

Unlocking Structured Pruning for Reasoning Language Models via Attention-Preserving MLP Trimming

Woosang Lim^{1,*}, Haneul Yoo^{2,*}, Vincent-Daniel Yun^{3,*}, Sungjoo Yoo^{1†}

¹Seoul National University, Republic of Korea

²New York University, United States

³University of Southern California, United States



Abstract

Structured pruning reduces LLM inference costs by removing model components to produce smaller dense models. However, existing methods can cause reasoning performance to collapse on long-form reasoning tasks such as AIME. We propose *LayerTrim*, a training-free structured pruning method that identifies redundant decoder layers and selectively trims their MLP features while preserving attention and decoder depth. LayerTrim is the first training-free structured pruning method designed to preserve long-form reasoning without gradients or recovery training. Across three reasoning language models and five benchmarks, LayerTrim outperforms existing structured pruning methods in most evaluated settings, improving average accuracy by up to 21.4 percentage points over RESP. On Qwen3-14B, LayerTrim achieves $1.24\times$ faster prefill and 19.7% lower peak GPU memory than the dense model, while better preserving reasoning ability. It also achieves the shortest average generation length among evaluated pruned models, while retaining the highest average accuracy. These results demonstrate that LayerTrim improves reasoning efficiency through both structural compression and reduced token generation.

1 Introduction

Reasoning language models (RLMs) achieve strong performance on complex tasks through chain-of-thought (CoT) reasoning (DeepSeek-AI et al., 2025; Yang et al., 2025). However, generating long reasoning sequences incurs high inference costs, motivating more efficient inference. Structured pruning reduces LLM inference costs by removing model components to produce smaller dense models (Ma et al., 2023; An et al., 2024). However, applying pruning methods developed for

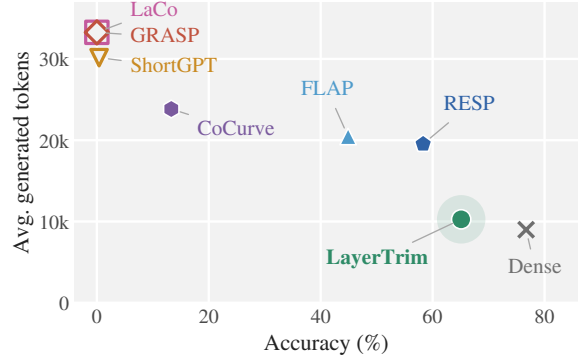


Figure 1: Accuracy versus generated tokens per response on Qwen3-14B with a pruning budget equivalent to 9 of its 40 decoder layers. Both metrics are averaged across five benchmarks, with a maximum generation length of 32K tokens.

non-reasoning LLMs to RLMs can cause performance to collapse on tasks requiring long chains of thought, such as AIME (Ding et al., 2026).

Figure 1 illustrates this failure, where existing structured pruning methods lead to longer reasoning traces and lower accuracy. To investigate this degradation, we analyze pruning strategies on Qwen3-14B (Yang et al., 2025). Our analysis reveals three observations (Figure 2). First, MLP trimming causes less representation error than attention pruning or entire module removal. Second, MLP feature importance is concentrated in a subset of features. Third, concentrating MLP trimming in redundant layers produces less error than distributing the same budget across all layers. These findings motivate selective MLP compression while preserving attention and decoder depth.

Based on these findings, we propose **LayerTrim**, a training-free structured pruning method that identifies redundant decoder **layers** and selectively **trims** their MLP features while preserving attention and decoder depth. LayerTrim is the first training-free structured pruning method designed to preserve long-form reasoning and code genera-

*Corresponding author: sungjoo.yoo@gmail.com.

*These authors contributed equally.

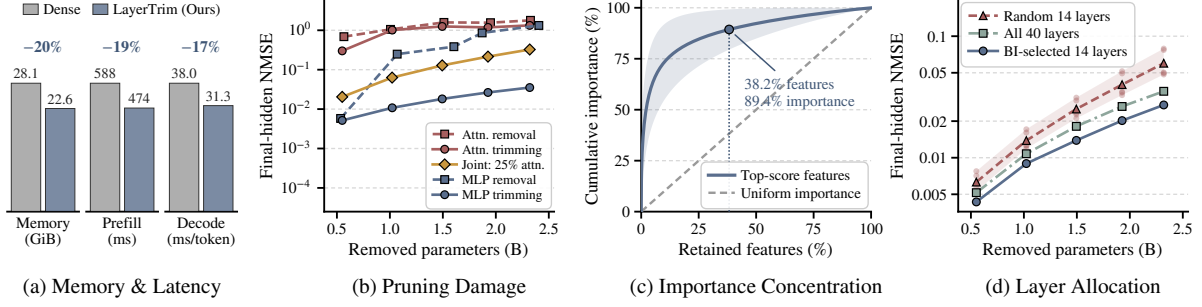


Figure 2: Observations motivating LayerTrim on Qwen3-14B. (a) LayerTrim reduces memory usage and inference latency with a pruning budget equivalent to 9 of 40 decoder layers. (b) MLP trimming causes less representation error than attention pruning, module removal, or joint pruning. (c) MLP feature importance is concentrated in a subset of features. (d) Block Influence (BI) (Men et al., 2025)-based layer selection reduces error compared with random or all-layer pruning. Diagnostic settings are given in Appendix D.

tion performance without recovery training. It uses dense-model reasoning traces (Lucas et al., 2026; Wang et al., 2025) to select layers using Block Influence (Men et al., 2025) and retain MLP features with the largest expected squared output contributions.

Across three RLMS and five benchmarks, LayerTrim outperforms existing structured pruning methods in most evaluated settings, improving average accuracy by up to 21.4 percentage points over RESP (Wang et al., 2025) under matched parameter budgets. On Qwen3-14B, LayerTrim achieves higher accuracy while generating fewer tokens on average than the other evaluated pruning methods (Figure 1). It also reduces peak GPU memory, prefill latency, and decode latency by approximately 20%, 19%, and 17%, respectively, relative to the dense model at a pruning budget equivalent to 9 of 40 decoder layers (Figure 2(a)).

Our contributions are as follows:

- We identify attention preservation and selective MLP trimming as strategies for reducing representation error compared with attention pruning or entire module removal.
- We introduce *LayerTrim*, a training-free structured pruning method for long-form reasoning without gradients or recovery training.
- We demonstrate improved reasoning efficiency, achieving higher accuracy with fewer generated tokens and better preserving reasoning completion.

2 Related Work

Structured pruning of LLMs. Unstructured pruning methods such as SparseGPT (Frantar and

Alistarh, 2023) and Wanda (Sun et al., 2024) remove individual weights, often requiring sparse kernels for practical acceleration. In contrast, structured pruning removes computational units to produce smaller dense models (Kim et al., 2026). LLM-Pruner (Ma et al., 2023) and HFPrune (Zhu et al., 2026) use gradient- or Taylor-based importance criteria, while FLAP (An et al., 2024) compensates for pruning errors. Sheared LLaMA (Xia et al., 2024) incorporates additional training, and CoCurve (Gong et al., 2026) considers interactions between structured units. Depth pruning methods such as ShortGPT (Men et al., 2025) identify redundant decoder layers using Block Influence, while LaCo (Yang et al., 2024) and GRASP (Liu et al., 2025) reduce model depth through layer merging or replacement. LoRP (Yun et al., 2026) exploits inter-layer redundancy and architectural locality to identify layers for training-free depth pruning. However, these methods do not explicitly consider long-form CoT reasoning, and removing entire layers can degrade reasoning performance.

Pruning RLMS. Pruning methods developed for non-reasoning LLMs can degrade reasoning performance (Ding et al., 2026). RAC (Lucas et al., 2026) uses on-policy reasoning traces for activation reconstruction, while OBC-Prune (Nguyen et al., 2026) extends reasoning-aware calibration through outcome-based token weighting. Both rely on unstructured pruning, requiring sparse kernel support for practical acceleration. RESP (Wang et al., 2025) uses self-generated reasoning traces, gradient-based importance, and progressive recalibration for structured pruning, but retains lower reasoning accuracy than dense models across the evaluated benchmarks.

Preserving long-form reasoning remains a challenge for existing structured pruning approaches. LayerTrim addresses this challenge by selectively trimming MLP features while preserving attention and decoder depth.

3 Method

3.1 Motivation

We first investigate three structural pruning decisions for RLMS: (1) which components to prune, (2) which MLP features to retain, and (3) how to allocate pruning across layers. Using Qwen3-14B and AIME reasoning traces generated by the dense model, we characterize MLP feature contributions and compare representation errors across pruning settings.

Pruning targets. Methods such as RESP prune attention and MLPs jointly. We separately compare attention and MLP removal, attention and MLP trimming, and joint trimming of both. Removal deletes modules selected by low Block Influence, whereas trimming reduces internal width across all 40 decoder layers.

