TELL-TALE: TASK EFFICIENT LLMS WITH TASK AWARE LAYER ELIMINATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper we introduce TALE, Task-Aware Layer Elimination, an inference-time algorithm that prunes entire transformer layers in an LLM by directly optimizing task-specific validation performance. We evaluate TALE on 9 tasks and 5 models, including LLaMA 3.1 8B, Qwen 2.5 7B, Qwen 2.5 0.5B, Mistral 7B, and Lucie 7B, under both zero-shot and few-shot settings. Unlike prior approaches, TALE requires no retraining and consistently improves accuracy while reducing computational cost across all benchmarks. Furthermore, applying TALE during finetuning leads to additional performance gains. Finally, TALE provides flexible user control over trade-offs between accuracy and efficiency. Mutual information analysis shows that certain layers act as bottlenecks, degrading task-relevant representations. TALE's selective layer removal remedies this problem, producing smaller, faster, and more accurate models that are also faster to fine-tune while offering new insights into transformer interpretability.

1 INTRODUCTION

While Large Language Models (LLMs) have achieved remarkable success, they come with large computational overheads. This may prevent users in resource limited organizations or with high-throughput applications from using more capable models. This limitation has spawned an important research area on model compression. We focus here on the branch known as model pruning. Although it often reduces computational overhead, pruning typically requires complex implementation procedures, extensive retraining or fine-tuning, and may result in unpredictable performance degradation (Section 2 details related work).

In contrast, having observed that not all layers contribute to a particular task, we propose a lightweight, greedy, iterative layer pruning algorithm, Task Aware Layer Elimination (TALE). TALE operates at inference time, is hardware agnostic, directly optimizes for task-specific accuracy at each pruning step and consistently offers improved results over the original model. This improvement persists in interactions with fine tuning on our tasks. As illustrated in Figure 1 and detailed in Section 3, TALE systematically evaluates all possible single-layer removals at each iteration, selecting the layer whose elimination results in the highest validation accuracy. This process continues iteratively until performance improvements fall below a predefined threshold, ensuring that only layers with minimal or negative impact on task performance are removed.

TALE improves task specific accuracy and provides moderate computational reductions with minimal implementation complexity. Our layer-wise pruning leverages the modular nature of transformer architectures, where each layer performs a complete transformation of the input representation through attention and feedforward mechanisms. This architectural property enables the removal of entire layers without requiring modifications to the remaining network structure.

We evaluate TALE with five LLMs, LLaMA 3.1 8B, Qwen 2.5 7B, Qwen 2.5 0.5B, Mistral 7B and Lucie 7B, on 9 diverse benchmark datasets (Sections 4 and 5) both in zero-shot and few-shot settings. We also investigated interactions between pruning with TALE finetuning. We find consistent improvements in both accuracy and computational efficiency in all cases. This suggests pre-trained LLMs contain redundant or even detrimental layers for a given downstream task. We analyze layer flow using the notion of mutual information (MI) to support this hypothesis (Section 6). Additionally, our experiments show TALE's potential as a tool for understanding layer function in and across models (Section 7), thus aiding model interpretability.

2 Related Work

Zhu et al. (2024) distinguishes four primary approaches to model compression: model pruning, quantization, low-rank approximation, and knowledge distillation. Our work focuses on pruning, which can be categorized into unstructured, structured, and semi-structured methods. Unstructured pruning removes individual parameters, resulting in irregular, sparse structures Han et al. (2015b); Chen et al. (2015); Srinivas & Babu (2015); structured pruning eliminates entire components such as neurons, attention heads, or layers while maintaining the overall network structure He et al. (2017); Voita et al. (2019); Lagunas et al. (2021); Men et al. (2024). Semi-structured pruning combines fine-grained control with structural regularity, and has been explored in recent work Li et al. (2023); Frantar & Alistarh (2023b); Sun et al. (2024). Early pruning methods leveraged second-order information for structured pruning LeCun et al. (1989); Hassibi et al. (1993), but the field has since shifted toward computationally simpler, magnitude-based approaches that prune parameters by importance scores Han et al. (2015a); See et al. (2016); Narang et al. (2017)

Model pruning has benefited from information-theory. Building on the information bottleneck view of deep networks, Tishby et al. (2000); Tishby & Zaslavsky (2015) conceptualize layers as filters that preserve task-relevant information while discarding irrelevant input. This has led to model pruning proposals based on mutual information (MI). Fan et al. (2021) propose a layer-wise strategy that leverages MI estimates to reduce hidden dimensionality in a top-down manner. MI-guided approaches demonstrate that information-theoretic criteria can balance representation quality with model compression (Ganesh et al., 2020; Westphal et al., 2024). A central challenge, however, is the difficulty of estimating MI. Neural estimators such as MINE Ishmael Belghazi et al. (2018) provide principled solutions for high-dimensional settings, but in practice, probing classifiers Belinkov (2022) remain the dominant tool due to their efficiency and interpretability.

