PRUNE BETTER BEFORE RECOVERING HARDER: TRAJECTORY-AWARE PRUNING AND RECOVERY FOR REASONING LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reasoning language models are expensive to serve because they generate long autoregressive reasoning traces. Structured layer pruning can reduce this cost by shortening model depth, but criteria based on hidden-state similarity may remove layers that are important for reasoning. We propose *trajectory-aware pruning and recovery*, a preservation-first framework for compressing large language models. For each candidate layer, we temporarily replace it with an identity mapping and measure how much the model's next-token predictions change on fixed reasoning trajectories generated by the original model. We prune layers with the smallest behavioral effect, then use on-policy distillation to correct the remaining distribution shift on contexts visited by the pruned model. This preserves reasoning behavior before compression and reserves distillation for the errors that remain. Experiments on mathematical reasoning, code generation, and open-ended generation show that our method consistently outperforms SOTA method demonstrating that better layer selection reduces the burden on recovery.

1 INTRODUCTION

Large language models (LLMs) often generate long chains of thought (CoTs) for reasoning tasks, making autoregressive inference computationally expensive. Structured layer pruning addresses this cost by removing entire Transformer layers (Kim et al., 2024; Xia et al., 2024; Men et al., 2025), directly reducing model depth and enabling practical inference speedups. Compared with unstructured methods such as Wanda and SparseGPT (Sun et al., 2023; Frantar & Alistarh, 2023), it removes hardware-aligned computation, but it also raises a central question for reasoning models: which layers can be removed without disrupting the sequence of computations that produces a valid reasoning trace? Removing a layer that appears redundant can instead collapse reasoning capability and leave the pruned model unable to generate coherent reasoning steps (Zhang et al., 2026).

Some layer-pruning methods estimate redundancy from hidden representations: a layer whose input and output states are similar is considered less important. This criterion is attractive because it is simple and local, but representation-level similarity is only an indirect proxy for functional importance. A layer can make a small immediate change to hidden states while still altering the probability of a critical reasoning token or supporting computations in later layers. Because generation is autoregressive, such a local error can change subsequent contexts and compound over a long reasoning trajectory. ShortOPD (Zhang et al., 2026) documents these failures and proposes a specialized on-policy distillation (OPD) procedure to restore performance after pruning. Yet when pruning causes a broad capability collapse, recovery can be nearly as difficult as training a new model from scratch, since OPD must relearn behavior that could have been preserved at pruning time.

This limitation motivates a different formulation of the problem. We therefore shift the central question from *how can we better recover a severely pruned reasoning model?* to *which layers can be removed without destroying its reasoning process?* Under this view, pruning is first a layer-selection problem: we should identify layers whose removal has a small effect on the model's reasoning trajectory and preserve the others before compression damage occurs. Recovery can then focus on correcting the smaller distribution shift that remains, rather than reconstructing reasoning behavior that pruning has already destroyed.

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

We address this problem with *trajectory-aware pruning and recovery*. For each candidate layer, we construct a counterfactual model by replacing that layer with an identity mapping, then evaluate the resulting next-token predictions on fixed reasoning trajectories generated by the original model. We combine the change in the next-token distribution with the drop in probability assigned to the teacher-generated token to obtain a trajectory-level behavioral score. We remove the layers with the smallest scores, preserving the computations that matter most for the original reasoning process. Finally, we apply OPD on contexts visited by the pruned model to correct the residual distribution shift introduced by structural compression. This implements a *preserve first, recover second* principle: layer selection protects reasoning behavior, while recovery refines the errors that remain. An overview of the framework is shown in Figure 1.

Our contributions are as follows:

- We identify a mismatch between representation-level redundancy and reasoning-trajectory redundancy in structured pruning for reasoning models. Based on this observation, we propose a *trajectory-aware pruning and recovery* framework that preserves reasoning behavior before compression and reserves recovery for the residual errors.
- We propose trajectory-aware layer selection. On fixed teacher-generated reasoning trajectories, we counterfactually replace each candidate layer with an identity mapping and rank layers using both next-token distributional change and degradation in the teacher-generated token probability.
- Experiments show that, at matched pruning ratios and on-policy recovery budgets, trajectory-aware selection provides a substantially stronger pruned initialization and improves the final accuracy–efficiency trade-off after recovery.

