworks or experts. For a given input x, the output y of an MoE module is computed as a weighted sum over all the active top-k experts: $y = \sum_{i=1}^{top-k} G(x)_i \cdot E_i(x)$, where $G(x) := \text{Softmax}(\text{TopK}[x \cdot W_g])$. The TopK$[\cdot]$ function selects the top-k experts with the highest gating scores, and Softmax normalizes their scores into a probability distribution. Each expert E_i is typically an FFN and constitutes the dominant portion of the parameters model.

Pruning LLMs Pruning is an established technique to reduce inference costs by removing less important parameters. Recent works such as SparseGPT (Frantar & Alistarh, 2023) and Wanda (Sun et al., 2023) demonstrated one-shot pruning methods that introduce unstructured or semi-structured sparsity, cutting up to 50% of parameters in GPT-scale models. These approaches solve layer-wise reconstruction or use weight magnitude heuristics to remove weights. However, the irregular weight sparsity often requires specialized hardware support to achieve speedups (Zhou et al., 2021), and may suffer efficiency on accelerators.

Expert Pruning and Compression in MoEs Since MoE models typically allocate the vast majority of their parameters to the expert sub-networks, pruning even a subset of experts can lead to substantial memory savings. NAEE (Lu et al., 2024) is a post-training expert pruning framework to permanently remove unimportant experts without needing to re-train the model. By evaluating each expert's contribution to the model's output on a small calibration set, NAEE identifies and permanently prunes the least significant experts. Furthermore, NAEE also introduced an inference-time policy to dynamically skip experts for certain tokens on the fly, effectively adjusting the active expert count based on the input token. Another recent approach is MoE-I2 (Yang et al., 2024), which introduces a two-stage compression pipeline tailored for MoEs. In the inter-expert pruning stage, MoE-I2 performs a layer-wise analysis to prune a fraction of experts to prune. The authors also introduce intra-expert compression to reduce the inner dimensionality of an expert's FFN. These advances in MoE-specific pruning highlight the growing interest in expert-level model trimming. Our proposed LExI method shares the overarching goal of exploiting expert redundancy to improve efficiency. However, rather than relying on static pruning, it focuses on adaptive expert utilization at inference time, offering a flexible and data-free alternative that preserves task performance while reducing computational cost.

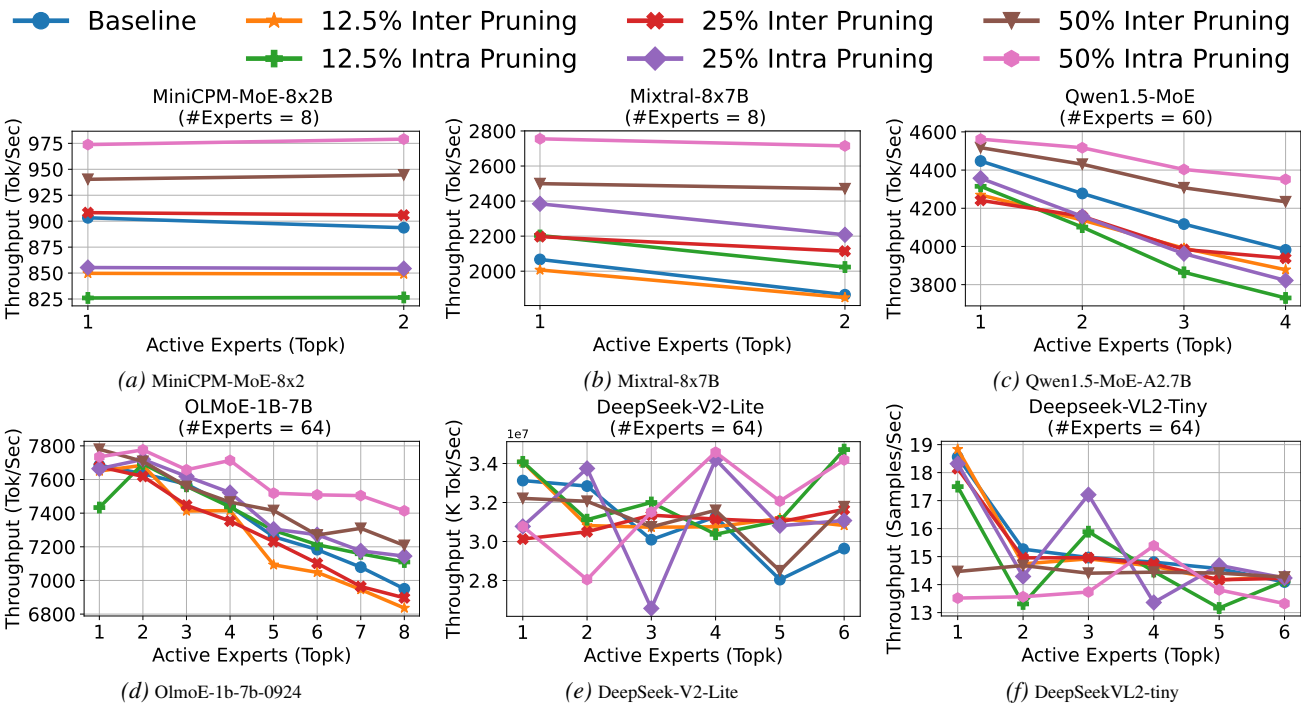
3. MoE Expert Latency Profiling

In this section, we evaluate the hardware performance of the MoE benchmarks to motivate our varying top-k solution.

