LOOP: RECOVERY-ALIGNED ITERATIVE LAYER PRUNING VIA ON-POLICY DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Layer-level pruning is attractive for large language model (LLM) deployment as it reduces the model size while preserving a dense architecture that can be efficiently executed on standard hardware. Current methods typically follow a one-shot paradigm, which first removes selected layers to reach a target model size and then performs additional training to recover model capabilities. However, pruning directly to the target size can substantially degrade reasoning capabilities, making subsequent recovery more difficult under a limited training budget. Moreover, this paradigm separates layer selection from recovery, limiting the use of recovery information to guide selection and potentially hindering the recovery of reasoning capabilities. To address these limitations, we propose LOOP, a **L**ayer-level Recovery-**O**riented **O**n-Policy **P**runing framework that interleaves pruning and recovery across multiple shots. In each shot, LOOP removes a single layer and performs short on-policy distillation, mitigating the abrupt degradation in reasoning capabilities associated with one-shot pruning. To guide layer selection in subsequent shots, LOOP reuses on-policy trajectories and optimization signals from recovery to jointly assess each remaining layer's reasoning contribution and recovery sensitivity, thereby making layer selection recovery-aware. Evaluations on several benchmarks show that LOOP consistently outperforms one-shot baselines in average accuracy across different model scales and pruning ratios.

1 INTRODUCTION

Large language models (LLMs) (Kaplan et al., 2020; OpenAI, 2023; 2026) have achieved remarkable capabilities through scaling. However, their increasing memory consumption and computational cost create substantial challenges for efficient deployment. These deployment challenges have motivated extensive research into model compression, including quantization (Yan et al., 2026), knowledge distillation (Shin et al., 2026), low-rank decomposition (Zhou et al., 2026), and pruning (Yang et al., 2026). These approaches reduce memory and computational requirements from different perspectives.

Among these approaches, layer-level pruning removes entire Transformer layers, producing a smaller, shallower model while preserving the remaining network's dense structure (Zhu et al., 2024). The resulting model can be efficiently executed on standard hardware using conventional dense computation libraries (Kim et al., 2024; Yang et al., 2024). Together, these properties make layer-level pruning a practical approach for reducing inference cost and memory usage while requiring only minimal changes to the existing model architecture and deployment stack.

Current layer-level pruning methods usually follow a one-shot paradigm that separates compression into two sequential stages: pruning and recovery. In this paradigm, layers are typically selected based on importance scores estimated on a calibration prompt set, for example, using changes in representations (Men et al., 2025). The selected layers are then removed together to reach the target model size. In the recovery stage, the pruned model undergoes additional training to recover reasoning performance (Muralidharan et al., 2024). Despite its practical advantages, layer-level pruning can introduce errors that propagate through subsequent steps of multi-step reasoning. The one-shot paradigm poses two key challenges to preserving such reasoning capabilities. First, pruning directly to the target size can substantially degrade reasoning performance and induce a large distributional shift in model predictions. As a result, subsequent recovery starts from a more degraded state, making it harder to restore the lost reasoning capabilities. Second, one-shot pruning finalizes layer selection

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.

before recovery begins. As a result, the importance scores used for pruning may not fully reflect each layer's role in the recovery process. In particular, scores computed on calibration prompts may fail to capture a layer's contribution along the complete on-policy trajectories encountered during recovery training. Moreover, criteria that measure only the immediate effect of removing a layer do not directly reflect its influence on the recovery objective. Such criteria may therefore overlook layers that are important for effective recovery. These limitations highlight the need to limit reasoning degradation during pruning while accounting for each layer's role in subsequent recovery.

To address these limitations, we propose **LOOP**, a **L**ayer-level Recovery-**O**riented **O**n-Policy **P**runing framework that interleaves pruning and recovery over multiple shots to preserve reasoning capabilities during compression. First, to mitigate the severe capability degradation caused by pruning directly to the target size, LOOP adopts an iterative pruning-and-recovery process that distributes layer removal across multiple shots. In each shot, LOOP removes a single layer and performs a short phase of on-policy recovery. The recovered model serves as the starting point for the next shot, where the remaining layers are re-evaluated before the next pruning decision. This process continues until the target model size is reached. By allowing the model to adapt after each layer removal, LOOP mitigates the abrupt degradation in reasoning capabilities associated with one-shot pruning and supports recovery throughout compression. Second, to better reflect each layer's role in subsequent recovery, LOOP combines two complementary importance signals: reasoning contribution and recovery sensitivity. Reasoning contribution captures a layer's importance to the reasoning process, while recovery sensitivity reflects its influence on the recovery objective. For reasoning contribution, LOOP measures the representational changes induced by each remaining layer. The measurement is performed over complete on-policy trajectories generated by the current model during recovery. For recovery sensitivity, LOOP uses gradient information produced during recovery training. The gradients are used to estimate how strongly the recovery loss depends on each remaining layer, thereby reflecting its influence on the recovery objective. These two signals are combined into a recovery-oriented importance score, which is updated after each recovery stage to guide layer selection. Together, these designs enable progressive pruning and recovery while incorporating both reasoning contribution and recovery sensitivity into layer selection.

Extensive experiments on mathematical and scientific reasoning benchmarks demonstrate the effectiveness of LOOP across different pruning ratios and model scales. On Qwen3-4B-Instruct, LOOP outperforms the strongest evaluated one-shot baseline by 7.37, 9.10, and 6.34 percentage points in average mathematical reasoning accuracy at pruning ratios of 15%, 20%, and 25%, respectively. At a 20% layer pruning ratio, LOOP consistently outperforms the baselines on Qwen3 models ranging from 1.7B to 8B parameters. LOOP also achieves higher average accuracy than the baseline on scientific reasoning benchmarks across all evaluated pruning ratios, suggesting that the effectiveness of LOOP extends beyond mathematical reasoning. Further analyses show that both iterative pruning and recovery-oriented layer selection contribute to the performance gains of LOOP.

2 Related Work

2.1 Model Compression and Structured Pruning

The rapid scaling of large language models (OpenAI, 2026; Yang et al., 2025) has led to substantial memory footprints and computational demands, making efficient deployment increasingly challenging. This challenge has motivated a broad range of model compression techniques, including quantization (Park et al., 2022; Liu et al., 2025), knowledge distillation (Jiang et al., 2023), low-rank decomposition (Wang et al., 2025b;c), and pruning (Ma et al., 2023; Muralidharan et al., 2024). These techniques reduce deployment overhead through different mechanisms: quantization lowers numerical precision, knowledge distillation transfers model capabilities to a compact student, and low-rank decomposition replaces large parameter matrices with compact factors. Pruning, in contrast, reduces model size by removing redundant weights or architectural components from a pretrained network. Existing pruning methods are commonly categorized as unstructured or structured according to the structure of the sparsity they introduce. Unstructured pruning sets individual weights to zero using criteria such as weight–activation statistics or local loss sensitivity (Frantar & Alistarh, 2023; Sun et al., 2024), while preserving the original tensor shapes. However, translating irregular sparsity into memory savings typically requires sparse storage formats, while inference speedups depend on efficient sparse kernels and suitable hardware support (Liu et al., 2023).

