

Orchestrating Hidden-Intermediate Pruning-and-Distill for MoEs Slimming

Yifu Ding¹, Yongcheng Jing^{1,†}, Chengyu Wang², Jun Huang², Dacheng Tao^{1,†}

¹Nanyang Technological University ²Alibaba Cloud Computing

¹N2506015D@e.ntu.edu.sg, yongcheng.jing@ntu.edu.sg, dacheng.tao@gmail.com

²chengyu.wcy@alibaba-inc.com, huangjun.hj@alibaba-inc.com

Abstract

The remarkable performance of Mixture-of-Experts (MoE) often comes with substantial architectural scale and resource overhead. Prior structural compression mainly focuses on expert-level or FFN intermediate pruning. However, we identify an emerging architectural trend that the FFN expansion ratio in state-of-the-art MoEs has steadily decreased, making intermediate-only pruning less effective and motivating us to explore other redundancies. Motivated by this, we propose an orchestrated hidden-intermediate prune-and-distill framework that, to the best of our knowledge, is the first to jointly prune expert FFNs along intermediate and hidden dimensions. It includes Budget-Constrained Coordinate Descent to efficiently search expert-specific pruning ratios across the two dimensions, and Attention Relation Transfer Distillation to provide stable supervision under hidden misalignment. Experiments across diverse benchmarks validates that hidden-dimension pruning is a promising direction for compressing modern MoEs.¹

1 Introduction

In recent years, Mixture-of-Experts (MoE) have become one of the mainstream architecture due to their ability to scale model capacity without a proportional increase in computational cost (Jiang et al., 2024; Team, 2025). Most prior work compresses MoEs at the expert level. For example, expert merging (Zhang et al., 2024; Miao et al., 2025), dropping (Lu et al., 2024; Chen et al., 2022), or skipping (hao Liu et al., 2024; Gao et al., 2024), as well as reducing only the intermediate width of of Feed-Forward Networks (FFNs) (He et al., 2025; Liang et al., 2025; Li et al., 2025c). Intermediate-only slimming was historically reasonable because FFNs used to be much wider than the hidden size,

[†] Corresponding authors.

¹Our code is available at <https://github.com/yifu-ding/MoE-Joint-Prune>.

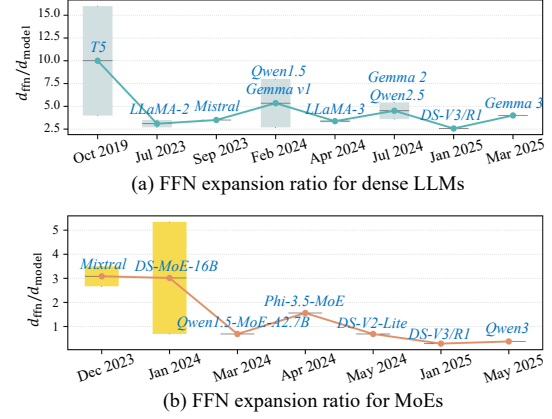


Figure 1: FFN expansion ratio $d_{\text{ffn}}/d_{\text{model}}$ ranges sorted by released time. It shows that the intermediate size is becoming smaller over time, inducing the necessity of compressing other dimensions (e.g., hidden width or attention modules).

meaning the FFN intermediate dimension was the main cause of the large parameter amount.

However, intermediate-only pruning delivers progressively smaller benefits. We discover an architectural shift in both dense and MoE models: the FFN expansion ratio ($d_{\text{ffn}}/d_{\text{model}}$) has been steadily decreasing. As illustrated in Figure 1, earlier MoEs such as Mixtral, typically adopt ratios in the range of 2.67 to 3.50, whereas recent models including DeepSeek-V3 and Qwen3 reduce this ratio to 0.29 and 0.38. This trend makes intermediate-only pruning increasingly less effective, motivating us to explore other sources of redundancy and redirect to hidden dimension pruning. Our scope therefore targets this modern low-expansion-ratio design choice of MoEs.

Nevertheless, hidden-dimension pruning is non-trivial, which raises three key challenges: (i) *Residual incompatibility*. The hidden width is shared by attention, FFNs, and normalization through residual connections, so removing hidden channels can violate shape constraints or break semantic com-

patibility to the residual stream. (ii) *Intermediate vs. hidden budget allocation*. Under an overall compression target, how to split pruning between d_{ffn} and d_{model} is critical, and the optimal allocation can differ across experts due to heterogeneous activation and redundancy patterns. (iii) *Distillation under misalignment*. Hidden pruning changes the representation basis, so teacher and student no longer share aligned hidden channels, making intermediate layer supervision hard to apply.

To address these challenges, we propose an **MoE-tailored Prune-and-Distill framework** that jointly slims expert FFNs across both intermediate and hidden dimensions. To the best of our knowledge, we are the first to study hidden-dimension redundancy in MoEs. It is a *Joint Pruning Framework* to preserve residual compatibility without introducing expensive alignment operators. To allocate sparsity between intermediate and hidden dimensions, we introduce *Budget-Constrained Coordinate Descent* (BCD) algorithm, which efficiently searches expert-wise pruning ratios by probing two budget-preserving coordinate actions with lightweight loss evaluations. To mitigate distillation difficulty under hidden channel misalignment, we adopt the *Relation Transfer Distillation* (RTD) using attention representations, avoiding feature projection for misaligned hidden dimensions while improving optimization stability and convergence.

We evaluate our method on two modern MoE architectures, DeepSeek-V2-Lite and Qwen3-30B-A3B-Thinking, across diverse benchmarks, and provide detailed ablations. Under both 25% and 50% structural pruning, our approach achieves the highest average performance, even surpassing unstructured pruning methods. Notably, on mathematical reasoning our pruned models remain on par with the original model. On Qwen3-30B-A3B-Thinking, our 25% pruned model achieves 96.3% on GSM8K and 93.9% on MATH500, closely matching the original model’s performance on these tasks. These results demonstrate that hidden-dimension pruning is a practical and effective direction for efficient MoE deployment.

Our main contributions are as follows:

- We identify a key trend in modern MoEs: the FFN expansion ratio is shrinking, making intermediate-only pruning less effective.
- To the best of our knowledge, we are the first to study an MoE-tailored joint pruning

scheme that prunes both the hidden and intermediate dimensions inside expert FFNs, enabling practical MoE slimming without introducing extra alignment projections.

- Budget-Constrained Coordinate Descent (BCD) efficiently searches the optimal pruning ratios for intermediate vs. hidden dimensions for each expert, and Relation Transfer Distillation (RTD) provides stable supervision under hidden misalignment.
- Experiments show that our method consistently outperforms intermediate-only pruning and other pruning methods across diverse benchmarks, validating the benefit of joint two-dimensional FFN pruning for efficient MoE deployment.

2 Related Work

Intermediate Dimension Pruning. Most structured pruning methods for LLMs and MoE models primarily target the intermediate dimension in FFNs, since mainstream Transformer and MoE backbones often adopt large FFN expansion ratios, making intermediate pruning effective for reducing parameters and FLOPs (Touvron et al., 2023a; Jiang et al., 2024). SEAP exploits clustering patterns in hidden states and activations to prune task-relevant structures without further training (Liang et al., 2025). SlimMoE estimates sensitivity via top- k logits KL divergence and prunes low-importance neurons in expert MLP projections (Li et al., 2025c). Concurrent to our work, Ding et al. (2026) also study intermediate-dimension pruning, reformulating prune-ratio allocation within experts as a channel-score coverage maximization problem solved via attribution-based approximation.

Hidden Dimension Pruning. Pruning hidden dimensions is challenging due to residual connections, and prior work is limited. Existing LLM methods can be roughly divided into three categories: (1) *Globally shared indices*: Xia et al. (2022, 2023); Liu et al. (2025) learn shared masks to preserve identical hidden indices across layers, maintaining residual shape compatibility. (2) *Extra projections on the residual path*: Ashkboos et al. (2024); Hu et al. (2024) introduce additional linear projections or equivalent transforms on residual streams to keep residual addition aligned after

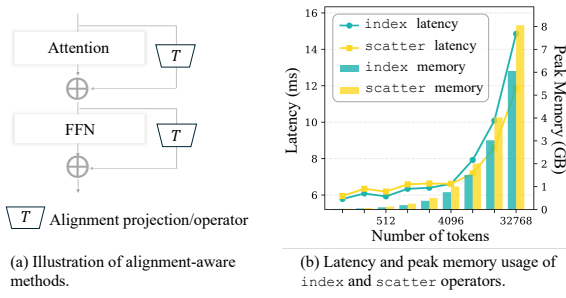


Figure 2: (a) Existing alignment-based methods in LLM hidden-dimension pruning, using extra operators or learned projections to map feature spaces. (b) Latency (ms) and memory usage (GB) of `index_add_` and `scatter_add_` operators in MoE expert aggregation under different sequence lengths.

hidden-dimension reduction. (3) *Index-based aggregation operators*: Gao et al. (2024) decouple structural constraints by computing on selected hidden dimensions and mapping outputs back to the full residual space via indexing operators, enabling layer-specific hidden pruning.

Most prior studies focus on dense LLMs, leaving hidden-dimension pruning for MoE models underexplored. Since experts have different redundancy patterns, shared hidden indices can hurt accuracy, while alignment projections introduce extra overhead. To the best of our knowledge, we are the first to systematically study MoE hidden-dimension pruning, enabling expert-specific hidden indices without additional alignment projections.

3 Pre-analysis Challenges in Hidden Pruning

As noted in Section 1, as the ratio $d_{\text{ffn}}/d_{\text{model}}$ shrinks, pruning only the FFN intermediate dimension yields progressively smaller gains. This motivates us to consider exploring other redundancies. However, structured pruning of the hidden dimension remains under-explored. It is difficult to prune hidden dimension even in dense LLMs, and the sparse characteristic of MoEs will enlarge the difficulty. We summarize three key challenges below.

3.1 Residual Misalignment

Transformers is fundamentally constructed based on residual connections. Therefore, hidden dimension is shared by almost all submodules, including attention, FFN, normalization, as well as the residuals alongside these modules. As a result, pruning hidden channels may cause shape or semantic misalignment between the module output and the resid-

ual path. One solution is to enforce an identical set of kept hidden indices as a global pruning strategy, as discussed in Section 2. However, this strategy is often overly restrictive and leads to larger accuracy drops, particularly in MoEs where experts are activated with highly non-uniform frequency and may favor different channel subsets.

An alternative is to introduce an additional alignment operator, such as a learned projection or a gather-based channel selection, as illustrated in Figure 2(a). However, these methods brings extra runtime overhead. In MoE layers, expert outputs are typically aggregated via a 1D `index_add_` over tokens, whereas hidden-channel pruning introduces an additional channel index, forcing a 2D restore operation (e.g. `scatter_add_`) to preserve residual compatibility. We benchmark these operators in Figure 2(b) under a top-8 routing policy. For sequence below 4096 tokens, 2D scatter not only incurs higher latency, but also requires approximately 33% more peak memory. While under long-context settings (≥ 4096 tokens), scatter yields faster speed, but it demands a substantial larger memory, exceeding 2GB for a single MoE layer at 32k context window.