2 Related Works

2.1 Pruning for Large Language Models

Pruning has become a common approach for reducing the memory and inference cost of large language models (LLMs). Unstructured methods such as SparseGPT and Wanda remove individual weights using calibration-time second-order information or simple weight–activation statistics, respectively (Frantar & Alistarh, 2023; Sun et al., 2023). Although these methods can achieve high sparsity, their irregular computation generally requires specialized kernels to translate sparsity into wall-clock speedups. Structured pruning instead removes hardware-aligned units, including layers, attention heads, channels, or rows and columns. LLM-Pruner performs gradient-informed structural pruning followed by recovery fine-tuning (Ma et al., 2023), while Sheared Llama incorporates structured pruning into continued pre-training to obtain a smaller target architecture (Xia et al., 2024). SliceGPT compresses the model by deleting rows and columns after a rotation of the representation space (Ashkboos et al., 2024); FLAP uses fluctuation-based statistics for adaptive structured pruning (An et al., 2024); and SoBP selects structures using global first-order information, a local greedy refinement, and module-wise reconstruction (Wei et al., 2024). WLR further studies retraining-free pruning through well-conditioned linear reconstruction (Li et al., 2026).

For depth pruning, Shortened Llama compares retraining strategies after deleting Transformer layers (Kim et al., 2024), and ShortGPT identifies removable layers with Block Influence (BI), measuring the change between a layer's input and output representations (Men et al., 2025). These approaches demonstrate that substantial depth reduction is possible, but their importance criteria remain primarily representation- or reconstruction-based. In particular, a small hidden-state change does not imply a small change in the next-token policy: a layer can have little average effect on representations while still altering a critical token decision in a long autoregressive reasoning trace. This distinction is especially important for reasoning models, where local deviations can change subsequent contexts and accumulate over many generation steps. Our work therefore evaluates layer removal by the behavioral effect of layer ablation on teacher-generated reasoning trajectories, directly targeting behavioral rather than representation-level redundancy.

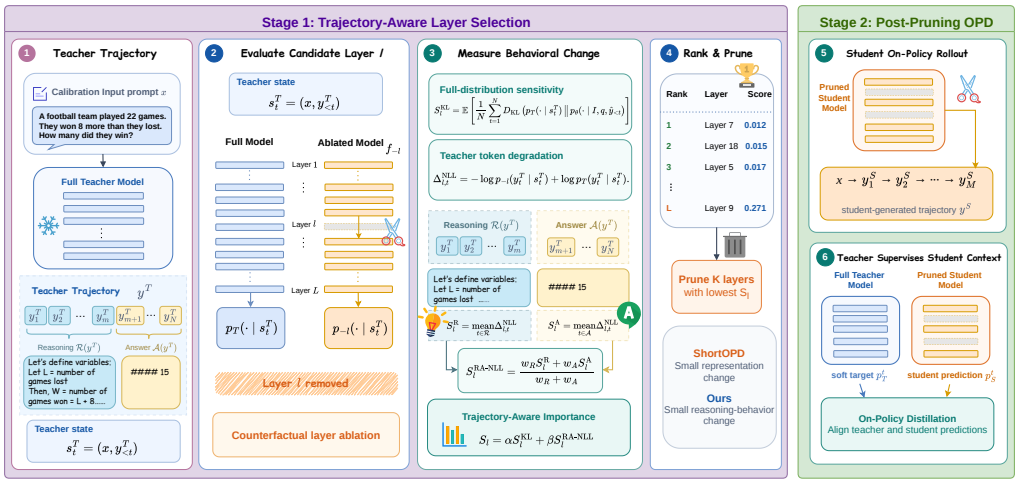

Figure 1: Overview of trajectory-aware pruning and recovery. We evaluate layer removal on fixed teacher reasoning trajectories, prune layers with the smallest behavioral effect, and use on-policy distillation to recover the residual trajectory shift.

2.2 Pruning with Post-Pruning Distillation or Reconstruction

Recent work has also explored explicit recovery procedures after structural pruning. ShortOPD observes that BI-based layer pruning can lead to repetition, generation collapse, and severe long-form reasoning degradation, and proposes Short-to-Long On-Policy Distillation: the pruned student is trained on states visited by its own rollouts, with a schedule that progresses from shorter to longer generations (Zhang et al., 2026). On-policy supervision addresses the distribution shift induced by autoregressive student behavior and can recover a substantial part of the lost capability. However, because layer selection is still inherited from BI, OPD is applied after potentially reasoning-critical layers have already been removed; recovery must consequently relearn behavior that could have been preserved during pruning.

