On the Limits of Layer Pruning for Generative Reasoning in Large Language Models

Anonymous ACL submission

Abstract

Recent work has shown that layer pruning can effectively compress large language models (LLMs) while retaining strong performance on classification benchmarks, often with little or no finetuning. In contrast, generative reasoning tasks, such as GSM8K and HumanEval$^+$, exhibit substantially weaker recovery. We show that beyond surface-level text degradation, pruning leads to a loss of key algorithmic capabilities, including arithmetic computation and balanced parenthesis generation. Under realistic post-training constraints, without access to pretraining-scale data or compute, we evaluate a minimal recovery strategy based on supervised finetuning with self-generated responses. This approach recovers up to 90% of baseline performance on classification tasks, but recovery for generative reasoning remains fundamentally limited. Notably, even models finetuned on $\sim$100B tokens after pruning fail to recover their original reasoning performance, suggesting that such capabilities are not as easily restored. This limitation persists even on simple tasks such as arithmetic, which do not require multi-step generation. Overall, we characterize the practical limits of layer pruning for generative reasoning and provide guidance on when depth reduction is effective under constrained post-training regimes.[1]

1 Introduction

Large Language Models (LLMs) have achieved remarkable performance across a wide range of tasks, a success often attributed to their large parameter counts and extensive training data (Hoffmann et al., 2022; Yang et al., 2025a; Grattafiori et al., 2024). However, the scale of modern LLMs raises significant concerns regarding efficiency and costs (LeCun et al., 1989; Wan et al., 2023; Song et al., 2024). These challenges have motivated a substantial body of work on model compression techniques aimed at reducing model size while preserving performance, including pruning at multiple granularities ranging from individual neurons to entire layers (Frantar and Alistarh, 2023; Sun et al., 2024; Muralidharan et al., 2024; Sreenivas et al., 2024; Song et al., 2024; Ashkboos et al., 2024; Ma et al., 2023; Ling et al., 2024).

Among these approaches, layer pruning has emerged as a particularly appealing strategy. By removing entire transformer blocks of contemporary decoder-only models, layer pruning offers a simple method for reducing model depth, often requiring minimal or no additional finetuning (Song et al., 2024; Yang et al., 2024; Lu et al., 2024; Men et al., 2025; Kim et al., 2024; Chen et al., 2024). This approach is further motivated by theoretical and empirical works suggesting redundancy across layers in LLMs (Sun et al., 2025; Lad et al., 2024; Men et al., 2025). Furthermore, layer pruning is largely orthogonal to other efficiency techniques such as quantization and sparsification, enabling it to be combined with complementary methods for additional computational savings (Song et al., 2024; Kim et al., 2024).

While this approach has achieved notable success in classification benchmarks, it has proven far less effective for reasoning-intensive generative tasks like math and coding, which require the model to generate a multi-step chain of thought to arrive at the correct solution (Lu et al., 2024; Chen et al., 2024; Men et al., 2025; Yang et al., 2024; Kim et al., 2024; Nepal et al., 2025). Prior work has largely attributed the failure of layer pruning on generative tasks to the importance of deeper layers for "reasoning," without explicitly characterizing how layer removal degrades model behavior (Wang et al., 2025; Song et al., 2025). Moreover, existing methods that partially recover generative performance typically require access to large-scale data (in billions of tokens) and compute, which can be

[1] Code available at https://anonymous.4open.science/r/limit-of-layer-pruning-0167

impractical (Muralidharan et al., 2024; Sreenivas et al., 2024). These limitations motivate a closer analysis of pruning-induced failure modes and an examination of how much recovery is achievable under realistic post-training constraints (e.g., limited to a single ∼80GB GPU and moderate-scale data). Rather than proposing a new pruning algorithm, our goal is to characterize the limits of layer pruning for generative reasoning and to identify practical regimes in which it remains viable.

In this paper, we make the following contributions:

- **We establish a controlled and realistic evaluation setting for post-pruning recovery.** Using self-generated responses (SGR) as a simple and strong baseline, we isolate the extent to which pruned models can recover capabilities under practical constraints (single 80GB GPU, no access to pretraining-scale data).
- **We systematically characterize the limitations of layer pruning for generative reasoning under realistic post-training constraints.** While prior work reports weaker performance on generative tasks, we show that this gap persists even with SGR across multiple model families, despite strong improvements on classification benchmarks.
- **We identify concrete failure modes underlying this degradation.** Beyond surface-level text degeneration, we show that pruning disrupts core algorithmic capabilities, including arithmetic computation and syntactic structure (e.g., parenthesis tracking), which are essential for multi-step reasoning.
- **We demonstrate that these limitations persist even under large-scale distillation.** Using the Minitron model trained on ∼100B tokens after pruning, we show that substantial additional training still fails to recover reasoning and even simple arithmetic capabilities.
- **We provide practical guidance on when layer pruning is viable.** Our results suggest that depth reduction is effective for classification and shallow tasks, but should be applied conservatively when generative reasoning or algorithmic capabilities are required.

Overall, our results clarify when and why layer pruning succeeds or fails, and provide practical guidance for its use in settings where preserving generative reasoning ability is a priority.

2 Layer Pruning and Recovery

Layer pruning has been widely studied as a simple and effective compression strategy for LLMs (Men et al., 2025; Kim et al., 2024; Song et al., 2024; Yang et al., 2024; Chen et al., 2024). In this work, we study a standard post-pruning pipeline under compute-constrained settings: (1) remove N layers according to a fixed strategy, and (2) recover performance through post-training. We consider two commonly used strategies for selecting layers: Block Influence (BI) and Reverse Order (Men et al., 2025; Muralidharan et al., 2024; Wang et al., 2025; Lu et al., 2024). BI removes layers whose output activations exhibit high cosine similarity with their inputs, as measured on a calibration dataset (e.g., C4 (Dodge et al., 2021)), while Reverse Order prunes layers from the tail end of the model. Experiments in this paper are primarily conducted on Gemma2-2B-It, Llama-3.1-8B-It, Qwen-2.5-7B-Instruct, and Mistral-7B-Instruct-v0.3.

Post-Training via Self-Distillation. Following pruning, performance is typically recovered via supervised finetuning on high-quality prompt–response datasets (Chen et al., 2024; Ma et al., 2023; Gromov et al., 2024; Lu et al., 2024; Wang et al., 2025; Xia et al., 2023). Prior work has also explored self-distillation approaches that refine or rewrite dataset responses using the base model (Thangarasa et al., 2025).

In this work, we adopt a simplified form of self-distillation in which we discard dataset responses entirely and instead use self-generated responses (SGR) on the prompts using the unpruned model. The resulting prompt–response pairs are then used to finetune the pruned model via QLoRA. This formulation removes dependence on reference outputs and isolates the extent to which the pruned model can recover capabilities from its own generated supervision. We use this approach as a simple and strong baseline for studying post-pruning recovery. Experiments are conducted on Alpaca-cleaned (Taori et al., 2023) and Dolci datasets (Ettinger et al., 2025).

Benchmarks. Following standard literature, in this paper, we evaluate on the following classification benchmarks: HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), MMLU (Hendrycks

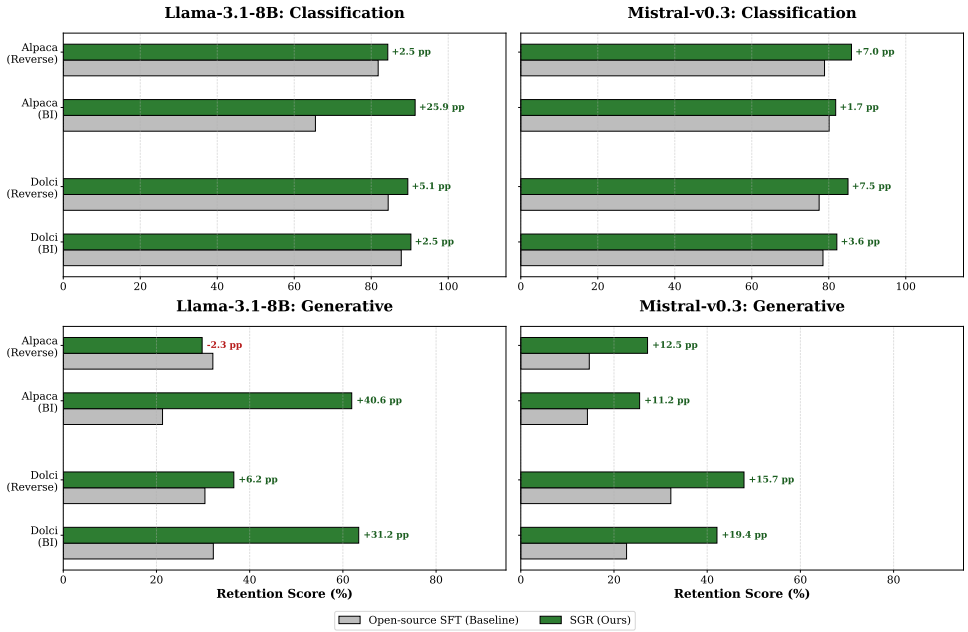


Figure 1: Comparison of performance retention (normalized to baseline) between SFT with open source data versus SGR (our method), with BI and Reverse pruning metrics at 25% pruning ratio. Full results are provided in Appendix A.9.