In contrast, structured pruning removes entire architectural components, such as attention heads, feed-forward neurons, hidden dimensions, or Transformer layers, producing smaller dense networks that are compatible with standard computation libraries. Among these granularities, layer-level pruning reduces model depth while preserving the dense structure of the remaining network, making it attractive for efficient deployment (Kim et al., 2024). Existing structured pruning methods can be further categorized according to how they assess the importance of candidate structures (Zhu et al., 2024). Loss-based methods estimate structural importance using the loss or its derivatives. For example, LLM-Pruner (Ma et al., 2023) uses gradient information to identify and remove less important coupled structures. At the layer level, Shortened LLaMA (Kim et al., 2024) removes entire Transformer layers and compares LoRA-based fine-tuning with continued pretraining for post-pruning recovery, highlighting the importance of recovery training at high pruning ratios. Other methods use activation or representation statistics to guide compression. FLAP (An et al., 2023) uses input-feature fluctuations to assess the importance of weight-matrix columns and compensates for their removal through bias adjustments. SliceGPT (Ashkboos et al., 2024) exploits computational invariance to rotate representations into principal-component bases before removing dimensions by deleting corresponding rows and columns of weight matrices. At the layer level, ShortGPT (Men et al., 2025) measures representational changes between layer inputs and outputs using Block Influence (BI) and removes layers with low scores computed on calibration data. Despite differences in their importance criteria, methods that finalize structural selection before recovery cannot use subsequent recovery signals to revise those decisions. This separation leaves room for approaches that update layer importance during recovery, allowing subsequent pruning decisions to account for both the model's evolving behavior and the recovery objective.

2.2 Post-Pruning Recovery and On-Policy Distillation

Since structural removal often degrades model performance, post-pruning recovery is commonly performed through supervised fine-tuning (Ma et al., 2023), continued training (Kim et al., 2024), or knowledge distillation (Wang et al., 2025a). Knowledge distillation transfers teacher knowledge to the compressed student through token-level output distributions or teacher-generated responses. The former provides supervision over alternative next tokens, while the latter supplies sequences that the student learns to reproduce (Kim & Rush, 2016). Conventional off-policy distillation trains on fixed reference or teacher-generated sequences, whose prefixes may differ from those encountered when the student generates responses at inference time. On-policy distillation addresses this mismatch by sampling trajectories from the current student and obtaining teacher feedback at the prefixes visited along these trajectories (Agarwal et al., 2024). As the student is updated, newly generated trajectories allow supervision to follow its evolving behavior. Recent work has applied on-policy distillation to post-pruning recovery. ShortOPD (Zhang et al., 2026) recovers one-shot depth-pruned models using on-policy distillation with a short-to-long rollout schedule. However, layer selection is completed before recovery, preventing recovery information from informing subsequent structural decisions. This separation makes it difficult to adapt layer selection to the model's behavior during recovery.

3 Preliminaries

Let $\mathcal{M}_{pre}$ denote a pretrained language model with L Transformer layers indexed by $\mathcal{I} = \{1, \ldots, L\}$, where $\mathcal{B}_l$ denotes the transformation of layer l, and $\mathcal{E}$ and $\mathcal{H}$ denote the input embedding and output mapping, respectively. The model can be expressed as

$$\mathcal{M}_{pre} = \mathcal{H} \circ \mathcal{B}_L \circ \cdots \circ \mathcal{B}_1 \circ \mathcal{E}, \tag{1}$$

where $\circ$ denotes function composition. Given a target layer pruning ratio ρ, layer-level pruning removes $K = \lceil \rho L \rceil$ layers, where $1 \leq K < L$, and retains $L - K$ layers indexed by the retained set $\mathcal{I}_{\text{keep}} = \{i_1, \ldots, i_{L-K}\} \subseteq \mathcal{I}$, with $i_1 < \cdots < i_{L-K}$. The resulting pruned model is

$$\mathcal{M}_{pruned} = \mathcal{H} \circ \mathcal{B}_{i_{L-K}} \circ \cdots \circ \mathcal{B}_{i_1} \circ \mathcal{E}. \tag{2}$$

The retained layers preserve their original order and are initialized with their pretrained parameters. Recovery training adapts the retained model parameters to mitigate pruning-induced degradation in reasoning performance. The task is to select the retained layers and adapt the model within a prescribed recovery budget, minimizing the degradation in reasoning performance relative to the pretrained model.

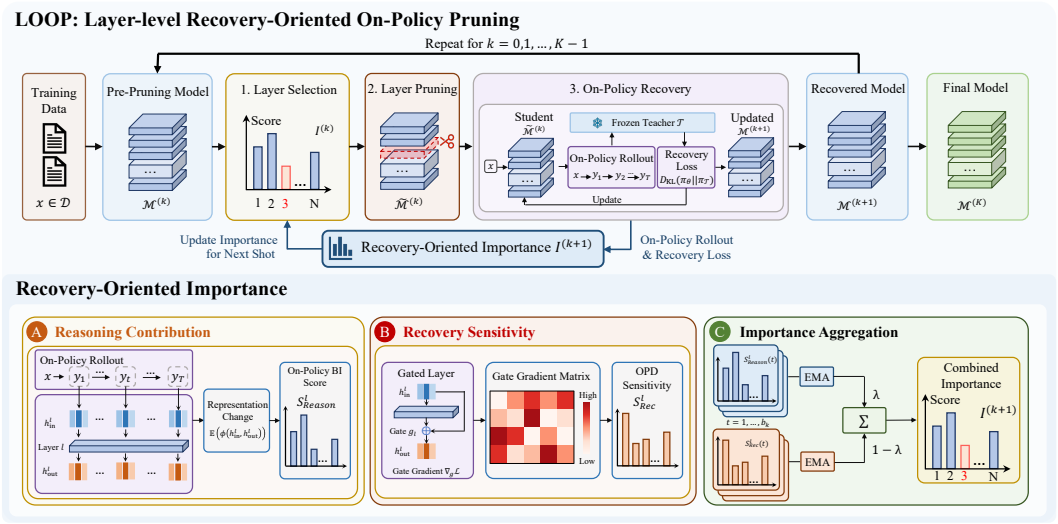

Figure 1: **Overview of LOOP.** The top shows the iterative pruning and recovery process, where each shot removes one layer, performs short on-policy recovery, and updates layer importance for the next shot. The bottom details the recovery-oriented importance criterion, which estimates reasoning contribution and recovery sensitivity, aggregates and combines them into an importance score.

4 METHOD

In this section, we introduce LOOP, a layer-level recovery-oriented on-policy pruning framework. As illustrated in Figure 1, LOOP iteratively removes one layer at a time and performs short on-policy recovery through OPD before the next pruning step, allowing the model to adapt progressively throughout compression. During recovery, LOOP reuses on-policy trajectories and recovery signals to re-evaluate the remaining layers, coupling subsequent layer selection with the recovery process.

4.1 RECOVERY-ORIENTED LAYER IMPORTANCE

Selecting a layer for removal requires considering not only its contribution to the current model's reasoning behavior, but also its relevance to the recovery objective. LOOP captures these two aspects using complementary signals collected during on-policy recovery. Specifically, reasoning contribution measures the representational change induced by a layer along on-policy reasoning trajectories, while recovery sensitivity measures the local sensitivity of the recovery objective to changes in the layer's contribution. The two signals are aggregated over recovery updates and combined into a recovery-oriented importance score to guide subsequent layer selection.

Reasoning Contribution. Reasoning contribution captures how important each layer is to the current student's reasoning process. LOOP estimates it using Block Influence (BI), which measures the change between a Transformer layer's input and output representations (Men et al., 2025). Whereas conventional BI is typically evaluated on a fixed calibration set, LOOP computes it along complete on-policy trajectories generated by the current student during recovery. This makes the score more representative of the current student's reasoning behavior. Specifically, consider the recovery stage of shot k, which consists of b_k recovery updates. At update u, let p_θ denote the current student policy with parameters θ. For a prompt $\mathbf{x} \sim \mathcal{D}$, where $\mathcal{D}$ is the training prompt distribution, the student generates a response

$$\mathbf{y} = (y_1, \ldots, y_T) \sim p_\theta(\cdot \mid \mathbf{x}), \tag{3}$$

where T is the response length. The prompt and response are concatenated into a complete trajectory $\boldsymbol{\tau} = [\mathbf{x}; \mathbf{y}]$. For each remaining layer l at update u, let $\mathbf{h}^l_{\text{in}}(\boldsymbol{\tau}; u, t)$ and $\mathbf{h}^l_{\text{out}}(\boldsymbol{\tau}; u, t)$ denote its input and output hidden states at token position t in trajectory $\boldsymbol{\tau}$, respectively. The reasoning contribution score at update u is defined as:

$$s^l_{\text{Reason}}(u) = \mathbb{E}_{\boldsymbol{\tau},t}\left[1 - \cos\left(\mathbf{h}^l_{\text{in}}(\boldsymbol{\tau}; u, t), \mathbf{h}^l_{\text{out}}(\boldsymbol{\tau}; u, t)\right)\right], \tag{4}$$

where the expectation denotes an average over all valid tokens in the trajectory batch collected at update u, including both prompt and response tokens. Generated reasoning traces are typically longer

than prompts in the collected trajectories and thus receive greater aggregate weight. To reduce fluctuations from stochastic rollouts and the influence of isolated score spikes, the reasoning contribution scores are aggregated within each shot using an exponential moving average (EMA, Tarvainen & Valpola (2017)):

$$\bar{s}^l_{\mathrm{Reason}}(u) = \beta \bar{s}^l_{\mathrm{Reason}}(u-1) + (1-\beta) s^l_{\mathrm{Reason}}(u), \qquad u = 2, \ldots, b_k, \tag{5}$$

where $\beta \in [0, 1)$ is the smoothing coefficient. The EMA is initialized as $\bar{s}^l_{\mathrm{Reason}}(1) = s^l_{\mathrm{Reason}}(1)$ at the first recovery update of each shot. After b_k steps of updates in shot k, the aggregated score for subsequent layer selection is

$$S^l_{\mathrm{Reason}}(k+1) = \bar{s}^l_{\mathrm{Reason}}(b_k). \tag{6}$$

A smaller aggregated score indicates less representational change along the sampled on-policy trajectories under the EMA weighting, suggesting a lower reasoning contribution as measured by BI.

Recovery Sensitivity. Recovery sensitivity (RS) assesses each layer's influence on the recovery objective by measuring the sensitivity of the recovery loss to layer gates. LOOP estimates RS using OPD Sensitivity, based on gate gradients of the on-policy distillation loss. During recovery, the student receives supervision from a frozen teacher $\mathcal{T}$, initialized as a copy of $\mathcal{M}_{pre}$, through on-policy distillation (Agarwal et al., 2024). For each response sampled according to Equation (3), the teacher additionally conditions on the associated ground-truth answer $a^\star$ and provides a next-token distribution at each prefix $(\mathbf{x}, \mathbf{y}_{<t})$. These distributions supervise the student through the following expected token-averaged reverse KL objective:

$$\mathcal{L}_{\mathrm{Rec}}(\theta) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D},\, \mathbf{y}\sim p_\theta(\cdot|\mathbf{x})} \left[\frac{1}{T} \sum_{t=1}^{T} D_{\mathrm{KL}} \left(p_\theta(\cdot \mid \mathbf{x}, \mathbf{y}_{<t}) \,\|\, p_{\mathcal{T}}(\cdot \mid \mathbf{x}, a^\star, \mathbf{y}_{<t}) \right) \right]. \tag{7}$$

To measure recovery sensitivity, a scalar gate g_l is introduced for each remaining layer l. For an input representation $\mathbf{h}^l_{\mathrm{in}}$, the gated output is

$$\widetilde{\mathbf{h}}^l_{\mathrm{out}} = \mathbf{h}^l_{\mathrm{in}} + g_l \left[\mathcal{B}_l(\mathbf{h}^l_{\mathrm{in}}) - \mathbf{h}^l_{\mathrm{in}} \right]. \tag{8}$$

Setting $g_l = 1$ reproduces the layer output, whereas $g_l = 0$ bypasses the layer. During OPD, all gates remain at 1 and participate in gradient computation but are excluded from optimizer updates. Let $\mathcal{L}_{\mathrm{Rec}}(u)$ denote the recovery loss on the trajectory batch at update u. The recovery sensitivity score of each layer is defined as

$$s^l_{\mathrm{Rec}}(u) = \left(\frac{\partial \mathcal{L}_{\mathrm{Rec}}(u)}{\partial g_l} \right)^2, \tag{9}$$

where the sampled trajectories are held fixed when computing the gate gradients. Squaring the gradient yields a nonnegative measure of local sensitivity, irrespective of the gradient's sign. We use this quantity to measure the local dependence of the recovery objective on each layer. The recovery sensitivity scores are aggregated within each shot using the same EMA rule as in Equation (5). After b_k recovery updates in shot k, the aggregated score is

$$S^l_{\mathrm{Rec}}(k+1) = \bar{s}^l_{\mathrm{Rec}}(b_k). \tag{10}$$

A larger aggregated score indicates greater local sensitivity of the recovery objective to changes in the layer's contribution, providing a complementary signal for subsequent layer selection.

Combined Importance. The aggregated reasoning contribution and recovery sensitivity scores from shot k are combined to guide layer selection in the next shot. To account for differences in their numerical scales, the two sets of EMA-aggregated scores are independently normalized across the remaining layers. For $q \in \{\mathrm{Reason}, \mathrm{Rec}\}$, the normalized score is

$$\widehat{S}^l_q(k+1) = \frac{S^l_q(k+1) - \min_j S^j_q(k+1)}{\max_j S^j_q(k+1) - \min_j S^j_q(k+1) + \epsilon}, \tag{11}$$

where the minimum and maximum are taken over all remaining layers $j \in \mathcal{I}^{(k+1)}$, and $\epsilon > 0$ is a small constant for numerical stability. The recovery-oriented importance score is then defined as

$$I^{(k+1)}_l = \lambda \widehat{S}^l_{\mathrm{Reason}}(k+1) + (1-\lambda) \widehat{S}^l_{\mathrm{Rec}}(k+1), \tag{12}$$

where $\lambda \in [0, 1]$ balances reasoning contribution and recovery sensitivity. A lower $I^{(k+1)}_l$ gives layer l higher priority for removal in the next shot.

4.2 Iterative Pruning and Recovery

To mitigate the degradation caused by pruning directly to the target size and to continuously align layer selection with the evolving pruned model, LOOP adopts an iterative pruning and recovery procedure. The procedure consists of three stages: warm-up, iterative pruning and recovery, and final recovery. Together, these stages allow recovery information to guide layer selection throughout compression while further adapting the target-size model after pruning. The complete procedure is summarized in Appendix B.

Warm-Up. Before the first pruning shot, LOOP performs b_{w} recovery updates on $\mathcal{M}_{pre}$ without removing any layers. During these updates, on-policy trajectories and gradients of the recovery loss with respect to layer gates are used to compute reasoning contribution and recovery sensitivity, respectively. Both scores are aggregated using EMA, initialized from the first warm-up update. After b_{w} updates, the resulting model is denoted as $\mathcal{M}^{(0)}$. The aggregated scores are then normalized and combined according to Equations (11) and (12) to obtain $I_l^{(0)}$ for the first layer selection.

Iterative Pruning and Recovery. At the beginning of shot $k \in \{0, \ldots, K-1\}$, LOOP selects a remaining layer with the lowest combined importance:

$$l_k^* \in \arg\min_{l \in \mathcal{I}^{(k)}} I_l^{(k)}. \tag{13}$$

The selected layer is removed from the current student, yielding

$$\widetilde{\mathcal{M}}^{(k)} = \mathrm{Prune}\left(\mathcal{M}^{(k)}, l_k^*\right), \qquad \mathcal{I}^{(k+1)} = \mathcal{I}^{(k)} \setminus \{l_k^*\}. \tag{14}$$

The pruned student then undergoes b_k recovery updates:

$$\mathcal{M}^{(k+1)} = \mathrm{Rec}\left(\widetilde{\mathcal{M}}^{(k)}, \mathcal{T}; b_k\right). \tag{15}$$

Here, $\mathrm{Rec}(\mathcal{M}, \mathcal{T}; b)$ denotes b updates to student $\mathcal{M}$ using the recovery objective in Equation (7) and the frozen teacher $\mathcal{T}$. During these updates, on-policy trajectories and gradients of the recovery loss with respect to layer gates are reused to compute reasoning contribution and recovery sensitivity for the remaining layers. The EMA statistics are initialized at the first recovery update of each shot and updated over the remaining recovery steps within each shot. The aggregated scores are normalized and combined to obtain $I_l^{(k+1)}$ for the layer selection in the next shot. This reuse of recovery information avoids collecting a separate set of trajectories for importance estimation. Each shot therefore removes a single layer, performs recovery, and updates layer importance before the next selection. After K shots, the resulting model $\mathcal{M}^{(K)}$ contains $L - K$ layers.