Lucas et al. study accurate pruning of reasoning models through chain-of-thought reconstruction (Lucas et al., 2026). Their approach uses generated reasoning traces as a reconstruction signal to identify or compensate for the computation removed by pruning, thereby retaining the model's chain-of-thought behavior more effectively than generic post-training. Nevertheless, the reconstruction objective remains a recovery-oriented surrogate and does not directly rank layers by their token-level behavioral effect under a common teacher trajectory. It can therefore conflate errors caused by layer removal with errors caused by regenerated or reconstructed contexts, and still places substantial burden on post-pruning training. In contrast, our framework uses fixed teacher trajectories and counterfactual identity ablations to select layers whose removal minimally changes next-token distributions before applying OPD. OPD consequently serves only as residual recovery, yielding a preservation-first complement to these recovery-centric methods.

3 Methods

We propose *trajectory-aware pruning and recovery*, a two-stage framework for compressing reasoning models while preserving their behavior, as shown in Figure 1. It combines behavior-guided layer selection with on-policy recovery; the following subsections detail each stage.

3.1 Trajectory-Aware Layer Selection

3.1.1 Counterfactual Layer Ablation

Our goal is to estimate the reasoning behavioral damage caused by removing a layer before any structural change is made. This isolates the selection problem from the later recovery problem: a

layer that can be compensated for after extensive training may still cause a severe transient collapse when removed. Let f_T denote the original model with L Transformer layers. For each input x in a calibration set $\mathcal{D}_{\text{cal}}$, the full model generates a teacher trajectory $y^T = (y_1^T, \ldots, y_N^T)$ of length N and its associated states $s_t^T = (x, y_{<t}^T)$. At every generation step t, the full model defines the teacher distribution:

$$p_T(\cdot \mid s_t^T) = f_T(s_t^T) \tag{1}$$

To evaluate layer l, we replace only that layer with an identity mapping and obtain the ablated model f_{-l}. This preserves the residual stream and all surrounding parameters, so the comparison isolates the computation performed by layer l and matches the eventual structured-pruning operation. The intervention therefore asks whether the original computation depends on this layer at pruning time, rather than whether a newly trained shallow model could eventually compensate for its absence. Every candidate is evaluated on the same teacher states s_t^T, rather than on a trajectory sampled from itself; otherwise, different prefixes would confound the measured layer effect. The ablated model distribution is:

$$p_{-l}(\cdot \mid s_t^T) = f_{-l}(s_t^T). \tag{2}$$

3.1.2 Distributional sensitivity on fixed trajectories

With the prefixes held fixed, we can now measure the immediate policy perturbation caused by each ablation. We use two complementary signals. First, the distribution score measures the overall change in next-token predictions. It is computed over the whole vocabulary because matching only the sampled token would miss probability mass transferred to competing tokens, even though that transfer can alter a later reasoning step:

$$S_l^{\text{KL}} = \mathbb{E}_{x \sim \mathcal{D}_{\text{cal}},\, y^T \sim p_T(\cdot|x)} \left[\frac{1}{N} \sum_{t=1}^{N} D_{\text{KL}}\big(p_T(\cdot \mid s_t^T) \,\big\|\, p_{-l}(\cdot \mid s_t^T)\big) \right]. \tag{3}$$

3.1.3 Teacher-token and region-aware sensitivity

The KL score treats all probability changes as relevant, but it does not distinguish a harmless change on an unlikely token from a change on the token that advances the current proof or completes the answer. We therefore add a teacher-token signal. For each token generated by the teacher, we define the per-step negative log-likelihood (NLL) degradation as

$$\Delta_{l,t}^{\text{NLL}} = -\log p_{-l}(y_t^T \mid s_t^T) + \log p_T(y_t^T \mid s_t^T). \tag{4}$$

Conditioned on s_t^T, the expectation of this quantity over $y_t^T \sim p_T(\cdot \mid s_t^T)$ equals the corresponding KL divergence. Thus, the teacher-token term is not a second estimate of a different divergence; it focuses that divergence on the concrete sequence the model actually used. Let $\mathcal{R}(y^T)$ and $\mathcal{A}(y^T)$ denote the reasoning and final-answer token sets, respectively. These sets are determined by the task's answer parser (e.g., explicit `</think>` or `Final Answer:` boundaries) and are kept fixed across all candidate layers. We compute region-normalized degradation scores

$$S_l^{\text{R}} = \mathbb{E}_{x,y^T} \left[\frac{1}{|\mathcal{R}(y^T)|} \sum_{t \in \mathcal{R}(y^T)} \Delta_{l,t}^{\text{NLL}} \right], \tag{5}$$

$$S_l^{\text{A}} = \mathbb{E}_{x,y^T} \left[\frac{1}{|\mathcal{A}(y^T)|} \sum_{t \in \mathcal{A}(y^T)} \Delta_{l,t}^{\text{NLL}} \right]. \tag{6}$$

We then combine the two regions as

$$S_l^{\text{RA-NLL}} = \frac{w_R S_l^{\text{R}} + w_A S_l^{\text{A}}}{w_R + w_A}, \tag{7}$$

where w_R is the reasoning weight and w_A is the answer weight. Region-wise normalization prevents the usually much longer reasoning span from numerically overwhelming the final answer span. The ratio $w_R : w_A$ therefore directly specifies the relative protection assigned to intermediate reasoning and final-answer behavior, instead of merely reflecting their token counts. This is a modeling

choice: reasoning tokens determine the intermediate state visited by the model, while answer tokens determine whether that state becomes a correct externally evaluated output.