et al., 2020), WinoGrande (Sakaguchi et al., 2021), OpenBookQA (Mihaylov et al., 2018), ARC-E (Clark et al., 2018), and ARC-C (Clark et al., 2018). All classification results are computed using the `lm-evaluation-harness` framework (Gao et al., 2021). We also consider a diverse set of generative benchmarks spanning multiple domains, including GSM8K for mathematical reasoning (Cobbe et al., 2021), HumanEval$^+$ and MBPP$^+$ for code generation (Liu et al., 2023), and XSUM for summarization (Narayan et al., 2018). While in classification benchmarks, evaluation reduces to scoring a fixed set of candidate outputs via log-likelihood comparison, in generative benchmarks, we require the model to produce a sequence of tokens to arrive at a valid solution, often involving multi-step reasoning or structured generation. These tasks place substantially different demands on the model than classification benchmarks.

This particular pipeline allows us to perform pruning and training with QLoRA (Dettmers et al., 2023) on a single 80GB GPU which is ideal for resource-constrained setups. Full experimental details are provided in Appendix A.6. Although we use QLoRA for efficiency, we perform extensive comparisons showing that it achieves performance comparable to LoRA and even full finetuning (Table 1, Figure 11, Section A.6.2).

2.1 Results

As shown in Figure 1, supervised finetuning with self-generated responses (SGR) consistently yields substantially stronger improvements than standard finetuning on fixed open-source datasets, in line with Thangarasa et al. (2025). In particular, we observe that direct supervised finetuning on such datasets provides limited recovery after pruning, motivating the use of SGR as a stronger baseline for evaluating recoverability.

These gains are particularly pronounced for classification tasks. For example, for LLaMA pruned using the Block Influence (BI) metric, SGR training on Alpaca achieves an average retention of 91.4%, corresponding to a +25.9 percentage point improvement over finetuning on Alpaca alone. SGR also improves performance on generative tasks. For instance, the same LLaMA model pruned with BI achieves a +31.2 percentage point gain when finetuned on Dolci SGR relative to finetuning on Dolci directly. However, a clear disparity remains: while classification performance typically retains close to or above 80%, generative performance remains below 65% in most cases. We observe similar trends across other model families, including Gemma and Qwen (see Appendix A.9).

Overall, these results demonstrate that while stronger supervision such as SGR can substantially

improve recovery, especially for classification, a significant gap persists between classification and generative performance. For example, while the same LLaMA model retains nearly 90% accuracy on classification benchmarks, its retention on generative benchmarks is only 63.4% (Figure 1). Although some degradation is expected due to the increased difficulty of generative reasoning tasks, these findings suggest that such capabilities are fundamentally harder to recover, even under stronger post-training strategies. ***If preserving generative reasoning is a primary objective, aggressive layer pruning may be impractical.***

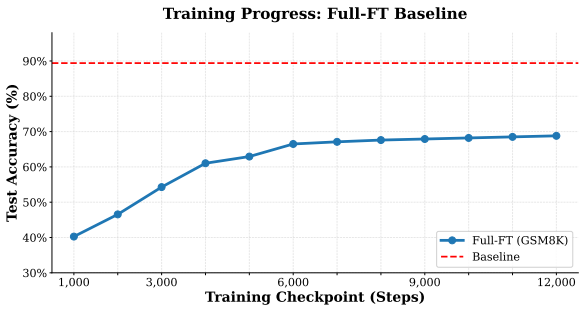


Figure 2: Full Finetuning with GSM8K dataset on the Qwen Pruned Model.

In addition, to estimate a reasonable upper bound on recoverable performance with layer pruning, we conduct an ablation experiment under highly favorable conditions. We perform full finetuning with SGR on the Qwen model, 7 layers pruned. We perform the finetuning on the GSM8K train split data and evaluate the model on the GSM8K test set. Despite these idealized conditions, Figure 2 shows that performance on GSM8K cannot be fully recovered after pruning (69% vs 90%). More details in Appendix A.6.2.

2.2 Case Study on Minitron

While our primary focus is to characterize recovery limits under resource-constrained post-training settings, we additionally analyze the `Llama-3.1-Minitron-4B-Depth` model (Sreenivas et al., 2024). This model is pruned at 50% where contiguous layers that lead to minimal loss in accuracy over a certain task are picked. Subsequently, the pruned model is trained via knowledge distillation on approximately 94B tokens, with the unpruned model as the teacher. To our knowledge, this is the only publicly available model that combines substantial layer pruning with large-scale post-training. Despite this extensive training, we further finetune the model using our SGR-based procedure on Dolci under the same experimental setup as earlier sections, ensuring it is adapted for reasoning-intensive tasks.

Nevertheless, performance remains significantly degraded. The model achieves 23.9% accuracy on GSM8K and 15.34% on MBPP$^+$, corresponding to relative retention of 49.2% and 34.3%, respectively, compared to the unpruned baseline. These results are striking: even after nearly 100B tokens of distillation-based training followed by additional supervised finetuning, the model fails to recover more than half of its original performance on reasoning benchmarks. In contrast to classification tasks, where recovery is largely achievable even in our minimalist setup, we see strong evidence that generative reasoning capabilities are intrinsically sensitive to depth reduction and are not easily restored. We might need even more data and compute to ensure that the model learns to show strong performance on reasoning tasks again, making recovery infeasible in practical settings. We note that Sreenivas et al. (2024) report stronger performance only after additional alignment stages with large scale data; we discuss this in Appendix A.1.

3 Layer-by-Layer Pruning for Generative Tasks

To better understand this discrepancy, we analyze the effects of removing individual layers in LLMs. Specifically, we perform single-layer pruning by removing one transformer layer at a time and evaluating the resulting model on a diverse set of generative benchmarks: GSM8K, HumanEval$^+$, and XSUM. Experiments are conducted on three models from distinct families: Qwen-2.5-7B-Instruct (Yang et al., 2025b), Llama-3.1-8B-Instruct (Grattafiori et al., 2024), and Mistral-7B-Instruct-v0.3 (Jiang et al., 2023).