Final Recovery. After K layers have been removed, LOOP performs b_{f} additional recovery updates without further changes to the model architecture:

$$\mathcal{M}_{\mathrm{final}} = \mathrm{Rec}\left(\mathcal{M}^{(K)}, \mathcal{T}; b_{\mathrm{f}}\right). \tag{16}$$

The total recovery budget B, measured in recovery updates, includes warm-up, recovery within each shot, and final recovery:

$$b_{\mathrm{w}} + \sum_{k=0}^{K-1} b_k + b_{\mathrm{f}} = B. \tag{17}$$

Together, the three stages allow recovery information to guide layer selection throughout pruning and further adapt the target-size model within the prescribed recovery budget.

5 Experiments

We evaluate LOOP on mathematical and scientific reasoning benchmarks across multiple model scales and layer pruning ratios. Controlled comparisons with one-shot pruning under matched recovery update budgets assess the effects of iterative pruning and recovery-oriented layer selection. Ablation studies and additional analyses examine the contributions of individual components and the efficiency of the pruning–recovery process.

Table 1: Performance of Qwen3-4B-Instruct under different layer-pruning ratios and recovery strategies on mathematical reasoning benchmarks. All results are measured by accuracy. *Mean* denotes the average accuracy across the four benchmarks, and T denotes the compressed model's mean performance relative to that of the original model.

Ratio	Method	Recovery	Metric	DAPO	Math-500	AMC	AIME	Mean	T (%)
0%	Original	–	–	0.472	0.784	0.650	0.256	0.5405	100.000
15%	Pruned	–	BI	0.032	0.138	0.000	0.000	0.0425	7.865
	SFT w/o KD	SFT	BI	0.143	0.464	0.175	0.033	0.2038	37.719
	KD	Off-Policy	BI	0.124	0.478	0.175	0.011	0.1970	36.454
	OPD	On-Policy	BI	0.332	0.604	0.450	0.122	0.3771	69.769
	LOOP (Ours)	On-Policy	BI + RS	**0.348**	**0.658**	**0.575**	**0.222**	**0.4508**	**83.404**
20%	Pruned	–	BI	0.004	0.014	0.000	0.000	0.0045	0.833
	SFT w/o KD	SFT	BI	0.072	0.312	0.050	0.011	0.1113	20.592
	KD	Off-Policy	BI	0.052	0.338	0.075	0.000	0.1163	21.512
	OPD	On-Policy	BI	0.139	0.468	0.325	0.011	0.2358	43.626
	LOOP (Ours)	On-Policy	BI + RS	**0.238**	**0.522**	**0.425**	**0.122**	**0.3268**	**60.463**
25%	Pruned	–	BI	0.001	0.006	0.000	0.000	0.0018	0.324
	SFT w/o KD	SFT	BI	0.023	0.190	0.050	0.000	0.0658	12.167
	KD	Off-Policy	BI	0.029	0.202	0.050	0.000	0.0703	13.000
	OPD	On-Policy	BI	0.175	0.386	0.125	0.044	0.1826	33.792
	LOOP (Ours)	On-Policy	BI + RS	**0.192**	**0.470**	**0.200**	**0.122**	**0.2460**	**45.522**

5.1 Experimental Setup

Models and Datasets. We evaluate LOOP on three models from the Qwen3 family (Yang et al., 2025): Qwen3-1.7B, Qwen3-4B-Instruct, and Qwen3-8B. Qwen3-4B-Instruct serves as the primary model for comparisons at target layer pruning ratios of 15%, 20%, and 25%. To assess performance across model scales, all three models are evaluated at a target layer pruning ratio of 20%. For mathematical reasoning, models are trained on DAPO-Math-17K (Yu et al., 2025) and evaluated on a held-out subset sampled from DAPO-Math, along with MATH-500 (Hendrycks et al., 2021), AMC, and AIME. For scientific reasoning, we use SciKnowEval (Feng et al., 2025) for both training and evaluation. The training set combines examples from its Biology, Chemistry, Physics, and Materials domains, while evaluation is conducted separately on the corresponding test sets. Accuracy is used as the evaluation metric for all benchmarks.

Baselines. LOOP is compared with baselines using the same pretrained model, training prompts, target layer pruning ratio, and evaluation protocol. All recovery methods share the same total update budget and common optimization hyperparameters, except for strategy-specific settings. **Pruned** removes layers selected by BI (Men et al., 2025) in a one-shot manner without subsequent recovery. Starting from the same BI-pruned model, **SFT w/o KD** performs supervised fine-tuning on fixed ground-truth responses, whereas **KD** trains on fixed teacher-generated responses. **OPD** uses the same BI-pruned initialization but performs on-policy distillation on student-generated responses (Agarwal et al., 2024). This baseline follows the one-shot pruning and OPD recovery pipeline of ShortOPD (Zhang et al., 2026), but uses a fixed maximum rollout length. The unpruned model **Original** serves as a reference.

5.2 Main Experimental Results

Performance under Different Pruning Ratios. To assess the effectiveness of LOOP at different levels of pruning, Table 1 compares mathematical reasoning performance on Qwen3-4B-Instruct at target layer pruning ratios of 15%, 20%, and 25%. Pruning without recovery causes severe performance degradation, demonstrating that directly removing entire Transformer layers substantially disrupts the model's capability. Under matched recovery update budgets, LOOP achieves the highest accuracy among the evaluated methods across all four benchmarks and pruning ratios. Compared with the best baseline, LOOP improves the average accuracy by 7.37, 9.10, and 6.34 percentage points at pruning ratios of 15%, 20%, and 25%, respectively. It also retains 83.40%, 60.46%, and 45.52% of the original model's average performance at the corresponding ratios. These results demonstrate the advantage of LOOP in preserving reasoning performance across different ratios.

Table 2: Performance of different-sized Qwen3 models under a 20% layer-pruning ratio on mathematical reasoning benchmarks. All results are measured by accuracy. The best and second-best results among the reported compression methods are highlighted in **bold** and underlined, respectively.

Model	Method	Recovery	Metric	DAPO	Math-500	AMC	AIME	Mean	T (%)
Qwen3-1.7B	Original	–	–	0.063	0.242	0.150	0.011	0.1165	100.000
	Pruned	–	BI	0.001	0.008	0.000	0.000	0.0023	1.931
	SFT w/o KD	SFT	BI	0.031	0.130	0.000	0.000	0.0403	34.542
	KD	Off-Policy	BI	0.028	0.122	0.025	0.011	0.0465	39.929
	OPD	On-Policy	BI	0.011	0.154	**0.075**	0.000	0.0600	51.491
	LOOP (Ours)	On-Policy	BI + RS	**0.032**	**0.190**	0.050	**0.011**	**0.0708**	**60.717**
Qwen3-4B	Original	–	–	0.472	0.784	0.650	0.256	0.5405	100.000
	Pruned	–	BI	0.004	0.014	0.000	0.000	0.0045	0.833
	SFT w/o KD	SFT	BI	0.072	0.312	0.050	0.011	0.1113	20.592
	KD	Off-Policy	BI	0.052	0.338	0.075	0.000	0.1163	21.512
	OPD	On-Policy	BI	0.139	0.468	0.325	0.011	0.2358	43.626
	LOOP (Ours)	On-Policy	BI + RS	**0.238**	**0.522**	**0.425**	**0.122**	**0.3268**	**60.463**
Qwen3-8B	Original	–	–	0.197	0.414	0.325	0.056	0.2479	100.000
	Pruned	–	BI	0.007	0.010	0.000	0.000	0.0043	1.714
	SFT w/o KD	SFT	BI	0.107	0.352	0.150	0.011	0.1550	62.539
	KD	Off-Policy	BI	0.084	0.260	0.150	0.011	0.1263	50.941
	OPD	On-Policy	BI	0.071	0.338	0.175	0.011	0.1488	60.007
	LOOP (Ours)	On-Policy	BI + RS	**0.108**	**0.440**	**0.350**	**0.156**	**0.2634**	**106.247**

Table 3: Performance of Qwen3-4B-Instruct under different layer-pruning ratios on scientific reasoning benchmarks (Feng et al., 2025). All results are measured by accuracy.