The final layer-importance score is

$$S_l = \alpha S_l^{\mathrm{KL}} + \beta S_l^{\text{RA-NLL}}, \tag{8}$$

where α and β are the coefficients for the KL and trajectory-aware NLL terms, respectively. The KL term penalizes broad policy redistribution, whereas the trajectory-aware NLL term makes the realized reasoning path and its answer endpoint explicit. Combining them avoids selecting a layer merely because it preserves the sampled token while changing alternative probabilities, or merely because it has a small average distributional effect while harming a semantically important token. A lower score therefore means that the counterfactual removal causes less observable disruption to the teacher's reasoning behavior.

3.1.4 Joint score and pruning decision

Given a pruning budget of K layers, we remove the K layers with the lowest scores. The ranking is computed before pruning so all candidates are judged against the same uncompressed reference. Since the region score is normalized by w_R+w_A, jointly scaling w_R and w_A has no effect; only their ratio controls the reasoning–answer trade-off. The coefficients α and β control the balance between broad policy preservation and preservation of the realized trajectory. This criterion therefore selects layers whose removal has little effect on reasoning behavior, rather than layers that only make a small change to hidden representations. The selected layers are then deleted to form the student model for the next stage, so on-policy recovery can focus on residual distribution shift.

The resulting ranking gives us a behaviorally safer student initialization. Nevertheless, the student will eventually encounter its own prefixes during generation, so a residual distribution shift can still accumulate. We address this remaining mismatch in the next subsection.

3.2 On-Policy Recovery

Trajectory-aware layer selection reduces the initial reasoning damage caused by pruning, but the pruned model can still move away from the full model during long generation. We use on-policy distillation to correct this remaining trajectory error. Let f_S be the model after pruning. For each input x, it generates a rollout and visits contexts s_t^S as follows:

$$y^S \sim \pi_S(\cdot \mid x), \qquad s_t^S = (x, y_{<t}^S), \tag{9}$$

The teacher provides a target distribution $p_T(\cdot \mid s_t^S)$ on these student-visited contexts. We optimize the student with:

$$\mathcal{L}_{\mathrm{OPD}} = \mathbb{E}_{x,t}\left[D_{\mathrm{KL}}\big(p_T(\cdot \mid s_t^S) \,\big\|\, p_S(\cdot \mid s_t^S)\big)\right]. \tag{10}$$

Because the supervision is provided on the student's own rollouts, this objective directly addresses contexts that arise after pruning. In this framework, on-policy distillation is not asked to rebuild severely damaged reasoning behavior. Instead, trajectory-aware layer selection preserves as much of that behavior as possible, and distillation recovers the residual shift caused by structural compression.

4 Experiments

4.1 Experimental Setup

Models and baselines. We conduct layer-pruning experiments on Qwen3-4B-Instruct and Qwen3-8B (Qwen Team, 2025). Our comparisons separate the effects of layer selection and post-pruning recovery. For layer selection, we compare the Block Influence (BI) criterion used by ShortGPT (Men et al., 2025) with our trajectory-aware criterion. For recovery, we consider supervised fine-tuning (SFT) and on-policy distillation (OPD), applying the recovery objectives to the corresponding pruned models. We also report the published results of ShortOPD (Zhang et al., 2026) as an external reference. Because its implementation and recovery training set are not publicly available, these results are not treated as a strictly controlled comparison.

Pruning calibration. To construct the teacher trajectories used for layer scoring, we form a mixed calibration pool from MATH Hendrycks et al. (2021), GSM8K Cobbe et al. (2021), OpenCodeInstruct Ahmad et al. (2025), Sharegpt Vicuna, and Ultrachat Ding et al. (2023). We randomly sample 96 examples from this mixture and let the unpruned model generate one teacher trajectory for each example. All candidate layer ablations are evaluated on this same fixed set of trajectories.