Challenge 1: Hidden-dimension pruning in MoEs must preserve residual compatibility while avoiding expensive alignment operators.

3.2 Intermediate vs. Hidden Budget Ratio

Given a target overall pruning ratio, one question naturally arises: *how should we allocate the budget between the intermediate and hidden dimensions?* We conduct an observation in Figure 3 on Qwen3-30B-A3B, where we evaluate both fixed pruning ratios (orange markers) and randomly sampled pruning ratios (blue markers). For each point, we prune the channels with the lowest scores in both intermediate and hidden dimensions (scores are measured following (Sun et al., 2023)), and collect the negative log-likelihood (NLL) of the pruned model on the same data batch.

Results shows that the prune ratio allocation between intermediate and hidden dimensions (p_I and p_H) is non-trivial. It suggests several inspirations: (1) Even under the same budget constraint (the yellow line on the loss surface), the resulting model can exhibit drastically different loss behavior. (2) Allocating the entire budget to either dimension (all-hidden or all-intermediate) is suboptimal, and hidden channels may structurally more sensitive

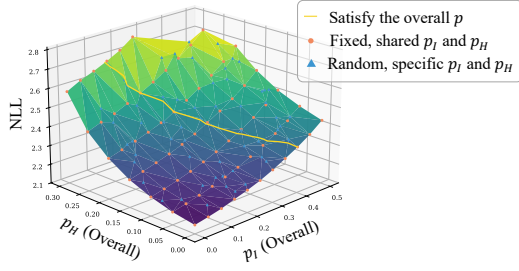


Figure 3: Loss surface for intermediate-hidden budget allocation. p_I and p_H denote pruning ratios of intermediate and hidden channels. The yellow curve indicates allocations satisfying the target compression budget, or orange dots are fixed preset ratios shared across experts, and blue dots are random generated expert-wise prune ratios (reported by averaged ratios).

since they defines the representation basis of the model. (3) Some randomly generated ratios outperform adjacent fixed ratios, indicating that the optimal expert shape can be heterogeneous rather than uniform across experts.

However, searching for the best p_I and p_H for each expert can substantially enlarge the search space. This challenge is more pronounced in MoEs than in dense models because experts are activated unevenly and exhibit heterogeneous redundancy patterns. Therefore, an efficient searching algorithm is required to obtain near-optimal expert-wise prune budget allocations.

Challenge 2: Different intermediate vs. hidden budget ration yields drastically different loss, while expert heterogeneity induces a combinatorially large search space.

3.3 Distillation Difficulty

Hidden-dimension pruning makes representation distillation substantially harder because teacher and student no longer share the same feature shape or basis. Since most intermediate-layer distillation objectives are defined on hidden states, prior practices often utilize logits-only distillation (Li et al., 2025c; Sreenivas et al., 2024; Wan et al., 2024; Wang et al., 2025) or introduces extra alignment modules, e.g., learned projections (Passban et al., 2021; Ashkboos et al., 2024), which increases training cost. Recent work (Suzuki et al., 2025) further suggests that even using LoRA adapters to mitigate the misalignment before distillation, intermediate-layer distillation may still provide limited gains over vanilla logits distillation.

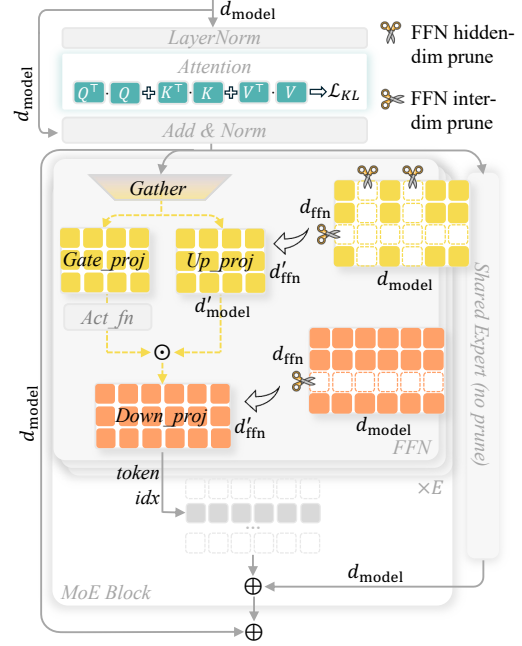


Figure 4: Overview of our joint structured pruning framework within an MoE Transformer block. We prune expert FFN intermediate and hidden dimensions, while distilling attention matrices using KL divergence for stable optimization.

Challenge 3: Hidden pruning makes representation distillation difficult because the teacher and student no longer share aligned hidden shapes and semantics.

4 Joint Prune-and-Distill Framework

To address the challenges of hidden-dimension redundancy and the teacher-student representation basis misalignment, we present an MoE-tailored prune-and-distill framework. In the following, we introduce the overall pruning framework in Section 4.1 (see Figure 4), the Budget-Constrained Coordinate Descent (BCD) in Section 4.2 (illustrated in Figure 5), and the Attention Relation Transfer Distillation (RTD) in Section 4.3.

4.1 Overall Pruning Framework

As shown in Figure 4, we propose a joint structural pruning framework for MoEs that prunes expert FFNs on both the intermediate dimension d_{ffn} and hidden dimension d_{model} , while preserving residual compatibility and inference efficiency without introducing exhausted operators.

Specifically, to enable expert-specific hidden patterns, *gate* and *up* projections in one expert share a lightweight 1D gather to select the kept hidden channels as input, so that, linear layers can be im-

plemented as smaller dense layers without custom sparse kernels. Meanwhile, for *down* projection, we solely prune its intermediate dimension while keep all hidden channels, since pruning its hidden dimension would require 2D indexing during expert output aggregation (e.g., scatter operators) to maintain semantic consistency, incurring higher overhead than standard 1D indexing.

4.2 Budget-Constrained Coordinate Descent

To allocate sparsity between intermediate and hidden pruning under a fixed compression target, we propose Budget-Constrained Coordinate Descent (BCD), as illustrated in Figure 5. Starting from an intermediate-only allocation, BCD iteratively probes two budget-preserving actions for selected experts: pruning more d_{ffn} or pruning more d_{model} , and applies the action with lower validation loss. Because the initial allocation already prunes only d_{ffn} , BCD shifts budget to d_{model} only when a hidden-pruning action is confirmed to lower the loss; in high-expansion-ratio MoEs, where hidden pruning offers little benefit, the search therefore naturally stays close to this intermediate-only initialization.

Why Coordinate Descent? We adopt coordinate descent for three reasons. (i) *Budget-preserving constraint*: the two-dimensional allocation problem is a constrained combinatorial search under a fixed global compression target, and each coordinate action is designed to be budget-neutral, so every update automatically satisfies the target ratio throughout the search. (ii) *Low evaluation cost*: each probe requires only an expert-local KL divergence computed on tokens routed to the activated expert, rather than a full-model forward pass, which keeps 100 search rounds per layer inexpensive—approximately 10.6 hours for DeepSeek-V2-Lite (1×NVIDIA H20) and 14.4 hours for Qwen3-30B-A3B (2×NVIDIA H20), substantially less than the subsequent fine-tuning stage and incurred only once per target compression ratio. (iii) *Expert heterogeneity*: redundancy patterns differ substantially across experts, and gradient-free, per-expert updates are more robust than a jointly optimized allocation, avoiding coupling of redundancy signals across experts with different sensitivities.

Initialization: Given the expert-wise pruning ratio $p_{\ell,e}$ from router weights and layerwise loss, we initialize each expert with kept channel counts $(k_{I,e}, k_{H,e})$ by assigning the pruning budget to d_{ffn}

while preserving all hidden channels. Each expert also maintains a step size Δ_e and a success counter success_e .

Coordinate Descent: At each round, we sample a calibration batch and compute the current expert-wise KL divergence loss $\mathcal{L}_0^e$ on tokens routed to expert e . As shown in Figure 5, we then select a subset of experts S and probe two actions. Next, we probe two budget-constrained coordinate actions on S : For Action A_1 , we adopt more pruning on intermediate dimension by reducing $k_{I,e}$ by current step size Δ_e for each $e \in S$. To satisfy overall parameter budget, we compensate on hidden dimension by increasing $k_{H,e}$ by $\frac{3d_{\text{model}}}{2d_{\text{ffn}}} \Delta_e$, producing a tentative state $(k_{I,e}^{A_1}, k_{H,e}^{A_1})$ for each selected experts. Action A_2 mirrors this procedure, we prune more hidden channels by decreasing $k_{H,e}$ by Δ_e for all $e \in S$. For both actions, we evaluate the corresponding expert-wise losses $\mathcal{L}_{A_1}^e$ and $\mathcal{L}_{A_2}^e$. We apply the action with lower loss if it improves over $\mathcal{L}_0^e$; otherwise, the state remains unchanged.

Expert Subset-size Decay: BCD starts with a large expert subset size m_{init} and gradually decays it to m_{min} . Using a large m early helps redistribute budgets quickly across many experts. As the search proceeds, we shrink m to reduce expert coupling in loss evaluation. This decay enables fast early budget redistribution while reducing expert coupling later, making the comparison between A_1 and A_2 more identifiable. Details about expert score penalization and subset size scheduling are provided in Appendix Section C.3.

Step-size Adaptation: We adapt the expert-wise step size Δ_e using a success-based rule (Figure 5, right). $\mathcal{L}_0^e$ is the loss at the start of the current round for expert e . If the chosen action improves the loss, i.e., $\min(\mathcal{L}_{A_1}^e, \mathcal{L}_{A_2}^e) < \mathcal{L}_0^e$, we increment $\text{success}_e \leftarrow \text{success}_e + 1$ for $e \in S$; otherwise we reset $\text{success}_e \leftarrow 0$. Two consecutive successes lead to a more aggressive exploration by doubling the step size $\Delta_e \leftarrow 2\Delta_e$, and resetting success_e to zero, whereas a failure halves the step size $\Delta_e \leftarrow \Delta_e/2$. This adaptive rule accelerates the progress, and reduces oscillations by shrinking the step size when approaching the local optimum.

We repeat this procedure layer by layer, producing the final expert-wise kept channel counts (k_I, k_H) for the intermediate and hidden dimensions, respectively.

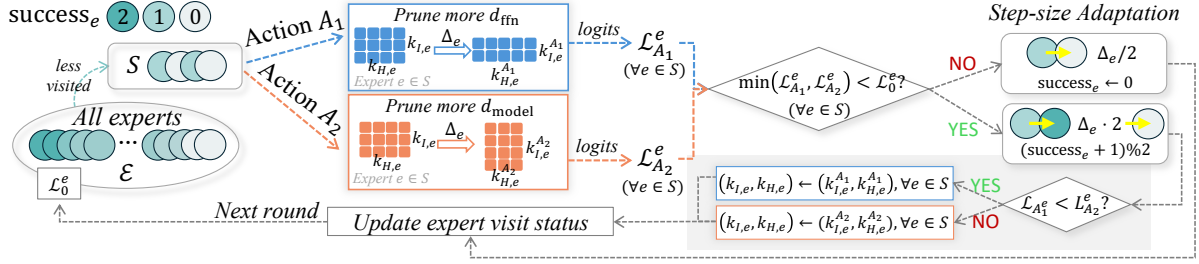


Figure 5: Overview of the Budget-Constrained Coordinate Descent (BCD) algorithm. It iteratively optimizes expert-wise budget allocation between intermediate and hidden dimensions through symmetric actions (A_1, A_2) and evolution-inspired step-size adaptation.