Ratio	Method	Metric	Biology	Chemistry	Materials	Physics	Mean	T (%)
0%	Original	–	0.200	0.138	0.574	0.400	0.3281	100.000
15%	OPD	BI	0.040	0.090	**0.521**	0.163	0.2036	62.040
	LOOP (Ours)	BI + RS	**0.180**	**0.126**	**0.521**	**0.188**	**0.2536**	**77.296**
20%	OPD	BI	0.040	0.090	0.447	**0.158**	0.1837	55.985
	LOOP (Ours)	BI + RS	**0.160**	**0.100**	**0.479**	0.150	**0.2222**	**67.714**
25%	OPD	BI	0.080	**0.111**	0.389	0.138	0.1795	54.704
	LOOP (Ours)	BI + RS	**0.120**	0.052	**0.468**	**0.163**	**0.2007**	**61.180**

Performance across Model Scales. The cross-scale comparison evaluates whether LOOP's effectiveness extends to models of different sizes. Table 2 reports results on Qwen3-1.7B, Qwen3-4B-Instruct, and Qwen3-8B at a target layer pruning ratio of 20%. LOOP achieves the highest average accuracy at each evaluated scale, outperforming the strongest baseline at each scale by 1.08, 9.10, and 10.84 percentage points on Qwen3-1.7B, Qwen3-4B-Instruct, and Qwen3-8B, respectively. These results support the effectiveness of LOOP across model scales.

Scientific Reasoning Performance. Experiments on the Biology, Chemistry, Physics, and Materials domains of SciKnowEval further assess the effectiveness of LOOP beyond mathematical reasoning. As shown in Table 3, LOOP outperforms the OPD baseline in terms of average accuracy across all pruning ratios. Specifically, LOOP improves the average accuracy over the OPD baseline by 5.00, 3.85, and 2.12 percentage points at pruning ratios of 15%, 20%, and 25%, respectively, while retaining 77.30%, 67.71%, and 61.18% of the original model's average performance.

Across the evaluated settings, LOOP consistently achieves higher average accuracy than the compared baselines. These results support the effectiveness of recovery-oriented multi-shot pruning across pruning ratios, model scales, and reasoning domains under the prescribed recovery budget.

5.3 Ablation Study and Analysis

Effect of Individual Components. Table 4 isolates the contributions of iterative pruning and recovery-oriented importance on Qwen3-4B-Instruct at a 20% layer pruning ratio. Starting from the one-shot BI baseline at 23.58% mean accuracy, introducing iterative pruning alone improves accuracy to 30.69%, while incorporating recovery sensitivity into BI under one-shot pruning raises it to 26.74%.

Table 4: Ablation study of LOOP using Qwen3-4B-Instruct at a 20% compression ratio. Combined Importance indicates whether recovery sensitivity is incorporated with BI for layer selection.

Iterative Pruning	Combined Importance	DAPO	Math-500	AMC	AIME	Mean
✓	✓	0.238	**0.522**	**0.425**	**0.122**	**0.3268**
✓	✗	**0.261**	0.514	0.375	0.078	0.3069
✗	✓	0.185	0.482	0.325	0.078	0.2674
✗	✗	0.139	0.468	0.325	0.011	0.2358

Table 5: Comparison of different importance fusion settings for Qwen3-4B-Instruct under a 15% layer-pruning ratio. All results are reported using accuracy.

Strategy	Setting	DAPO	Math-500	AMC	AIME	Mean
Two-stage	BI → RS	0.258	0.510	0.425	0.122	0.329
Weighted	$\lambda = 0.8$	0.283	0.590	0.450	0.133	0.364
	$\lambda = 0.5$	**0.348**	**0.658**	**0.575**	**0.222**	**0.451**
	$\lambda = 0.2$	0.344	0.598	0.550	0.156	0.412
RS only	$\lambda = 0$	0.301	0.600	0.550	0.144	0.399

Table 6: Comparison of the efficiency of LOOP and OPD on Qwen3-4B-Instruct under different pruning ratios.

Method	Time (s)	Output Tokens
LOOP-20%	48.888	**13678.75**
OPD-20%	53.496	15662.13
LOOP-25%	49.878	**14014.75**
OPD-25%	54.648	16990.00

Combining both components yields the highest mean accuracy of 32.68%, exceeding iterative BI by 1.99 percentage points and one-shot pruning with combined importance by 5.94 percentage points. These results show that each component improves performance both individually and when added to the other, supporting their complementary roles in post-pruning recovery.

Importance Fusion Strategy. To examine how reasoning contribution and recovery sensitivity should be combined for layer selection, Table 5 compares fusion strategies on Qwen3-4B-Instruct at a 15% layer pruning ratio. The two-stage strategy first filters candidate layers using BI and then selects a layer using recovery sensitivity. Weighted fusion combines the two normalized scores according to Equation (12), with λ controlling the weight of reasoning contribution. Equal weighting ($\lambda = 0.5$) achieves the highest accuracy on all four benchmarks, with an average accuracy of 0.451, compared with 0.329 for two-stage selection and 0.399 for recovery sensitivity alone. Unequal weighting also yields lower average accuracy, indicating that the relative weighting of the two signals matters. These results favor balanced fusion for layer selection in this setting.

Efficiency Analysis. Table 6 compares LOOP with the one-shot OPD baseline in terms of measured training time and response length. Per-step time is averaged over the full training run and includes the complete wall-clock duration of each training step. At layer pruning ratios of 20% and 25%, the average per-step time decreases from 53.50 to 48.89 seconds and from 54.65 to 49.88 seconds, respectively. Under matched evaluation settings, the final LOOP models also generate responses that are 12.67% and 17.51% shorter, respectively. With the same retained depth at each pruning ratio, shorter responses require fewer autoregressive decoding steps. Together, these results show that LOOP improves final-model performance without increasing average training-step time or response length under the evaluated settings.

6 Conclusion

In this work, we presented **LOOP**, a layer-level recovery-oriented on-policy pruning framework designed to limit reasoning degradation during compression. LOOP alternates single-layer pruning with short on-policy distillation and uses recovery signals to guide subsequent layer selection. Its layer-selection criterion combines reasoning contribution, estimated from representational changes along complete on-policy trajectories, with recovery sensitivity, estimated from squared gradients of the recovery loss with respect to layer gates. Experiments on mathematical and scientific reasoning benchmarks show that LOOP achieves higher average accuracy than the evaluated one-shot baselines under matched recovery update budgets across multiple pruning ratios and model scales. Ablation studies support the contributions of both iterative pruning and recovery-oriented importance. These findings highlight the value of accounting for recovery in layer selection to better preserve reasoning performance under a limited recovery budget.