With regards to large transformers, Zhang & Papyan (2025) proposes a pruning strategy based on matrix approximations. Xia et al. (2023) demonstrates that pruning 7B models into 1.3B and 2.7B variants via structured layer and hidden-dimension pruning, combined with dynamic batch loading for fine-tuning, can yield submodels outperforming comparably sized models trained from scratch, though not matching the original model. Complementary work Frantar & Alistarh (2023a); Zhang et al. shows that contiguous blocks, especially attention layers, can often be pruned with limited performance loss, although critical "cornerstone" layers Zhang et al. remain essential for reasoning. Kim et al. (2024) investigates lock-level pruning based on weight importance, but these methods typically require fine-tuning techniques to recover baseline accuracy.

Most pruning methods compress models while attempting to match the original performance but often suffer accuracy degradation, which requires retraining or fine-tuning Xia et al. (2024). Improvements are generally measured relative to small models rather than original unpruned baselines. To the best of our knowledge, TALE is the only pruning method reported to improve over the original model's performance without requiring any additional training or fine-tuning.

3 TALE , our Greedy-Selection Algorithm

3.1 Basics and Intuitions

A transformer maps a sequence of input vectors $(x_1, \cdots, x_n)$ to a corresponding sequence of output vectors through a stack of L layers. Each layer ℓ transforms the hidden representations $X^{(\ell)} = (x_1^{(\ell)}, \ldots, x_n^{(\ell)})$ into $X^{(\ell+1)}$ through attention and feedforward blocks, connected by residual pathways. Removing layer ℓ from this pipeline simply redirects the flow such that $X^{(\ell-1)} \to X^{(\ell+1)}$, a property that makes the architecture naturally amenable to layer-wise pruning.

Our initial intuition for TALE came from examining the behavior of partial forward passes. Let $h^{(k)}$ denote the hidden representation after k layers. Instead of always decoding from the final representation $h^{(\mathrm{L})}$, we projected intermediate representations $h^{(k)}$ for $k < \mathrm{L}$ directly into the vocabulary space using the output projection W_{out}, i.e.,

$$\hat{y}^{(k)} = \mathrm{softmax}(W_{\text{out}} h^{(k)}).$$

We then compared the performance of $\hat{y}^{(k)}$ across different values of k. Surprisingly, we observed that for many tasks, intermediate layers ($k < \mathrm{L}$) achieved higher accuracy than the final layer L