Figure 3 summarizes the results. In a few cases, certain layers appear redundant; for example, early layers in Qwen and some deeper layers in Llama can be removed with minimal effect. But largely, across models, even single-layer removal can substantially impact performance. Pruning certain layers can even lead to sharp drops on GSM8K and HumanEval$^+$. While prior work has noted performance degradation in mathematical reasoning under layer pruning (Chen et al., 2024; Wang et al., 2025; Nepal et al., 2025), we additionally

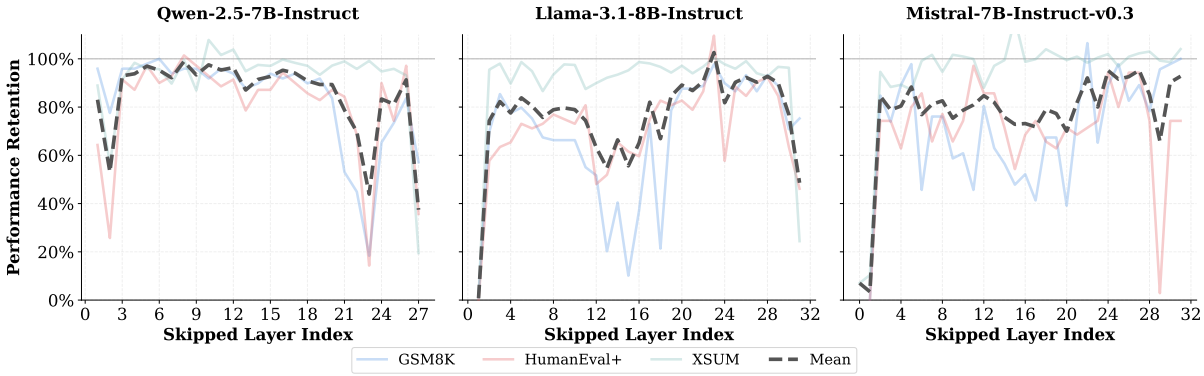


Figure 3: Effect of removing a single layer on model performance across generative benchmarks. Reasoning-intensive tasks such as GSM8K and HumanEval+ exhibit severe performance degradation at specific layers, while XSUM remains comparatively robust except for layers whose removal induces general text degeneration. (We generally skip layer 0 for its poor results.)

observe similar sensitivity in code generation, with the locations of the sharp drops varying across model families and tasks, as seen in Qwen versus Mistral. Task-specific effects are also pronounced: reasoning-intensive tasks such as mathematics and coding are far more sensitive to depth reduction, whereas summarization tasks like XSUM remain largely stable. ***These results indicate that layer pruning exhibits strong model- and task-dependent effects, contrasting with its relative robustness on classification benchmarks.***

Although earlier studies have suggested that pruning affects reasoning (Gromov et al., 2024; Men et al., 2025), detailed characterization of the resulting errors remains limited. In the following sections, we analyze a subset of key failure modes that arise in generative reasoning after pruning.

3.1 Text Degeneration

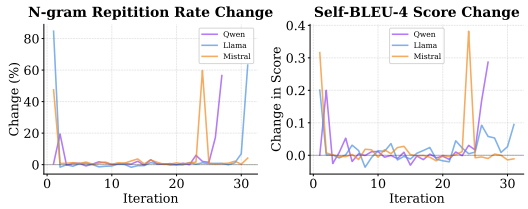


Figure 4: Text degeneration under single-layer pruning, measured using 4-gram repetition (left) and Self-BLEU4 averaged across responses and normalized relative to the baseline.

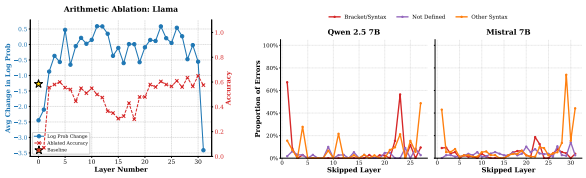
Text degeneration (Holtzman et al., 2019) is a commonly observed failure mode in pruned language models and can hinder instruction following and coherent generation. We quantify degeneration using two complementary metrics computed with 4-grams: *4-gram repetition* and *Self-BLEU4*, where higher values indicate increased repetition and reduced diversity (Holtzman et al., 2019). We additionally report the average number of generated tokens per prompt in Appendix A.2.

As shown in Figure 4, layer pruning often amplifies degenerative behaviors. In agreement with earlier findings (Men et al., 2025), we observe that both early and late layers are particularly important for maintaining stable text generation. In some cases, elevated repetition metrics align with sharp performance drops, such as layer 2 in Qwen.

Prior work has primarily attributed pruning-induced performance degradation to looping and repetitive outputs (Wang et al., 2025). However, degeneration alone does not fully account for failures in generative reasoning tasks. Near the points of sharp drops in Qwen and Llama on the math and coding tasks, text generation quality remains largely intact. Conversely, in Mistral, we observe a pronounced spike in degeneration metrics at layer 24 without a corresponding drop in task performance. Manual inspection reveals that removing this layer causes the model to continue rambling after producing a valid response. ***Overall, these findings indicate that while text degeneration is a significant side effect of layer pruning, it is not a sufficient explanation for the loss of reasoning ability in generative tasks.***

3.2 Degradation of Arithmetic

Beyond high-level reasoning behaviors, solving mathematical word problems requires reliable exe-



(a) Effect of single-layer pruning on the arithmetic ability of Llama.

(b) Distribution of syntactic error types induced by single-layer pruning.

Figure 5: Analysis of Model Performance and Error Types under Single-Layer Pruning.

cution of basic arithmetic operations. We find that pruning leads to a pronounced degradation of arithmetic ability, even on elementary computations. In particular, removing deeper layers in Qwen and mid-depth layers in Llama results in frequent arithmetic failures during qualitative inspection, where models fail to correctly perform simple calculations (see Appendix A.3).

Since standard generative evaluations can obscure true abilities due to auxiliary demands such as multi-token generation (Schaeffer et al., 2023; Hu and Frank, 2024; Song et al., 2025), to isolate arithmetic competence, we design a controlled evaluation that probes the model's ability to produce the *first answer token* in simple arithmetic prompts. Given inputs of the form "`Question: What is (7 + 5) - 6? Answer:`", we measure (i) the change in logprob of the correct answer token relative to the unpruned baseline, and (ii) top-1 accuracy, i.e., whether the correct token is assigned the highest logprob. Full experimental details are provided in Appendix A.4.

Figure 5a reports results over 200 arithmetic problems. We observe substantial drops in both logprob and accuracy after pruning layers near the middle region, despite the absence of any generation requirement in this task. ***This demonstrates that layer pruning induces a structural loss of arithmetic capability, rather than merely degrading Chain of Thought generation.*** Moreover, the correspondence between arithmetic failures and performance drops on GSM8K (Figure 3) suggests that a significant fraction of mathematical reasoning errors stem directly from impaired arithmetic abilities. Results for Qwen and Mistral are reported in Appendix A.4.

3.3 Degradation of Parenthesis Tracking

Figure 3 shows that layer pruning can cause substantial performance drops on coding benchmarks such as HumanEval^{+}. Similar to arithmetic in mathematical reasoning, correct syntax generation is a necessary prerequisite for code reasoning. ***We observe that removing specific layers significantly degrades the model's ability to maintain syntactic consistency, particularly in tracking and closing parentheses.*** Representative examples are provided in Appendix A.5.

Unlike math benchmarks, code evaluation allows errors to be categorized based on execution feedback. Leveraging this property, Figure 5b reports the prevalence of common syntactic failures under single-layer pruning for Qwen and Mistral, including *unbalanced parentheses*, *undefined variables*, and *other invalid syntax*. Notably, pruning certain layers (e.g., layer 23 in Qwen) leads to a sharp increase in parenthesis-matching errors, indicating a severe loss in syntactic skills. In Mistral, we also observe spikes in auxiliary syntax failures, such as malformed markdown code blocks (e.g., "`python (code)"`), which can directly disrupt downstream evaluation pipelines.

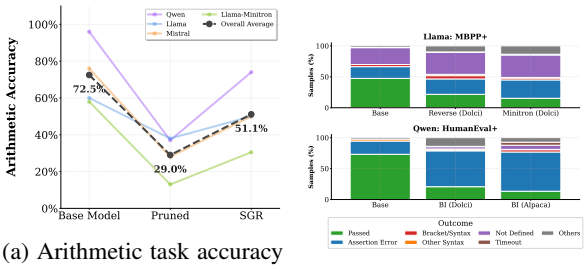
Taken together, these results indicate that layer pruning disrupts not only surface-level text generation but also internal mechanisms responsible for important algorithmic capabilities, such as arithmetic execution and parenthesis tracking. Tasks such as classification or summarization are less dependent on specialized capabilities, which helps explain their relative robustness to depth reduction. In contrast, the pronounced sensitivity observed under single-layer removal highlights why aggressive depth pruning poses a fundamental challenge for generative reasoning tasks, making performance retention without continued training almost impossible.

4 Post-Recovery Analysis

4.1 Arithmetic Ability

Building on the analysis in Section 3.2, we examine how arithmetic ability is affected by pruning and the extent to which it can be recovered through SGR finetuning. Figure 6a reports results for three model families at a fixed pruning ratio of 25%.

Across all models, pruning leads to a substantial drop in arithmetic accuracy, with average performance decreasing to 29%. Finetuning with SGR partially recovers this loss, but performance remains well below that of the base model. Notably, this arithmetic task is considerably simpler than full



(a) Arithmetic task accuracy for baseline, pruned, and finetuned models (50% ratio for Minitron, 25% for rest).