Across all tested budgets, MLP-only trimming produces lower final-hidden representation error than module removal, attention trimming, or joint trimming of attention and MLPs (Figure 2(b)). These results motivate preserving attention and decoder depth while reducing MLP width.

Importance concentration within MLPs. To select features for width reduction, we express the MLP output as a sum of intermediate feature terms. For a SwiGLU MLP (Shazeer, 2020) with intermediate width I , let $\phi^{(\ell)}(x) = \text{SiLU}(W_{\text{gate}}^{(\ell)}x) \odot (W_{\text{up}}^{(\ell)}x) \in \mathbb{R}^I$ denote its intermediate activation vector. The j th feature contributes $c_j^{(\ell)}(x) = \phi_j^{(\ell)}(x)W_{\text{down}}^{(\ell)}[:, j]$ to the MLP output, yielding $\text{MLP}^{(\ell)}(x) = \sum_{j=1}^I c_j^{(\ell)}(x)$.

We define the feature importance score as the expected squared norm of $c_j^{(\ell)}(x)$:

$$\begin{aligned} s_j^{(\ell)} &= \mathbb{E}_{\mathcal{D}} \left[\|c_j^{(\ell)}(x)\|_2^2 \right] \\ &= \mathbb{E}_{\mathcal{D}} \left[(\phi_j^{(\ell)}(x))^2 \|W_{\text{down}}^{(\ell)}[:, j]\|_2^2 \right] \end{aligned} \quad (1)$$

Here, $\mathcal{D}$ denotes calibration token positions, and x is the dense-model MLP input. Figure 2(c) shows that the top 38.2% of features account for 89.4% of the total feature importance, supporting their retention during MLP width reduction.

Redundancy-aware budget allocation. Feature importance determines which dimensions to retain within an MLP, while pruning budget allocation determines which layers undergo width reduction. Figure 2(d) compares trimming all 40 MLPs, 14 randomly selected MLPs, and 14 MLPs selected by low Block Influence under matched parameter budgets. Trimming the MLPs selected by low Block Influence produces lower final-hidden representation error than either alternative. The comparison with random selection indicates that layer selection matters beyond the number of trimmed MLPs.

3.2 LayerTrim

Guided by these observations, LayerTrim reduces MLP width while preserving attention and decoder depth. It retains high-importance features and concentrates pruning in less influential layers. Figure 3 summarizes five stages: reasoning-trace calibration, decoder-layer selection, MLP feature ranking, structured MLP trimming, and the resulting compressed model. Algorithm 1 in Appendix A gives the complete procedure.

Reasoning-trace calibration. Following prior work on pruning RLMS (Lucas et al., 2026; Wang et al., 2025), we collect statistics through teacher-forced forward passes over reasoning traces generated by the dense model. For each layer, we record decoder-block inputs and outputs and MLP intermediate activations. These statistics support layer selection and feature ranking without gradients.

Decoder-layer selection. To concentrate pruning in less influential layers, we use *Block Influence* (BI) (Men et al., 2025):

$$\text{BI}_{\ell} = 1 - \mathbb{E}_{\mathcal{D}} \left[\cos \left(h_{\text{in}}^{(\ell)}, h_{\text{out}}^{(\ell)} \right) \right] \quad (2)$$

where $h_{\text{in}}^{(\ell)}$ and $h_{\text{out}}^{(\ell)}$ are decoder block inputs and outputs. Low BI indicates a small directional change across a block. For a budget of N layer-equivalents, we select the $K = 2N$ lowest-BI layers, denoted $\mathcal{L}_K$, and modify only their MLPs. This choice spreads the pruning budget across more MLPs rather than applying stronger width reduction to fewer layers.

MLP feature ranking. Let m denote the number of features retained in each selected MLP. Let C and I be the hidden and MLP intermediate widths, and P_{layer} the prunable parameter count of one decoder layer. Each removed feature eliminates

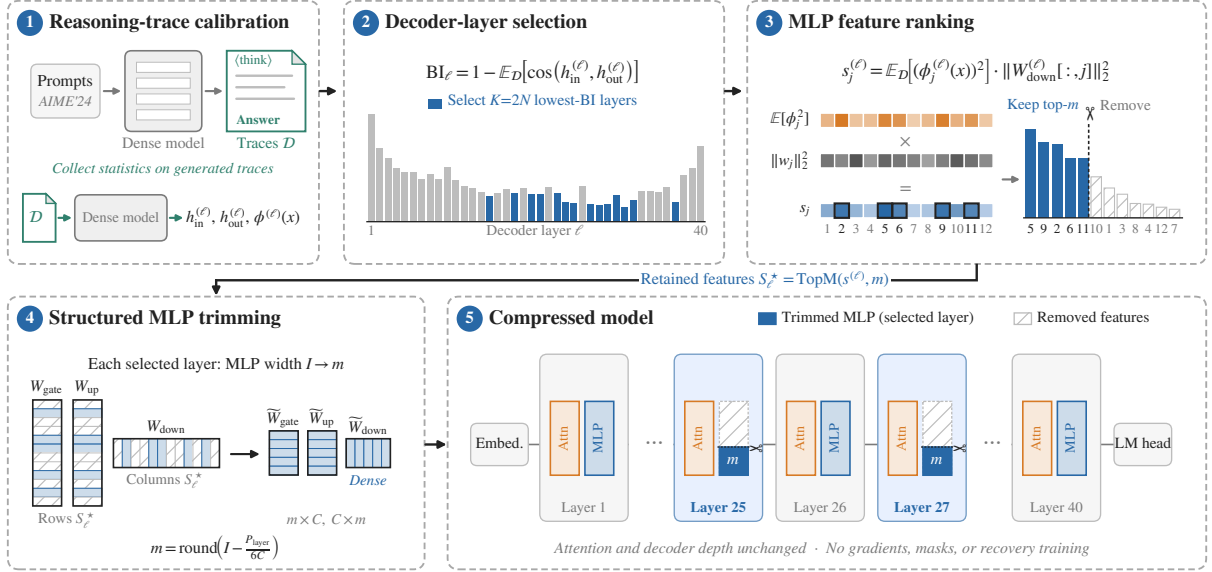


Figure 3: Overview of LayerTrim. LayerTrim collects layer-level and MLP feature-level statistics from reasoning traces generated by the dense model. It selects redundant decoder layers and retains MLP features with larger output contributions. The selected MLPs are reduced to smaller dense layers while preserving attention and decoder depth.

C weights from each of the gate, up, and down projections, totaling $3C$ parameters per feature and $3CK(I - m)$ across the K selected MLPs. Setting this count equal to the budget, $3CK(I - m) = NP_{\text{layer}}$, and using $K = 2N$ gives

$$m = \text{round}\left(I - \frac{P_{\text{layer}}}{6C}\right) \quad (3)$$

For each selected MLP, we compute feature importance $s_j^{(\ell)}$ using Equation 1. We rank features by this score and retain the top m , $S_\ell^* = \text{TopM}(s^{(\ell)}, m)$. To reduce calibration overhead, we approximate the squared MLP output change using the summed scores of removed features.

Structured MLP trimming. We physically remove the remaining $I - m$ features from each selected MLP. For each $\ell \in \mathcal{L}_K$, we retain rows S_ℓ^* of the gate and up projections and columns S_ℓ^* of the down projection. The reduced projections have shapes $m \times C$ and $C \times m$. The retained weights form a compressed checkpoint with smaller dense MLPs. Attention modules, unselected MLPs, and decoder depth remain unchanged. No inference-time masks, fine-tuning, distillation, or recovery training are required.

4 Experimental Results

4.1 Experimental Settings

Models and benchmarks. We use three RLMS and five benchmarks:

- **Models:** Qwen3-14B (Yang et al., 2025), Phi-4-reasoning-plus (Abdin et al., 2025), Magistral-Small-2506 (Mistral-AI et al., 2025);
- **Benchmarks:** AIME2025 (Zhang and Math-AI, 2025), AIME2026 (Zhang and Math-AI, 2026), GPQA-Diamond (Rein et al., 2024), HumanEval (Chen et al., 2021), and MBPP+ (Liu et al., 2023).

We report exact numerical-answer accuracy on AIME, multiple-choice accuracy on GPQA-Diamond, and pass@1 on HumanEval and MBPP+ using their original and extended test suites, respectively. For each model, we use its original dense checkpoint as the reference to measure performance retention after pruning.

Baselines and pruning budgets. We compare LayerTrim with three depth pruning methods and three width pruning methods:

- **Depth:** ShortGPT (Men et al., 2025), LaCo (Yang et al., 2024), GRASP (Liu et al., 2025);
- **Width:** FLAP (An et al., 2024), RESP (Wang et al., 2025), CoCurve (Gong et al., 2026).