4.3 Attention Relation Transfer Distillation

Pruning the hidden dimension makes layerwise distillation nontrivial because it alters the underlying feature basis of hidden representations, leading to representation misalignment between teacher and student. Consequently, naive block-output matching may provide unstable supervision. Moreover, prior work reports that directly distilling layer outputs in LLMs can be overly strict, as it implicitly enforces strong alignment constraints (Wang et al., 2025; Zeng et al., 2025; Wan et al., 2024).

Distillation Objective: To improve the optimization stability and convergence of compressed MoEs, we adopt attention relation transfer distillation inspired by previous studies (Qin et al., 2022; Wu et al., 2025). Instead of matching layer outputs, we distill the relations from query, key, and value matrices in attention layers.

Let $\mathbf{A}_{\ell,a}^T$ and $\mathbf{A}_{\ell,a}^S$ denote the per-head representations of $\mathbf{A} \in \{\mathbf{Q}, \mathbf{K}, \mathbf{V}\}$ at layer ℓ and head a for the teacher and student, respectively. We first apply ℓ_2 normalization $\|\cdot\|_2$ on the last dimension, and then form a token-to-token relation matrix by multi-head scaled dot-product followed by Softmax and head dimension d_{head} :

$$\mathbf{R}_{\mathbf{A},\ell,a,t}^* = \text{Softmax} \left(\frac{\|\mathbf{A}_{\ell,a,t}^*\|_2 \cdot \|\mathbf{A}_{\ell,a,t}^*\|_2^\top}{\sqrt{d_{\text{head}}}} \right), \quad (1)$$

where $* \in \{T, S\}$, meaning that we compute relations in the same way for both the teacher and student. The distillation objective then minimizes the teacher-to-student KL divergence, averaged over query, key, value, the selected layers Φ , and heads:

$$\mathcal{L}_{\text{RTD}} = \frac{1}{N_{\text{head}} |\Phi|} \sum_{\ell \in \Phi} \sum_{\mathbf{A} \in \{\mathbf{Q}, \mathbf{K}, \mathbf{V}\}} \sum_{a=1}^{N_{\text{head}}} D_{\text{KL}}(\mathbf{R}_{\mathbf{A},\ell,a}^T \| \mathbf{R}_{\mathbf{A},\ell,a}^S). \quad (2)$$

Here Φ is the layers selected for distillation. Following studies (Wang et al., 2020; Wu et al., 2025; He et al., 2024), we conduct distillation from middle to later stages, where attention structures are more stable and more directly tied to high-level reasoning behavior.

We combine RTD and task loss, and decay the distillation weight γ over training so that optimization gradually shifts from distillation to solely task supervision. Ablations on enabling logits distillation and the choice of distilled layers are reported in Section 5.3.

5 Experiments

In this section, we evaluate our joint prune-and-distill framework on representative MoE models.

5.1 Settings

Models and Datasets. We evaluate on representative open-source MoEs, DEEPSEEK-V2-LITE and QWEN3-30B-A3B, which have small FFN expansion ratios, i.e., $d_{\text{ffn}}/d_{\text{model}}=0.69$ and 0.38 , respectively. This makes them suitable for evaluating joint intermediate and hidden pruning beyond intermediate-only compression. We calibrate BCD with task-related data, including C4 (Raffel et al., 2019), GSM8K (Cobbe et al., 2021), and OpenCode (Ahmad et al., 2025), and perform all post-pruning fine-tuning and distillation on Alpaca (Taori et al., 2023), without any additional math or code training data.

Comparisons. We compare against Wanda (Sun et al., 2023), SparseGPT (Frantar and Alistarh, 2023), LLM-Shearing (Xia et al., 2023), and SliceGPT (Ashkboos et al., 2024) at overall pruning ratios of 25% and 50%. We evaluate all baselines without post-pruning fine-tuning and likewise use the original unpruned checkpoint without ad-

Method	$p\%$	ARC-C	ARC-E	Hella	PiQA	BoolQ	Wino	MMLU	GSM8K	HumanEval	Avg
Full model	0	46.93	78.37	77.98	80.52	79.82	71.35	45.58	30.94	32.30	68.60
LLM-Shearing	25	40.18	71.17	63.24	78.92	75.68	68.65	53.03	27.47	27.72	51.78
Wanda	25	46.84	76.64	58.44	79.65	79.39	71.51	53.84	13.12	35.24	57.19
SparseGPT	25	47.01	76.73	58.50	79.24	80.06	71.90	51.76	19.24	34.80	57.69
SliceGPT	25	46.50	74.03	54.96	78.89	73.12	67.32	43.88	19.33	33.91	54.66
Ours	25	46.13	75.14	64.68	79.28	73.15	69.37	52.05	34.65	37.62	59.12
LLM-Shearing	50	28.83	59.28	53.15	69.37	63.96	61.98	34.33	10.99	10.89	43.64
Wanda	50	31.40	50.84	35.41	64.15	63.18	59.27	40.81	1.14	2.44	38.96
SparseGPT	50	33.69	54.66	55.43	68.13	66.30	61.85	42.88	10.15	31.71	47.20
SliceGPT	50	27.05	43.76	35.16	63.92	59.49	61.24	29.18	10.52	11.43	37.97
Ours	50	34.12	60.36	58.74	74.10	66.49	62.88	41.17	28.15	32.67	50.96

Table 1: Comparison on DeepSeek-V2-Lite under diverse tasks and benchmarks. $p\%$ is the overall prune ratio.

Method	$p\%$	ARC-C	ARC-E	Hella	PiQA	BoolQ	Wino	MMLU	GSM8K	MATH500	Avg
Full model	0	52.70	79.30	78.50	79.60	88.70	72.85	77.80	89.99	93.90	79.26
LLM-Shearing	25	46.13	69.91	58.38	78.74	83.60	67.21	71.46	43.21	77.55	66.24
Wanda	25	26.45	33.71	25.43	52.12	38.50	51.14	22.95	0	0.20	27.83
Ours	25	58.02	80.9	67.21	80.1	81.98	72.64	74.56	96.30	93.88	78.40
LLM-Shearing	50	30.27	49.55	46.31	65.95	63.24	52.43	39.12	29.63	59.18	48.41
Wanda	50	24.57	27.23	26.48	53.75	40.55	50.12	22.93	0	0	27.29
Ours	50	42.06	68.20	63.60	76.77	78.92	69.60	54.91	92.08	83.67	69.98

Table 2: Comparison on Qwen3-30B-A3B-Thinking. $p\%$ is the overall prune ratio.

ditional fine-tuning as the reference (Yang et al., 2024b; Xie et al., 2024).

Benchmarks. We use the EleutherAI LM Evaluation Harness², covering general knowledge and reasoning tasks (math and code). We report zero-shot performance on ARC-Challenge (Clark et al., 2018), ARC-Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2019), BoolQ (Clark et al., 2019), WinoGrande (Sakaguchi et al., undefined) and MATH500 (Lightman et al., 2023), 5-shot results on MMLU (Hendrycks et al., 2020), 8-shot results on GSM8K (Cobbe et al., 2021) and HumanEval (Chen et al., 2021).

Fine-tuning. We apply parameter-efficient fine-tuning with LoRA on routers and FFNs. The LoRA for routers uses a smaller rank and a reduced learning rate scaled by 0.01. Router auxiliary loss is added with coefficient 0.001 for MoE routing stability. For distillation, we initialize γ to 10 and linearly decay it to 0 over training, so that RTD provides guidance at early stages and gradually phased out, with the final stage optimized solely by $\mathcal{L}_{\text{task}}$. Distillation is performed on Alpaca (Taori et al.,

2023) for 2 epochs; no additional math or code training data are used.

5.2 Overall Results

Tables 1 and 2 compare 25% and 50% pruning on DeepSeek-V2-Lite and Qwen3-30B-A3B-Thinking. Results indicate that enforcing a globally shared hidden index can cause substantial accuracy degradation, particularly on reasoning tasks where experts exhibit highly heterogeneous redundancy patterns. While unstructural pruning with Wanda and SparseGPT is competitive at 25% pruning on DeepSeek-V2-Lite, performance drops sharply at 50% pruning because a uniform pruning ratio across experts is suboptimal for MoEs. In contrast, our method preserves performance and remains on par with the original full model in reasoning tasks even under substantial compression, suggesting that the original MoE carries redundant capacity relative to these tasks. This highlights the importance of allowing each expert choose its own hidden capacity and representation basis.

Inference Efficiency. Table 3 reports GPU memory, on-disk storage, and decoding throughput for the original and 50%-pruned models. Memory is one of the main deployment bottlenecks for large MoEs, and 50% pruning reduces peak memory by

²<https://github.com/EleutherAI/lm-evaluation-harness>

Model	$p\%$	Memory (GB)	Storage (GB)	Throughput (tok/s)
Qwen3-30B-A3B	0	57.24	61.06	29.39
	50	32.02	32.07	29.16
DeepSeek-V2-Lite	0	31.19	31.41	75.83
	50	20.16	16.57	75.90

Table 3: Inference efficiency of the original ($p\% = 0$) and 50%-pruned models. Throughputs are measured on a single NVIDIA H20 GPU.

Method	ARC-C	PIQA	GSM8K	Avg
Full model	58.36	80.90	89.99	76.42
$p_I = 0.5$ Only	35.84	66.00	80.25	60.90
$p_H = 0.5$ Only	35.07	59.19	61.67	51.98
Preset (Avg.)	30.49 _{3.6}	66.17 _{2.8}	60.83 _{12.5}	52.50
Random (Avg.)	31.57 _{0.0}	68.34 _{0.3}	66.64 _{5.0}	55.52
Mixed Top-k	38.74	67.52	52.86	53.04
BCD (no FT)	37.81	70.88	86.70	65.13
BCD + FT	39.33	75.03	89.41	67.92
BCD + FT + RTD (Ours)	42.06	76.77	92.08	70.30

Table 4: Ablation study of pruning allocation strategies on Qwen3-30B-A3B under a 50% global pruning ratio. ‘Preset’ uses a fixed p_H shared across all layers and experts, while ‘Random’ samples expert-specific ratios. ‘Mixed Top-k’ ranks intermediate and hidden channel scores jointly in a single list. ‘BCD (no FT)’ applies BCD without any post-pruning fine-tuning or distillation, isolating the contribution of the allocation strategy itself from subsequent recovery.

44% on Qwen3-30B-A3B and 35% on DeepSeek-V2-Lite, enabling deployment on fewer or smaller GPUs. Throughput is preserved (within $\sim 1\%$), because the 1D gather that implements hidden pruning is applied once when the pruned model is constructed (Section 4.1); decoding then runs standard dense matrix multiplications over the smaller *gate/up* projections, without any runtime gather/scatter operator or non-contiguous memory access.