(b) Code evaluation outcomes (MBPP$^+$ and HumanEval$^+$) across model families.

Figure 6: Comparison of model performance across arithmetic and coding benchmarks under pruning and recovery.

generative mathematical reasoning, as it does not require multi-step reasoning or long-form generation. The persistent gap between base and finetuned pruned models, even on this minimal objective, indicates that ***pruning degrades core computational capabilities in a manner that is difficult to recover.***

While some models, such as LLaMA, exhibit smaller absolute drops (e.g., from 60% to 50%), this residual gap represents a lower bound on the error introduced by pruning. For more complex reasoning tasks, which combine arithmetic with multi-step generation, this degradation is likely to be further amplified.

We further corroborate this trend using the Minitron model introduced in Section 2.2. The base model (LLaMA-3.1-8B) achieves 58% accuracy on this task, which drops sharply to 13% after pruning. Strikingly, even after nearly 100B tokens of post-pruning training followed by supervised finetuning with SGR, the model fails to recover simple arithmetic performance. This reinforces our earlier findings and highlights a fundamental limitation: even extensive post-training is insufficient to restore basic computational abilities once depth has been reduced.

4.2 Syntax Ability

As discussed in Section 3.3, coding benchmarks are particularly sensitive to layer pruning, as they require strict syntactic correctness in addition to semantic reasoning. Figure 6b summarizes post-recovery code evaluation results across model families. ***Even after finetuning, pruned models continue to have difficulty generating syntactically valid code.*** This issue is more pronounced on MBPP$^+$, where code must be generated from natural language descriptions, compared to HumanEval$^+$, which provides a function signature as a prefix. Even the Minitron model, after significant training, fails to handle logical and syntactically sound code generation.

Overall, finetuning only partially restores syntactic abilities after pruning. Across models, we observe persistent errors such as undefined variables and unbalanced parentheses, consistent with the failure modes identified in Section 3.3. The prevalence of invalid and logically incorrect code indicates that pruning disrupts structural mechanisms required to maintain syntactic and state consistency during generation, which are not fully recoverable through training.

Although for Qwen in HumanEval$^+$, we see higher rates of syntactically valid code, the fraction of functionally correct solutions remains substantially lower. This disparity highlights that syntactic validity alone does not imply recovery of coding competence. Analogous to arithmetic serving as a lower bound for mathematical reasoning, these results suggest that residual syntactic errors provide a lower bound on the degradation of coding ability, with deeper semantic reasoning likely affected to an even greater extent.

5 Pruning at Moderate Ratios

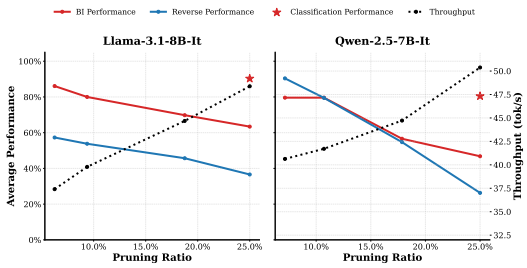


Figure 7: Average accuracy on generative tasks for Qwen and LLaMA across pruning strategies and ratios, alongside model throughput. Throughput (tokens/s) is shown on the secondary axis (dotted line). Even at moderate pruning ratio (25%), a substantial gap remains in generative vs classification tasks. Additional details are provided in Appendix A.10.

While prior sections highlight the limitations of aggressive pruning, moderate depth reduction remains appealing in practice due to its simplicity and direct impact on inference latency. Unlike other compression techniques, removing entire transformer blocks yields immediate throughput gains without requiring specialized hardware or kernel-level optimizations.

To better understand this trade-off, we evaluate performance across varying pruning ratios (Figure 7; full results in Table 5). With only two layers removed, both models retain approximately 85–90% of their original performance while achieving a **1.1–1.15**× increase in throughput. However, performance degrades steadily beyond this point, particularly for generative tasks, making more aggressive pruning less practical when reasoning capabilities are required.

We also observe that pruning strategies optimized for classification do not necessarily generalize to generative reasoning. For example, while Reverse pruning is competitive with Block Influence (BI) on classification tasks for LLaMA, it underperforms on generative benchmarks, reinforcing the earlier observation that classification metrics alone are insufficient for guiding pruning decisions.

Other approaches such as quantization (Frantar et al., 2022; Dettmers and Zettlemoyer, 2023; Lin et al., 2024) and sparsification (Frantar and Alistarh, 2023; Sun et al., 2024) are effective at reducing memory footprint, but often yield limited throughput gains under realistic settings and may require specialized support. In contrast, layer pruning provides a simple and hardware-agnostic mechanism for improving latency, and can be combined with these techniques for complementary benefits. Our results suggest that such approaches are most effective when applied conservatively, as increasing depth reduction leads to disproportionate degradation in generative reasoning performance.

6 Background & Related Work

Layer Pruning in LLMs. Recent work suggests that layers in large language models, particularly deeper ones, exhibit redundancy, motivating layer pruning as an effective compression strategy (Gromov et al., 2024; Sun et al., 2025; Men et al., 2025; Yin et al., 2023; Lad et al., 2024). On classification benchmarks, pruning has achieved strong results, with models retaining over 80% of baseline accuracy even after removing 20–25% of layers (Yang et al., 2024; Men et al., 2025; Gromov et al., 2024; Song et al., 2024). However, these gains do not consistently extend to generative reasoning tasks such as GSM8K, which require multi-step generation (Chen et al., 2024; Men et al., 2025; Gromov et al., 2024; Wang et al., 2025). This discrepancy suggests that layer redundancy may be task-dependent.

Limitations for Generative Reasoning. Prior work largely attributes the degradation in generative performance to the importance of deeper layers for reasoning (Song et al., 2025; Wang et al., 2025). Some studies highlight changes in evaluation or qualitative differences in chain-of-thought outputs after pruning (Song et al., 2025; Wang et al., 2025), but a systematic analysis of failure modes remains limited. To mitigate performance loss, existing approaches rely on continued pretraining (Gromov et al., 2024; Sreenivas et al., 2024; Muralidharan et al., 2024; Xia et al., 2023) or architectural modifications (Chen et al., 2024), often requiring large-scale data and compute. In more constrained settings, standard finetuning on open-source datasets has been commonly used, but yields limited recovery for generative tasks (Sreenivas et al., 2024; Muralidharan et al., 2024; Gromov et al., 2024; Wang et al., 2025).

7 Conclusion

In this work, we study the limits of layer pruning for generative reasoning in large language models under realistic post-training constraints. While prior work has shown that pruning can preserve classification performance, we demonstrate that generative reasoning degrades substantially and remains difficult to recover, even with stronger supervision such as self-generated responses.

Through controlled analysis, we show that this degradation extends beyond surface-level text quality to core algorithmic capabilities, including arithmetic computation and syntactic structure. Notably, these failures persist even under large-scale post-training, suggesting that scale alone is insufficient to restore such capabilities once disrupted.

Our results indicate that the effectiveness of layer pruning is inherently task-dependent. While moderate depth reduction offers a practical trade-off for classification and shallow tasks, aggressive pruning leads to disproportionate loss in reasoning ability. These findings highlight a fundamental limitation of depth reduction and suggest that preserving generative reasoning requires more principled approaches than post-hoc recovery alone.

We hope this work clarifies the role of depth in enabling algorithmic reasoning and informs the design of future compression strategies for large language models.

Limitations

Our study focuses on a specific post-pruning recovery regime: a single 80GB GPU, moderate-scale instruction data, and QLoRA-based finetuning. While we show that QLoRA is competitive with full finetuning in our setting, we cannot rule out that substantially different compute budgets or training procedures could achieve better recovery. Additionally, our failure mode analysis (arithmetic and parenthesis tracking) covers important but not exhaustive aspects of generative reasoning. Future work could extend this analysis to other structured generation tasks.

References

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

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Xiaodong Chen, Yuxuan Hu, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. 2024. Streamlining redundant layers to compress large language models. *arXiv preprint arXiv:2403.19135*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115.