We evaluate budgets of $N \in \{7, 9\}$ layer-equivalents, each corresponding to the parameter count of N decoder layers. We match the pruning

		$L_p/L_t = 7/40$						$L_p/L_t = 9/40$					
Method	Type	AIME25	AIME26	GPQA-D	HumanEval	MBPP+	Avg.	AIME25	AIME26	GPQA-D	HumanEval	MBPP+	Avg.
<i>Qwen3-14B</i> (Dense)		72.2 ± 5.1	75.6 ± 10.2	62.3 ± 0.6	93.5 ± 0.9	80.0 ± 1.3	76.7	72.2 ± 5.1	75.6 ± 10.2	62.3 ± 0.6	93.5 ± 0.9	80.0 ± 1.3	76.7
ShortGPT	Depth	0.0	0.0	5.6 ± 0.5	19.5 ± 1.2	6.0 ± 0.6	6.2	0.0	0.0	0.3 ± 0.6	1.4 ± 0.4	0.2 ± 0.2	0.4
LaCo	Depth	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
GRASP	Depth	0.0	0.0	0.5 ± 0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0
FLAP	Width	44.4 ± 3.8	51.1 ± 8.4	44.1 ± 0.8	76.8 ± 3.2	51.6 ± 0.8	53.6	38.9 ± 6.9	41.1 ± 6.9	35.4 ± 3.9	59.8 ± 2.8	49.3 ± 0.6	44.9
RESP	Width	57.8 ± 8.4	62.2 ± 3.8	48.0 ± 2.2	84.3 ± 2.7	72.8 ± 1.9	65.0	52.2 ± 5.1	67.8 ± 11.7	46.0 ± 2.2	62.0 ± 2.1	63.7 ± 1.1	58.3
CoCurve	Width	1.1 ± 1.9	5.6 ± 1.9	24.6 ± 1.5	53.9 ± 2.1	50.4 ± 0.7	27.1	0.0	0.0	7.7 ± 0.6	25.0 ± 0.6	34.0 ± 1.4	13.3
LayerTrim (Ours)	Width	64.4 ± 5.1	60.0 ± 5.8	49.7 ± 2.9	90.2 ± 1.6	77.2 ± 1.8	68.3	56.7 ± 3.3	63.3 ± 3.3	42.8 ± 3.5	88.0 ± 2.7	74.8 ± 0.3	65.1
<i>Phi-4-reasoning-plus</i> (Dense)		72.2 ± 5.1	67.8 ± 5.1	67.7 ± 1.7	78.7 ± 2.4	68.8 ± 2.2	71.0	72.2 ± 5.1	67.8 ± 5.1	67.7 ± 1.7	78.7 ± 2.4	68.8 ± 2.2	71.0
ShortGPT	Depth	0.0	0.0	3.0 ± 0.5	2.8 ± 1.4	5.4 ± 1.1	2.2	0.0	0.0	9.1 ± 1.3	0.0	0.0	1.8
LaCo	Depth	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
GRASP	Depth	0.0	0.0	0.5 ± 0.5	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0
FLAP	Width	22.2 ± 1.9	27.8 ± 5.1	41.4 ± 1.3	68.9 ± 5.8	51.5 ± 0.7	42.4	7.8 ± 3.8	7.8 ± 1.9	24.1 ± 0.8	3.7 ± 0.0	2.5 ± 0.9	9.2
RESP	Width	62.2 ± 1.9	50.0 ± 5.8	33.5 ± 0.6	50.6 ± 2.1	53.2 ± 0.9	49.9	40.0 ± 5.8	45.6 ± 5.1	15.7 ± 0.5	34.1 ± 1.2	37.6 ± 1.1	34.6
CoCurve	Width	8.9 ± 3.8	3.3 ± 3.3	17.0 ± 0.8	57.3 ± 2.2	45.9 ± 1.3	26.5	3.3 ± 0.0	3.3 ± 0.0	6.2 ± 0.3	26.6 ± 3.0	24.9 ± 1.6	12.9
LayerTrim (Ours)	Width	60.0 ± 3.3	60.0 ± 3.3	53.0 ± 0.5	74.6 ± 3.0	63.3 ± 1.0	62.2	43.3 ± 3.3	46.7 ± 0.0	43.3 ± 0.6	56.9 ± 2.0	54.3 ± 1.4	48.9
<i>Magistral-Small-2506</i> (Dense)		56.7 ± 3.3	61.1 ± 8.4	62.3 ± 3.0	94.3 ± 0.9	77.2 ± 1.1	70.3	56.7 ± 3.3	61.1 ± 8.4	62.3 ± 3.0	94.3 ± 0.9	77.2 ± 1.1	70.3
ShortGPT	Depth	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
LaCo	Depth	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
GRASP	Depth	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
FLAP	Width	3.3 ± 0.0	2.2 ± 1.9	21.9 ± 0.8	40.0 ± 2.5	36.5 ± 2.3	20.8	1.1 ± 1.9	0.0	12.6 ± 2.7	21.5 ± 0.7	17.8 ± 0.9	10.6
RESP	Width	40.0 ± 3.3	37.8 ± 5.1	38.4 ± 1.3	53.7 ± 1.8	51.3 ± 1.6	44.2	18.9 ± 6.9	15.6 ± 6.9	24.1 ± 2.6	27.6 ± 0.9	35.5 ± 0.9	24.3
CoCurve	Width	1.1 ± 1.9	1.1 ± 1.9	15.8 ± 2.0	56.3 ± 0.9	50.7 ± 2.1	25.0	0.0	0.0	3.7 ± 0.8	19.5 ± 1.8	25.0 ± 1.2	9.6
LayerTrim (Ours)	Width	30.0 ± 8.8	42.2 ± 1.9	52.2 ± 1.5	90.7 ± 3.7	73.8 ± 1.2	57.8	18.9 ± 8.4	18.9 ± 1.9	44.3 ± 1.6	76.0 ± 3.1	70.5 ± 2.5	45.7

Table 1: Main results across three RLMS under matched parameter budgets. The two column groups correspond to $L_p/L_t = 7/40$ and $9/40$, where L_p is the layer-equivalent pruning budget and L_t is the original decoder depth. Type indicates depth or width pruning. Benchmark entries are mean $\pm$ sample std. over three seeds, and Avg. is the unweighted mean accuracy (%) across the five benchmarks. **Bold** marks the best pruned-model accuracy within each budget for each benchmark and the average.

parameter budget across methods. Budget allocation and method-specific implementation details are provided in Appendix B.

Calibration. For each model, we use the same calibration set across all pruning methods, consisting of 24 AIME 2024 reasoning traces generated by its dense checkpoint; following its original procedure, RESP additionally regenerates these traces during pruning. Traces are generated with temperature 0.6, seed 0, and a maximum generation length of 16K tokens. LayerTrim collects layer redundancy and MLP feature importance statistics from each complete prompt and up to its first 2K generated tokens, without updating its weights.

Evaluation settings. We evaluate dense and pruned models using their native chat templates with reasoning enabled. All models are evaluated with temperature 0.6 and top- p 0.95. The effective maximum output length is 32K tokens for all three models. We evaluate each configuration with generation seeds 0, 1, and 2, generating one response per problem for each seed. We report the mean and sample standard deviation over the three seeds.

4.2 Main Results

Table 1 compares LayerTrim with existing pruning methods under matched parameter budgets. LayerTrim achieves the highest average benchmark score among the pruning methods for all three models at both budgets. Its lead over the strongest baseline grows as the budget increases from $N = 7$ to $N = 9$. At $N = 9$, LayerTrim exceeds the strongest baseline average by 6.8, 14.3, and 21.4 percentage points on Qwen3-14B, Phi-4-reasoning-plus, and Magistral-Small-2506, respectively.

Although calibration uses only AIME 2024 reasoning traces, LayerTrim’s advantages extend to science and coding. Compared with the strongest baseline on each benchmark, its gains reach up to 20.2 percentage points on GPQA-Diamond, 48.4 on HumanEval, and 35.0 on MBPP+.