Training set. Our training set contains 49,973 examples spanning general instruction following, code, and mathematics. The general subset contains 15,000 examples, evenly divided between Ultrachat and ShareGPT Vicuna (7,500 each). The code subset contains 15,000 examples from OpenCodeInstruct. The mathematical subset consists of all 7,473 training examples from GSM8K and 12,500 examples from MATH. Training and evaluation examples are kept disjoint, with standard quality filtering and deduplication applied. All recovery methods use the same prompt mixture: SFT learns from the corresponding reference responses, whereas OPD samples student rollouts and obtains token-level targets from the frozen, unpruned teacher on the student-visited trajectories.

Evaluation. We evaluate free-form generation on six task families in three domains. For mathematics, we use GSM8K (Cobbe et al., 2021) and MATH-500 (Lightman et al., 2023), scored by answer-match accuracy. MATH-500 uses Qwen-style long decoding with a 16k-token generation cap and the recommended sampling parameters. For code, we use HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021), scored by execution pass@1. For open-ended generation, we evaluate only Alpaca and MT-Bench (Zheng et al., 2023); following ShortOPD, their responses are judged on a 1–10 scale by GPT-5.5. Tables report mathematics and code scores as percentages and open-ended judge scores on the 1–10 scale. The overall average rescales each judge score by a factor of 10 and then averages the six task scores.

4.2 Main Results

We organize the comparison by pruning rate and, within each rate, explicitly separate the layer-selection criterion from the recovery objective. The unpruned model provides the reference point, and ShortOPD is included as a published external baseline. Since ShortOPD's code and training data are not publicly available, comparisons with it should be viewed as a reference rather than a strictly controlled evaluation.

Table 1 reports results for Qwen3-4B-Instruct. At a 25% pruning rate, our trajectory-aware selection followed by OPD reaches 87.33% on GSM8K, 81.00% on MATH-500, 53.6% on MBPP, and 65.85% on HumanEval, while also obtaining the strongest open-ended scores. Its overall score is 68.35, compared with 56.85 for BI selection followed by the same OPD recovery. In contrast, SFT after the same trajectory-aware pruning leads to a near-collapse, highlighting the importance of the recovery objective. At the high sparsity of 50%, BI nearly completely collapses, with an overall score of only 7.19, whereas TA still retains meaningful capability and achieves 30.63.

Table 1: Main results for Qwen3-4B-Instruct, grouped by pruning rate. TA denotes our trajectory-aware layer-selection criterion, and BI denotes Block Influence. Bold denotes the best result within each pruning-rate group.

Pruning rate	Layer selection	Recovery	Math		Code		Open-ended		Overall
			GSM8K	MATH-500	MBPP	HumanEval	Alpaca	MT-Bench	
0%	—	—	84.31	94.60	71.00	81.70	8.66	7.69	82.52
25%	BI	ShortOPD	62.70	42.00	49.81	43.90	4.68	4.28	48.00
	TA	SFT	1.66	3.80	5.40	1.22	2.41	1.37	8.32
	TA	OPD	**87.33**	**81.00**	**53.60**	**65.85**	**6.44**	**5.79**	**68.35**
	BI	OPD	78.99	70.40	44.00	46.95	5.59	4.49	56.85
30%	TA	SFT	0.90	1.20	3.60	1.82	2.15	1.23	6.88
	TA	OPD	**85.14**	**76.00**	**48.20**	**54.26**	**6.15**	**4.85**	**62.27**
	BI	OPD	80.13	63.60	37.40	39.02	5.13	4.14	52.13
50%	TA	SFT	0.38	0.60	0.00	0.00	1.00	1.00	3.50
	TA	OPD	**59.59**	**29.80**	**24.40**	**21.90**	**2.89**	**1.92**	**30.63**
	BI	OPD	1.51	4.60	4.60	7.30	1.39	1.13	7.19

Table 2: Main results for Qwen3-8B, grouped by pruning rate. TA denotes our trajectory-aware layer-selection criterion, and BI denotes Block Influence.

Pruning rate	Layer selection	Recovery	Math		Code		Open-ended		Overall
			GSM8K	MATH-500	MBPP	HumanEval	Alpaca	MT-Bench	
0%	—	—	85.29	83.00	65.60	62.20	8.79	7.41	76.34
25%	BI	ShortOPD	75.74	40.80	42.80	40.24	3.38	3.29	44.38
	TA	SFT	2.35	5.40	0.00	0.00	1.68	1.21	6.09
	TA	OPD	**86.80**	**70.80**	**43.40**	**39.63**	**5.91**	**5.18**	**58.60**
	BI	OPD	75.96	49.60	23.80	7.31	4.94	3.99	41.00
30%	TA	SFT	1.06	3.80	0.00	0.00	1.29	1.04	4.70
	TA	OPD	**89.38**	**64.60**	**37.40**	**25.00**	**5.80**	**4.43**	**53.11**
	BI	OPD	60.57	39.40	22.60	5.49	4.36	3.45	34.37

Table 3: Ablation of the Stage-1 scoring components for Qwen3-4B-Instruct at 25% pruning. All rows use the same OPD recovery and balanced region weights. Bold denotes the best value in each column.