AI USE STATEMENT

In this work, we used generative AI tools for assist with translation and polishment. We have not used generative AI tools for interpret results, propose or refine hypotheses, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for identify relevant literature. We have reviewed all AI-assisted work. We have ensured the authenticity of the relevant literature provided by AI and the correctness of the citations. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We assert that the methodology and applications of this study do not involve human subjects, generate harmful insights, or engage in applications that could lead to discrimination, bias, or social harm. All datasets used in this research are publicly available benchmark datasets widely used in academic studies. We commit to adhering to all applicable data privacy and intellectual property laws and regulations. We confirm that we have read and understood the ICLR Code of Ethics and pledge to comply with its guidelines in all aspects of our research, paper submission, and subsequent interactions.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of LOOP's iterative pruning and recovery procedure and recovery-oriented layer importance criterion in Section 4, together with pseudocode for the main algorithms. The models, datasets, evaluation protocols, baseline configurations, and training hyperparameters are documented in Section 5 and Section C.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes, 2024. URL https://arxiv.org/abs/2306.13649.

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Fluctuation-based adaptive structured pruning for large language models, 2023. URL https://arxiv.org/abs/2312.11983.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns, 2024. URL https://arxiv.org/abs/2401.15024.

Kehua Feng, Xinyi Shen, Weijie Wang, Xiang Zhuang, Yuqi Tang, Qiang Zhang, and Keyan Ding. Sciknoweval: Evaluating multi-level scientific knowledge of large language models, 2025. URL https://arxiv.org/abs/2406.09098.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot, 2023. URL https://arxiv.org/abs/2301.00774.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

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, 2021. URL https://arxiv.org/abs/2103.03874.

Yuxin Jiang, Chunkit Chan, Mingyang Chen, and Wei Wang. Lion: Adversarial distillation of proprietary large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3134–3154, 2023.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

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, 2024. URL https://arxiv.org/abs/2402.02834.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation, 2016. URL https://arxiv.org/abs/1606.07947.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: Llm quantization with learned rotations. In *International Conference on Learning Representations*, volume 2025, pp. 92009–92032, 2025.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Ré, and Beidi Chen. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, 2023. URL https://api.semanticscholar.org/CorpusID:260815690.

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 Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20192–20204, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1035. URL https://aclanthology.org/2025.findings-acl.1035/.

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. Compact language models via pruning and knowledge distillation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 41076–41102. Curran Associates, Inc., 2024. doi: 10.52202/079017-1299. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/4822991365c962105b1b95b1107d30e5-Paper-Conference.pdf.

OpenAI. GPT-4 Technical Report. *ArXiv preprint*, abs/2303.08774, 2023. URL https://arxiv.org/abs/2303.08774.

OpenAI. Openai gpt-5 system card, 2026. URL https://arxiv.org/abs/2601.03267.

Gunho Park, Baeseong Park, Minsub Kim, Sungjae Lee, Jeonghoon Kim, Beomseok Kwon, Se Jung Kwon, Byeongwook Kim, Youngjoo Lee, and Dongsoo Lee. Lut-gemm: Quantized matrix multiplication based on luts for efficient inference in large-scale generative language models. *ArXiv preprint*, abs/2206.09557, 2022. URL https://arxiv.org/abs/2206.09557.

Donghyeok Shin, Yeongmin Kim, Suhyeon Jo, Byeonghu Na, and Il-Chul Moon. Amid: Knowledge distillation for llms with α -mixture assistant distribution. In *International Conference on Learning Representations*, volume 2026, pp. 51469–51493, 2026.

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

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017.

Fei Wang, Li Shen, Liang Ding, Chao Xue, Ye Liu, and Changxing Ding. Layer as puzzle pieces: Compressing large language models through layer concatenation. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 6791–6824. Curran Associates, Inc., 2025a. doi: 10.52202/085713-0235. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/0a4096f09ab230082df548f901d4f79e-Paper-Conference.pdf.

Qinsi Wang, Jinghan Ke, Masayoshi Tomizuka, Kurt Keutzer, and Chenfeng Xu. Dobi-svd: Differentiable svd for llm compression and some new perspectives. In *International Conference on Learning Representations*, volume 2025, pp. 12561–12590, 2025b.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. Svd-llm: Truncation-aware singular value decomposition for large language model compression. In *International Conference on Learning Representations*, volume 2025, pp. 19299–19319, 2025c.

Xianglong Yan, Chengzhu Bao, Zhiteng Li, Tianao Zhang, Kaicheng Yang, Haotong Qin, Ruobing Xie, Samm Sun, and Yulun Zhang. PT^2-LLM: Post-training ternarization for large language models. In *International Conference on Learning Representations*, volume 2026, pp. 11677–11689, 2026.

An Yang, Anfeng Li, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Liangwei Yang, Yuhui Xu, Juntao Tan, Doyen Sahoo, Caiming Xiong, Huan Wang, Shelby Heinecke, et al. Entropy-based block pruning for efficient large language models. In *International Conference on Learning Representations*, volume 2026, pp. 103724–103740, 2026.

Yifei Yang, Zouying Cao, and Hai Zhao. LaCo: Large language model pruning via layer collapse. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 6401–6417, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.372. URL https://aclanthology.org/2024.findings-emnlp.372/.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Qingyu Zhang, Qianhao Yuan, Hongyu Lin, Yaojie Lu, Xianpei Han, Le Sun, Ming Xu, and Jiarui Li. Shortopd: Recovering pruned llms with short-to-long on-policy distillation, 2026. URL https://arxiv.org/abs/2607.13124.

Changhai Zhou, Qian Qiao, Yuhua Zhou, Yuxin Wu, Shichao Weng, Weizhong Zhang, and Cheng Jin. Large language model compression with global rank and sparsity optimization. In *International Conference on Learning Representations*, volume 2026, pp. 115013–115044, 2026.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 12:1556–1577, 2024. doi: 10.1162/tacl_a_00704. URL https://aclanthology.org/2024.tacl-1.85/.

A Interpretation of Recovery Sensitivity

Recovery sensitivity is designed to characterize how strongly the recovery objective locally depends on the contribution of each layer. Consider the recovery loss as a function of the layer gate g_l. Around the operating point $g_l = 1$, a first-order Taylor expansion gives

$$\mathcal{L}_{\mathrm{Rec}}(1 + \Delta g_l) \approx \mathcal{L}_{\mathrm{Rec}}(1) + \frac{\partial \mathcal{L}_{\mathrm{Rec}}}{\partial g_l} \Delta g_l. \tag{18}$$

Thus, the magnitude of $\partial \mathcal{L}_{\mathrm{Rec}} / \partial g_l$ reflects how strongly a local change in the contribution of layer l affects the recovery objective. A gradient close to zero indicates weak local dependence, whereas a gradient with large magnitude indicates that the recovery objective is more sensitive to that layer.

Because recovery sensitivity measures the strength of this dependence rather than its direction, LOOP uses the squared gate gradient. For a small zero-mean perturbation Δg_l with variance σ^2,

$$\mathbb{E}\left[(\Delta \mathcal{L}_{\mathrm{Rec}})^2\right] \approx \sigma^2 \left(\frac{\partial \mathcal{L}_{\mathrm{Rec}}}{\partial g_l} \right)^2. \tag{19}$$

This gives the squared gate gradient a direct interpretation as a nonnegative measure of local recovery sensitivity. It also prevents gradients with opposite directions from canceling when sensitivity is aggregated across on-policy recovery updates. Therefore, recovery sensitivity serves as a local recovery-oriented importance signal rather than an estimate of the exact loss change caused by removing a layer.

B Algorithm

Algorithm 1 summarizes the three-stage procedure of LOOP. The RecoverAndScore subroutine is defined in Algorithm 2, where on-policy trajectories and gate gradients are reused to estimate layer importance. The operator $\mathrm{Rec}(\mathcal{M}, \mathcal{T}; b)$ denotes the same OPD recovery updates without importance estimation. Here, $\mathcal{M}_{\mathrm{pre}}$ denotes the pretrained model, while $\mathcal{M}^{(0)}$ denotes the student after warm-up.