Tim Dettmers and Luke Zettlemoyer. 2023. The case for 4-bit precision: k-bit inference scaling laws. In *International Conference on Machine Learning*, pages 7750–7774. PMLR.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 1286–1305.

Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, and 1 others. 2025. Olmo 3. *arXiv preprint arXiv:2512.13961*.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. *arXiv preprint arXiv:2301.00774*.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, and 1 others. 2021. A framework for few-shot language model evaluation. *Zenodo*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. 2024. The unreasonable ineffectiveness of the deeper layers. *arXiv preprint arXiv:2403.17887*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, and 1 others. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Jennifer Hu and Michael C Frank. 2024. Auxiliary task demands mask the capabilities of smaller language models. *arXiv preprint arXiv:2404.02418*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

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

Vedang Lad, Jin Hwa Lee, Wes Gurnee, and Max Tegmark. 2024. The remarkable robustness of llms: Stages of inference? *arXiv preprint arXiv:2406.19384*.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. *Advances in neural information processing systems*, 2.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6:87–100.

Gui Ling, Ziyang Wang, and Qingwen Liu. 2024. Slimgpt: Layer-wise structured pruning for large language models. *Advances in Neural Information Processing Systems*, 37:107112–107137.

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. *Advances in Neural Information Processing Systems*, 36:21558–21572.

Yao Lu, Hao Cheng, Yujie Fang, Zeyu Wang, Jiaheng Wei, Dongwei Xu, Qi Xuan, Xiaoniu Yang, and Zhaowei Zhu. 2024. Reassessing layer pruning in llms: New insights and methods. *arXiv preprint arXiv:2411.15558*.

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

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.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. 2024. Compact language models via pruning and knowledge distillation. *Advances in Neural Information Processing Systems*, 37:41076–41102.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*.

Aadim Nepal, Safal Shrestha, Anubhav Shrestha, Minwu Kim, Jalal Naghiyev, Ravid Shwartz-Ziv, and Keith Ross. 2025. Layer importance for mathematical reasoning is forged in pre-training and invariant after post-training. *arXiv preprint arXiv:2506.22638*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. 2023. Are emergent abilities of large language models a mirage? *Advances in neural information processing systems*, 36:55565–55581.

Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. 2024. Sleb: Streamlining llms through redundancy verification and elimination of transformer blocks. *arXiv preprint arXiv:2402.09025*.

Xinyuan Song, Keyu Wang, PengXiang Li, Lu Yin, and Shiwei Liu. 2025. Demystifying the roles of llm layers in retrieval, knowledge, and reasoning. *arXiv preprint arXiv:2510.02091*.

Sharath Turuvekere Sreenivas, Saurav Muralidharan, Raviraj Joshi, Marcin Chochowski, Ameya Sunil Mahabaleshwarkar, Gerald Shen, Jiaqi Zeng, Zijia Chen, Yoshi Suhara, Shizhe Diao, and 1 others. 2024. Llm pruning and distillation in practice: The minitron approach. *arXiv preprint arXiv:2408.11796*.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2024. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

Wenfang Sun, Xinyuan Song, Pengxiang Li, Lu Yin, Yefeng Zheng, and Shiwei Liu. 2025. The curse of depth in large language models. *arXiv preprint arXiv:2502.05795*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpaca: A Strong, Replicable Instruction-Following Model. Stanford Center for Research on Foundation Models (CRFM).

Vithursan Thangarasa, Ganesh Venkatesh, Mike Lasby, Nish Sinnadurai, and Sean Lie. 2025. Self-data distillation for recovering quality in pruned large language models. *Proceedings of Machine Learning and Systems*, 7.

Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, and 1 others. 2023. Efficient

large language models: A survey. *arXiv preprint arXiv:2312.03863*.

Keyu Wang, Tian Lyu, Guinan Su, Jonas Geiping, Lu Yin, Marco Canini, and Shiwei Liu. 2025. When fewer layers break more chains: Layer pruning harms test-time scaling in llms. *arXiv preprint arXiv:2510.22228*.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2025b. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Yifei Yang, Zouying Cao, and Hai Zhao. 2024. Laco: Large language model pruning via layer collapse. *arXiv preprint arXiv:2402.11187*.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Jaiswal, Mykola Pechenizkiy, Yi Liang, and 1 others. 2023. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity. *arXiv preprint arXiv:2310.05175*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

A Appendix

A.1 Additional Alignment in Minitron

Sreenivas et al. (2024) report final results after multiple post-distillation alignment stages, including supervised finetuning on math and code data, instruction tuning, and preference optimization. These stages are applied after the distillation phase.

The publicly released `Llama-3.1-Minitron-4B-Depth` checkpoint corresponds to the model after distillation (trained on ∼100B tokens), prior to these alignment stages. As this checkpoint is not explicitly aligned for instruction following, we perform additional finetuning using SGR before evaluating recovery, ensuring the model is adapted for reasoning-intensive tasks.

The results in Sreenivas et al. (2024) suggest that on top of 100 billion tokens of training, we need additional amount of curated data and multi-stage instruction tuning pipelines. Consequently, under realistic post-training settings without access to such scale of data and compute, recovering generative reasoning after pruning remains challenging.

A.2 Tokens with Layer Pruning

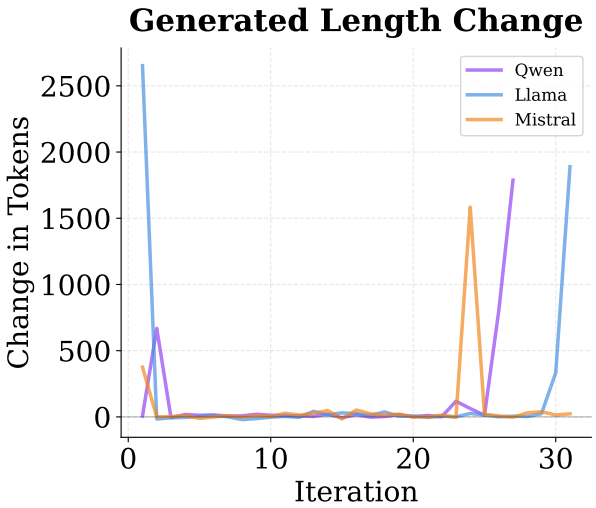


Figure 8: Text degeneration results with layer pruning using N-gram repetition (left) and Self-BLEU4 score relative to baseline.

A.3 Arithmetic Mistake

Example: Weekly Egg Production

Question. Gary buys 4 chickens. After two years, he has 8 times as many chickens as he started with. If each chicken lays 6 eggs per day, how many eggs does Gary currently collect every week?

Pruned Model Response. Initially, Gary buys 4 chickens. After two years, he has eight times as many chickens:

$$4 \times 8 = 32 \text{ chickens.}$$

Each chicken lays 6 eggs per day. The total number of eggs laid per day is computed as:

$$32 \times 6 = 364 \text{ eggs.} \qquad \textbf{(Arithmetic error)}$$

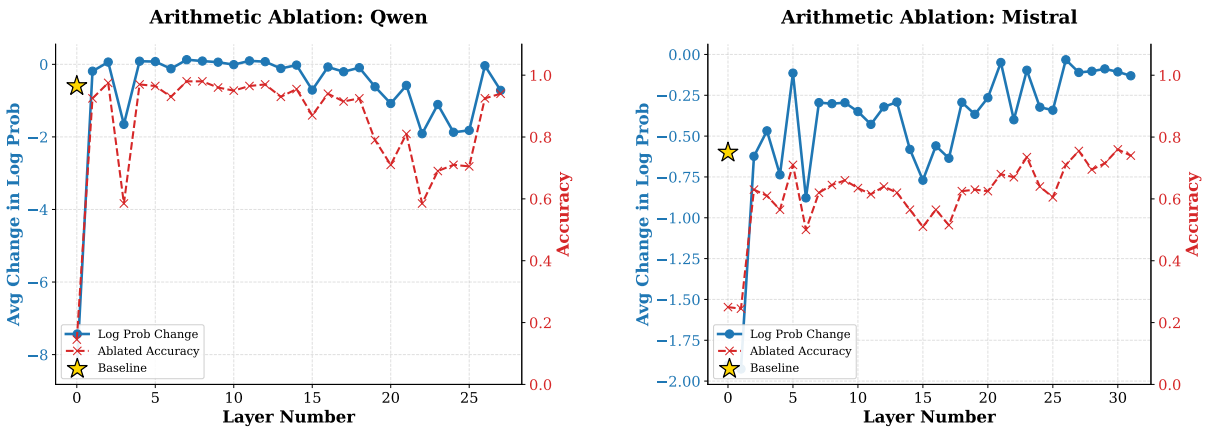
To determine the weekly total, the daily number of eggs is multiplied by 7:

$$364 \times 7 = 2448 \text{ eggs.}$$

Thus, the solution concludes that Gary collects $\boxed{2448}$ eggs per week.