Mixture-of-Experts Benchmarks. We evaluate our method on a diverse set of MoE models spanning both language and vision-language domains, including LLMs such as Mixtral-8x7B-Instruct (Jiang et al., 2024), Qwen1.5-MoE-A2.7B-Chat (Team, 2024), OLMoE-1B-7B-0924-Instruct (Muennighoff et al., 2024), MiniCPM-MoE-8x2B (Hu et al., 2024), and DeepSeek-V2-Lite-Chat (Liu et al., 2024a), as well as the DeepSeekVL2-Tiny VLM (Wu et al., 2024). As shown in Table 1, these models exhibit substantial architectural diversity, ranging from smaller models like DeepSeek VL2-Tiny (3B parameters, 12 layers, 64 experts, Top-6, FFN dimension 896) to larger architectures such as Mixtral-8x7B-Instruct-v0.1 (46.7B parameters, 32 layers, 8 experts, Top-2, FFN dimension 14336), with intermediate models including OLMoE-1B-7B-0125-Instruct (6.92B, 16 layers), Qwen1.5-MoE-A2.7B-Chat (14.3B, 24 layers), DeepSeek-V2-Lite-Chat (15.7B, 27 layers), and MiniCPM-MoE-8x2B (17B, 40 layers). This heterogeneous set of models with varying numbers of experts, active experts (*TopK*), and architectural choices enables a robust evaluation of our proposed method across different MoE configurations.

Hardware and Software Setup. All inference performance evaluations are conducted on *NVIDIA H100 GPUs* with 80GB memory per GPU, with Tensor Cores for optimized matrix operations. We use `vLLM` (Kwon et al., 2023) as our inference engine, which is a high-performance framework with native support for MoEs via *FusedMoE*, which fuses expert computation and routing to improve efficiency. Unless otherwise specified, all LLMs are deployed on 4 GPUs, while DeepSeek-V2-Lite-Chat and DeepSeekVL2-Tiny use 2 GPUs. We use tensor parallelism across devices for all models. During inference, we use a batch size of 16, with input and output sequence lengths varied across models to comply with each model's maximum context length constraints. We report *throughput* as the performance metric, defined as the total number of tokens (input + output) processed per second. We first measure the *end-to-end latency*, defined as the time elapsed from input prompt submission to the generation of the final output token, and then convert this latency into throughput. The metric for VLMs is the number of input (image + text) samples process per second.

Inter and Intra Expert Pruning. Inter-pruning (Lu et al., 2024) prunes experts and their routing weights, while maintaining the same number of active experts. Intra-pruning (Yang et al., 2024) targets the FFN intermediate size within each expert, preserving the expert count while reducing individual expert complexity. We consider the following pruning percentages: $\{12.5\%, 25\%, 50\%\}$. 12.5% inter pruning removes $1/8$th of the experts in each layer, whereas



(a) MiniCPM-MoE-8x2 *(b)* Mixtral-8x7B *(c)* Qwen1.5-MoE-A2.7B

(d) OlmoE-1b-7b-0924 *(e)* DeepSeek-V2-Lite *(f)* DeepSeekVL2-tiny

Figure 2. Throughput vs. Active Experts under Inter and Intra Expert Pruning

25% intra pruning prunes $1/4$th of the FFN dimension in each expert. The top-k search space in our paper includes every integer from 1 up to the baseline pretrained top-k: $1, 2, \ldots,$ top-k_{baseline}.

Figure 2 illustrates the throughput of the six MoEs under varying degrees of pruning and top-k. Models with fewer active experts (e.g., MiniCPM, Mixtral) show marginal gains with aggressive pruning, while models with more active experts (e.g., Qwen, OLMoE) exhibit complex interactions where pruning can improve or degrade throughput depending on token-to-expert routing balance and compute saturation. DeepSeek-VL2-Tiny shows throughput instability under pruning, suggesting higher sensitivity to expert load balance. The performance degradation comes from load imbalance across experts, leading to an increased number of tokens processed by each expert.

4. LExI

LExI implements a two-stage pipeline to determine the optimal number of active experts per layer. In the first stage, it performs a one-time profiling to assess each layer's sensitivity to different top-k values. The sensitivity profiling methodology extends beyond expert allocation, serving as a foundation for diverse optimization problems such as layer-specific mixed-precision quantization or layer-wise pruning. The second stage employs a low-cost evolutionary search algorithm leveraging these sensitivity values as efficient proxies to identify the best performing top-k for each layer.