5.3 Ablation Study

Effect of Budget-Constrained Coordinate Descent. Table 4 compares methods for hidden and intermediate dimensions pruning under the same 50% prune ratio. *Preset* fixes a global p_H that shared across experts. *Random* samples expert-specific $p_{H,e}$ from the uniform distribution. Both compute the corresponding $p_{I,e}$ to meet the overall pruning ratio. ‘Mixed Top-k’ jointly ranks intermediate and hidden channels together.

Overall, the results suggest that (i) pruning only one dimension is insufficient, and hidden-only pruning can be more sensitive, (ii) pruning requires expert-specific intermediate vs. hidden allocation, uniform pruning is suboptimal, and (iii) a good al-

KD Loss	$ \Phi $	ARC-C	PIQA	Wino	Avg.
Full model	–	58.36	80.90	72.85	70.70
No distill	0	39.22	75.35	67.65	60.74
$\mathcal{L}_{\text{logits}}$	0	40.00	75.46	67.75	61.07
$\mathcal{L}_{\text{RTD}}$	1	39.02	75.32	67.25	60.53
$\mathcal{L}_{\text{RTD}}$	2	40.02	75.68	68.60	61.43
$\mathcal{L}_{\text{RTD}}$	4	40.05	75.73	68.39	61.39
$\mathcal{L}_{\text{RTD}}$	7	42.06	76.77	69.19	62.67
$\mathcal{L}_{\text{RTD}}, \mathcal{L}_{\text{logits}}$	7	39.85	75.52	68.40	61.26
$\mathcal{L}_{\text{RTD}}$	12	40.10	75.52	67.85	61.16

Table 5: Comparison of distillation settings. $|\Phi|$ denotes the number of layers that are selected for distillation. $\mathcal{L}_{\text{logits}}$ is logits distillation, and $\mathcal{L}_{\text{RTD}}$ is relation transfer distillation.

location strategy should be data-driven, while naive preset or random rules can drop heavily. We additionally report ‘BCD (no FT)’, which applies BCD without any post-pruning fine-tuning or distillation. Since Wanda, SparseGPT, LLM-Shearing, and SliceGPT are likewise evaluated without post-pruning fine-tuning (Section 5.1), this row isolates the contribution of the BCD allocation strategy under a comparable, fine-tuning-free setting, before LoRA fine-tuning and RTD are added.

Effect of Relation Transfer Distillation. To show the effectiveness of RTD, we compare it with logits-only distillation and no distillation fine-tuning. Table 5 shows that RTD is consistently more effective than logits-only, which brings only marginal improvement over no distillation on the common knowledge tasks. On GSM8K, the selected seven-layer RTD configuration achieves 92.08, compared with 90.51 for logits-only distillation and 89.41 without distillation, showing that RTD also benefits mathematical reasoning. As for the layer selection, a moderate number of layers selected from middle to later stages performs best ($|\Phi| = 7$), which achieves the highest accuracy, outperforming fewer layers ($|\Phi| = 2, 4$) and more layers ($|\Phi| = 12$). The index of selected layers is provided in Appendix Section D.3. Combining logits distillation with RTD does not further improve results, therefore, we only use RTD in our main results as default.

Effect of Calibration Corpus. BCD calibration only computes the expert-wise KL divergence between pruned and original models on calibration tokens (Section 4.2), with no labels or weight updates, so it cannot teach the model to solve downstream tasks (Appendix Section C.2). Table 6 ablates the

Calibration Corpus	ARC-c	GSM8K	HumanEval
WikiText2	36.11	27.15	2.44
C4	35.94	9.18	3.05
Pile	39.92	42.51	5.17
GSM8K (train)	37.13	61.08	15.85
OpenCodeReasoning	33.95	54.49	29.27
RedPajama	36.51	23.35	3.05

Table 6: Effect of BCD calibration corpus on Qwen1.5-MoE-A2.7B under 25% pruning. Each row calibrates BCD using only the listed corpus.

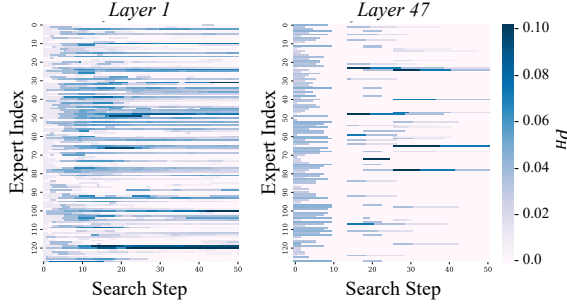


Figure 6: Search progress on Qwen3-30B-A3B. Heatmaps track p_H over steps across experts, revealing layer-dependent hidden-pruning preferences. More layer-wise visualizations are provided in Appendix Section D.2 (Figures 8 and 9).

calibration corpus on Qwen1.5-MoE-A2.7B at 25% pruning: corpora close to a target task improve that task’s score most, e.g., GSM8K-train gives the best GSM8K result and OpenCodeReasoning gives the best HumanEval result, while generic corpora like C4 and WikiText2 lag far behind on GSM8K. This gap reflects a better structural match between the retained channels and the target task rather than answer leakage. The substantially lower GSM8K score obtained with C4 calibration further supports this interpretation.

5.4 Visualizations and Analysis

Tracing of coordinate descent process. To investigate the progress of BCD, we plot the evolution of pruning ratios p_H throughout the search process in Figure 6. It shows that the BCD algorithm automatically balances the pruning allocation between the intermediate and hidden dimensions. We highlight two findings: (1) *Layer-wise dynamics*: p_H is layer-dependent, with shallow layers pruning hidden dimension aggressively, while deeper layers are more sensitive and retain more hidden channels. (2) *Expert heterogeneity*: p_H varies across experts, confirming that experts within the same layer possess distinct redundancy patterns, validating the necessity of the expert-specific pruning allocation.

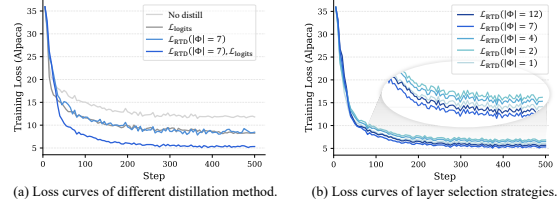


Figure 7: (a) Training loss under different distillation objectives, showing RTD improves convergence and final loss. (b) Layer selections, where a moderate number of layers chosen from the middle to later stages outperforms too few or too many layers.

Loss curves during training. Figure 7 shows the loss curves throughout the fine-tuning. RTD accelerates convergence and reduces the final loss compared to no distillation or logits-only distillation. In addition, the layer selection study in Figure 7(b) is consistent with the final performance. $|\Phi| = 7$ leads to the lowest loss in the later training stage, while selecting too few or too many layers is less effective.

6 Conclusion

In this work, we present an MoE-tailored prune-and-distill framework that slims expert FFNs along the intermediate and hidden dimensions. We highlight that the FFN expansion ratio is continuously shrinking in recent MoEs, which makes intermediate-only pruning less effective. To our knowledge, we are the first to design hidden pruning in MoE architectures. Experiments and ablations show consistent improvements by jointly pruning multiple dimensions, establishing hidden-dimension pruning as a viable new direction for efficient MoE deployment.

Limitations

Our evaluation covers only two text MoEs with low FFN expansion ratios; future work should test high-expansion models, routing designs, and model scales. BCD is sensitive to the calibration distribution, motivating general-purpose or adaptive calibration for unseen domains. RTD uses a fixed seven-layer selection and linear decay schedule, leaving automatic layer and schedule selection for future work. Finally, expert-specific pruning creates heterogeneous weight shapes, requiring specialized kernels and scheduling for efficient multi-GPU serving.

Acknowledgments

This research is supported by the RIE2025 Industry Alignment Fund – Industry Collaboration Projects (IAF-ICP) (Award I2301E0026), administered by A*STAR, as well as supported by Alibaba Group and NTU Singapore through Alibaba-NTU Global e-Sustainability CorpLab (ANGEL).

References

- Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, and 1 others. 2024. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*.
- Wasi Uddin Ahmad, Sean Narenthiran, Somshubra Majumdar, Aleksander Ficek, Siddhartha Jain, Jocelyn Huang, Vahid Noroozi, and Boris Ginsburg. 2025. Opencodereasoning: Advancing data distillation for competitive coding. *arXiv preprint arXiv:2504.01943*.
- Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024. [Slicept: Compress large language models by deleting rows and columns](#). In *International Conference on Learning Representations*.
- Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.
- Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language. In *AAAI Conference on Artificial Intelligence*.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. [Evaluating large language models trained on code](#).
- Tianyu Chen, Shaohan Huang, Yuan Xie, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. 2022. [Task-specific expert pruning for sparse mixture-of-experts](#). In *arXiv.org*.
- Christopher Clark, Kenton Lee, Ming-Wei Chang, T. Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *North American Chapter of the Association for Computational Linguistics*.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. In *arXiv.org*.
- K. Cobbe, Vineet Kosaraju, Mo Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. In *arXiv.org*.
- Marta R Costa-Jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, and 1 others. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.
- Damai Dai, Chengqi Deng, Chenggang Zhao, R. Xu, Huazuo Gao, Deli Chen, Jiasli Li, Wangding Zeng, Xingkai Yu, Yu Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and W. Liang. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. pages 1280–1297.
- DeepSeek-AI. 2024. [Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model](#). Preprint, arXiv:2405.04434.
- Yifu Ding, Jiacheng Wang, Ge Yang, Yongcheng Jing, Jinyang Guo, Xianglong Liu, and Dacheng Tao. 2026. Attribution-guided and coverage-maximized pruning for structural moe compression. *arXiv preprint arXiv:2606.18304*. ICML 2026 (Spotlight).
- Zican Dong, Han Peng, Peiyu Liu, Wayne Xin Zhao, Dong Wu, Feng Xiao, and Zhifeng Wang. 2025. Domain-specific pruning of large mixture-of-experts models with few-shot demonstrations.
- Elias Frantar and Dan Alistarh. 2023. [Sparsegpt: Massive language models can be accurately pruned in one-shot](#). In *International Conference on Machine Learning*.
- Shangqian Gao, Chi-Heng Lin, Ting Hua, Tang Zheng, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. 2024. [Disp-llm: Dimension-independent structural pruning for large language models](#). In *Neural Information Processing Systems*.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- En hao Liu, Junyi Zhu, Zinan Lin, Xuefei Ning, Matthew B. Blaschko, Shengen Yan, Guohao Dai, Huazhong Yang, and Yu Wang. 2024. [Efficient expert pruning for sparse mixture-of-experts language models: Enhancing performance and reducing inference costs](#). In *arXiv.org*.