4.3 Ablation Studies

We examine four design choices in LayerTrim: the pruning target, feature-importance score, layer selection, and number of selected MLPs. Table 2 reports results on Qwen3-14B using AIME25 and GPQA-Diamond. Detailed experimental settings are provided in Appendix C. Additional calibration-length, calibration-source, and cross-feature selec-

Variant	AIME25	GPQA-D	Avg.
Pruning targets			
Attention removal	0.0 $\pm$ 0.0	0.7 $\pm$ 0.3	0.4
Attention trimming	0.0 $\pm$ 0.0	1.3 $\pm$ 0.3	0.7
MLP removal	0.0 $\pm$ 0.0	17.5 $\pm$ 1.1	8.8
Joint trimming (25% attn.)	43.3 $\pm$ 3.3	53.2 $\pm$ 4.2	48.3
MLP trimming	62.2 $\pm$ 8.4	55.4 $\pm$ 2.4	58.8
Feature-importance score			
$\mathbb{E}[\phi_j^2]$	61.1 $\pm$ 5.1	48.1 $\pm$ 3.3	54.6
$\ w_j\ _2^2$	22.2 $\pm$ 1.9	41.1 $\pm$ 3.0	31.7
$\text{Var}(\phi_j)\ w_j\ _2^2$ (FLAP)	60.0 $\pm$ 8.8	49.5 $\pm$ 4.5	54.8
$\mathbb{E}[\phi_j^2]\ w_j\ _2^2$ (Ours)	64.4 $\pm$ 5.1	49.7 $\pm$ 2.9	57.1
Layer selection and distribution			
MLP-only BI	56.7 $\pm$ 5.8	51.2 $\pm$ 1.5	54.0
Random	50.0 $\pm$ 8.8	46.3 $\pm$ 0.8	48.2
All-layer ($K = 40$)	50.0 $\pm$ 8.8	53.5 $\pm$ 0.9	51.8
Block BI (Ours)	64.4 $\pm$ 5.1	49.7 $\pm$ 2.9	57.1
Number of selected MLPs			
$K = 1.5N$	66.7 $\pm$ 3.3	46.6 $\pm$ 5.4	56.7
$K = 3N$	57.8 $\pm$ 3.8	54.9 $\pm$ 1.5	56.4
$K = 2N$ (Ours)	64.4 $\pm$ 5.1	49.7 $\pm$ 2.9	57.1

Table 2: Ablations on Qwen3-14B. Scores (%) are mean $\pm$ sample std over three seeds. Avg. averages AIME25 and GPQA-Diamond; bold marks the best average in each group. We remove approximately 1.5B parameters in the first group and 2.312B ($N = 7$) in the others. See Appendix C for experimental settings.

tion ablations appear in Appendix C.1.

Pruning targets. Figure 2(b) shows that MLP trimming produces the lowest final-hidden NMSE. The accuracy results in Table 2 show the same pattern: MLP-only trimming preserves substantially higher average accuracy than whole-module removal or attention pruning. It also outperforms joint attention-MLP trimming, which allocates 25% of the pruning budget to attention. These results show that MLP-only trimming minimizes pruning damage among the tested choices in both representation error and task accuracy.

Feature-Importance score. We compare alternative criteria for selecting MLP features during width reduction. Scores based on activations achieve higher average accuracy than weight-only ranking. Our feature-importance score achieves the highest average accuracy among the evaluated criteria, outperforming activation-only ranking and FLAP’s variance-based score.

Layer selection and distribution. The accuracy results in Table 2 support the BI-based allocation in Figure 2(d): selecting low-BI layers outperforms random selection and all-layer trimming in average accuracy. Block BI also achieves higher average accuracy than MLP-only BI, which measures changes

in the residual stream before and after the MLP update. This supports measuring redundancy across the full decoder block rather than the MLP update alone.

Number of selected MLPs. Under a fixed budget, $K = 1.5N$ achieves the highest AIME25 accuracy, while $K = 3N$ achieves the highest GPQA-Diamond accuracy. The default $K = 2N$ gives the highest average accuracy by a small margin, providing a practical balance between the two tasks.

4.4 Efficiency Measurement

Measurement settings. We measure prefill latency, decode latency, and peak GPU memory in bfloat16 at batch size 1 on a single NVIDIA RTX 6000 Ada (48 GB). Prefill latency is measured on a 2,048-token prompt, and decode latency is averaged over 128 greedy decoding steps with a static KV cache. Each configuration runs two warm-up iterations followed by ten timed runs, synchronized using `torch.cuda.synchronize()`. We report median latency and peak GPU memory during inference, excluding model loading and pruning overhead. All methods are evaluated under matched parameter budgets.

Efficiency results. Table 3 compares inference efficiency across three RLMS at two pruning budgets. At the higher pruning budget, LayerTrim achieves prefill speedups of 1.24 $\times$, 1.22 $\times$, and 1.27 $\times$, and decode speedups of 1.21 $\times$, 1.22 $\times$, and 1.24 $\times$ on Qwen3-14B, Phi-4-reasoning-plus, and Magistral-Small-2506, respectively. Peak GPU memory is reduced by 19.7%, 20.3%, and 20.8% on the corresponding models compared with their dense counterparts. While achieving inference costs comparable to existing structured pruning methods, LayerTrim retains the highest average accuracy across all three models at both pruning budgets. LayerTrim also requires only 27.9 s of pruning preparation on Qwen3-14B, 94 $\times$ and 140 $\times$ faster than RESP at $N = 7$ and $N = 9$, respectively (Appendix B.1).

5 Discussions

5.1 Analysis of LayerTrim

Attention preservation. We analyze how well MLP trimming preserves attention patterns on reasoning traces. Panel 4a shows that MLP trimming produces the lowest attention KL divergence across the tested budgets. At approximately 1.5B removed

Method	Type	$L_p/L_t = 7/40$				$L_p/L_t = 9/40$			
		Prefill (ms)	Decode (ms/tok)	Peak Mem. (GiB)	Acc. (%)	Prefill (ms)	Decode (ms/tok)	Peak Mem. (GiB)	Acc. (%)
<i>Qwen3-14B</i> (Dense)		588 (1.00×)	38.0 (1.00×)	28.1 (100%)	76.7 (100.0%)	588 (1.00×)	38.0 (1.00×)	28.1 (100%)	76.7 (100.0%)
ShortGPT	Depth	502 (1.17×)	31.8 (1.19×)	23.8 (84%)	6.2 (8.1%)	470 (1.25×)	30.1 (1.26×)	22.5 (80%)	0.4 (0.5%)
LaCo	Depth	497 (1.18×)	31.8 (1.19×)	23.8 (84%)	0.0 (0.0%)	466 (1.26×)	30.1 (1.26×)	22.5 (80%)	0.0 (0.0%)
GRASP	Depth	498 (1.18×)	34.0 (1.12×)	23.8 (85%)	0.1 (0.1%)	466 (1.26×)	32.8 (1.16×)	22.6 (80%)	0.0 (0.0%)
FLAP	Width	513 (1.15×)	32.6 (1.16×)	23.6 (84%)	53.6 (69.9%)	482 (1.22×)	31.2 (1.22×)	22.4 (80%)	44.9 (58.5%)
RESP	Width	523 (1.12×)	32.9 (1.15×)	23.8 (85%)	65.0 (84.7%)	493 (1.19×)	31.4 (1.21×)	22.6 (80%)	58.3 (76.0%)
CoCurve	Width	509 (1.16×)	32.8 (1.16×)	23.8 (85%)	27.1 (35.3%)	508 (1.16×)	31.4 (1.21×)	22.5 (80%)	13.3 (17.3%)
LayerTrim (Ours)	Width	510 (1.15×)	32.8 (1.16×)	23.8 (85%)	68.3 (89.0%)	474 (1.24×)	31.3 (1.21×)	22.6 (80%)	65.1 (84.9%)
<i>Phi-4-reasoning-plus</i> (Dense)		635 (1.00×)	37.7 (1.00×)	28.2 (100%)	71.0 (100.0%)	635 (1.00×)	37.7 (1.00×)	28.2 (100%)	71.0 (100.0%)
ShortGPT	Depth	539 (1.18×)	31.5 (1.20×)	23.6 (84%)	2.2 (3.1%)	497 (1.28×)	29.8 (1.27×)	22.3 (79%)	1.8 (2.5%)
LaCo	Depth	540 (1.17×)	31.5 (1.20×)	23.6 (84%)	0.0 (0.0%)	505 (1.26×)	29.8 (1.27×)	22.3 (79%)	0.0 (0.0%)
GRASP	Depth	548 (1.16×)	32.6 (1.16×)	23.7 (84%)	0.1 (0.1%)	514 (1.23×)	31.1 (1.21×)	22.4 (80%)	0.0 (0.0%)
FLAP	Width	521 (1.22×)	32.4 (1.16×)	23.5 (83%)	42.4 (59.7%)	498 (1.27×)	30.8 (1.22×)	22.2 (79%)	9.2 (13.0%)
RESP	Width	544 (1.17×)	32.4 (1.16×)	23.7 (84%)	49.9 (70.3%)	514 (1.23×)	30.9 (1.22×)	22.4 (80%)	34.6 (48.7%)
CoCurve	Width	533 (1.19×)	32.6 (1.15×)	23.6 (84%)	26.5 (37.3%)	515 (1.23×)	30.9 (1.22×)	22.3 (79%)	12.9 (18.2%)
LayerTrim (Ours)	Width	557 (1.14×)	32.3 (1.17×)	23.7 (84%)	62.2 (87.6%)	519 (1.22×)	30.8 (1.22×)	22.4 (80%)	48.9 (68.9%)
<i>Magistral-Small-2506</i> (Dense)		996 (1.00×)	57.4 (1.00×)	44.7 (100%)	70.3 (100.0%)	996 (1.00×)	57.4 (1.00×)	44.7 (100%)	70.3 (100.0%)
ShortGPT	Depth	836 (1.19×)	47.9 (1.20×)	37.4 (84%)	0.0 (0.0%)	779 (1.28×)	45.2 (1.27×)	35.3 (79%)	0.0 (0.0%)
LaCo	Depth	836 (1.19×)	47.9 (1.20×)	37.4 (84%)	0.0 (0.0%)	781 (1.28×)	45.2 (1.27×)	35.3 (79%)	0.0 (0.0%)
GRASP	Depth	797 (1.25×)	49.2 (1.17×)	37.5 (84%)	0.0 (0.0%)	763 (1.31×)	46.9 (1.23×)	35.4 (79%)	0.0 (0.0%)
FLAP	Width	783 (1.27×)	48.8 (1.18×)	37.4 (84%)	20.8 (29.6%)	789 (1.26×)	46.4 (1.24×)	35.3 (79%)	10.6 (15.1%)
RESP	Width	810 (1.23×)	48.8 (1.18×)	37.5 (84%)	44.2 (62.9%)	806 (1.24×)	46.3 (1.24×)	35.4 (79%)	24.3 (34.6%)
CoCurve	Width	778 (1.28×)	48.8 (1.18×)	37.4 (84%)	25.0 (35.6%)	789 (1.26×)	46.4 (1.24×)	35.3 (79%)	9.6 (13.7%)
LayerTrim (Ours)	Width	834 (1.19×)	48.8 (1.18×)	37.5 (84%)	57.8 (82.2%)	781 (1.27×)	46.3 (1.24×)	35.4 (79%)	45.7 (65.0%)