4.1. Stage 1: Per Layer MoE Top-K Perturbation

We use a Monte Carlo-based method (Algorithm 1) to evaluate the sensitivity of each layer under varying top-ks. For each layer, we sample a random synthetic input tensor $\mathbf{X} \sim \mathcal{N}(0,1)^{B\times L\times H}$ from the standard normal distribution. We compute the baseline output using the default top-k configuration, followed by computing outputs for each top-k in the target search space. The perturbation induced by each top-k is quantified using the Frobenius norm between the baseline output and the corresponding perturbed output. This process is repeated over millions of random input samples to obtain a statistically robust estimate of the average deviation for each candidate top-k. This norm serves as a metric to capture output magnitude shifts in high-dimensional space, while Monte Carlo ensures diverse inputs. This sensitivity analysis provides an estimate of how active expert selection affects per-layer behavior.

Why Gaussian? We purposefully used Gaussian inputs with zero mean and unit variance for sensitivity profiling to approximate the post-LayerNorm outputs in the decoder layer. In all LLMs, a LayerNorm precedes the MoE module, ensuring that input activations are standardized to have mean zero and variance one. Although these normalized activations do not strictly follow a Gaussian distribution, they are nonetheless centered and scaled in a comparable manner, making Gaussian inputs a reasonable proxy. This choice provides

a consistent and unbiased basis for assessing the intrinsic behavior of each layer under different top-k configurations.

4.2. Top-k Perturbation Sensitivity Analysis

Figure 6 visualizes the normalized sensitivity of different MoEs under various top-ks. Higher values of Δ_k indicate greater deviation from the baseline behavior, suggesting stronger sensitivity to changes in top-k. For Mixtral-8x7B, early layers demonstrate greater sensitivity to reductions in number of active experts, as indicated by lower perturbation loss, while later layers are more sensitive under top-k perturbation. Interestingly, this finding diverges from prior works (Dong et al., 2019), which suggests that early layers are typically more susceptible to number of active experts. In contrast, Qwen1.5-MoE-A2.7B exhibits a reversed pattern where early layers are particularly sensitive to top-k perturbations. DeepSeek model displays a bell-shaped sensitivity profile, with both initial and final layers demonstrating higher perturbation loss, while intermediate layers remain relatively stable. These findings have practical implications for adaptive expert selection strategies suggesting that latency can be can be optimized by reducing the number of active experts in more robust (low-sensitivity) layers.

4.3. Stage 2: Evolutionary Search with Proxy

LExI utilizes the proxies generated in the previous step to guide the evolutionary algorithm to allocate layer-wise top-k (Algorithm 2). We define the active expert budget (total active experts across all layers) B, minimum ($k_{\min}$) and maximum ($k_{\max}$) number of top-k per layer. The objective is to find a feasible allocation $k^* = (k_1^* \ldots, k_L)$ which minimizes the total layer-wise loss $\sum_{j=1}^{L} \mathcal{D}j(k_j)$ (sum of TopK Perturbed Frobenius Norm losses) across L layers, subject to the budget constraint $\sum j = 1^L k_j = B$. We initialize a population of N_{pop} allocations (each satisfying the constraints) and then evolve this population over $G_{\max}$ generations. In each generation, every candidate solution is evaluated by its fitness $\phi(k) = \sum_{j=1}^{L} \mathcal{D}j(k_j)$, and parent solutions are chosen (e.g. via tournament selection) to produce offspring. A pair of parents is recombined using uniform crossover, wherein each layer's allocation k_j in the offspring is inherited from one of the two parents. The offspring is mutated and ensured that the total budget B remains unchanged. After mutation, the new solution is added to the population. This evolutionary loop of selection, crossover, mutation, and repair is repeated for up to $G_{\max}$ generations and the optimal allocation k^* is returned. By maintaining the proxies, our LExI algorithm effectively navigates the combinatorial search space of discrete allocations and finds solutions fast without needing to load the actual model, making it well-suited for optimizing top-k selection under various global active expert budgets.

5. Results and Evaluation

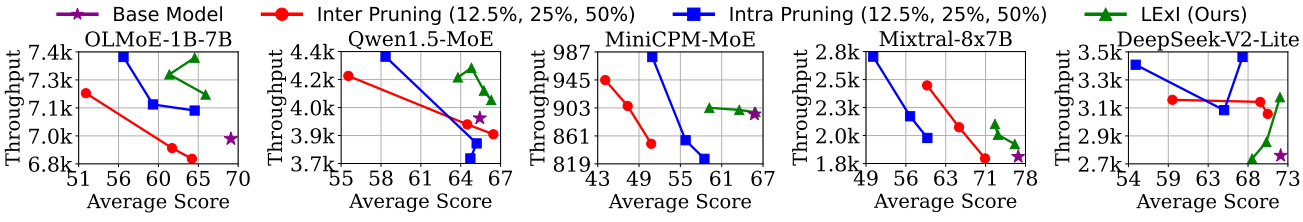
This section presents our evaluation results and key insights from these runs. We compare our results with the baseline model, Inter Pruning (Lu et al., 2024) and Intra Pruning (Yang et al., 2024).