- Jinghan He, Haiyun Guo, Kuan Zhu, Zihan Zhao, Ming Tang, and Jinqiao Wang. 2024. Seekr: Selective attention-guided knowledge retention for continual learning of large language models. *arXiv preprint arXiv:2411.06171*.
- Shwai He, Daize Dong, Liang Ding, and Ang Li. 2025. [Towards efficient mixture of experts: A holistic study of compression techniques](#). *Transactions on Machine Learning Research*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, D. Song, and J. Steinhardt. 2020. Measuring massive multitask language understanding. *ArXiv*, abs/2009.03300.
- Yuxuan Hu, Jing Zhang, Zhe Zhao, Chen Zhao, Xiaodong Chen, Cuiping Li, and Hong Chen. 2024. Sp3: Enhancing structured pruning via pca projection. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3150–3170.
- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L  lio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timoth  e Lacroix, and William El Sayed. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.
- Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, and 1 others. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.
- Jaeseong Lee, Seung won Hwang, Aurick Qiao, Daniel Campos, Zhewei Yao, and Yuxiong He. 2024. Stun: Structured-then-unstructured pruning for scalable moe pruning. *ArXiv*, abs/2409.06211.
- Aonian Li, Bangwei Gong, Bo Yang, Boji Shan, Chang Liu, Cheng Zhu, Chunhao Zhang, Congchao Guo, Da Chen, Dong Li, and 1 others. 2025a. Minimax-01: Scaling foundation models with lightning attention. *arXiv preprint arXiv:2501.08313*.
- Lujun Li, Qiyuan Zhu, Jiacheng Wang, Wei Li, Hao Gu, Sirui Han, and Yike Guo. 2025b. Sub-moe: Efficient mixture-of-expert llms compression via subspace expert merging. In *arXiv.org*.
- Zichong Li, Chen Liang, Zixuan Zhang, Ilgee Hong, Young Jin Kim, Weizhu Chen, and Tuo Zhao. 2025c. Slimmoe: Structured compression of large moe models via expert slimming and distillation. *arXiv preprint arXiv:2506.18349*.
- Xun Liang, Hanyu Wang, Huayi Lai, Simin Niu, Shichao Song, Jiawei Yang, Jihao Zhao, Feiyu Xiong, Bo Tang, and Zhiyu Li. 2025. [Seap: Training-free sparse expert activation pruning unlock the brain-power of large language models](#). In *arXiv.org*.
- Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, and 1 others. 2024. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*.
- Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.
- Liyuan Liu, Young Jin Kim, Shuohang Wang, Chen Liang, Yelong Shen, Hao Cheng, Xiaodong Liu, Masahiro Tanaka, Xiaoxia Wu, Wenxiang Hu, and 1 others. 2024b. Grin: Gradient-informed moe. *arXiv preprint arXiv:2409.12136*.
- Yijiang Liu, Huanrui Yang, Youxin Chen, Rongyu Zhang, Miao Wang, Yuan Du, and Li Du. 2025. [Pat: Pruning-aware tuning for large language models](#). In *AAAI Conference on Artificial Intelligence*.
- Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. 2024. [Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models](#). In *Annual Meeting of the Association for Computational Linguistics*.
- Ruijie Miao, Yilun Yao, Zihan Wang, Zhiming Wang, Bairen Yi, LingJun Liu, Yikai Zhao, and Tong Yang. 2025. Mergemoe: Efficient compression of moe models via expert output merging. *arXiv preprint arXiv:2510.14436*.
- Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, and 1 others. 2024. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*.
- Alexandre Muzio, Alex Sun, and Churan He. 2024. [Seer-moe: Sparse expert efficiency through regularization for mixture-of-experts](#). In *arXiv.org*.
- Peyman Passban, Yimeng Wu, Mehdi Rezagholizadeh, and Qun Liu. 2021. Alp-kd: Attention-based layer projection for knowledge distillation. In *Proceedings of the AAAI Conference on artificial intelligence*, volume 35, pages 13657–13665.
- Haotong Qin, Yifu Ding, Mingyuan Zhang, Qinghua Yan, Aishan Liu, Qingqing Dang, Ziwei Liu, and Xianglong Liu. 2022. Bibert: Accurate fully binarized bert. In *International Conference on Learning Representations (ICLR)*.

- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.
- Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. undefined. Winogrande: An adversarial winograd schema challenge at scale. *Proceedings of the AAAI Conference on Artificial Intelligence*.
- Yikang Shen, Zhen Guo, Tianle Cai, and Zengyi Qin. 2024. Jetmoe: Reaching llama2 performance with 0.1 m dollars. *arXiv preprint arXiv:2404.07413*.
- Sharath Turuvekere Sreenivas, Saurav Muralidharan, Raviraj Joshi, Marcin Chochowski, Ameya Sunil Mahabaleshwarkar, Gerald Shen, Jiaqi Zeng, Zijia Chen, Yoshi Suhara, Shizhe Diao, and 1 others. 2024. Llm pruning and distillation in practice: The minitron approach. *arXiv preprint arXiv:2408.11796*.
- Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.
- Xingwu Sun, Yanfeng Chen, Yiqing Huang, Ruobing Xie, Jiaqi Zhu, Kai Zhang, Shuaipeng Li, Zhen Yang, Jonny Han, Xiaobo Shu, and 1 others. 2024. Hunyuan-large: An open-source moe model with 52 billion activated parameters by tencent. *arXiv preprint arXiv:2411.02265*.
- Takeshi Suzuki, Hiroaki Yamada, and Takenobu Tokunaga. 2025. Aligning sizes of intermediate layers by lora adapter for knowledge distillation. *The Sixth Workshop on Insights from Negative Results in NLP*.
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.
- Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.
- Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, and 1 others. 2024a. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.
- Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 2024b. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.
- Qwen Team. 2024. [Qwen1.5-moe: Matching 7b model performance with 1/3 activated parameters](#).
- Qwen Team. 2025. [Qwen3 technical report](#). *Preprint*, arXiv:2505.09388.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023a. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. 2024. [Knowledge fusion of large language models](#). In *International Conference on Learning Representations*.
- Chengyu Wang, Junbing Yan, Yuanhao Yue, and Jun Huang. 2025. Distilqwen2.5: Industrial practices of training distilled open lightweight language models.
- Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems*, 33:5776–5788.
- Tianwen Wei, Bo Zhu, Liang Zhao, Cheng Cheng, Biye Li, Weiwei Lü, Peng Cheng, Jianhao Zhang, Xiaoyu Zhang, Liang Zeng, and 1 others. 2024. Skywork-moe: A deep dive into training techniques for mixture-of-experts language models. *arXiv preprint arXiv:2406.06563*.
- Xun Wu, Shaohan Huang, Wenhui Wang, Ting Song, Li Dong, Yan Xia, and Furu Wei. 2025. Bitnet distillation. *arXiv preprint arXiv:2510.13998*.
- Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. [Sheared llama: Accelerating language model pre-training via structured pruning](#). In *International Conference on Learning Representations*.
- Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. [Structured pruning learns compact and accurate models](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Yanyue Xie, Zhi Zhang, Ding Zhou, Cong Xie, Ziang Song, Xin Liu, Yanzhi Wang, Xue Lin, and An Xu. 2024. [Moe-pruner: Pruning mixture-of-experts large language model using the hints from its router](#). In *arXiv.org*.

Fuzhao Xue, Zian Zheng, Yao Fu, Jinjie Ni, Zangwei Zheng, Wangchunshu Zhou, and Yang You. 2024. Openmoe: an early effort on open mixture-of-experts language models. In *Proceedings of the 41st International Conference on Machine Learning, ICML’24*. JMLR.org.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Huaran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Cheng Yang, Yang Sui, Jinqi Xiao, Lingyi Huang, Yu Gong, Yuanlin Duan, Wenqi Jia, Miao Yin, Yu Cheng, and Bo Yuan. 2024b. 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*, pages 10456–10466, Miami, Florida, USA. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

Runjia Zeng, J. Liang, Cheng Han, Zhiwen Cao, Jiahao Liu, Xiaojun Quan, Y. Chen, Lifu Huang, Tong Geng, Qifan Wang, and Dongfang Liu. 2025. Probabilistic token alignment for large language model fusion.

Zeliang Zhang, Xiaodong Liu, Hao Cheng, Chenliang Xu, and Jianfeng Gao. 2024. [Diversifying the expert knowledge for task-agnostic pruning in sparse mixture-of-experts](#). In *arXiv.org*.

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. 2024. Llama-moe: Building mixture-of-experts from llama with continual pre-training. *arXiv preprint arXiv:2406.16554*.

Contents of Appendix

A Potential Risks 13

B Trend of Shrinking FFN Expansion Ratio 13

C Detailed Implementations about Our Method 14

C.1 Preserve Efficient Attentions 14

C.2 Initialization of Expert-wise Prune Ratio 14

C.3 Additional Details about Budget-Constrained Coordinate Descent (BCD) Algorithm 15

D Experiment 16

D.1 Detailed Experiment Settings 16

D.2 Additional Visualizations for BCD .. 17

D.3 Layer Selection for Distillation 18

A Potential Risks

This paper presents work whose goal is to advance the field of machine learning. There are many potential societal consequences of our work. But model compression may affect the reliability, robustness, and fairness of the compressed model. In particular, pruning may amplify existing failure modes of the original model, reduce performance on underrepresented data distributions, or cause uneven degradation across tasks and modalities. We recommend careful evaluation before deployment in high-stakes applications.

B Trend of Shrinking FFN Expansion Ratio

Slimming the FFN intermediate dimension has long been a primary research focus for reducing parameters and FLOPs in Transformer blocks. However, this is becoming less effective in recent models, because modern architectures have already shrunk the FFN inner width over the years. As summarized in Table 7, we observe a clear architectural evolution in both dense and MoE models: the FFN expansion ratio $d_{\text{ffn}}/d_{\text{model}}$ is steadily decreasing across model families and generations. This implies that the remaining reducible capacity in the FFN intermediate dimension is progressively limited.