Configuration	GSM8K	MATH-500	MBPP	HumanEval	Alpaca	MT-Bench	Overall
KL + RA-NLL (default)	**87.33**	81.00	**53.60**	**65.85**	6.44	**5.79**	**68.35**
KL only (w/o NLL scoring)	86.65	**82.60**	51.80	**65.85**	**6.53**	5.35	67.61
RA-NLL only (w/o KL scoring)	86.65	80.80	52.60	65.24	6.48	5.52	67.54

Table 2 reports the corresponding results for the 8B model. At 25% pruning, trajectory-aware selection with OPD outperforms both ShortOPD and BI with OPD on all six benchmarks, achieving an overall score of 58.60 versus 41.00 for BI. At 30% pruning, it also leads across the benchmarks, outperforming both SFT recovery and BI-based pruning. Although performance relative to the unpruned model varies across tasks, these results consistently favor trajectory-aware selection over BI across model sizes and pruning rates under the same OPD recovery setting.

4.3 Layer-Selection Patterns

Figure 2 compares the layer-selection patterns at a 25% pruning rate. Across both models, BI mainly removes consecutive layers near the end of the model, whereas TA removes a combination of early and late layers. For Qwen3-4B-Instruct, BI removes layers 25–33 consecutively, while TA retains layers 28–30 and 33 by pruning early layers 2, 3, 4, and 7 instead. These patterns reflect different judgments of layer redundancy. BI treats small changes between a layer's input and output representations as evidence of low importance, but such changes do not necessarily imply a small behavioral effect when the layer is removed.

The difference becomes more pronounced at 50% pruning, as shown in Figure 3. BI's removals extend further into the middle of the model, including consecutive layers 15–21. In contrast, TA removes more early layers and retains 10 of the 12 layers in the middle interval 12–23, compared with only five retained by BI. This pattern is consistent with the hypothesis that some middle layers play an important role in maintaining the model's reasoning behavior. The retention of these middle layers suggests their greater relative importance under the trajectory-based criterion.

These differences in layer selection are accompanied by better performance after the same OPD recovery, as reported in Table 1. For the Qwen3-4B-Instruct model, TA achieves an overall score of 68.35 at 25% pruning, compared with 56.85 for BI. At 50% pruning, the gap increases, with TA reaching 30.63 versus 7.19 for BI. Together, these results show that post-recovery performance depends critically on which layers are removed, not solely on specialized OPD recovery methods. Although this comparison does not isolate middle-layer retention as the cause of the improvement, it supports selecting layers by their effect on predictions rather than representation similarity alone.

Additionally, Figure 4 shows the per-layer scores assigned to Qwen3-4B-Instruct by the two selection criteria. The differing score profiles explain why the criteria select different layers at the same pruning budget.

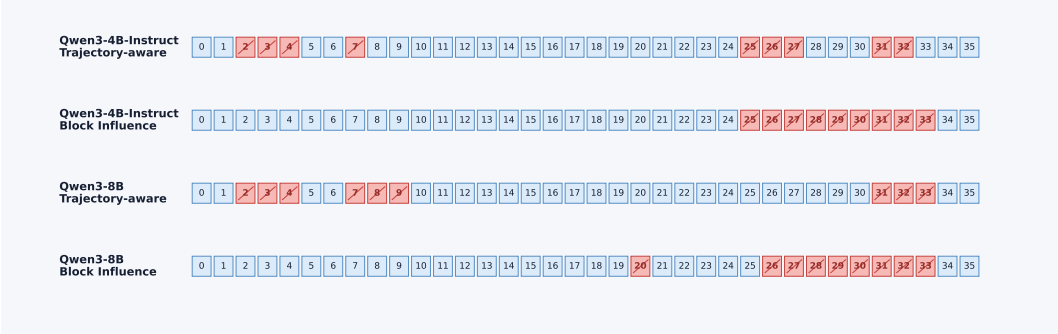

Figure 2: Layer-selection patterns at a 25% pruning rate. TA denotes our trajectory-aware layer-selection criterion, and BI denotes Block Influence.