5.1. LM-Eval Results

Figure 3 presents a comparison of average accuracy versus throughput across MoE models across nine language modeling benchmarks. Traditional pruning methods, inter-expert (red) and intra-expert (blue), consistently demonstrate a trade-off: reducing the number of parameters improves throughput but significantly degrades accuracy. In contrast, our proposed method, LExI (green), achieves a more favorable accuracy-throughput balance across all models. On OLMoE-1B-7B (Figure 3a), LExI with the active expert budget (B) = 100 achieves the same throughput as 50% intra-pruning while delivering $+10\%$ higher accuracy, and outperforms the 50% inter-pruning baseline by +15% higher accuracy. On Qwen1.5-MoE, LExI offers at least $+5.1\%$ higher throughput compared to both inter and intra pruning, with a consistent $+0.5$ % accuracy gain. For MiniCPM-MoE, LExI achieves $+15\%$ higher accuracy than 25% inter-pruned models at equivalent throughput, demonstrating superior accuracy-throughput tradeoffs. For Mixtral-8x7B, LExI surpasses the inter-pruned baseline by $+10\%$ accuracy at nearly identical throughput. On DeepSeekV2-Lite, LExI outperforms inter pruning with $+6.5\%$ higher throughput at equal accuracy, and recovers $+6$ accuracy points over intra pruning with only a minor throughput compromise. Notably, in Qwen1.5-MoE and MiniCPM-MoE, LExI not only avoids accuracy degradation but also achieves throughput comparable to or better than 50% inter- or intra-expert pruning. These results highlight the robustness of LExI's expert budget reallocation in preserving model performance while improving inference efficiency.

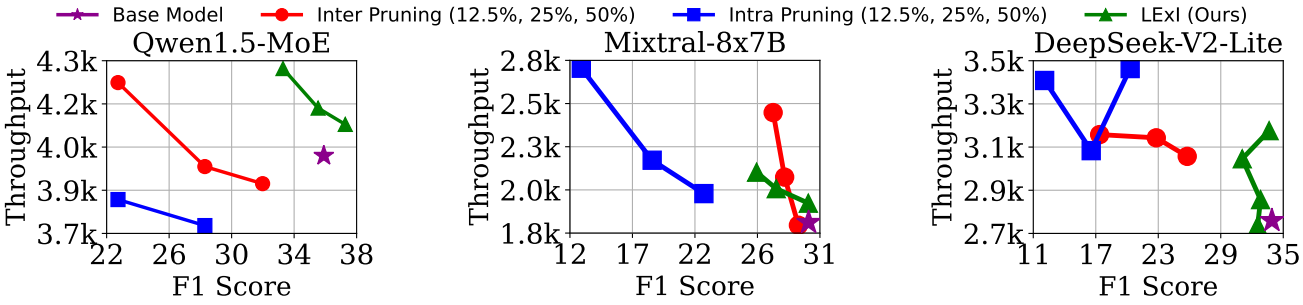
5.2. Long Context Evaluation

Figure 4 presents an accuracy-throughput comparison on the Qasper dataset. Inter and intra pruning methods consistently reduce throughput but often come at a sharp cost in F1 score, particularly under higher pruning ratios. In contrast, LExI achieves a more favorable Pareto optimality (F1 score and throughput). On Qwen1.5 and DeepSeek models, LExI achieves higher throughput than the base model while maintaining a competitive F1 score, demonstrating its efficacy in preserving task accuracy while enhancing inference efficiency. On Qwen1.5-MoE-A2.7B, LExI achieves a score of 35.5 at ~ 4.1k throughput, outperforming inter pruning (F1 34 at ~ 3.9k) and intra pruning (F1 30 at ~ 3.75k) with a $+0.5$–5.5 F1 gain and $+5.1\%$–9.3% higher throughput.



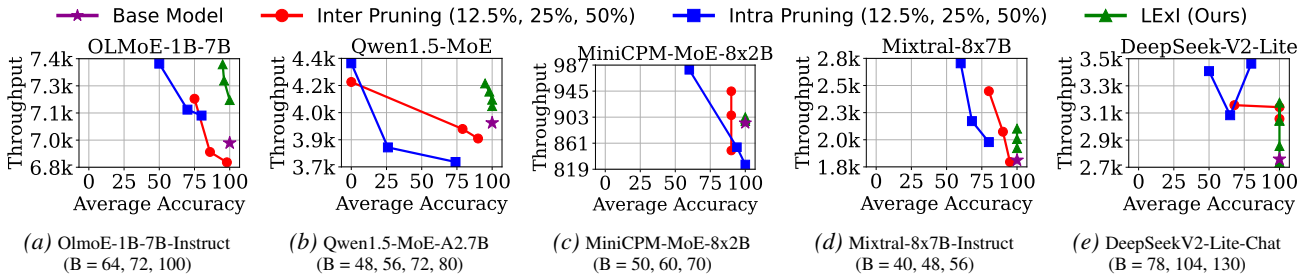
(a) OlmoE-1B-7B-Instruct (B = 64, 72, 100) *(b)* Qwen1.5-MoE-A2.7B (B = 48, 56, 72, 80) *(c)* MiniCPM-MoE-8x2B (B = 50, 60, 70) *(d)* Mixtral-8x7B-Instruct (B = 40, 48, 56) *(e)* DeepSeekV2-Lite-Chat (B = 78, 104, 130)