C Implementation Details

Training Configuration. The experiments use vLLM for rollout generation. The training batch size and optimization mini-batch size are both 32, with one response sampled per prompt. The learning rate is 2×10^{-5}, with a learning-rate warm-up of 10 updates. The total training budget is fixed at 200 update steps. OPD and LOOP share the same maximum rollout length and total update budget to isolate the effects of iterative pruning and layer selection.

Rollout and Pruning Configuration. Training responses are sampled with a temperature of 0.6, top-p of 0.9, and top-k of 50. Pruning is restricted to complete Transformer layers. We use $b_w = 20$ warm-up updates before the first pruning shot and reserve a nominal budget of $\bar{b}_f = 40$ updates for final recovery. For a total budget of $B = 200$ updates and K pruning shots, each shot is assigned

$$b_k = \left\lceil \frac{B - b_w - \bar{b}_f}{K} \right\rceil, \qquad k = 0, \ldots, K-1. \tag{20}$$

Any additional updates introduced by rounding up are deducted from the final recovery budget, giving

$$b_f = B - b_w - \sum_{k=0}^{K-1} b_k. \tag{21}$$

This allocation ensures that the total number of updates remains exactly B. For LOOP, weighted fusion is enabled with equal weights of 0.5 for reasoning contribution (measured by BI) and recovery sensitivity. The EMA decay β is set to 0.9. The one-shot comparison uses BI-based layer selection with an EMA decay of 0.9.

Algorithm 1 LOOP: Layer-level Recovery-Oriented On-Policy Pruning

Require: Pretrained model $\mathcal{M}_{\mathrm{pre}}$ with L layers; prompt distribution $\mathcal{D}$; pruning ratio ρ; update budgets $b_{\mathrm{w}}, \{b_k\}_{k=0}^{K-1}, b_{\mathrm{f}}$; EMA coefficient β; fusion weight λ; normalization constant $\epsilon > 0$.
Ensure: Compressed model $\mathcal{M}_{\mathrm{final}}$.

$K \leftarrow \lceil \rho L \rceil$, with $1 \le K < L$
Require $b_{\mathrm{w}} \ge 1$, $b_k \ge 1$, $b_{\mathrm{f}} \ge 0$, and $b_{\mathrm{w}} + \sum_{k=0}^{K-1} b_k + b_{\mathrm{f}} = B$
$\mathcal{T} \leftarrow \mathrm{Copy}(\mathcal{M}_{\mathrm{pre}})$; freeze $\mathcal{T}$
$\mathcal{M} \leftarrow \mathrm{Copy}(\mathcal{M}_{\mathrm{pre}})$
$\mathcal{I}^{(0)} \leftarrow \{1, \dots, L\}$

Stage 1: Warm-up
$(\mathcal{M}^{(0)}, \mathbf{I}^{(0)}) \leftarrow \textsc{RecoverAndScore}(\mathcal{M}, \mathcal{T}, \mathcal{D}, \mathcal{I}^{(0)}, b_{\mathrm{w}}, \beta, \lambda, \epsilon)$

Stage 2: Iterative pruning and recovery
for $k = 0, \dots, K-1$ **do**
    Select $l_k^* \in \arg\min_{l \in \mathcal{I}^{(k)}} I_l^{(k)}$ using a fixed tie-breaking rule
    $\widetilde{\mathcal{M}}^{(k)} \leftarrow \mathrm{Prune}(\mathcal{M}^{(k)}, l_k^*)$
    $\mathcal{I}^{(k+1)} \leftarrow \mathcal{I}^{(k)} \setminus \{l_k^*\}$
    $(\mathcal{M}^{(k+1)}, \mathbf{I}^{(k+1)}) \leftarrow \textsc{RecoverAndScore}(\widetilde{\mathcal{M}}^{(k)}, \mathcal{T}, \mathcal{D}, \mathcal{I}^{(k+1)}, b_k, \beta, \lambda, \epsilon)$
end for

Stage 3: Final recovery
$\mathcal{M}_{\mathrm{final}} \leftarrow \mathrm{Rec}(\mathcal{M}^{(K)}, \mathcal{T}; b_{\mathrm{f}})$
return $\mathcal{M}_{\mathrm{final}}$

Algorithm 2 RECOVERANDSCORE: On-Policy Recovery with Importance Estimation

procedure $\textsc{RecoverAndScore}(\mathcal{M}, \mathcal{T}, \mathcal{D}, \mathcal{I}, b, \beta, \lambda, \epsilon)$
    Freeze $\mathcal{T}$; attach differentiable gates $\mathbf{g} = \mathbf{1}$ excluded from optimizer updates
    for $u = 1, \dots, b$ **do**
        Sample $(\mathbf{x}, a^\star) \sim \mathcal{D}$ and generate $\mathbf{y} \sim p_\theta(\cdot \mid \mathbf{x})$; form $\boldsymbol{\tau} = [\mathbf{x}; \mathbf{y}]$
        Compute $\mathcal{L}_{\mathrm{Rec}}(u)$ using the teacher conditioned on $a^\star$
        Obtain the parameter and gate gradients of $\mathcal{L}_{\mathrm{Rec}}(u)$
        Compute $\{s^l_{\mathrm{Reason}}(u), s^l_{\mathrm{Rec}}(u)\}_{l \in \mathcal{I}}$ using Equation (4) and Equation (9)
        if $u = 1$ **then**
            $\bar{s}^l_q \leftarrow s^l_q(u)$ for all $l \in \mathcal{I}$, $q \in \{\mathrm{Reason}, \mathrm{Rec}\}$
        else
            $\bar{s}^l_q \leftarrow \beta \bar{s}^l_q + (1-\beta) s^l_q(u)$ for all l, q
        end if
        Update $\mathcal{M}$ using the OPD gradients; keep $\mathbf{g} = \mathbf{1}$
    end for
    $S^l_q \leftarrow \bar{s}^l_q$; normalize each score type using Equation (11)
    $I_l \leftarrow \lambda S^l_{\mathrm{Reason}} + (1-\lambda) S^l_{\mathrm{Rec}}$ for all $l \in \mathcal{I}$
    Remove auxiliary gates
    return $(\mathcal{M}, \mathbf{I})$
end procedure

Teacher Configuration. During on-policy distillation, the student generates responses using only the original problem prompt. For teacher inference, the original prompt is augmented with the ground truth using a fixed instruction template. The student-generated response is then appended to this augmented prompt, and the teacher computes token distributions at the corresponding student-generated positions.

D Computational Complexity and Scalability

For a model with L Transformer layers and target pruning ratio ρ, LOOP performs $K = \lceil \rho L \rceil$ pruning shots. The total number of recovery updates is controlled by the fixed budget

$$b_w + \sum_{k=0}^{K-1} b_k + b_f = B. \tag{22}$$

For this depth-based comparison, we assume matched token workloads. Let $C_{\mathrm{stu}}(m)$ denote the student-side cost of one recovery update when the student contains m layers, including on-policy generation and optimization, and let $C_{\mathrm{tea}}(L)$ denote the teacher inference cost. Since the teacher remains fixed while the student becomes progressively shallower, the total recovery computation is

$$\begin{aligned} C_{\mathrm{LOOP}} = b_w C_{\mathrm{stu}}(L) + \sum_{k=0}^{K-1} b_k C_{\mathrm{stu}}(L-k-1) + b_f C_{\mathrm{stu}}(L-K) \\ + B C_{\mathrm{tea}}(L). \end{aligned} \tag{23}$$

For comparison, a one-shot OPD pipeline that first estimates layer importance on a calibration set, prunes directly to $L - K$ layers, and then performs B recovery updates has cost

$$C_{\mathrm{OneShot}} = C_{\mathrm{cal}}(L) + B C_{\mathrm{stu}}(L-K) + B C_{\mathrm{tea}}(L), \tag{24}$$

where $C_{\mathrm{cal}}(L)$ denotes the cost of the separate calibration stage. Therefore, the difference in computational cost between the two methods is

$$\begin{aligned} C_{\mathrm{LOOP}} - C_{\mathrm{OneShot}} =& b_w \left(C_{\mathrm{stu}}(L) - C_{\mathrm{stu}}(L-K) \right) \\ &+ \sum_{k=0}^{K-1} b_k \left(C_{\mathrm{stu}}(L-k-1) - C_{\mathrm{stu}}(L-K) \right) \\ &- C_{\mathrm{cal}}(L) \end{aligned} \tag{25}$$