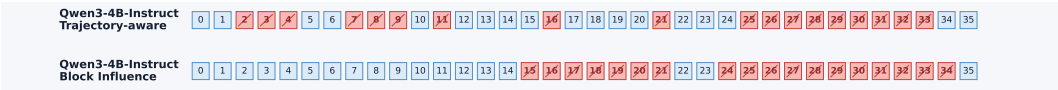

Figure 3: Layer-selection patterns for Qwen3-4B-Instruct at a 50% pruning rate. TA denotes our trajectory-aware layer-selection criterion, and BI denotes Block Influence.

4.4 Ablation Studies

We next examine the design choices inside trajectory-aware layer selection. Specifically, we ablate the scoring components in the Stage-1 score defined by Eqs. equation 7–equation 8, while keeping the pruning budget, calibration trajectories, OPD recovery objective, and evaluation protocol fixed. Thus, the results isolate how the scoring signal changes the layers presented to the same recovery procedure. We report the six task scores used in the main experiments and compute *Overall* by converting the two 1–10 judge scores to percentages before averaging all six values.

4.4.1 Contributions of KL and Region-Aware NLL

Our default score combines the trajectory-level KL divergence term with a region-aware NLL term. In the final importance score of Eq. equation 8, we use equal top-level coefficients $(\alpha, \beta) = (1, 1)$; in the region-aware NLL term of Eq. equation 7, we use balanced reasoning/answer weights $(w_R, w_A) = (1, 1)$. To separate their roles, we compare this configuration with KL-only scoring (*w/o NLL scoring*) and RA-NLL-only scoring (*w/o KL scoring*). The latter names refer to the layer-selection score only; in particular, they do not remove the KL term from the subsequent OPD objective.

Table 3 shows that the KL and region-aware NLL terms provide complementary signals for layer selection. Using both terms achieves the best overall score of 68.35. The combined score performs particularly well on GSM8K, MBPP, HumanEval, and MT-Bench, while KL-only scoring obtains the best results on MATH-500 and Alpaca. One possible explanation is that these tasks benefit more from preserving the broader output distribution: MATH-500 allows diverse intermediate reasoning expressions, while Alpaca is open-ended and has less distinct reasoning/answer regions. In such cases, KL may better capture the behavioral effect of layer removal, whereas RA-NLL provides a more targeted signal on specific reasoning and answer tokens. Neither component consistently dominates across tasks, and their combination yields a more balanced criterion across reasoning, code generation, and open-ended generation.

4.4.2 Sensitivity to Reasoning–Answer Weights

We next fix $(\alpha, \beta) = (1, 1)$ and vary only the relative region weights inside RA-NLL. Because the term is normalized by $w_R + w_A$, these settings change the reasoning/answer share rather than the overall NLL scale. The balanced setting is compared with reasoning-heavy, answer-heavy, answer-focused, and reasoning-focused variants.

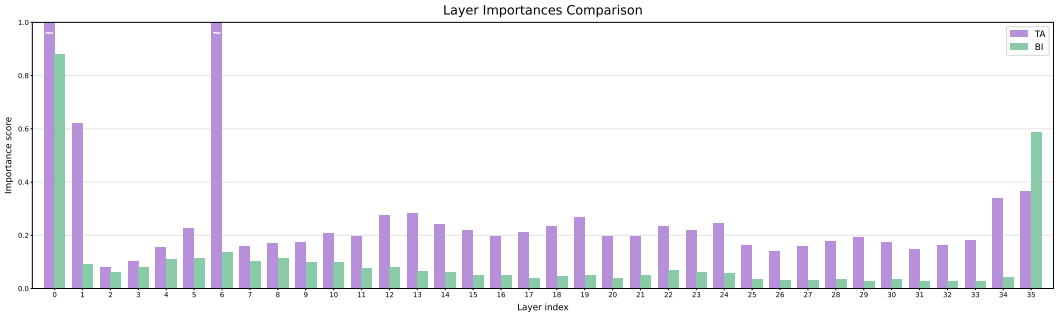

Figure 4: Per-layer importance scores for Qwen3-4B-Instruct under the BI and TA layer-selection criteria.

Table 4: Sensitivity to reasoning–answer weights in RA-NLL for Qwen3-4B-Instruct at 25% pruning. The pair in parentheses is (w_R, w_A); all rows use the same KL coefficient and OPD recovery.