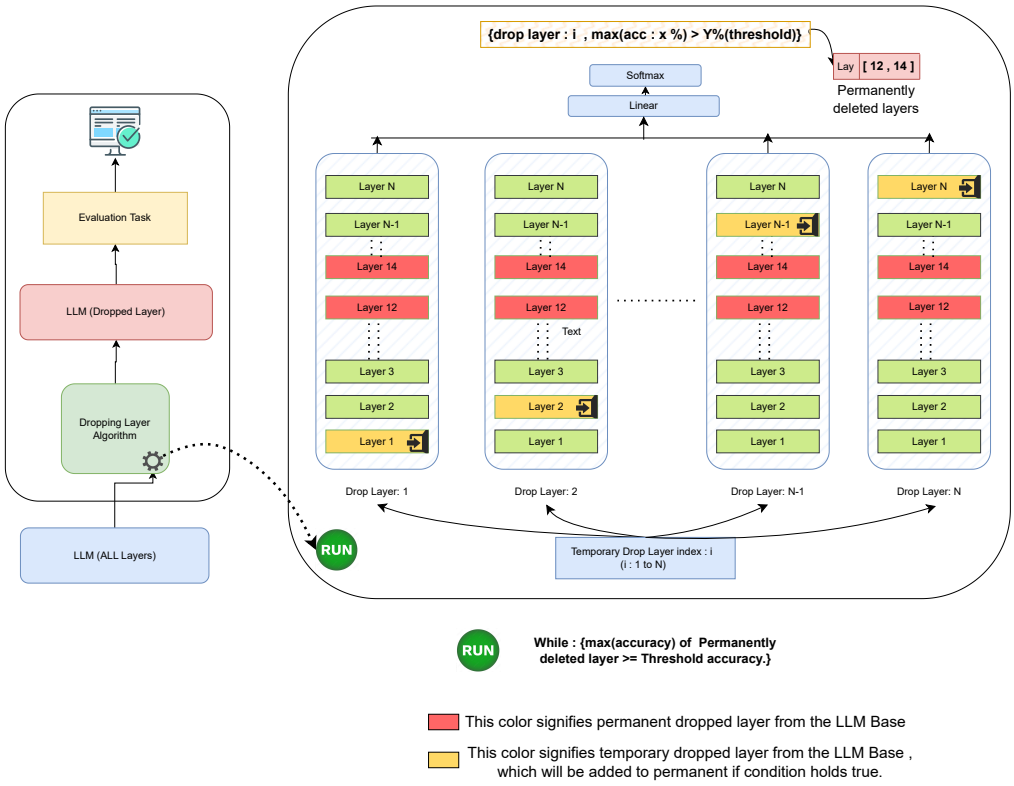


Figure 1: Illustration of TALE layer elimination. Candidate layers (yellow) are tested for removal, and the best-performing ones above the threshold are permanently dropped (red) until no further improvement is possible.

(Figure 5). This indicated that additional depth does not always translate into better task-specific performance: some layers contribute marginally, while others introduce representational noise.

This experiment motivated our central hypothesis: *not all layers in an LLM are equally useful, and selectively removing redundant layers can preserve—or even improve—downstream accuracy*. TALE (Task-Aware Layer Elimination) formalizes this intuition into a principled, iterative pruning strategy.

3.2 TALE

TALE is a greedy iterative layer pruning algorithm for pre-trained open-weights LLM compression that systematically removes layers while preserving or even improving model performance (Algorithm 6). Starting with a full pre-trained model, TALE evaluates all possible single-layer removals at each iteration, computing the validation accuracy for each candidate pruned architecture. The layer whose removal results in the highest accuracy is permanently eliminated from the model, and this compressed architecture becomes the baseline for the next iteration. This process continues iteratively until the performance improvement falls below a predefined threshold, at which point the algorithm terminates and returns the most compressed model that maintains performance above the specified threshold. Our approach directly optimizes for task-specific accuracy at each pruning step, ensuring that only layers with minimal impact on the target objective are removed. This exhaustive evaluation strategy, while computationally intensive during the pruning phase, provides strong empirical guarantees about the optimality of each pruning decision within the greedy framework.

Algorithm 1 TALE : Greedy Iterative Layer Pruning

Require: Pre-trained model $\mathcal{M}$ with L layers; validation set $\mathcal{D}_{val}$; performance threshold ϵ
Ensure: Compressed model $\mathcal{M}^*$

```
Initialize M* ← M
repeat
    for each layer ℓ ∈ {1, ..., L} of M* do
        Construct candidate model M_{-ℓ} by removing layer ℓ
        Compute validation accuracy A_ℓ = Acc(M_{-ℓ}, D_val)
    end for
    Select ℓ* = arg max_ℓ A_ℓ
    if A_{ℓ*} ≥ Acc(M*, D_val) − ε then
        Update M* ← M_{-ℓ*}
    else
        break
    end if
until All Accuracies below threshold
return M*
```

4 BENCHMARKS AND DATASETS

We evaluate TALE across a diverse suite of nine benchmarks spanning reasoning, language understanding, and commonsense knowledge. For mathematical reasoning, we include **GSM8K-Hard**, a curated subset of GSM8K Cobbe et al. (2021) with more than five premises per question to increase difficulty, and **MATH500** Hendrycks et al. (2021b), a benchmark for symbolic and arithmetic reasoning (for evaluation details see Appendix A). For language understanding, we consider **MMLU** Hendrycks et al. (2021a) and **BoolQ** Clark et al. (2019), while **Winogrande** Sakaguchi et al. (2021), **CommonsenseQA** Talmor et al. (2019), and **BIG-Bench** Srivastava et al. (2023) capture commonsense and multi-task generalization. Finally, we include both **ARC-Easy** and **ARC-Challenge** Clark et al. (2018), which evaluate scientific and factual reasoning at varying difficulty levels. Together, these nine datasets cover a broad spectrum of downstream challenges and allow us to assess both the generality and task-specific benefits of our pruning strategy.