Specifically, refer to Table 7, earlier dense LLMs typically adopt ratios around 2.7 to 3.5 (e.g., LLaMA 2 and Mistral), and early mainstream

Model family	Time	Para.	Experts	#L	#H	d_{model}	d_{ffn}	FFN Ratio	Is MoE
T5 (3B) (Raffel et al., 2020)	2019.10	3B	–	24	32	768–1024	3072–16384	4.00–16.00	✗
NLLB (Costa-Jussà et al., 2022)	2022.07	54B	2/64/0	24	16	1024	8192	8.00	✓
LLaMA-2 (7B/70B) (Touvron et al., 2023b)	2023.07	7B/70B	–	32/80	32/64	4096–8192	11008–28672	2.69–3.50	✗
Mistral (7B) (Jiang et al., 2023)	2023.09	7B	–	32	32	4096	14336	3.50	✗
Qwen2-57B-A14B (Yang et al., 2024a)	2023.05	57.4B	8/64/0	28	28	3584	2560	0.71	✓
Mixtral-8×7B (Jiang et al., 2024)	2023.12	46.7B	2/8/0	32	32	4096	14336	3.50	✓
OpenMoE (Xue et al., 2024)	2023.12	34B	2/16/0	12	12	768	2048	2.67	✓
DeepSeek-MoE-16B (Dai et al., 2024)	2024.01	16.4B	6/64/2	28	16	2048	1408	0.69	✓
Gemma 1 (2B/7B) (Team et al., 2024a)	2024.02	2B/7B	–	18/28	8/16	2048–3072	16384–24576	8.00	✗
Qwen1.5 (7B/32B) (Bai et al., 2023)	2024.02	7B/32	–	32/64	32/40	4096–5120	11008–27392	2.69–5.35	✗
Qwen1.5-MoE-A2.7B (Team, 2024)	2024.02	14.3B	4/60/0	24	16	2048	1408	0.69	✓
JetMoE (Shen et al., 2024)	2024.03	8.52B	2/8/0	24	32	2048	5632	2.75	✓
Jamba (Lieber et al., 2024)	2024.03	51.6B	2/16/0	32	32	4096	14336	3.50	✓
DBRX ³	2024.03	132B	4/16/0	40	48	6144	10752	1.75	✓
Grok-1 ⁴	2024.03	314B	2/8/0	64	48	6144	–	–	✓
Phi-3.5-MoE (Abdin et al., 2024)	2024.08	41.9B	2/16/0	32	32	4096	6400	1.56	✓
GRIN-MoE (Liu et al., 2024b)	2024.09	41.9B	2/16/0	32	32	4096	6400	1.56	✓
Arctic ⁵	2024.04	482B	2/128/0	35	56	7168	4864	0.68	✓
Mixtral-8×22B (Jiang et al., 2024)	2024.04	141B	2/8/0	56	48	6144	16384	2.67	✓
DeepSeek-V2 (DeepSeek-AI, 2024)	2024.04	236B	6/160/2	60	128	5120	1536	0.30	✓
DeepSeek-V2-Lite (DeepSeek-AI, 2024)	2024.05	16B	6/64/2	27	16	2048	1408	0.69	✓
Skywork-MoE (Wei et al., 2024)	2024.05	13B	2/16/0	52	36	4608	12288	2.67	✓
Yuan2 ⁶	2024.05	40B	2/32/0	24	16	2048	8192	4.00	✓
LLaMA-MoE (Zhu et al., 2024)	2024.06	6.7B	2/8/0	32	32	4096	11008	2.69	✓
OLMoE (Muennighoff et al., 2024)	2024.07	6.92B	8/64/0	16	16	2048	1024	0.50	✓
LLaMA 3 (8B/70B/405B) (Grattafiori et al., 2024)	2024.07	8B/70B/405B	–	32/80/126	32/64/128	8192–16384	28672–53248	3.25–3.50	✗
Gemma 2 (2B/9B) (Team et al., 2024b)	2024.07	2B/9B	–	26/42	8/16	2304–3584	9216–14336	4.00	✗
Qwen2.5 (7B/72B) (Yang et al., 2024a)	2024.09	7.6B/72B	–	28/80	28/64	896–8192	4864–29658	3.60–5.43	✗
Hunyuan-Large (Sun et al., 2024)	2024.11	389B	1/16/1	64	80	6400	18304	2.86	✓
DeepSeek-V3 (Team, 2025)	2024.12	671B	8/256/1	61	128	7168	2048	0.29	✓
Gemma 3 (12B/27B) (Team et al., 2025)	2025.03	12B/27B	–	48/62	16/32	3840–5376	15360–21504	4.00	✗
DeepSeek-V3/R1 (Liu et al., 2024a)	2025.01	671B	8/256/1	61	128	7168	2048	0.29	✓
MiniMax-Text-01 (Li et al., 2025a)	2025.01	456B	2/32/0	80	64	6144	9216	1.50	✓
Qwen3 (30B/235B) (Team, 2025)	2025.05	30B/235B	8/128/0	48/94	32/64	2048–4096	768–1536	0.38	✓

Table 7: Model architecture summary: model family, release time, and key statistics. **Para.** denotes the parameter count; **Experts** reports the expert configuration (activated/total/shared); **#L** is the number of layers; **#H** is the number of attention heads. We also report the hidden width d_{model} , the intermediate width d_{ffn} , and the FFN expansion ratio. For MoE models that expose dense/shared and routed expert FFNs, we report d_{ffn} as the expert FFN dimension. Accordingly, **FFN Ratio** is defined as $\frac{d_{\text{ffn,expert}}}{d_{\text{model}}}$ for MoEs and $\frac{d_{\text{ffn,dense}}}{d_{\text{model}}}$ for dense models. The ratio is reported as a single value or a range across representative sizes within each model family.

MoEs such as Mixtral remain in a similar range (2.67 to 3.50). In contrast, several recent MoEs substantially compress the routed expert FFN width. For example, DeepSeek-MoE and Qwen1.5-MoE-A2.7B use a ratio of 0.69, and the ratio further drops to 0.29 for DeepSeek-V3/R1 (MoE) and to 0.38 for Qwen3 released in 2025. This indicates that when $d_{\text{ffn}}/d_{\text{model}}$ is continuously becoming smaller, pruning the FFN intermediate dimension alone cannot deliver the same compression benefits as in earlier architectures. This motivates us to shifting the compression target to other structures and dimensions that are comparatively underexplored, such as hidden width (the d_{model} dimension), which can provide complementary slimming gains beyond intermediate-dimension pruning, especially for modern LLMs and MoEs.

C Detailed Implementations about Our Method

C.1 Preserve Efficient Attentions

We do not prune self-attention in this work, because modern MoE backbones widely adopt efficient attention designs such as grouped-query attention (GQA) in Qwen3 or multi-head latent attention (MLA) in DeepSeek, where the K/V pathway is already compact and the redundancy is limited. As a result, attention pruning yields minor parameter and FLOPs reduction compared to pruning the expert FFNs, yet may require expensive training to recover accuracy.

C.2 Initialization of Expert-wise Prune Ratio

We follow (He et al., 2025; Lee et al., 2024; Xie et al., 2024; Muzio et al., 2024; Dong et al., 2025; Li et al., 2025b) to derive expert-wise pruning ratios using a lightweight calibration procedure on C4 (Raffel et al., 2019), GSM8K (Cobbe et al., 2021) and OpenCode (Ahmad et al., 2025). We

sample a fixed calibration set with approximately 1.6M tokens for each dataset, and then conduct the layer importance measurement by NLL and expert-wise importance estimation by routers' output. The detailed process is demonstrated as follow.

Layer-wise initialization by NLL. To measure layer-wise sensitivity, we apply a small perturbation to one target layer at a time and evaluate the induced negative log-likelihood (NLL) change on the calibration set. Layers that exhibit larger NLL degradation are treated as more sensitive and thus should be pruned less.

Given a target overall prune ratio $p\% \in (0, 1)$, our goal is to assign a layer-wise pruning ratio p^ℓ such that the global pruning level matches p . Let $\Delta\mathcal{L}_\ell \geq 0$ denote the measured NLL degradation when perturbing layer ℓ . We first normalize layer sensitivity to $[0, 1]$:

$$\tilde{s}_\ell = \frac{\Delta\mathcal{L}_\ell - \min_j \Delta\mathcal{L}_j}{\max_j \Delta\mathcal{L}_j - \min_j \Delta\mathcal{L}_j + \epsilon}. \quad (3)$$

We then convert sensitivity into a layer pruning weight (higher sensitivity $\Rightarrow$ smaller weight):

$$w_\ell = (1 - \tilde{s}_\ell)^\eta, \quad \eta \geq 1, \quad (4)$$

where η controls the smoothness of reweighting. Finally, we distribute the target pruning budget across layers by normalizing w_ℓ :

$$p^\ell = p \cdot L \cdot \frac{w_\ell}{\sum_{j=1}^L w_j}, \quad (5)$$

where L is the total hidden layers, so that the average layer-wise prune ratio matches the global target, i.e., $\frac{1}{L} \sum_{\ell=1}^L p^\ell = p$. We use p^ℓ as the layer-wise target pruning ratio, which is further used to derive the per-expert initialization at the beginning of BCD algorithm.

Expert-wise initialization by routing weights.

For expert-wise allocation within a layer, we estimate expert importance from the router outputs. For each calibration batch, we collect the router probabilities after the Softmax for routed experts. Let $g_{\ell,e}(x_t) \in [0, 1]$ denote the routing weight of expert e at layer ℓ for token t . We accumulate

routing weights across all calibration tokens from multiple batches:

$$\Omega_{\ell,e} = \sum_{t=1}^T g_{\ell,e}(x_t), \quad (6)$$

where T is the total number of calibration tokens processed across all batches. We then normalize $\Omega_{\ell,e}$ within each layer to obtain the expert importance:

$$\bar{\omega}_{\ell,e} = \frac{\Omega_{\ell,e}}{\sum_{j=1}^E \Omega_{\ell,j}}, \quad \sum_{e=1}^E \bar{\omega}_{\ell,e} = 1. \quad (7)$$

As in the layer-wise initialization, we convert importance into an expert pruning weight (higher importance $\Rightarrow$ smaller weight):

$$v_{\ell,e} = 1 - \bar{\omega}_{\ell,e}. \quad (8)$$

Finally, given the layer-wise pruning ratio p^ℓ , we normalize $v_{\ell,e}$ to distribute the pruning budget across experts:

$$p_{\ell,e} = \text{clip} \left(p^\ell E \cdot \frac{v_{\ell,e}}{\sum_{j=1}^E v_{\ell,j}}, 0, 1 \right). \quad (9)$$

Thus, experts with larger routing importance receive smaller pruning weights and are pruned less. Before clipping, the mean expert-wise pruning ratio remains p^ℓ .

Role of calibration data. We emphasize that calibration data (including GSM8K) is used only to compute the expert-wise KL divergence $\mathcal{L}_0^e$ between the pruned and original model on the routed tokens (Section 4.2); no labels or supervised loss are involved, and no model weights are updated on this data. Calibration therefore only determines *which channels are kept*, not what the model learns, and is mechanistically analogous to activation-based calibration in post-training quantization (e.g., GPTQ, AWQ, SparseGPT). Our GSM8K calibration uses only the training split, which is disjoint from the GSM8K test split used throughout our evaluation. Section 5.3 in the main text reports an ablation over calibration corpus choice.

C.3 Additional Details about Budget-Constrained Coordinate Descent (BCD) Algorithm

³<https://www.databricks.com/blog/introducing-dbrx-new-state-art-open-llm>

⁴<https://github.com/xai-org/grok-1>

⁵<https://github.com/Snowflake-Labs/snowflake-arctic>

⁶<https://github.com/IEIT-Yuan/Yuan2.0-M32>

Algorithm pipeline. For clarity and reproducibility, we provide the full algorithm in Algorithm 1, which is consistent with the description in Section 4.2 in main text and the pipeline illustration in Figure 5.

Wall-clock search cost. Since each BCD probe only requires an expert-local forward pass and KL divergence evaluation rather than a full-model forward pass, the search is inexpensive relative to fine-tuning. Running 100 rounds over all layers takes approximately 10.6 hours for DeepSeek-V2-Lite on 1×NVIDIA H20 and 14.4 hours for Qwen3-30B-A3B on 2×NVIDIA H20. This cost is incurred once per target compression ratio and is substantially smaller than the subsequent LoRA fine-tuning stage.