Table 3: Inference efficiency under matched parameter budgets across three RLMS. We report prefill latency, decode latency per token, peak GPU memory, and average task accuracy. Parentheses indicate speedup over the dense model for latency, relative memory usage for peak GPU memory, and accuracy retention relative to the dense model. All methods are evaluated using their deployed pruned structures.

parameters, panel 4b also shows the highest overlap with the dense model’s top-8 attended tokens across all query–key distance bins, including 16K–32K. These results suggest that MLP-only trimming better preserves attention to earlier tokens during long reasoning, providing an attention-level explanation for why MLP trimming better preserves reasoning performance.

Feature-Importance Surrogate. To reduce calibration overhead, we approximate the squared MLP output change by summing feature importance scores without computing cross-feature terms. For a set R of removed features, we compare this approximation with the squared output change measured on the same dense-model inputs:

$$\Delta_\ell(R) = \mathbb{E}_x \left[\left\| \sum_{j \in R} c_j^{(\ell)}(x) \right\|_2^2 \right] \approx \sum_{j \in R} s_j^{(\ell)} \quad (4)$$

Panel 4c shows that the scores rank the measured changes across held-out layer–width conditions, with Spearman $\rho = 0.999$. Table 7 in Appendix C.1 reports comparable average accuracy to full-error greedy selection with $13.2\times$ faster calibration and selection. These results support using the low-overhead approximation to guide feature

selection.

5.2 Quantitative Analysis of Reasoning Traces

Figure 5 complements the experimental results in Table 1 with diagnostics of the generated outputs and reasoning paths. Panel 5a examines GPQA-Diamond responses from Phi-4-reasoning-plus at $L_p/L_t = 9/40$, measuring the ratio of responses reaching the max token limit and exhibiting strong repetition. Here, we flag strong repetition when two consecutive, non-overlapping 512-token windows each satisfy $1 - U_8/N_8 \geq 0.8$, where U_8 and N_8 denote the numbers of unique and total token 8-grams. LaCo exhibits 100% token-limit hits despite 0% detected strong repetition: its outputs continue as incoherent token fragments without forming the sustained exact-token repetition captured by this criterion (see Appendix E). LayerTrim achieves token-limit hits and strong repetition rates of 24.4% and 21.9%, respectively, compared with 70.2% and 49.2% for FLAP. Together with its higher accuracy, these results show that LayerTrim combines stronger task performance with substantially fewer repetitive and budget-exhausting outputs.

Panel 5b evaluates code formulation on HumanEval using Qwen3-14B at $L_p/L_t = 9/40$. The compilable-code rate measures the ratio of re-

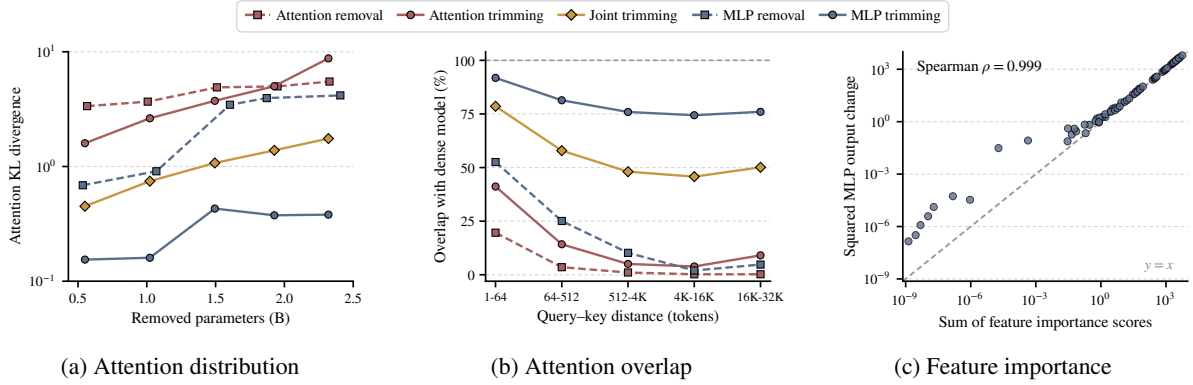


Figure 4: Attention preservation and feature importance on Qwen3-14B. (a) Attention KL divergence across pruning budgets. (b) Overlap with the dense model’s top-8 attended tokens by query-key distance, with approximately 1.5B parameters removed. (c) Summed feature importance scores versus squared MLP output change. Experimental details are provided in Appendix D.

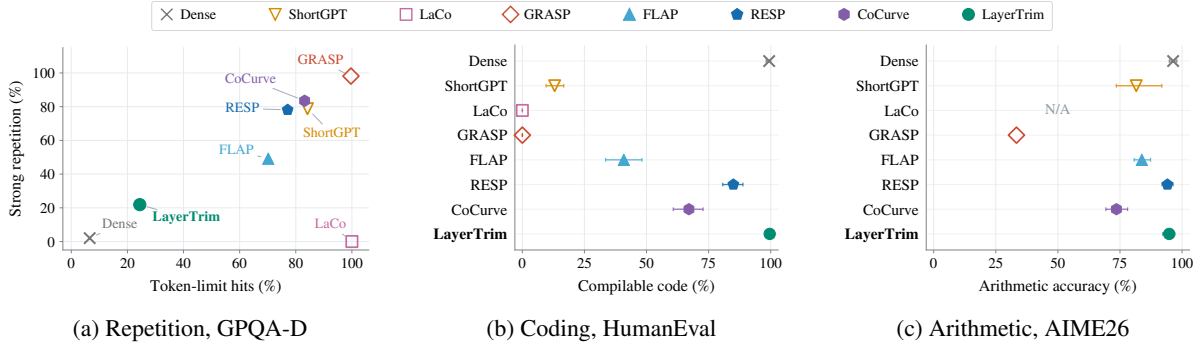


Figure 5: Quantitative analysis results comparing LayerTrim to existing pruning methods

sponses containing at least one extracted Python code that compiles and defines the requested function. LayerTrim yields 99.6%, close to Dense’s 99.4%, while the second-best RESP yields 85.0%. Its advantage also extends to functional problem solving: among responses containing a qualifying candidate, benchmark correctness is 88.4% for LayerTrim vs. 72.7% for RESP. Thus, the overall HumanEval accuracy gap in Table 1 (88.0% versus 62.0%) is accompanied by advantages in both forming compilable code and producing correct solutions when such code is present.

Panel 5c examines arithmetic consistency on AIME26 using Phi-4-reasoning-plus at $L_p/L_t = 7/40$. We broadly extract numeric equalities, including those embedded in prose or chained expressions, deduplicate them within each response, and report the fraction that hold under exact arithmetic evaluation. N/A in LaCo denotes that no eligible equality was extracted. LayerTrim attains 94.8% consistency, within 1.6pp of Dense’s 96.4%, compared with 83.8% for FLAP and 73.6% for CoCurve. While RESP exhibits similar arith-

metic consistency of 94.1%, LayerTrim achieves substantially higher accuracy (60.0% vs. 50.0%). This suggests that LayerTrim’s advantage extends beyond preserving individual arithmetic operations to better preserving the problem-level reasoning needed to integrate locally correct calculations into a correct final solution.

6 Conclusion

LayerTrim reduces the inference cost of reasoning language models by selectively trimming MLP features while preserving attention and decoder depth. It uses dense-model reasoning traces to identify redundant layers and retain high-importance features, without gradients or recovery training. Across three models and five benchmarks, LayerTrim achieves higher average accuracy than the evaluated structured pruning baselines at both pruning budgets, while achieving comparable inference speedups. LayerTrim offers a step toward more efficient deployment of reasoning language models through training-free structured pruning.