A.4 Arithmetic Ablation Experiment Details

For this experiment, we rely on the EleutherAI/arithmetic dataset. We use the single digit, three operations subset. We restrict the output space to individual digits from 0 to 9.



(a) Qwen (b) Mistral

Figure 9: Effect of single-layer pruning on the arithmetic ability of various models.

A.5 Balanced Parenthesis Error

Example: Code Generation with a Syntax Error

Prompt. Given an array of integers, compute the minimum number of elements that must be changed to make the array palindromic.

Generated Response.

```
def smallest_change(arr):
    n = len(arr)
    changes = 0
    for i in range(n // 2):
        if arr[i] != arr[n - 1 - i):
            changes += 1
        else:
            continue
    return changes
```

Highlighted Error.

`arr[n - 1 - i)` **(parenthesis mismatch)**

A.6 Finetuning Details

A.6.1 QLoRA with BI and Reverse Pruning Strategies

Experimental Results. We post-train our models using QLoRA with 4-bit NF4 quantization, a learning rate of 2×10^{-4}, batch size 8, constant learning rate with 50 warmup steps, bf16 training, and sequence length 8192 with gradient checkpointing on a single A100 80GB GPU. We use the TRL package for training. For experiments with the Alpaca dataset (∼50K), we train for 2 epochs; for Dolci (Ettinger et al., 2025) (∼90K), 1 epoch. We focus on Dolci since our broader objective is to assess whether post-pruning training can preserve generative reasoning performance. We rely on QLoRA because it is comparable to even LoRA trained models in our experiments (see Table 1). We also show in Appendix A.6.3 that QLoRA closely matches performance of full finetuning for recovery in our settings as well.

Model	HeSw	PIQA	MMLU	Wino	OBQA	ARC-E	ARC-C	Mean
Gemma2-2B-It								
Reverse-order*	0.844	0.893	0.925	0.941	0.747	0.785	0.736	<u>0.839</u>
PPL*	0.859	0.948	0.616	0.837	0.826	0.867	0.706	0.808
Magnitude-L1*	0.844	0.951	0.475	0.804	0.854	0.895	0.730	0.793
Magnitude-L2*	0.791	0.918	0.424	0.789	0.669	0.812	0.594	0.714
BI*	0.796	0.880	0.874	0.957	0.758	0.805	0.725	0.828
Taylor*	0.846	0.890	0.955	0.932	0.848	0.787	0.723	**0.854**
QLoRA (Reverse + Dolci)	*0.736*	*0.857*	*0.930*	*0.945*	*0.702*	*0.816*	*0.708*	*0.813*
QLoRA (BI + Dolci)	*0.729*	*0.857*	*0.856*	*0.961*	*0.927*	*0.774*	*0.698*	*0.83*
LLaMA-3.1-8B-It								
Reverse-order*	0.679	0.875	0.934	0.844	0.870	0.754	0.769	0.818
PPL*	0.834	0.953	0.496	0.783	0.781	0.891	0.735	0.782
Magnitude-L1*	0.446	0.676	0.369	0.660	0.402	0.348	0.389	0.470
Magnitude-L2*	0.446	0.676	0.368	0.659	0.396	0.347	0.389	0.469
BI*	0.710	0.897	0.356	0.729	0.598	0.746	0.549	0.655
Taylor*	0.840	0.892	0.421	0.965	0.811	0.837	0.807	0.796
QLoRA (Reverse + Dolci)	*0.787*	*0.896*	*0.895*	*1.000*	*0.651*	*0.863*	*0.813*	***0.844***
QLoRA (BI + Dolci)	*0.689*	*0.862*	*0.925*	*0.915*	*0.858*	*0.786*	*0.780*	<u>*0.831*</u>

Table 1: Performance retention on classification benchmarks across Gemma and Llama with various pruning strategies. LoRA-trained results from (Lu et al., 2024) are marked with an asterisk (*). QLoRA results largely match or outperform LoRA trained models across various pruning strategies showing consistent >80% performance retention. HeSw = HellaSwag, Wino = Winogrande, OBQA = OpenBookQA.

Pruning Strategies. In our experiments, we mainly deal with two commonly used layer pruning strategies, Block Influence (BI) and Reverse Order (Men et al., 2025; Lu et al., 2024; Sreenivas et al., 2024). We show that QLoRA with simple pruning metrics like BI and Reverse perform comparably with techniques like LoRA composed with other various pruning metrics (see Table 1). Additionally, we consider an *iterative* pruning procedure. The iterative procedure extends the single-layer pruning analysis in Figure 3 by greedily removing layers based on observed redundancy. Specifically, we first prune the layer whose removal leads to the smallest performance drop. With this layer removed, we then determine the layer whose removal leads to the smallest drop, and additionally prune that layer. We continue to repeat this process. This pruning strategy allows us to examine whether selectively removing redundant layers affects the recovery behavior observed in generative reasoning tasks. The full procedure is described in Algorithm 1.

Algorithm 1 Greedy Iterative Pruning via Benchmark Performance

Input: Model $\mathcal{M}$ with L layers, benchmark dataset $\mathcal{D}$, number of layers to prune N
Output: Pruned layer set $\mathcal{P}$
$\mathcal{P} \leftarrow \emptyset$
for $k = 1$ to N **do**
  $\ell^\star \leftarrow \arg\max_{\ell \in \{1,\ldots,L\} \setminus \mathcal{P}} \ \text{Score}\big(\mathcal{M}_{-(\mathcal{P} \cup \{\ell\})}, \mathcal{D}\big)$
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{\ell^\star\}$
end for
Return $\mathcal{P}$

A.6.2 Upper-Bound Recovery on GSM8K

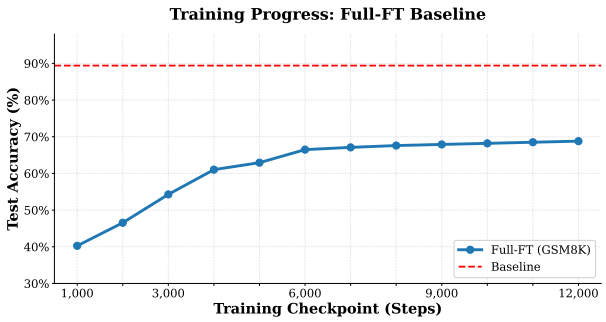


Figure 10: Full Finetuning with GSM8K dataset on the Qwen Iterative Pruned Model.

To estimate an upper bound on recoverable performance under layer pruning, we conduct an experiment under highly favorable conditions. We apply Iterative pruning to the Qwen model until 7 layers using GSM8K exclusively as the calibration dataset, and subsequently finetune the pruned model only on the GSM8K training split. For each training example, we generate eight responses from the unpruned Qwen model to increase output diversity. Finetuning is performed for approximately two epochs, followed by evaluation on the GSM8K test set.

Despite these idealized conditions, Figure 10 shows that performance on GSM8K cannot be fully recovered after pruning. This result underscores the difficulty of restoring generative reasoning abilities even when the pruning metric, training data, and evaluation task are perfectly aligned.

A.6.3 QLoRA vs. Full Finetuning

We further compare QLoRA against full-parameter supervised finetuning (Full-FT) using self-generated Dolci data. Both methods are applied to the same Iteratively pruned Qwen model with seven layers removed, and evaluated on GSM8K as a proxy for generative reasoning quality.