Figure 3. Average Accuracy (↑) vs Throughput (↑) on 9 LM-Eval Tasks (ARC-c, ARC-e, BoolQ, HellaSwag, MMLU, OBQA, RTE, WinoGrande). B: Active Expert Budget



(a) Qwen1.5-MoE-A2.7B (B = 48, 56, 72, 80) *(b)* Mixtral-8x7B-Instruct (B = 40, 48, 56) *(c)* DeepSeekV2-Lite-Chat (B = 78, 104, 130)

Figure 4. F1 Score (↑) vs Throughput (↑) on Qasper Dataset in LongBench. B: Active Expert Budget



(a) OlmoE-1B-7B-Instruct (B = 64, 72, 100) *(b)* Qwen1.5-MoE-A2.7B (B = 48, 56, 72, 80) *(c)* MiniCPM-MoE-8x2B (B = 50, 60, 70) *(d)* Mixtral-8x7B-Instruct (B = 40, 48, 56) *(e)* DeepSeekV2-Lite-Chat (B = 78, 104, 130)

Figure 5. Passkey Retrieval Average (↑) vs Throughput (↑) Comparison. B: Active Expert Budget

5.3. Passkey Retrieval Task

Figure 5 illustrates the trade-off between throughput and accuracy for the Passkey Retrieval across five MoEs. This task evaluates a model's ability to extract precise key information embedded in distractive or noisy contexts, demanding both precision and robustness. Traditional inter- and intra-pruning approaches generally degrade performance, with noticeable accuracy drops even at moderate pruning levels. In contrast, our proposed LExI method consistently outperforms these baselines by achieving higher or comparable accuracy while significantly improving throughput. Notably, in models like OLMoE-1B-7B and Qwen1.5-MoE, LExI nearly restores or surpasses the base model's accuracy while offering improved efficiency. These results highlight LExI's strength in preserving critical retrieval capabilities under expert budget constraints, making it a superior choice for precision-sensitive tasks like passkey extraction.

6. Conclusion

In this paper, we propose LExI, a framework to search for the optimal number of active experts per layer of a pre-trained MoE. LExI achieves better throughput than both inter and intra expert pruning methods. Unlike uniform expert pruning, which can yield speedups only at the cost of substantial accuracy loss, our method delivers substantial gains while preserving model accuracy. For example, on OLMoE-1B-7B, Mixtral-8x7B, and DeepSeek, LExI maintains accuracy nearly identical to the unpruned model yet attains higher throughput, often surpassing the accuracy of pruned models. In certain scenarios, this layer-adaptive approach achieves higher throughput than the base model, highlighting the robustness of our approach.

References

Cai, W., Jiang, J., Wang, F., Tang, J., Kim, S., and Huang, J. A survey on mixture of experts in large language models. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

Deitke, M., Clark, C., Lee, S., Tripathi, R., Yang, Y., Park, J. S., Salehi, M., Muennighoff, N., Lo, K., Soldaini, L., et al. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models. *arXiv preprint arXiv:2409.17146*, 2024.

Dong, Z., Yao, Z., Gholami, A., Mahoney, M. W., and Keutzer, K. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 293–302, 2019.

Frantar, E. and Alistarh, D. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pp. 10323–10337. PMLR, 2023.

Guo, Y., Cheng, Z., Tang, X., Tu, Z., and Lin, T. Dynamic mixture of experts: An auto-tuning approach for efficient transformer models. *arXiv preprint arXiv:2405.14297*, 2024.

Hu, S., Tu, Y., Han, X., He, C., Cui, G., Long, X., Zheng, Z., Fang, Y., Huang, Y., Zhao, W., et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., Casas, D. d. l., Hanna, E. B., Bressand, F., et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023.

Liu, A., Feng, B., Wang, B., Wang, B., Liu, B., Zhao, C., Dengr, C., Ruan, C., Dai, D., Guo, D., et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.

Liu, E., Zhu, J., Lin, Z., Ning, X., Blaschko, M. B., Yan, S., Dai, G., Yang, H., and Wang, Y. Efficient expert pruning for sparse mixture-of-experts language models: Enhancing performance and reducing inference costs. *arXiv preprint arXiv:2407.00945*, 2024b.

Lu, X., Liu, Q., Xu, Y., Zhou, A., Huang, S., Zhang, B., Yan, J., and Li, H. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. *arXiv preprint arXiv:2402.14800*, 2024.

Muennighoff, N., Soldaini, L., Groeneveld, D., Lo, K., Morrison, J., Min, S., Shi, W., Walsh, P., Tafjord, O., Lambert, N., et al. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*, 2024.

Sun, M., Liu, Z., Bair, A., and Kolter, J. Z. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

Team, Q. Qwen1.5-moe: Matching 7b model performance with 1/3 activated parameters", February 2024. URL https://qwenlm.github.io/blog/qwen-moe/.

Wu, Z., Chen, X., Pan, Z., Liu, X., Liu, W., Dai, D., Gao, H., Ma, Y., Wu, C., Wang, B., et al. Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding. *arXiv preprint arXiv:2412.10302*, 2024.