Limitations

Our evaluation focuses on three reasoning language models and five benchmarks covering mathematical reasoning, scientific reasoning, and code generation. Although LayerTrim outperforms the evaluated pruning baselines in most settings, experiments on additional model architectures, scales, and reasoning tasks would help assess its generalizability. We also evaluate fixed pruning budgets and layer allocation settings. Adapting these choices to individual models and tasks may further improve the trade-off between model compression and reasoning accuracy.

Ethical considerations

Structured pruning modifies the internal architecture of language models and may affect their reasoning behavior. Although LayerTrim aims to preserve reasoning accuracy, pruned models may still produce incorrect answers, incomplete reasoning traces, or unexpected outputs. These risks should be considered when deploying compressed models in applications where reasoning accuracy is critical. Our experiments use publicly available models and benchmarks and do not involve human participants or the collection of personal information.

References

- Marah Abdin, Sahaj Agarwal, Ahmed Awadallah, Vidhisha Balachandran, Harkirat Behl, Lingjiao Chen, Gustavo de Rosa, Suriya Gunasekar, Mojan Javaheripi, Neel Joshi, Piero Kauffmann, Yash Lara, Caio César Teodoro Mendes, Arindam Mitra, Besmira Nushi, Dimitris Papailiopoulos, Olli Saarikivi, Shital Shah, Vaishnavi Shrivastava, and 4 others. 2025. [Phi-4-reasoning technical report](#). *arXiv preprint arXiv:2504.21318*.
- Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. 2024. [Fluctuation-based adaptive structured pruning for large language models](#). In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*. AAAI Press.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. [Evaluating large language models trained on code](#). *arXiv preprint arXiv:2107.03374*.
- DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shiron Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 176 others. 2025. [Deepseek-r1 incentivizes reasoning in llms through reinforcement learning](#). *Nature*, 645(8081):633–638.
- Longwei Ding, Anhao Zhao, Fanghua Ye, Ziyang Chen, and Xiaoyu Shen. 2026. [From LLMs to LRMs: Rethinking pruning for reasoning-centric models](#). *arXiv preprint arXiv:2601.18091*.
- Elias Frantar and Dan Alistarh. 2023. [SparseGPT: Massive language models can be accurately pruned in one-shot](#). In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10323–10337. PMLR.
- Zhiren Gong, Zihao Zeng, Tiantong Wang, Yixin Wang, Honoka Anada, Zijie Wang, Ming Xiao, Chau Yuen, and Wei Yang Bryan Lim. 2026. [CoCurve: Cross-module co-pruning curvature for structured LLM pruning](#). *arXiv preprint arXiv:2607.17568*.
- Minkyu Kim, Vincent-Daniel Yun, Youngra Kim, Suin Cho, Woosang Lim, and Sunwoo Lee. 2026. [Rethinking layer redundancy: Calibration matters more than search in LLM depth pruning](#). *arXiv preprint arXiv:2604.24938*.
- Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. [Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 21558–21572. Curran Associates, Inc.
- Kainan Liu, Yong Zhang, Ning Cheng, Zhitao Li, Shaojun Wang, and Jing Xiao. 2025. [GRASP: Replace redundant layers with adaptive singular parameters for efficient model compression](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 26333–26348, Suzhou, China. Association for Computational Linguistics.
- Ryan Lucas, Kayhan Behdin, Zhipeng Wang, Qingquan Song, Shao Tang, and Rahul Mazumder. 2026. [Reasoning models can be accurately pruned via chain-of-thought reconstruction](#). In *International Conference on Learning Representations*, volume 2026, pages 142840–142852.
- Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. [LLM-Pruner: On the structural pruning of large language models](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 21702–21720. Curran Associates, Inc.
- Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2025. [ShortGPT: Layers in large language models are more redundant than you](#)

- [expect](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20192–20204, Vienna, Austria. Association for Computational Linguistics.
- Mistral-AI, Abhinav Rastogi, Albert Q. Jiang, Andy Lo, Gabrielle Berrada, Guillaume Lample, Jason Rute, Joep Barmantlo, Karmesh Yadav, Kartik Khandelwal, Khyathi Raghavi Chandu, Léonard Blier, Lucile Saulnier, Matthieu Dinot, Maxime Darrin, Neha Gupta, Roman Soletskyi, Sagar Vaze, Teven Le Scao, and 81 others. 2025. [Magistral](#). *arXiv preprint arXiv:2506.10910*.
- Ha Lan Nguyen, Huy Hoang Tran, Trac-Duy Tran, and Dung D. Le. 2026. [OBC-Prune: Outcome-based calibration for large reasoning model pruning](#). *arXiv preprint arXiv:2609.17890*.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. [GPQA: A graduate-level Google-Proof Q&A benchmark](#). In *First Conference on Language Modeling*.
- Noam Shazeer. 2020. [GLU variants improve transformer](#). *arXiv preprint arXiv:2002.05202*.
- Mingjie Sun, Zhuang Liu, Anna Bair, and Zico Kolter. 2024. [A simple and effective pruning approach for large language models](#). In *International Conference on Learning Representations*, volume 2024, pages 4942–4964.
- Ziyan Wang, Enmao Diao, Qi Le, Pu Wang, Minwoo Lee, Shu-ping Yeh, Evgeny Stupachenko, Hao Feng, and Li Yang. 2026. [From local to global: Revisiting structured pruning paradigms for large language models](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 35720–35739, San Diego, California, United States. Association for Computational Linguistics.
- Ziyan Wang, Enmao Diao, Qi Le, Pu Wang, Guanchu Wang, Minwoo Lee, Shu-ping Yeh, and Li Yang. 2025. [Think before you prune: Self-reflective structured pruning for reasoning language models](#). *arXiv preprint arXiv:2512.02185*. Preprint.
- Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2024. [Sheared LLaMA: Accelerating language model pre-training via structured pruning](#). In *International Conference on Learning Representations*, volume 2024, pages 5385–5409.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. [Qwen3 technical report](#). *arXiv preprint arXiv:2505.09388*.
- Yifei Yang, Zouying Cao, and Hai Zhao. 2024. [LaCo: Large language model pruning via layer collapse](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6401–6417, Miami, Florida, USA. Association for Computational Linguistics.
- Vincent-Daniel Yun, Youngra Kim, Woosang Lim, YoungJin Heo, Minkyu Kim, and Sunwoo Lee. 2026. [Locality-aware redundancy pruning for LLM depth compression](#). *arXiv preprint arXiv:2605.27786*.
- Yifan Zhang and Team Math-AI. 2025. [American invitational mathematics examination \(AIME\) 2025](#).
- Yifan Zhang and Team Math-AI. 2026. [American invitational mathematics examination \(AIME\) 2026](#).
- Yijun Zhu, Jianxin Wang, and Chengchao Shen. 2026. [High-fidelity pruning for large language models](#). *arXiv preprint arXiv:2603.08083*.

Appendix

A LayerTrim Procedure

Algorithm 1 summarizes the procedure in Section 3.2 for uniform gated MLPs. Here, L is the decoder depth, C and I are the hidden and intermediate widths, and P_{layer} counts the attention and MLP projection parameters in one original block. Calibration averages are taken over token positions.

Algorithm 1 LayerTrim: reasoning-calibrated structured MLP pruning.

Require: Dense model $\mathcal{M}$; calibration prompts $\mathcal{Q}$; layer-equivalent budget N

Ensure: Compressed model $\tilde{\mathcal{M}}$ with unchanged attention and depth

- 1: Generate reasoning traces $\mathcal{T} \leftarrow \text{GENERATE}(\mathcal{M}, \mathcal{Q})$.
 - 2: Form $\mathcal{D}$ from each complete prompt and up to 2,048 generated tokens of its trace in $\mathcal{T}$.
 - 3: With gradients disabled, teacher-force $\mathcal{M}$ on $\mathcal{D}$ and accumulate block cosine similarities and feature second moments $\hat{a}_j^{(\ell)} = \mathbb{E}_{\mathcal{D}}[(\phi_j^{(\ell)})^2]$.
 - 4: **for** $\ell = 1, \dots, L$ **do**
 - 5: $\text{BI}_\ell \leftarrow 1 - \mathbb{E}_{x \sim \mathcal{D}} [\cos(h_{\text{in}}^{(\ell)}(x), h_{\text{out}}^{(\ell)}(x))]$.
 - 6: **end for**
 - 7: $K \leftarrow 2N$; require $K \leq L$.
 - 8: $\mathcal{L}_K \leftarrow \text{BOTTOMK}(\{\text{BI}_\ell\}_{\ell=1}^L, K)$.
 - 9: $m \leftarrow \text{round}(I - P_{\text{layer}}/(6C))$.
 - 10: Require $0 < m \leq I$.
 - 11: $\tilde{\mathcal{M}} \leftarrow \text{COPY}(\mathcal{M})$.
 - 12: **for all** $\ell \in \mathcal{L}_K$ **do**
 - 13: **for** $j = 1, \dots, I$ **do**
 - 14: $\hat{s}_j^{(\ell)} \leftarrow \hat{a}_j^{(\ell)} \|W_{\text{down}}^{(\ell)}[:, j]\|_2^2$.
 - 15: **end for**
 - 16: $S_\ell^* \leftarrow \text{TOPM}(\{\hat{s}_j^{(\ell)}\}_{j=1}^I, m)$.
 - 17: $\tilde{W}_{\text{gate}}^{(\ell)} \leftarrow W_{\text{gate}}^{(\ell)}[S_\ell^*, :]; \tilde{W}_{\text{up}}^{(\ell)} \leftarrow W_{\text{up}}^{(\ell)}[S_\ell^*, :]$.
 - 18: $\tilde{W}_{\text{down}}^{(\ell)} \leftarrow W_{\text{down}}^{(\ell)}[:, S_\ell^*]$.
 - 19: Install these projections in $\tilde{\mathcal{M}}$ and set this MLP’s width to m .
 - 20: **end for**
 - 21: **return** $\tilde{\mathcal{M}}$
-

TOPM returns the indices of the m highest-scoring features. The same indices select gate and up rows and down-projection columns.