5 RESULTS

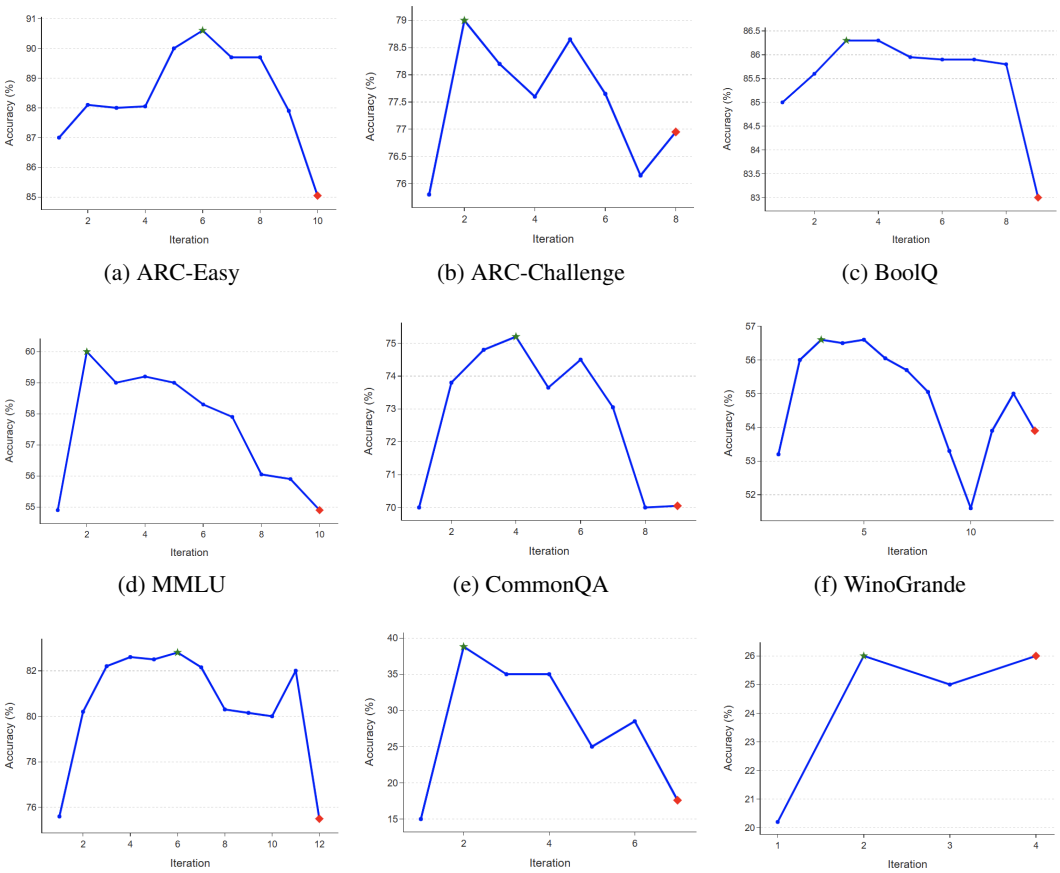
We evaluate TALE across five medium-scale models (LLaMA 3.1 8B, Mistral 7B, Lucie 7B, Qwen 2.5 7B) and one smaller model (Qwen 2.5 0.5B), spanning nine benchmarks that cover commonsense reasoning, reading comprehension, and mathematical problem solving. All experiments are conducted in the zero-shot setting unless otherwise noted.[1] Table 3 summarizes model configurations.

Iterative pruning trajectories. Figure 2 visualizes the iterative layer-pruning process for LLaMA 3.1 8B. Each curve tracks accuracy as layers are progressively removed. The ⋆ denotes the best-performing pruned model (*Best*), while the □ highlights the *Best Speedup with Baseline Accuracy* (BSBA) model—the pruned configuration achieving maximum inference speedup without falling below baseline accuracy.