Expert subset-size decay to reduce coupling. Beyond the main procedure, BCD adopts a scheduled subset size m when selecting experts to update in each round, which decouples the effect of experts in each round. We start from a relatively large subset m_{init} and gradually shrink it to a small m_{min} . The schedule is piecewise: during the first 20% rounds, we keep $m = m_{\text{init}}$ fixed; during the remaining rounds, we linearly anneal m from m_{init} to m_{min} . In our setting, each expert is visited about 20 times on average. Specifically, we set $m_{\text{init}} = 32$ and $m_{\text{min}} = 4$ for QWEN3-30B-A3B, and $m_{\text{init}} = 30$ and $m_{\text{min}} = 2$ for DEEPSEEK-V2-LITE, with a total of 100 rounds.

Using a large m in early rounds accelerates the search by redistributing budget across many experts in parallel. In later rounds, we shrink m to mitigate expert coupling in the loss measurement. If too many experts are updated simultaneously, routed tokens can overlap across experts, so the loss change reflects coupled effects and cannot be cleanly attributed to any single expert’s action. Reducing m in late rounds limits the number of experts that change in one round, making the per-expert comparison between actions more identifiable.

Expert scheduling with score penalization. At each round, we rank experts by a selection score and choose the top- m experts for update. The score encourages exploration while penalizing repeatedly selected or already-converged experts:

$$s_e = \gamma_{\text{explore}} s_e^{\text{explore}} + s_e^{\text{router}} - \gamma_{\text{det}} d_e - \gamma_{\text{rec}} r_e - \gamma_{\text{vis}} v_e, \quad (10)$$

where v_e is the total visit count of expert e , r_e is a recent-selection exponential moving average

(EMA), and $d_e \in \{0, 1\}$ indicates whether an expert is determined (converged). We set $d_e = 1$ if an expert has been visited more than 10 times and its step size has reached the minimum Δ_{min} ; such experts are less visited in the following rounds to avoid wasting updates. The EMA r_e is updated each round by decaying all experts and pushing selected ones toward 1:

$$r_e \leftarrow \eta r_e + (1 - \eta) \mathbb{I}[e \in S], \quad (11)$$

where $\eta \in (0, 1)$ is the EMA decay factor and S is the selected subset.

The exploration term s_e^{explore} is defined as the following equation to favor rarely visited experts:

$$s_e^{\text{explore}} = \sqrt{\frac{\log(1+t)}{1+v_e}}, \quad (12)$$

where t is the current round index. The router term s_e^{router} is derived from the router outputs indicating the importance of the expert. In practice, we additionally filter out experts that are not activated on the current batch to avoid spending updates on inactive experts. Overall, this score penalization balances expert visit times and importance.

Remark. The score penalization is introduced for accelerating the search progress within a limited number of rounds. In practice, if we use sufficient search rounds, BCD can achieve the similar results without penalties; thus, this design is optional and can be simplified when the search round is large.

D Experiment

D.1 Detailed Experiment Settings

All models, datasets, and code artifacts used in this work are used in accordance with their original licenses and intended research purposes. These are public research artifacts, and we only use them for model compression, evaluation, and reproducibility studies.

Ablation Setup. In Table 4, we compare BCD with several prune ratio allocation strategies. *Pre-set* fixes a global hidden pruning ratio p_H that shared across experts, and computes the corresponding expert-wise intermediate prune ratios $p_{I,e}$; we sweep $p_H \in \{0.1, 0.2, 0.3\}$ and report the mean_{std} across these settings. *Random* samples expert-specific $p_{H,e}$ from the uniform distribution; we report results averaged over three distributions, $p_H \sim U(0, 0.1)$, $U(0, 0.2)$, and $U(0, 0.3)$,

and then calculate expert-wise $p_{I,e}$. “Mixed Top-k” jointly ranks intermediate and hidden channels in 1D-vector, and keep the channels with highest scores. However, these naive methods all lag behind our BCD algorithm, which achieves the best performance with a clear margin.

Training hyperparameters. We summarize the key training hyperparameters in Table 8. We distill the pruned model on Alpaca with a base learning rate of 1×10^{-4} . To stabilize training, we scale the learning rate for the attention LoRA adapters and router adapters by 0.5 and 0.01, respectively. We also use smaller LoRA ranks for the router, setting it to 4, while using larger ranks for the FFN and attention modules (32 and 16, respectively). We do not validate whether increasing the router LoRA rank would further improve performance.

Category	Setting
Base lr	1×10^{-4}
Max grad norm	0.5
$\mathcal{L}_{\text{RTD}}$ coef	10
$\mathcal{L}_{\text{RTD}}$ temperature	1.0
$\mathcal{L}_{\text{logits}}$ coef	1.0
$\mathcal{L}_{\text{logits}}$ temperature	3.0
Router aux loss coef	0.001
FFN LoRA rank	32
FFN LoRA alpha	128
FFN LoRA dropout	0.05
Gate LoRA rank	4
Gate LoRA alpha	16
lr_{gate}	1×10^{-6}
Attention LoRA rank	16
Attention LoRA alpha	64
lr_{attn}	5×10^{-5}

Table 8: Training and distillation hyperparameters. The gate and attention learning rates are scaled as $\text{lr}_{\text{gate}} = \text{lr} \cdot 0.01$ and $\text{lr}_{\text{attn}} = \text{lr} \cdot 0.5$.

Comparison baselines. We compare our method against representative pruning methods that prune hidden dimension for LLMs, covering both unstructural and structural pruning techniques: (1) Wanda (Sun et al., 2023) and SparseGPT (Frantar and Alistarh, 2023) are used as unstructured pruning baselines, applying a uniform pruning ratio across all layers and experts. (2) LLM-Shearing (Xia et al., 2023) performs structured slimming with a globally shared hidden-dimension pruning, while allowing the intermediate pruning to vary across layers and experts. We do not adopt its Dynamic Batch Loading component, as our study

does not consider on-the-fly reweighting of training data across domains during fine-tuning. (3) SliceGPT (Ashkboos et al., 2024) introduces additional linear transformations on the residual path to align feature dimensions, enabling different hidden pruning indices. For all methods, we evaluate under overall pruning ratios of 25% and 50%.

Models and Datasets. We evaluate on representative open-source MoEs, DEEPSEEK-V2-LITE and QWEN3-30B-A3B, which both have small FFN expansion ratios, i.e., $d_{\text{ffn}}/d_{\text{model}}=0.69$ or 0.38, respectively. Their small FFN expansion ratios also limit the gains from intermediate-only pruning, making them well suited for evaluating joint intermediate and hidden pruning, as well as our distillation strategy. We calibrate BCD using task-related data, including C4 (Raffel et al., 2019) for general tasks, GSM8K (Cobbe et al., 2021) for math, and OpenCode (Ahmad et al., 2025) for code. We perform all post-pruning fine-tuning and distillation on Alpaca (Taori et al., 2023) for 2 epochs, without any additional math or code training data. For channel scoring for calibration, we sample a fixed calibration set with approximately 1.6M tokens for each dataset.

D.2 Additional Visualizations for BCD

Layer-wise evolution of hidden pruning on Qwen3-30B-A3B-Thinking. Figure 8 visualizes how the hidden pruning ratio p_H evolves over the coordinate descent steps for multiple layers from shallow to deep. A consistent pattern is that p_H is highly expert-dependent within each layer: some experts quickly updates to relatively larger p_H values (darker bands), while many others remain close to zero (nearly white). Moreover, hidden pruning is more aggressive in shallow layers, while deeper layers end up with lower p_H , meaning they retain more hidden channels, indicating that deeper layers may be more sensitive to hidden dimension pruning.

Most experts stabilize early, while a small subset keeps adjusting. In Figure 8, many experts exhibit rapid changes in the first few search steps followed by long plateaus, which appear as horizontal bands that persist in the remaining steps. At the same time, a smaller subset of experts continues to receive updates throughout the search, visible as late-stage color changes at specific expert indices. This contrast highlights that the search dynamics

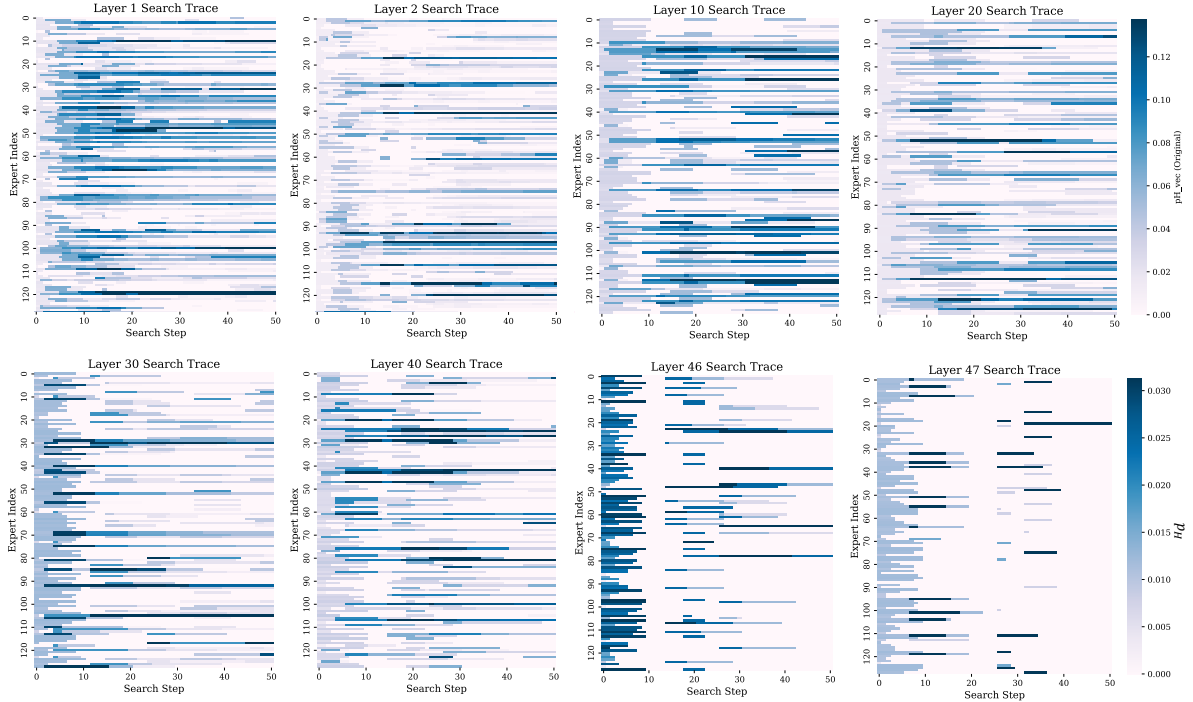


Figure 8: Visualization of the coordinate descent search process with Qwen3-30B-A3B-Thinking. The heatmaps illustrate the evolution of hidden pruning ratio p_H across search steps for various layers from shallow to deep.

are not uniform across experts even within the same layer.

Visit counts on DeepSeek-V2-Lite. The visit-count heatmaps in Figure 9(c) become progressively darker from left to right, indicating that visit counts accumulate as the search proceeds. At the same time, the intensity is not uniform across expert indices, suggesting that some experts are visited more frequently than others by score penalization.