B Main Evaluation Implementation Details

Pruning budgets. For each model, we set the pruning budget to $B_N = NP_{\text{layer}}$, where P_{layer} is the number of attention and MLP projection parameters in one original decoder layer and $N \in \{7, 9\}$. All methods use the same target budget, with small differences in actual removal counts because heads and channels are removed in whole units.

LayerTrim allocation. LayerTrim selects $K = 2N$ MLPs using Block Influence and retains the

same width m in each selected MLP. Removing one gated-MLP feature affects a gate row, an up row, and a down-projection column, totaling $3C$ parameters. The realized budget is $3CK(I - m)$, with m set by Equation 3.

Calibration settings. All methods use the same 24 AIME 2024 reasoning traces generated by the corresponding dense model at temperature 0.6, top- p 0.95, and seed 0. LayerTrim uses the complete prompt and up to the first 2,048 generated tokens from each trace for statistics collection.

Baseline implementations. For all baselines except RESP, we use the authors’ official implementations, adapted to our models and matched-budget. We reimplement RESP using the public GISP (Wang et al., 2026) repository, following the GISP-based structural pruning procedure described in the RESP paper (Wang et al., 2025), with decode-only scoring and calibration regeneration. All baselines use the shared calibration data described above.

Generation settings. Dense and pruned models use the corresponding native reasoning-enabled chat template. Magistral uses its native Tekken tokenizer and official system prompt. Sampling uses temperature 0.6, top- p 0.95, and seeds 0, 1, and 2. Each seed produces one response per problem. We set the maximum output length to 32K tokens. Inference uses BF16 in vLLM.

Answer and code scoring. We evaluate AIME and GPQA-Diamond by exact matching of the extracted final answers. For coding benchmarks, we report pass@1 using HumanEval’s original tests and MBPP+’s extended tests.

B.1 Calibration Overhead

We compare LayerTrim and RESP on Qwen3-14B in BF16 using a single NVIDIA RTX 6000 Ada (48 GB). Both use the same 24 AIME 2024 calibration traces, with up to 2,048 generated tokens per trace. We measure calibration time from model loading until the pruning masks are computed. RESP’s trace regeneration and required intermediate checkpoint preparation are included. Initial trace generation and final weight export are excluded.

LayerTrim requires substantially less preparation time than RESP at both pruning budgets (Table 4).

Method	$N = 7$	$N = 9$
RESP	$2,626.1 \pm 18.2$	$3,898.1 \pm 18.0$
LayerTrim (Ours)	27.9 ± 0.6	27.9 ± 0.6

Table 4: Pruning preparation time (seconds) on Qwen3-14B. Values are mean $\pm$ sample std over three runs.

C Ablation Experimental Details

Evaluation and reporting. Table 2 evaluates Qwen3-14B on AIME25 and GPQA-Diamond using the main evaluation protocol. The repeated default entries reuse the same evaluated configuration.

Pruning targets. The first group uses the pruning configurations of Figure 2(b). All three trimming conditions remove approximately 1.494B parameters; attention and MLP removal delete approximately 1.510B and 1.604B, respectively, because whole-module removal has discrete budgets. Joint trimming assigns 25% of its budget to attention and 75% to MLPs. Attention trimming removes selected dimensions within each attention head.

Design-ablation budget and calibration. The remaining three groups fix the budget at $N = 7$ (approximately 2.312B removed parameters) and use the default reasoning-trace calibration in Section 4.1.

Feature-importance scores. The feature-score group fixes the 14 Block-BI-selected MLPs and their retained widths. Only the within-MLP ranking score changes. Here, ϕ_j denotes an intermediate activation and $w_j = W_{\text{down}}[:, j]$ denotes its down-projection column. The default score is $\mathbb{E}[\phi_j^2] \|w_j\|_2^2$. We compare it with activation-only ranking using $\mathbb{E}[\phi_j^2]$, weight-only ranking using $\|w_j\|_2^2$, and variance-based ranking using $\text{Var}(\phi_j) \|w_j\|_2^2$.

Layer selection and distribution. The layer-selection group fixes the feature score and total removal budget. Block BI, MLP-only BI, and random selection each choose $K = 14$ MLPs. MLP-only BI measures cosine distance between the residual stream before and after the MLP update. Random selection uses one random fixed mask. All-layer allocation distributes the same removal budget across all 40 MLPs.

Number of selected MLPs. The allocation group fixes the Block BI ranking and default feature score

Prefix	AIME25	GPQA-D	HumanEval	Avg.
512	52.2 ± 3.8	52.9 ± 1.3	87.8 ± 2.7	64.3
1,024	62.2 ± 6.9	53.5 ± 1.8	89.0 ± 0.6	68.3
2,048	64.4 ± 5.1	49.7 ± 2.9	90.2 ± 1.6	68.1
4,096	68.9 ± 6.9	51.5 ± 5.0	91.9 ± 0.4	70.8

Table 5: Calibration-length ablation on Qwen3-14B at $N = 7$. Prefix is the maximum number of generated reasoning tokens per trace; each complete prompt is also included. HumanEval reports pass@1 (%); the other benchmarks report accuracy (%). The pruning budget and $K = 14$ are fixed.

while varying $K \in \{11, 14, 21\}$. Retained widths are adjusted to preserve the same total removal budget; $K = 1.5N$ is rounded to 11. Smaller K concentrates pruning in fewer MLPs.

C.1 Additional Accuracy Ablations

These experiments follow the main evaluation protocol and reuse previously evaluated controls. Avg. averages the displayed benchmark means before rounding.

Calibration length. We vary the number of generated reasoning tokens used to collect statistics on the same 24 AIME 2024 traces, retaining each complete prompt. For Qwen3-14B at $N = 7$, we recompute Block BI and feature importance at each length while keeping $K = 14$ and the parameter budget fixed. Accuracy is stable beyond 1,024 tokens (Table 5); we use 2,048 as the default.

Calibration source	AIME25	GPQA-D	Avg.
WikiText-2	0.0 ± 0.0	20.0 ± 2.3	10.0
AIME prompts only	22.2 ± 6.9	38.2 ± 4.7	30.2
AIME reasoning (Ours)	64.4 ± 5.1	49.7 ± 2.9	57.1

Table 6: Calibration-source ablation on Qwen3-14B at $N = 7$. Entries are accuracy (%); Avg. averages the two benchmarks. WikiText-2 and reasoning traces are token-matched; prompt-only calibration uses fewer tokens.

Calibration source. We compare AIME 2024 reasoning traces, token-matched WikiText-2 text, and the same 24 AIME prompts without generated continuations on Qwen3-14B at $N = 7$. We recompute Block BI and feature importance for each source, keeping the pruning budget and $K = 14$ fixed. Reasoning traces yield higher average accuracy than WikiText-2 or prompts alone (Table 6).

Selection criterion	AIME25	GPQA-D	HumanEval	Avg.
Squared MLP output change	60.0 $\pm$ 6.7	52.5 $\pm$ 1.5	91.9 $\pm$ 1.5	68.1
Feature importance (Ours)	64.4 $\pm$ 5.1	49.7 $\pm$ 2.9	90.2 $\pm$ 1.6	68.1

Table 7: Comparison of feature-selection criteria on Qwen3-14B at $N = 7$. HumanEval reports pass@1 (%); the other benchmarks report accuracy (%). Both methods use the same removal budget.

Cross-feature terms in feature selection. We compare feature-importance ranking with greedy selection based on the squared MLP output change (Equation 4). Both use Qwen3-14B at $N = 7$ with the same calibration traces, BI-selected layers, and retained widths. The squared output change includes cross-feature terms $\mathbb{E}[\phi_i \phi_j] w_i^\top w_j$ ($i \neq j$), which feature importance omits. Both methods achieve 68.1% average accuracy (Table 7), supporting feature importance as a lower-cost selection criterion with comparable average accuracy.

Full-error calibration and selection take $13.2\times$ as long as diagonal ranking on the same hardware.