Xie, Y., Zhang, Z., Zhou, D., Xie, C., Song, Z., Liu, X., Wang, Y., Lin, X., and Xu, A. Moe-pruner: Pruning mixture-of-experts large language model using the hints from its router. *arXiv preprint arXiv:2410.12013*, 2024.

Yang, C., Sui, Y., Xiao, J., Huang, L., Gong, Y., Duan, Y., Jia, W., Yin, M., Cheng, Y., and Yuan, B. Moe-i^2: Compressing mixture of experts models through inter-expert pruning and intra-expert low-rank decomposition. *arXiv preprint arXiv:2411.01016*, 2024.

Zhou, A., Ma, Y., Zhu, J., Liu, J., Zhang, Z., Yuan, K., Sun, W., and Li, H. Learning n: m fine-grained structured sparse neural networks from scratch. *arXiv preprint arXiv:2102.04010*, 2021.

A. Appendix

A.1. Algorithms

This subsection summarizes the two-stage optimization procedure used by LExI. Algorithm 1 computes a layer-wise sensitivity proxy by measuring the perturbation induced by changing top-k at each layer, without requiring calibration data. The resulting sensitivity table $\mathcal{D}$ provides a fast estimate of how costly each top-k choice is for model behavior. Algorithm 2 then uses this proxy inside an evolutionary search to identify a feasible per-layer top-k allocation under a global active expert budget. Together, these algorithms provide a data-free pipeline that balances inference efficiency and model quality through layer-adaptive expert activation.

Algorithm 1 LExI Stage 1: Per Layer Top-k Perturbation Loss Computation

Input: $\mathcal{M}_{\text{moe}}$ (Mixture of Experts module), T (target top-k values), k_{base} (baseline top-k), B (batch size), H (hidden size), L (sequence length), N_{iter} (number of iterations)
Output: Average Frobenius norm per top-k
$\mathcal{D} \leftarrow \{k : \emptyset \; \forall k \in T\}$
for $i = 1$ to N_{iter} **do**
  Sample $\mathbf{X} \sim \mathcal{N}(0,1)^{B \times L \times H}$
  `UpdateTopk`($\mathcal{M}_{\text{moe}}$, k_{base})
  $\mathbf{Y}_{\text{base}} \leftarrow f_{\text{moe}}(\mathbf{X})$
  for all $k \in T$ **do**
    `UpdateTopk`($\mathcal{M}_{\text{moe}}$, k)
    $\mathbf{Y}^{j}_{\text{perturbed}} \leftarrow f_{\text{moe}}(\mathbf{X})$
    $\Delta \leftarrow \|\mathbf{Y}^{j}_{\text{perturbed}} - \mathbf{Y}_{\text{base}}\|_F$
    $\mathcal{D}[k] \leftarrow \mathcal{D}[k] \cup \{\Delta\}$
  end for
end for
for all $k \in T$ **do**
  $\bar{\Delta}_k \leftarrow \frac{1}{N_{\text{iter}}} \sum_{\delta \in \mathcal{D}[k]} \delta$
  $\mathcal{D}[k] \leftarrow \bar{\Delta}_k$
end for
return $\mathcal{D}$

A.2. Limitations

Our approach has two primary limitations. First, it does not reduce the memory footprint of the MoE model. Unlike prior expert pruning methods that explicitly target improving memory efficiency by removing parameters, our method focuses solely on optimizing computational performance during inference. This means that while our approach can significantly speed up inference by reducing the number of expert computations, it does not reduce model size. As a result, it is less effective in memory-constrained deployment scenarios. Nevertheless, our method can be effectively combined with existing MoE pruning methods, enabling a joint optimization of both computational efficiency and model memory. Second, our method may underperform in settings where the top-k expert search space is inherently limited. Our approach relies on selectively reducing the number of active experts during inference to gain computational efficiency. However, in architectures such as Llama-4, where each MoE layer is pretrained with only a single active expert, there is no flexibility to reduce active experts further. In such scenarios, our method becomes inapplicable.

Algorithm 2 LExI Stage 2: Evolutionary Top-k Allocation Optimization with Proxy

Input: $\mathcal{D}$, B, $k_{\min}$, $k_{\max}$, N_{pop}, $G_{\max}$, η_{mut}, L
Output: Optimal topk allocation $\mathbf{k}^* = (k_1, ..., k_L)$
Initialize population $\mathcal{P} \leftarrow \{\mathbf{k}_i\}$ such that $\sum_{j=1}^{L} k_j = B$ and $k^j_{\min} \le k_j \le k^j_{\max}$, $\forall j$
for $g = 1$ to $G_{\max}$ **do**
  Evaluate fitness: $\phi(\mathbf{k}) = \sum_{j=1}^{L} \mathcal{D}_j(k_j)$
  Select parents via tournament: $\mathbf{p}_1, \mathbf{p}_2 \leftarrow \arg\min_{\mathbf{k} \in \mathcal{P}} \phi(\mathbf{k})$
  Crossover: $k'_j \leftarrow \alpha_j p_{1,j} + (1 - \alpha_j) p_{2,j}$, where $\alpha_j \sim \text{Bernoulli}(0.5)$
  Mutation: $k''_j \leftarrow k'_j + \Delta_j$, where $\Delta_j \in \{-1, 0, +1\}$ and $\sum_j \Delta_j = 0$
  Project to feasible space: $\mathbf{k}''' \leftarrow \text{Proj}(\mathbf{k}'')$
  Update population: $\mathcal{P} \leftarrow \mathcal{P} \cup \{\mathbf{k}'''\}$
end for
return $\mathbf{k}^* = \arg\min_{\mathbf{k} \in \mathcal{P}} \phi(\mathbf{k})$