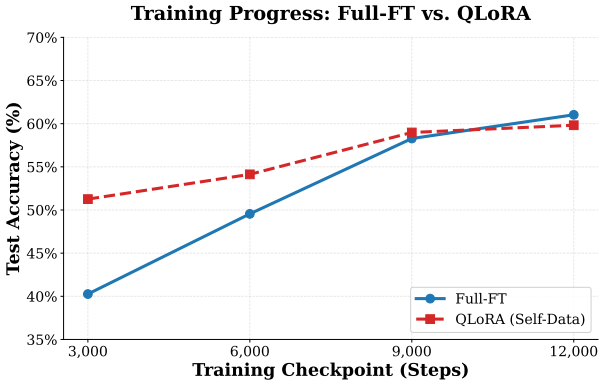


Figure 11: Comparison between full supervised finetuning (Full-FT) and QLoRA on self-generated Dolci data.

At the scale of our experiments, QLoRA achieves recovery comparable to Full-FT, despite operating under significantly reduced memory and compute requirements. While full finetuning may yield additional gains when scaled further, doing so incurs substantially higher computational cost and hardware demands. Given our focus on recovery under resource-constrained post-training settings, we adopt QLoRA throughout the paper as a practical and representative finetuning approach for studying the limits of generative reasoning recovery after layer pruning.

A.7 RLVR-Based Recovery on GSM8K

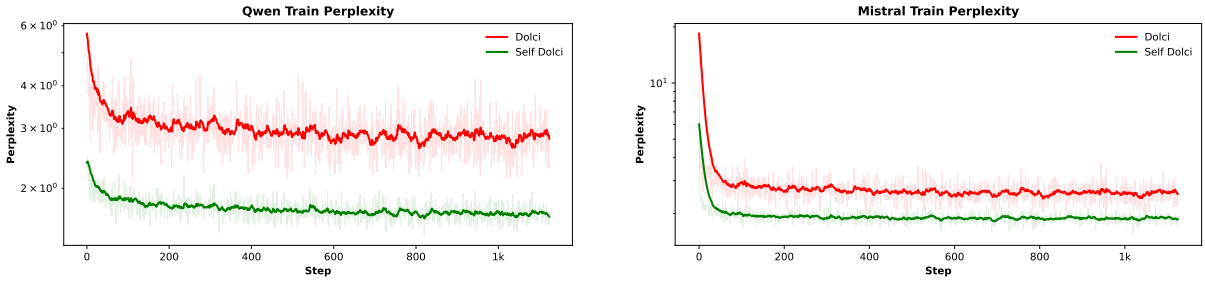
To assess whether reinforcement-based alignment can further recover capabilities lost through depth reduction, we apply RLVR finetuning on GSM8K on top of the SGR-recovered Qwen2.5-7B-Instruct model pruned with BI at 7 layers.

As shown in Table 2, RLVR provides only marginal additional gains over SGR alone (0.418 vs. 0.329 retention on GSM8K), with performance remaining well below the unpruned baseline.

Model / Strategy	GSM8K Retention
Qwen2.5-7B-Instruct (unpruned)	1.000
+ BI pruning, SFT on Dolci	0.270
+ BI pruning, SGR on S.Dolci	0.329
+ BI pruning, SGR + RLVR on GSM8K	0.418

Table 2: GSM8K retention under increasing post-pruning recovery for Qwen2.5-7B-Instruct (BI pruning, 7 layers removed). Retention is normalized to the unpruned baseline. RLVR is applied on top of the SGR-recovered model.

A.8 Perplexity Curves



(a) Qwen

(b) Mistral

Figure 12: Perplexity curves during training for both standard finetuning and for SGR for the Qwen and Mistral models (Both are BI pruned: 25%).

A.9 Full Results with Self-Generated Responses

We also observe that, in Table 3, the effectiveness of pruning strategies varies by model. While Iterative pruning does not provide consistent gains over standard metrics such as BI and Reverse Order for Llama and Mistral, it performs better for Qwen. This suggests that greedily selecting layers based on empirical redundancy does not generalize uniformly across architectures, potentially reflecting differences in internal layer organization, as previously noted for Qwen models (Gromov et al., 2024).

	Classification								Generative				
Model	**HeSw**	**PIQA**	**MMLU**	**Wino**	**OBQA**	**ARC-E**	**ARC-C**	**Mean**	**GSM8K**	**HumEval$^+$**	**MBPP$^+$**	**XSUM**	**Mean**
Gemma2-2B-It													
Open-source data													
Reverse + Alpaca	0.844	0.893	0.925	0.941	0.747	0.785	0.736	0.839	0.040	0.119	0.029	0.814	0.251
BI + Alpaca	0.796	0.880	0.874	0.957	0.758	0.805	0.725	0.828	0.042	0.153	0.099	0.809	0.276
Reverse + Dolci	0.736	0.857	0.930	0.945	0.702	0.816	0.708	0.813	0.129	0.219	0.121	0.763	0.308
BI + Dolci	0.729	0.857	0.856	0.961	0.927	0.774	0.698	0.829	0.154	0.095	0.113	0.778	0.285
Self-Generated Responses													
SGR (Reverse + S.Alpaca)	0.783	0.861	0.930	0.984	0.702	0.866	0.849	0.854 ↑	0.047	0.103	0.212	0.856	0.304 ↑
SGR (BI + S.Alpaca)	0.756	0.847	0.874	0.888	0.843	0.866	0.849	0.846 ↑	0.067	0.186	0.164	0.851	0.317 ↑
SGR (Reverse + S.Dolci)	0.732	0.805	0.937	0.940	0.758	0.845	0.824	0.834 ↑	0.164	0.256	0.121	0.768	0.327 ↑
SGR (BI + S.Dolci)	0.729	0.857	0.904	0.945	0.955	0.821	0.800	**0.859** ↑	0.259	0.389	0.298	0.876	**0.455** ↑
LLaMA-3.1-8B-It													
Open-source data													
Reverse + Alpaca*	0.679	0.875	0.934	0.844	0.870	0.754	0.769	0.818	0.453	0.111	0.084	0.638	0.321
BI + Alpaca	0.710	0.897	0.356	0.729	0.598	0.746	0.549	0.655	0.318	0.344	0.128	0.062	0.213
Reverse + Dolci	0.787	0.896	0.895	1.000	0.651	0.863	0.813	0.844	0.405	0.434	0.302	0.077	0.304
BI + Dolci	0.793	0.886	0.943	1.027	0.710	0.896	0.887	0.878	0.469	0.444	0.308	0.068	0.322
Iterative + Dolci	0.787	0.896	0.789	0.926	0.828	0.893	0.747	0.838	0.328	0.301	0.245	0.338	0.303
Self-Generated Responses													
SGR (Reverse + S.Alpaca)	0.682	0.860	0.909	0.926	0.947	0.807	0.772	0.843 ↑	0.561	0.290	0.162	0.179	0.298 ↓
SGR (BI + S.Alpaca)	0.838	0.913	0.967	1.049	0.769	0.950	0.912	**0.914** ↑	0.724	0.412	0.481	0.860	0.619 ↑
SGR (Reverse + S.Dolci)	0.809	0.920	0.963	0.979	0.799	0.917	0.879	0.895 ↑	0.628	0.556	0.251	0.029	0.366 ↑
SGR (BI + S.Dolci)	0.826	0.907	0.984	1.033	0.769	0.924	0.879	0.903 ↑	0.758	0.634	0.390	0.754	**0.634** ↑
SGR (Iterative + S.Dolci)	0.792	0.907	0.835	1.022	0.947	0.875	0.780	0.880 ↑	0.647	0.423	0.312	0.763	0.536 ↑
Qwen2.5-7B-Instruct													
Open-source data													
Reverse + Alpaca	0.710	0.832	0.765	0.854	0.488	0.720	0.570	0.706	0.012	0.025	0.020	0.591	0.162
BI + Alpaca	0.867	0.982	0.480	0.789	0.927	0.874	0.708	0.804	0.097	0.167	0.274	0.768	0.327
Reverse + Dolci	0.681	0.794	0.703	0.826	0.488	0.650	0.524	0.666	0.059	0.159	0.120	0.541	0.220
BI + Dolci	0.820	0.958	0.507	0.854	0.878	0.843	0.626	0.784	0.270	0.200	0.278	0.817	0.391
Iterative + Dolci	0.827	0.910	0.700	0.859	0.732	0.828	0.727	0.798	0.294	0.333	0.358	0.812	0.449
Self-Generated Responses													
SGR (Reverse + S.Alpaca)	0.740	0.856	0.848	0.846	0.488	0.741	0.633	0.736 ↑	0.025	0.051	0.024	0.605	0.176 ↑
SGR (BI + S.Alpaca)	0.891	0.982	0.540	0.826	0.732	0.852	0.748	0.796 ↓	0.202	0.183	0.288	0.846	0.380 ↑
SGR (Reverse + S.Dolci)	0.678	0.788	0.731	0.732	0.516	0.652	0.576	0.668 ↑	0.153	0.142	0.157	0.600	0.263 ↑
SGR (BI + S.Dolci)	0.851	0.982	0.545	0.841	0.854	0.867	0.693	**0.805** ↑	0.329	0.283	0.398	0.861	0.468 ↑
SGR (Iterative + S.Dolci)	0.852	0.934	0.699	0.852	0.610	0.869	0.775	0.799 ↑	0.581	0.325	0.374	0.822	**0.525** ↑
Mistralv0.3-7B-Instruct													
Open-source data													
Reverse + Alpaca	0.832	0.863	0.769	0.886	0.523	0.849	0.799	0.789	0.087	0.158	0.246	0.096	0.147
BI + Alpaca	0.827	0.848	0.823	0.896	0.705	0.843	0.663	0.801	0.056	0.244	0.223	0.048	0.143
Reverse + Dolci	0.792	0.834	0.844	0.941	0.591	0.821	0.601	0.775	0.375	0.316	0.488	0.108	0.322
BI + Dolci	0.813	0.848	0.877	0.951	0.614	0.786	0.608	0.785	0.236	0.316	0.285	0.072	0.227
Iterative + Dolci	0.806	0.886	0.685	0.851	0.682	0.804	0.621	0.762	0.293	0.244	0.262	0.096	0.224
Self-Generated Responses													
SGR (Reverse + S.Alpaca)	0.876	0.890	0.891	0.866	0.773	0.903	0.812	**0.859** ↑	0.182	0.282	0.569	0.054	0.272 ↑
SGR (BI + S.Alpaca)	0.860	0.873	0.924	0.856	0.614	0.851	0.751	0.818 ↑	0.102	0.264	0.569	0.084	0.255 ↑
SGR (Reverse + S.Dolci)	0.861	0.863	0.929	0.906	0.750	0.883	0.758	0.850 ↑	0.421	0.526	0.875	0.095	**0.479** ↑
SGR (BI + S.Dolci)	0.854	0.857	0.908	0.911	0.614	0.831	0.772	0.821 ↑	0.395	0.509	0.715	0.066	0.421 ↑
SGR (Iterative + S.Dolci)	0.841	0.871	0.718	0.836	0.682	0.856	0.657	0.780 ↑	0.230	0.402	0.492	0.042	0.292 ↑