Configuration	GSM8K	MATH-500	MBPP	HumanEval	Alpaca	MT-Bench	Overall
Balanced (1,1)	87.33	81.00	**53.60**	**65.85**	6.44	**5.79**	**68.35**
Reasoning-heavy (2,1)	87.49	79.00	52.40	64.63	6.49	5.46	67.17
Answer-heavy (1,2)	86.80	80.00	50.20	64.02	6.50	5.47	66.79
Answer-focused (0,1)	87.11	**82.60**	53.20	64.02	6.48	5.32	67.48
Reasoning-focused (1,0)	**87.79**	81.00	51.40	62.80	**6.78**	5.42	67.49

Table 4 shows that the relative weighting of reasoning and answer tokens affects layer selection differently across tasks. The balanced setting $(w_R, w_A) = (1, 1)$ achieves the best overall score suggesting that considering both regions provides a more robust estimate of a layer's behavioral importance. In contrast, emphasizing a single region can benefit specific tasks: answer-focused scoring performs best on MATH-500, where correctness is strongly determined by the final answer tokens, while reasoning-focused scoring achieves the best results on GSM8K and Alpaca. This is likely because GSM8K relies heavily on multi-step intermediate reasoning, while Alpaca-style open-ended generation has no sharply localized final-answer span, making the broader generation trajectory a more informative signal of behavioral importance. Overall, no single region consistently dominates across tasks, while balanced weighting provides the most robust performance across benchmarks.

5 Conclusion

We presented *trajectory-aware pruning and recovery*, a preservation-first framework for structured layer pruning of reasoning language models. Rather than estimating layer redundancy from hidden-state similarity, our method directly measures the behavioral effect of layer removal along fixed teacher-generated trajectories, allowing layers to be selected according to their impact on autoregressive reasoning. The resulting pruned model is then refined with on-policy distillation to correct the residual distribution shift on student-visited contexts. Experiments across mathematical reasoning, code generation, and open-ended generation, with different model sizes and pruning ratios, consistently show that trajectory-aware layer selection provides a stronger initialization for subsequent recovery than representation-based pruning. Our ablations further show that distribution-level KL and region-aware token degradation provide complementary signals for identifying behaviorally important layers. Overall, these results highlight that effective compression of reasoning models depends not only on how the pruned model is recovered, but also critically on preserving the right computation before recovery begins.

AI use statement

We used a large language model (LLM) solely to polish and improve the clarity, grammar, and overall presentation of the manuscript. The LLM was not used to generate the research ideas, conduct the

experiments, analyze the results, or draw the scientific conclusions. All final content was reviewed and verified by the authors, who take full responsibility for the final manuscript.

ETHICS STATEMENT

This work does not involve human participants, personal data, or other ethically sensitive materials. We therefore identify no ethical concerns associated with this research.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide the pruning parameters in the appendix, together with the training settings for OPD and a detailed description of the composition of the training dataset. These details enable researchers to reproduce our experimental setup and results.

REFERENCES

Wasi Uddin Ahmad, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Vahid Noroozi, Somshubra Majumdar, and Boris Ginsburg. Opencodeinstruct: A large-scale instruction tuning dataset for code llms. 2025. URL https://arxiv.org/abs/2504.04030.

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 10865–10873, 2024.

Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*, 2024.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

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, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

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

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations, 2023.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. Shortened llama: Depth pruning for large language models with comparison of retraining methods. *arXiv preprint arXiv:2402.02834*, 2024.

Siqi Li, Jingyang Xiang, Jiateng Wei, Chengrui Zhu, Jiandang Yang, Jun Chen, Jian Yang, Xiaobin Wei, Yunliang Jiang, and Yong Liu. Wlr: Well-conditioned linear reconstruction for retraining-free pruning of llms. *Neural Networks*, pp. 108840, 2026.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Ryan Lucas, Kayhan Behdin, Zhipeng Wang, Qingquan Song, Rahul Mazumder, et al. Reasoning models can be accurately pruned via chain-of-thought reconstruction. In *International Conference on Learning Representations*, volume 2026, pp. 142840–142852, 2026.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20192–20204, 2025.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

Jiateng Wei, Quan Lu, Ning Jiang, Siqi Li, Jingyang Xiang, Jun Chen, and Yong Liu. Structured optimal brain pruning for large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 13991–14007, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.775. URL https://aclanthology.org/2024.emnlp-main.775/.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared llama: Accelerating language model pre-training via structured pruning. In *International Conference on Learning Representations*, volume 2024, pp. 5385–5409, 2024.

Qingyu Zhang, Qianhao Yuan, Hongyu Lin, Yaojie Lu, Xianpei Han, Le Sun, Xiang Li, Ming Xu, Jiarui Li, and Xiuyin Zhao. Shortopd: Recovering pruned llms with short-to-long on-policy distillation. *arXiv preprint arXiv:2607.13124*, 2026.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.