Adaptive step sizes over steps. Figure 9(d) visualizes the adaptive step sizes Δ_e used during the search. The step-size patterns are clearly expert-dependent. And the step adaptation is not constant over time. Most experts adopt larger step sizes in the middle stage of the search, and then shrink them at the end.

D.3 Layer Selection for Distillation

Layer selection Φ for RTD. In Table 9, we explicitly list the distillation layer set Φ used for each setting. Following the layer selection heuristic adopted in BitNet (Wu et al., 2025), we primarily distill the middle-to-late layers, which are often more aligned with high-level reasoning and task behaviors. In addition, we annotate whether the logits distillation loss $\mathcal{L}_{\text{logits}}$ is enabled.

KD Loss	$ \Phi $	Φ (layer indices)	$\mathcal{L}_{\text{logits}}$
No distill	0	$\emptyset$	$\times$
$\mathcal{L}_{\text{logits}}$	0	$\emptyset$	$\checkmark$
$\mathcal{L}_{\text{RTD}}$	1	{40}	$\times$
$\mathcal{L}_{\text{RTD}}$	2	{40, 44}	$\times$
$\mathcal{L}_{\text{RTD}}$	4	{32, 36, 40, 44}	$\times$
$\mathcal{L}_{\text{RTD}}$	7	{20, 24, 28, 32, 36, 40, 44}	$\times$
$\mathcal{L}_{\text{RTD}}, \mathcal{L}_{\text{logits}}$	7	{20, 24, 28, 32, 36, 40, 44}	$\checkmark$
$\mathcal{L}_{\text{RTD}}$	12	{0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44}	$\times$

Table 9: Layer selection Φ for different distillation settings. $|\Phi|$ is the number of selected layers. $\mathcal{L}_{\text{logits}}$ denotes logits distillation and $\mathcal{L}_{\text{RTD}}$ denotes relation transfer distillation.

Empirical finding and current scope. As reported in the ablation study (Section 5.3) in main text, using $|\Phi| = 7$ yields the best overall outcome among the tested settings, while using fewer layers ($|\Phi| = 1, 2, 4$) or more layers ($|\Phi| = 12$) leads to suboptimal results. However, this analysis is preliminary. We do not evaluate distilling *only* shallow layers, nor do we explore *denser* layer selections beyond our tested grids (e.g., every layer or every two layers). A more comprehensive study of the distillation, including layer selection, and finer-grained distillation schedules is left for future work.

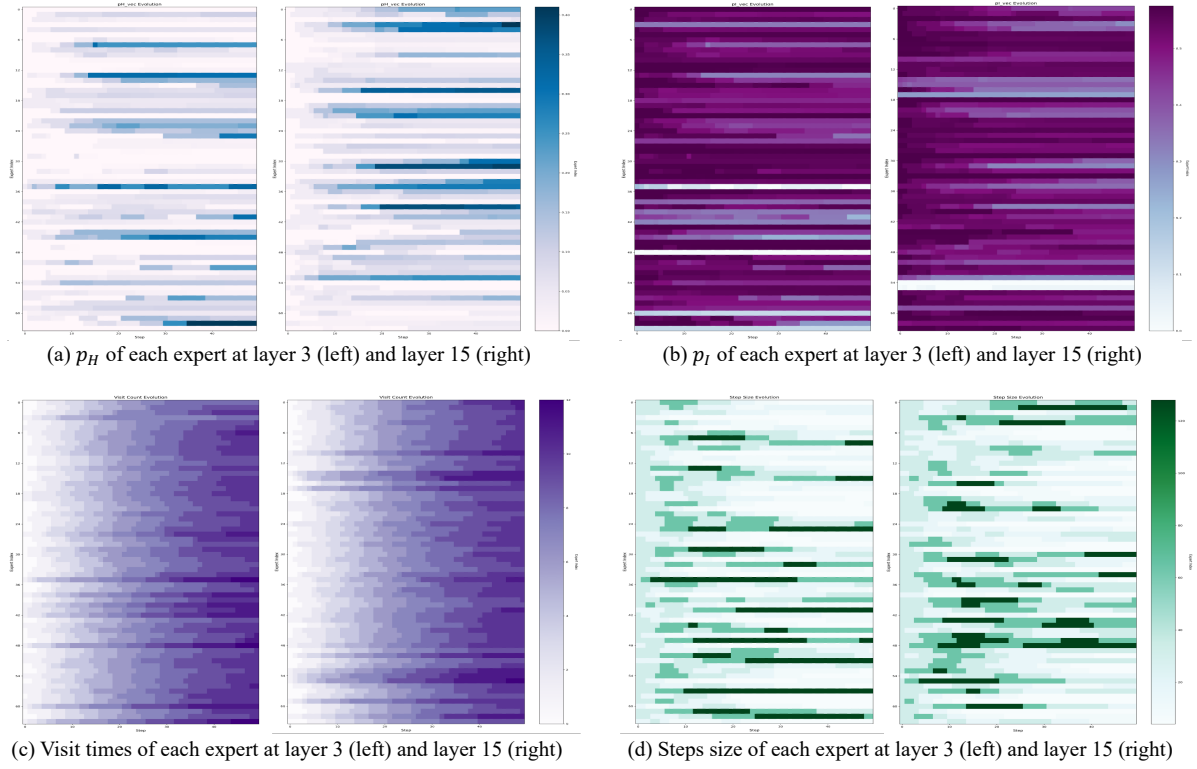


Figure 9: Visualization of the BCD search dynamics on DeepSeek-V2-Lite. Heatmaps show the evolution of (a) hidden pruning ratio (p_H) and (b) intermediate pruning ratio (p_I) across search steps for each expert at a shallow layer (Layer 3, left) and a deeper layer (Layer 15, right); (c) expert visit counts and (d) adaptive step sizes Δ_e over the same search process.

Algorithm 1 Budget-Constrained Coordinate Descent (BCD) at Layer ℓ

```

1: Input: target prune ratio  $p_\ell \in \mathbb{R}^E$  for experts  $\mathcal{E}$  at layer  $\ell$ ; expert count  $E$ ; intermediate size  $d_{\text{ffn}}$ ; hidden size  $d_{\text{model}}$ ; initial step sizes  $\Delta \in \mathbb{R}_+^E$ ; model to be pruned  $\theta$ ; calibration batches  $\{B_r\}_{r=1}^R$ ; total rounds  $R$ ; minimal step size  $\text{step}_{\min}$ .
2: Output: kept channels  $(k_I, k_H) \in \mathbb{Z}_+^E \times \mathbb{Z}_+^E$  for layer  $\ell$ .
3: Constants: budget exchange rate  $\alpha \leftarrow \frac{3 d_{\text{model}}}{2 d_{\text{ffn}}}$ .
4: // Initialization: intermediate-only allocation
5:  $k_{I,e} \leftarrow \max(1, \lfloor d_{\text{ffn}} \cdot (1 - p_{\ell,e}) \rfloor)$ ,  $\forall e \in \mathcal{E}$  {keep channels}
6:  $k_{H,e} \leftarrow d_{\text{model}}$ ,  $\forall e \in \mathcal{E}$ 
7:  $\text{success}_e \leftarrow 0$ ,  $\text{visit}_e \leftarrow 0$ ,  $\forall e \in \mathcal{E}$ 
8: for  $r \leftarrow 1$  to  $R$  do
9:    $B \leftarrow B_r$ 
10:  // Evaluate current expert-wise loss on routed tokens
11:   $\mathcal{L}_0^e \leftarrow \text{ExpertKL}(\theta, k_I, k_H; B, e)$ ,  $\forall e \in \mathcal{E}$ 
12:  // Select experts to update: lowest-visit, then filter rarely-activated ones
13:   $S \leftarrow \text{SelectExperts}(\text{visit}, B, r)$   $\{S \subseteq \mathcal{E}\}$ 
14:  // Probe Action A1: prune more intermediate ( $d_{\text{ffn}}$ ), compensate hidden ( $d_{\text{model}}$ )
15:   $(k_I^{A_1}, k_H^{A_1}) \leftarrow (k_I, k_H)$ 
16:  for each  $e \in S$  do
17:     $k_{I,e}^{A_1} \leftarrow \max(1, k_{I,e} - \Delta_e)$ 
18:     $k_{H,e}^{A_1} \leftarrow \min(d_{\text{model}}, k_{H,e} + \lfloor \alpha \Delta_e \rfloor)$ 
19:  end for
20:   $\mathcal{L}_{A_1}^e \leftarrow \text{ExpertKL}(\theta, k_I^{A_1}, k_H^{A_1}; B, e)$ ,  $\forall e \in S$ 
21:  // Probe Action A2: prune more hidden ( $d_{\text{model}}$ ), compensate intermediate ( $d_{\text{ffn}}$ )
22:   $(k_I^{A_2}, k_H^{A_2}) \leftarrow (k_I, k_H)$ 
23:  for each  $e \in S$  do
24:     $k_{H,e}^{A_2} \leftarrow \max(1, k_{H,e} - \Delta_e)$ 
25:     $k_{I,e}^{A_2} \leftarrow \min(d_{\text{ffn}}, k_{I,e} + \lfloor \Delta_e / \alpha \rfloor)$ 
26:  end for
27:   $\mathcal{L}_{A_2}^e \leftarrow \text{ExpertKL}(\theta, k_I^{A_2}, k_H^{A_2}; B, e)$ ,  $\forall e \in S$ 
28:  // Coordinate descent update (per-expert)
29:  for each  $e \in S$  do
30:     $L^* \leftarrow \min(\mathcal{L}_{A_1}^e, \mathcal{L}_{A_2}^e)$ 
31:    if  $L^* < \mathcal{L}_0^e$  then
32:      if  $\mathcal{L}_{A_1}^e \leq \mathcal{L}_{A_2}^e$  then
33:         $(k_{I,e}, k_{H,e}) \leftarrow (k_{I,e}^{A_1}, k_{H,e}^{A_1})$ 
34:      else
35:         $(k_{I,e}, k_{H,e}) \leftarrow (k_{I,e}^{A_2}, k_{H,e}^{A_2})$ 
36:      end if
37:    end if
38:     $\text{visit}_e \leftarrow \text{visit}_e + 1$ 
39:  end for
40:  // Step-size adaptation (per-expert, success-based)
41:  for each  $e \in S$  do
42:    if  $\min(\mathcal{L}_{A_1}^e, \mathcal{L}_{A_2}^e) < \mathcal{L}_0^e$  then
43:       $\text{success}_e \leftarrow \text{success}_e + 1$ 
44:      if  $\text{success}_e = 2$  then
45:         $\Delta_e \leftarrow 2\Delta_e$ ;  $\text{success}_e \leftarrow 0$ 
46:      end if
47:    else
48:       $\Delta_e \leftarrow \max(\Delta_e / 2, \text{step}_{\min})$ 
49:       $\text{success}_e \leftarrow 0$ 
50:    end if
51:  end for
52: end for
53: return  $(k_I, k_H)$ 

```