A.3. Model Setup

A.4. Sensitivity Graphs

Figure 6 provides layer-wise sensitivity heatmaps for all evaluated MoE models. Each cell reports the output deviation when the top-k configuration is perturbed at a specific layer, allowing us to identify layers that are robust to expert reduction and layers that are highly sensitive. These patterns motivate the layer-adaptive allocation used by LExI, where more constrained expert budgets are assigned to robust layers and larger budgets are preserved for sensitive layers.

A.5. Additional Results

This subsection reports complementary evidence beyond the main paper metrics. We include perplexity results on language modeling corpora and accuracy-throughput tradeoffs on vision-language benchmarks to test whether the same top-k allocation principles generalize across domains. In both settings, LExI preserves task quality more reliably than pruning-based baselines while maintaining competitive or better throughput.

Table 1. LLM and VLM MoE Models

Model	#P (B)	#Layers	#Experts	TopK	FFN Dim
DeepSeek VL2-Tiny	3	12	64	6	896
OLMoE-1B-7B-0125-Instruct	6.92	16	64	8	1024
Qwen1.5-MoE-A2.7B-Chat	14.3	24	60	4	1408
DeepSeek-V2-Lite-Chat	15.7	27	64	6	1408
MiniCPM-MoE-8x2B	17	40	8	2	5760
Mixtral-8x7B-Instruct-v0.1	46.7	32	8	2	14336

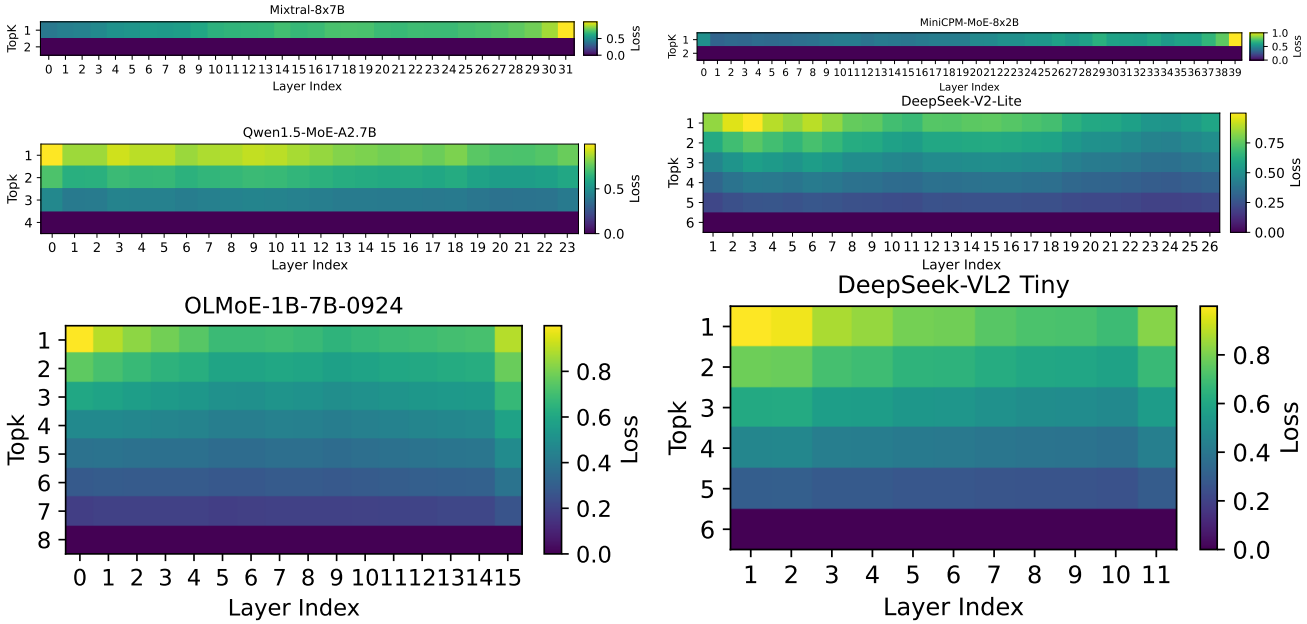


Figure 6. Top-k sensitivity analysis. The heatmap plots depict the layer-wise output deviation with respect to changing the top-k. The initial layers in Mixtral model are less sensitive to top-k perturbation than deeper layers, while OLMoE exhibits a bell curve pattern where initial and last layers are more sensitive.