D Representation error and Attention Diagnostics Experimental Details

Representation error. Figure 2(b),(d) compares dense and pruned models on the same four held-out AIME25 traces. Each trace is truncated to 2,048 total tokens, with 256 continuation positions sampled for measurement. Final-hidden NMSE is

$$\frac{\sum_t \|h_t^{\text{pruned}} - h_t^{\text{dense}}\|_2^2}{\sum_t \|h_t^{\text{dense}}\|_2^2}$$

where t ranges over measured positions across the traces. In panel (b), we apply trimming to all 40 layers.

Attention measurements. Panel 4(a)–(b) measures attention in the final two decoder layers (38 and 39) using three dense-model AIME25 traces truncated to 32K total tokens. Panel 4(a) reports KL divergence from dense to pruned attention distributions across pruning budgets, measured by the number of removed parameters. Both panels exclude attention to the first token and rescale the remaining probabilities to sum to one.

Attention overlap. For each query token and attention head, we select the eight earlier tokens that receive the highest attention weights from that query. We select these tokens separately in the dense and pruned models. Panel 4 (b) shows the fraction of the dense model’s top-eight tokens that

remain in the pruned model’s top eight, grouped by how far each token precedes the query. Higher overlap means that pruning better preserves which earlier tokens the model attends to.

Pruning budgets. Panel 4(b) uses approximately 1.5B removed parameters. The component configurations and discrete-budget differences are the same as the first group of Table 2, described in Appendix C.

Feature importance and output change. Panel 4(c) compares summed feature importance with the measured MLP output change across 70 combinations of layers and retained widths on four held-out AIME25 traces. Each point corresponds to a feature set R removed using the calibration ranking. For activation ϕ_j and output weight vector w_j , the horizontal axis sums the importance of the removed features:

$$\sum_{j \in R} \mathbb{E}[\phi_j^2] \|w_j\|_2^2$$

The vertical axis measures the squared MLP output change when these features are removed together, using the same inputs:

$$\mathbb{E} \left\| \sum_{j \in R} w_j \phi_j \right\|_2^2$$

Both quantities are averaged over the held-out inputs. Feature importance ignores cross-feature terms, while the measured output change includes them. Their strong rank agreement shows that feature importance reliably identifies which pruning choices cause larger output changes.

E Qualitative Analysis

The following examples supplement Figure 5 with excerpts from the evaluated reasoning and coding outputs. They illustrate repetition, incoherent token fragments, code-format failures, and arithmetic errors; they are not prevalence estimates. In particular, lexical collapse need not satisfy the exact-token repetition criterion in the main text, explaining LaCo’s low detected repetition despite frequent token-limit hits.

[illegible]

Method	Original trace excerpts and qualitative observations
Dense 1,348 tokens Terminated	Reasoning: [...] So for each i in range(len(numbers)), and for each j in range(i+1, len(numbers)), compare numbers[i] and numbers[j]. Final response: [...] <pre>def has_close_elements(numbers: List[float], threshold: float) -> bool: for i in range(len(numbers)): for j in range(i + 1, len(numbers)): if abs(numbers[i] - numbers[j]) < threshold: return True return False</pre> [...] <i>Observation:</i> Distinct-index comparisons are preserved in the final function.
ShortGPT 337 tokens Terminated	Reasoning: [...] The function should check if any two numbers in the list are within the given threshold. The function should be written in a Python block with the proper indentation. Final response: [...] The function uses the '**' operator to check if any two numbers in the list are within the given threshold. The function returns 'True' if any two numbers are within the given threshold, and 'False' otherwise. The function is as described, and the code is correct. <i>Observation:</i> The response claims to provide an implementation, but supplies no comparison logic.

LaCo 32,256 tokens Limit hit	[...] imer Spo aggregateTimelineaslasca Jenner官TC户攀筹progressbar共创的高度GD ? 栳cket clam命运随手oidantom迎接系列产 品movement踩春节路上功课NPCast人性化刹档埠勿官- [...] <i>Observation:</i> These mixed-script fragments do not form an interpretable plan or function.
GRASP 32,256 tokens Limit hit	from typing import List, float, 2.0, 3.0, 4.0, 0.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 [...] 0.0: 0.0: 0.0: 0.0: 0.0: 0.0: 0.0: 0.0: 0.0: 0.0: 0.0: 0.0: 0.0: 0.0: 0.0: 0.0: 0.0: 0.0: 0.0: <i>Observation:</i> A malformed import leads into numeric repetition, rather than a function body.
FLAP 4,122 tokens Terminated	Reasoning: [...] So, I can sort the list first, then check each consecutive pair. If any pair has a difference less than the threshold, then the answer is True. Otherwise, it's False. Final response: [...] def has_close_elements(numbers: List[float], threshold: float) -> bool: sorted_numbers = sorted(numbers) for i in range(len(sorted_numbers) - 1): if sorted_numbers[i+1] - sorted_numbers[i] < threshold: return True return False [...] <i>Observation:</i> The intended algorithm survives, but the final if and return statements are incorrectly indented.
RESP 3,176 tokens Terminated	Reasoning: [...] Therefore, if the list is sorted, then the minimum difference between any two elements is the minimum of the adjacent differences. [...] if len(numbers) < 2: return False numbers_sorted = sorted(numbers) Final response: [...] if len(numbers) < 2: return False numbers_sorted = sorted(numbers) [...] <i>Observation:</i> Correct relative indentation is lost when the full function is converted to a body-only completion.
CoCurve 115 tokens Terminated	Reasoning (empty): <think> </think> Final response: [...] numbers = set(numbers) # Remove duplicates to consider each number only once for num in numbers: if any(abs(num - x) < threshold for x in numbers): # Check if any number is within the threshold of the given number return True return False [...] <i>Observation:</i> The code compiles, but compares each value with itself; it incorrectly accepts the prompt's first example.
LayerTrim (Ours) 2,699 tokens Terminated	Reasoning: [...] But the problem says "closer to each other than given threshold". So, if the difference is exactly equal to the threshold, then it's not considered. For example, if threshold is 0.5, and two elements are 0.5 apart, then it's not considered. So the condition is strictly less than. Final response: [...] if len(numbers) < 2: return False sorted_numbers = sorted(numbers) for i in range(len(sorted_numbers) - 1): if sorted_numbers[i+1] - sorted_numbers[i] < threshold: return True return False [...] <i>Observation:</i> The final body preserves strict comparison and consistent indentation in the original prompt context.

<i>Question:</i> Let N be the number of positive integer divisors of 17017^{17} that leave a remainder of 5 when divided by 12. Find the remainder when N is divided by 1000.	
Method	Original trace excerpts and qualitative observations
Dense 3,907 tokens Terminated	[...] $7 * 11 = 77$, $77 * 13 = 1001$, $1001 * 17 = 17017$. [...] So indeed, the valid cases are: (a even, d odd, b even) and (a odd, d even, b odd). And each gives 9 possibilities for a, 9 for b, 9 for d. So total $9^3 + 9^3 = 2 * 729 = 1458$. And then times 18 choices for c yields $1458 * 18 = 26244$. Then remainder mod 1000 is $26244 \bmod 1000 = 244$. [...] <i>Observation:</i> The two parity cases and independent exponent choices produce the correct count.

Table 10: Qualitative analysis of example reasoning traces on AIME 2026 using Phi-4-reasoning-plus at $L_p/L_t = 7/40$ and seed 0. [...] denotes omitted text.

F Licenses and Terms of Use

All models and datasets used in this study are publicly available for research use. We follow their respective licenses and usage conditions and use these resources solely for non-commercial research.

Models. Qwen3-14B (Yang et al., 2025) and Magistral-Small-2506 (Mistral-AI et al., 2025) are released under the Apache 2.0 License. Phi-4-reasoning-plus (Abdin et al., 2025) is released under the MIT License.

Evaluation datasets. AIME2025 (Zhang and Math-AI, 2025) and AIME2026 (Zhang and Math-AI, 2026) are distributed through the Math-AI benchmark releases under the Apache 2.0 License. The original AIME problems are subject to the copyright of the Mathematical Association of America. GPQA-Diamond (Rein et al., 2024) is released under the CC BY 4.0 License, with additional dataset access and redistribution conditions. HumanEval (Chen et al., 2021) is distributed under the MIT License, while MBPP+ (Liu et al., 2023) is distributed through EvalPlus under the Apache 2.0 License.

Calibration data. We use AIME2024 problems for calibration, with reasoning traces generated by the original dense models. The original examination problems are subject to the copyright and usage conditions of the Mathematical Association of America. The generated reasoning traces are used for calibration and diagnostic analysis. The calibration-source ablation also uses WikiText-2, whose dataset card lists CC BY-SA 3.0 and GFDL licenses.¹

Software. Hugging Face Transformers is released under the Apache 2.0 License, while PyTorch is distributed under a BSD-style license. All software is used in accordance with its respective licensing terms.

¹<https://huggingface.co/datasets/Salesforce/wikitext>