Table 3: Performance retention (normalized to baseline). Results marked with (*) are sourced from (Lu et al., 2024). SGR is our method with the pruning metric in parentheses. Reverse, BI, and Iterative (described in Algorithm 1) indicate the pruning order, while Alpaca and Dolci denote the training data source; S.Alpaca and S.Dolci refer to self-generated variants of the corresponding datasets. ↑ / ↓ indicate improvement or degradation in mean retention of our SGR approach with respect to the standard approach of doing SFT with the open-source prompts and responses.

A.10 Throughput under Different Pruning Ratios

To complement Section 5, we report decoding throughput under moderate pruning ratios using the Block Influence (BI) strategy. Following prior work (e.g., (Frantar et al., 2022)), we measure tokens-per-second on WikiText-2 with a fixed context length of 512 tokens, which provides a lightweight proxy for

Pruning Level	Throughput (tok/s)	Speedup	GPU Mem (GB)
LLaMA-3.1-8B-Instruct			
Base	35.73	1.00×	14.96
∼10% pruned	40.37	1.13×	13.74
∼18% pruned	43.75	1.23×	12.52
∼25% pruned	46.30	1.30×	11.71
Qwen2.5-7B-Instruct			
Base	38.47	1.00×	14.19
∼10% pruned	41.71	1.08×	12.88
∼18% pruned	44.72	1.16×	12.01
∼25% pruned	50.38	1.31×	11.15

Table 4: Throughput and memory under BI pruning at moderate pruning ratios. Pruning levels are grouped into approximate percentages of removed layers.

autoregressive generation efficiency.

We group pruning configurations into three regimes based on the proportion of removed layers: approximately 10%, 18%, and 25%. All measurements are conducted under identical hardware and inference settings, and we report throughput, relative speedup, and GPU memory usage.

We observe a consistent increase in decoding throughput as the pruning ratio increases. For both model families, removing approximately 25% of layers yields a speedup of around 1.3×, while also reducing GPU memory usage by roughly 20–25%.

Importantly, we find that these throughput trends closely match those observed during evaluation on generative benchmarks (e.g., GSM8K, HumanEval+, MBPP+, XSUM), indicating that WikiText-2 serves as a reliable proxy for relative efficiency comparisons.

However, as discussed in Section 5, these gains come with a noticeable degradation in generative reasoning performance. While moderate pruning (e.g., ∼10–18%) provides a favorable trade-off between efficiency and performance, more aggressive pruning (e.g., ∼25%) leads to disproportionately larger drops in reasoning capability relative to the additional speedup.

A.11 Pruning at Different Ratios

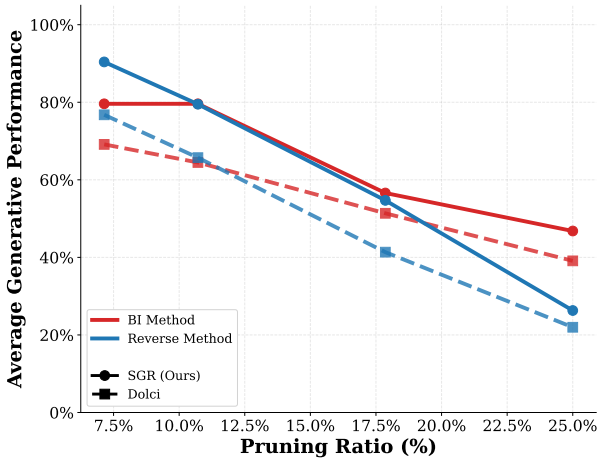


Figure 13: Differences between finetuning with Self-Generated Responses (SGR) vs on Dolci dataset directly for the Qwen Model. At all pruning ratios, SGR is consistently better than the raw dataset.

Method	GSM8K	HumanEval	MBPP	XSUM	Average
Qwen					
2 Layers Pruned					
BI	0.878	0.692	0.709	0.906	0.796
Iterative	0.940	0.817	0.736	0.930	0.856
Reverse	0.946	0.950	0.793	0.925	**0.904**
3 Layers Pruned					
BI	0.878	0.692	0.709	0.906	**0.796**
Iterative	0.896	0.633	0.421	0.910	0.715
Reverse	0.785	0.817	0.685	0.891	0.795
5 Layers Pruned					
BI	0.538	0.409	0.496	0.822	0.566
Iterative	0.819	0.658	0.458	0.853	**0.697**
Reverse	0.422	0.476	0.455	0.837	0.547
7 Layers Pruned					
BI	0.329	0.283	0.398	0.861	0.468
Iterative	0.591	0.350	0.358	0.797	**0.524**
Reverse	0.153	0.142	0.157	0.600	0.263
LLaMA					
2 Layers Pruned					
BI	0.932	0.811	0.726	0.977	**0.861**
Iterative	0.938	0.855	0.631	0.968	0.848
Reverse	0.871	0.855	0.530	0.037	0.573
3 Layers Pruned					
BI	0.926	0.755	0.558	0.963	0.800
Iterative	0.946	0.800	0.603	0.958	**0.827**
Reverse	0.893	0.733	0.502	0.023	0.538
6 Layers Pruned					
BI	0.818	0.655	0.530	0.788	**0.698**
Iterative	0.806	0.566	0.497	0.894	0.691
Reverse	0.762	0.700	0.341	0.025	0.457
8 Layers Pruned					
BI	0.758	0.634	0.390	0.754	**0.634**
Iterative	0.671	0.466	0.312	0.758	0.552
Reverse	0.628	0.556	0.251	0.029	0.366

Table 5: Retention on generative benchmarks under increasing pruning levels. We report results on GSM8K, HumanEval, MBPP, and XSUM. Average denotes the mean recovery across benchmarks.