Perplexity Evaluation. Figure 7 compares perplexity and throughput across multiple MoEs. Across models, LExI consistently offers a better accuracy-efficiency benefit, outperforming both the pruning baselines. Notably, pruning methods often yield modest throughput gains but at the cost of substantial perplexity degradation, especially evident on OLMoE and Mixtral. LExI achieves throughput improvements close to aggressive pruning levels while almost preserving the perplexity of the base model. For example, on Mixtral-8x7B, LExI achieves $\sim$2.4k throughput at $\sim$23 perplexity on C4 dataset and $\sim$2.2k throughput at $\sim$10 perplexity on WikiText, whereas inter pruning attains the same speedup with double the perplexity. This indicates that static expert reduction via LExI enables smarter compute allocation, unlike pruning, which disrupts the expert-token mapping and leads to suboptimal routing and load imbalance.

Vision Language Domain. Figure 8 evaluates the LExI method on DeepSeekVL2-Tiny across four vision-language tasks. While intra-pruning peaks at 25% intra pruning at the cost of sharp accuracy drops, its performance is highly unstable and inconsistent across tasks. Inter-pruning, on the other hand, exhibits a flat trade-off curve that fails to provide significant speedups. In contrast, LExI consistently achieves superior accuracy and throughput balance, yielding improvements without compromising performance. Unlike pruning, which disrupts expert specialization and introduces fragility, LExI leverages structured top-k expert budget selection that preserves model capacity while enabling efficient routing.

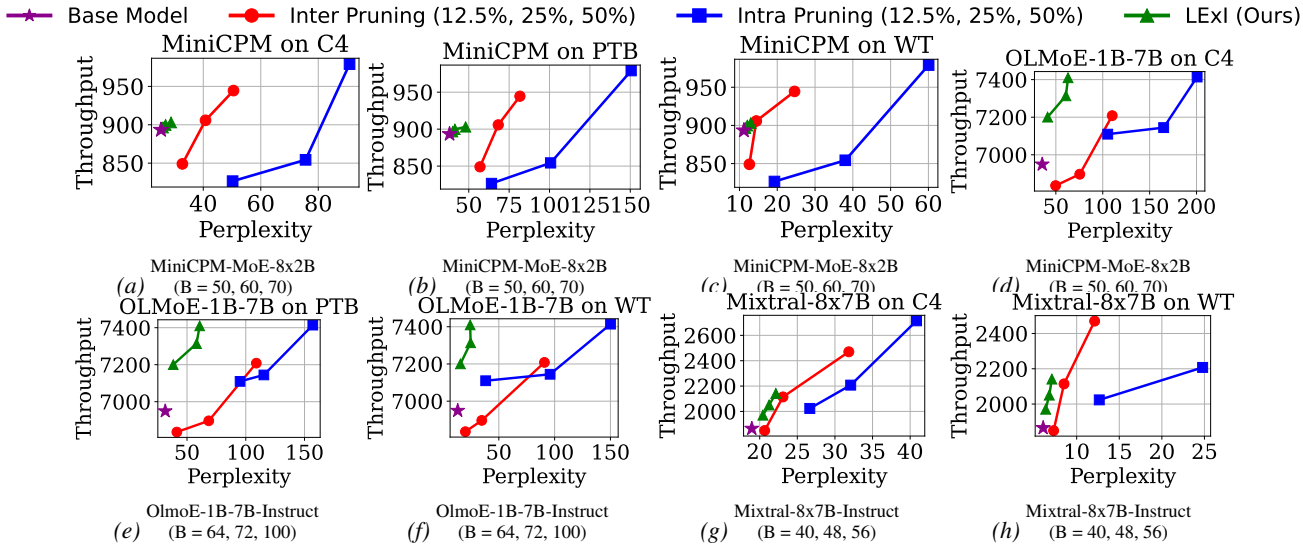


Figure 7. Perplexity (↓) vs Throughput (↑) on C4, PTB & WikiText(WT) B: Active Expert Budget

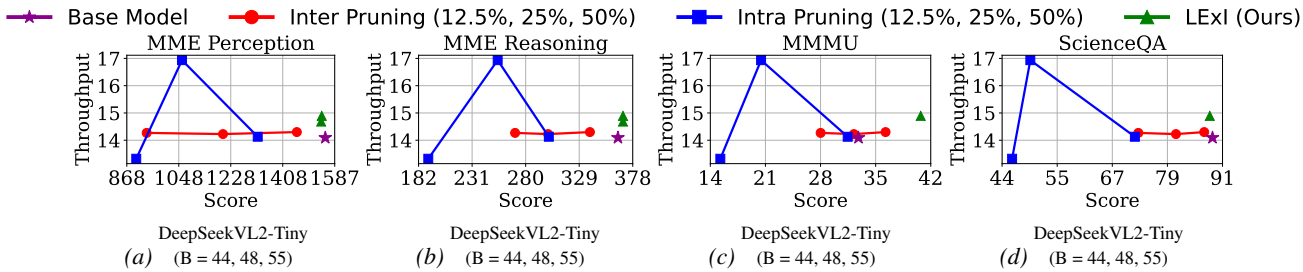


Figure 8. DeepSeekVL2-Tiny: Average Accuracy (↑) vs Throughput (↑) on Vision tasks. B: Active Expert Budget