Assuming that the student-side computation of each recovery update scales approximately linearly with model depth, we write $C_{\mathrm{stu}}(m) = cm$, where c denotes the average computation per Transformer layer. The difference between LOOP and one-shot OPD can then be expressed as

$$\begin{aligned} C_{\mathrm{LOOP}} - C_{\mathrm{OneShot}} &= c \left[b_w K + \sum_{k=0}^{K-1} b_k (K-k-1) \right] - C_{\mathrm{cal}}(L) \\ &\approx c \left[b_w K + \frac{(B - b_w - b_f)(K-1)}{2} \right] - C_{\mathrm{cal}}(L). \end{aligned} \tag{26}$$

This analysis highlights the additional student-side computation from retaining more layers during early recovery, offset in part by avoiding separate calibration. Actual wall-clock time also depends on rollout lengths and implementation details. Table 6 reports lower average training-step time for LOOP under the evaluated settings.

E Experiments

E.1 Extended Recovery Analysis

To examine whether the performance gap persists with additional recovery training, we compare LOOP with a one-shot BI baseline on Qwen3-4B-Instruct at a 20% layer pruning ratio. The one-shot baseline in these figures corresponds to the OPD baseline in the main experiments, where all target layers are first pruned using BI and the resulting model is then recovered with on-policy distillation. Figure 2 compares LOOP with this baseline at evaluation checkpoints from 200 to 500 updates, where OOD Avg denotes the mean accuracy on MATH-500, AMC and AIME. LOOP achieves higher DAPO accuracy and higher average accuracy on the out-of-distribution benchmarks at every reported checkpoint. Although accuracy fluctuates during training, the one-shot baseline does not close the gap within the evaluated update range. These results suggest that the advantage of LOOP persists

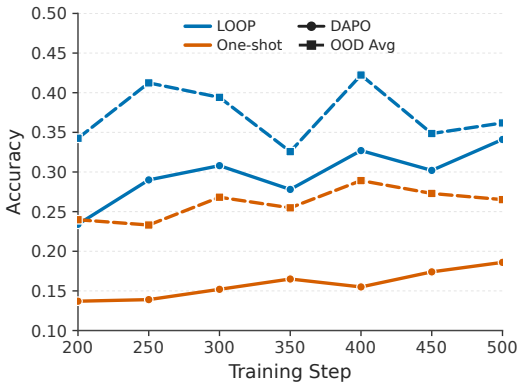

Figure 2: Accuracy of LOOP and the one-shot BI baseline during extended recovery training.

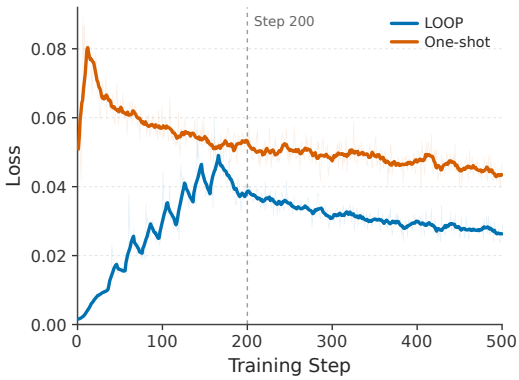

Figure 3: Recovery loss of LOOP and the one-shot BI baseline over 500 updates.

Table 7: Comparison of OPD training without pruning and pruning-based recovery methods on Qwen3-8B. Pruned methods use a 20% layer-pruning ratio.

Model	Method	Recovery	Metric	DAPO	Math-500	AMC	AIME	Mean	T (%)
Qwen3-8B	Original	–	–	0.197	0.414	0.325	0.056	0.2479	100.00
	OPD Trained	–	–	0.273	0.510	0.425	0.211	0.3548	143.120
	Pruned	–	BI	0.007	0.010	0.000	0.000	0.0043	1.71
	OPD	On-Policy	BI	0.071	0.338	0.175	0.011	0.1488	60.01
	LOOP (Ours)	On-Policy	BI + RS	**0.108**	**0.440**	**0.350**	**0.156**	**0.2634**	**106.25**

beyond the initial recovery budget. Figure 3 compares the recovery loss of LOOP and the one-shot BI baseline throughout training. Although LOOP's loss initially rises with fluctuations, it remains below that of the one-shot baseline throughout the observed training period. After 200 updates, both losses show an overall downward trend, while the gap persists. Together with LOOP's consistently higher accuracy at the reported evaluation checkpoints, these results suggest that its pruning strategy facilitates the recovery of reasoning capabilities.

E.2 Effect of OPD Without Pruning

To disentangle the effect of on-policy distillation from that of pruning and recovery, we further apply OPD directly to the unpruned Qwen3-8B model under the same training budget. As shown in Table 7, OPD improves the mean accuracy of the unpruned model from 0.2479 to 0.3548, indicating that on-policy training itself provides substantial task adaptation. Nevertheless, the pruned models remain considerably more challenging to recover: the one-shot OPD baseline reaches only 0.1488 mean accuracy, while LOOP achieves 0.2634. Importantly, both methods use on-policy distillation for recovery, yet LOOP outperforms the one-shot OPD baseline by 11.46 percentage points. This comparison suggests that the gains of LOOP over one-shot pruning cannot be attributed to on-policy distillation alone, and instead support the benefit of coupling iterative pruning with recovery-oriented layer selection.

E.3 Cross-Domain Generalization

Table 8: Cross-domain scientific reasoning performance of Qwen3-4B-Instruct at a 20% layer-pruning ratio. All results are reported using accuracy.

Ratio	Method	Recovery Data	Metric	Biology	Chemistry	Materials	Physics	Mean	T (%)
0%	Original	–	–	0.200	0.138	0.574	0.400	0.3281	100.000
20%	OPD	Science	BI	0.040	0.090	0.447	**0.158**	0.1837	55.985
	LOOP (Ours)	Science	BI + RS	**0.160**	**0.100**	**0.479**	0.150	**0.2222**	**67.714**
	OPD	Math	BI	0.000	0.000	0.064	0.050	0.0285	8.673
	LOOP (Ours)	Math	BI + RS	0.000	0.000	0.255	0.125	0.0951	28.977

To assess cross-domain performance retention after recovery on mathematical data, we evaluate models pruned and recovered on DAPO-Math on scientific reasoning benchmarks without additional training on scientific data. As shown in Table 8, LOOP achieves a mean accuracy of 0.0951, compared with 0.0285 for the one-shot OPD baseline, with improvements on Materials and Physics. These results suggest that LOOP mitigates cross-domain performance degradation relative to one-shot OPD.

E.4 Results on Llama-3.1-8B.

Table 9: Performance of Llama-3.1-8B on mathematical reasoning benchmarks. All results are measured by accuracy.

Method	Recovery	Metric	DAPO	Math-500	AMC	AIME	Mean	T (%)
Original	–	–	0.160	0.452	0.275	0.022	0.2273	100.000
OPD	On-Policy	BI	0.015	0.194	0.100	0.000	0.0773	33.985
LOOP (Ours)	On-Policy	BI + RS	**0.097**	**0.200**	**0.150**	**0.005**	**0.1130**	**49.713**

To assess whether the benefits of LOOP extend beyond the Qwen3 family, we additionally evaluate it on Llama-3.1-8B (Grattafiori et al., 2024) at a 20% layer pruning ratio. As shown in Table 9, LOOP outperforms the one-shot OPD baseline on all four benchmarks, improving mean accuracy from 7.73% to 11.30%. These results provide additional evidence of LOOP's effectiveness beyond Qwen3, although performance remains below that of the unpruned model.