From these trajectories, three consistent patterns emerge: (i) TALE identifies compressed models that *outperform* the original across diverse tasks, with ⋆ markers lying strictly above baseline. (ii) Accuracy improvements persist across multiple pruning steps before diminishing returns, showing that substantial redundancy exists even in carefully tuned pretrained models. (iii) Pruning dynamics are task-specific: datasets such as ARC-Easy and MMLU tolerate deeper pruning while continuing to improve, whereas reasoning-heavy tasks like GSM8K-Hard converge earlier, reflecting heterogeneous layer importance across domains.

Best vs. BSBA models. Table 1 compares baseline models against their pruned counterparts under both *Best* and *BSBA* configurations. Across all benchmarks, the Best models yield consistent

[1] Code available at `https://anonymous.4open.science/r/tale/`



(a) ARC-Easy (b) ARC-Challenge (c) BoolQ

(d) MMLU (e) CommonQA (f) WinoGrande

(g) BIG-Bench (h) GSM8K-Hard (i) Math500

Figure 2: Accuracy progression of TALE across 9 benchmark datasets for LLaMA 3.1 8B. Each curve represents the accuracy at successive iterations. The ⋆ denotes the best-performing layer drop configuration, while the □ highlights the Best Speed up with at least Baseline Accuracy (BSBA) configuration.

accuracy gains—up to $+146\%$ (LLaMA 8B on GSM8K-Hard), $+101\%$ (Lucie 7b on MMLU) and $+244\%$ (Qwen 7B on GSM8k-Hard)—while also delivering moderate speedups. BSBA models, by construction, trade smaller gains in accuracy for more aggressive speedups, offering practical operating points when inference cost is the dominant concern.

Few-shot setting. We tested TALE under the few-shot regime for Lucie and LLaMA models (Appendix Tables 4–5). Few-shot prompting improves baselines on reasoning tasks such as GSM8K and Math500, yet TALE-pruned variants still achieve higher accuracy in nearly all settings. This shows that pruning-induced improvements are largely complementary to gains from in-context learning.

Takeaways. TALE consistently uncovers Pareto-optimal models along the accuracy–efficiency frontier. By balancing task fidelity with computational savings, it enables both accuracy-focused and efficiency-focused deployment. The observed diversity in pruning profiles across datasets underscores the importance of adaptive pruning, rather than one-size-fits-all heuristics, for effective model compression. A tunable selection metric for choosing among candidate trade-offs is in Appendix E.

5.1 TALE AND FINE-TUNING: HOW DOES PRUNING INTERACT WITH TRAINING?

A natural question is whether pruning layers before or after fine-tuning harms the model's ability to learn. Intuitively, one might expect that removing layers reduces representational capacity and thus limits downstream fine-tuning performance compared to baseline instruct-tuned models. Sur-

Dataset	LLaMA 3.1 8B (zero-shot)							Qwen 2.5 7B (zero-shot)						
	Baseline	Best Model			BSBA			Baseline	Best Model			BSBA		
	Perf.	Perf.	#D	Sp.	Perf.	#D	Sp.	Perf.	Perf.	#D	Sp.	Perf.	#D	Sp.
ARC-Easy	87.00	**90.55**(+4.08% ↑)	5	1.3	87.82	8	1.4	90.04	**94.40**(+4.84% ↑)	3	1.2	90.08	7	1.5
ARC-Challenge	75.86	**78.62**(+3.63% ↑)	4	1.3	76.90	7	1.4	86.55	**92.00**(+6.45% ↑)	2	1.3	86.55	6	1.4
BoolQ	85.00	**86.20**(+1.40% ↑)	3	1.1	85.70	7	1.4	81.90	**83.90**(+2.44% ↑)	4	1.3	82.70	5	1.2
MMLU	54.87	**59.90**(+9.17% ↑)	1	1.1	54.87	9	1.4	68.10	**71.00**(+4.26% ↑)	5	1.2	68.13	6	1.3
CommonQA	72.20	**75.30**(+4.29% ↑)	3	1.2	73.10	6	1.3	80.30	**84.40**(+5.11% ↑)	2	1.1	80.50	6	1.3
Winogrande	53.83	**56.67**(+5.28% ↑)	4	1.3	53.83	12	1.5	62.04	**67.25**(+8.40% ↑)	3	1.2	62.19	6	1.5
BIG-Bench	75.20	**83.60**(+11.17% ↑)	5	1.3	75.20	11	1.6	79.20	**81.60**(+3.03% ↑)	6	1.5	81.60	6	1.5
GSM8K-HARD	15.07	**37.08**(+146.05% ↑)	1	1.1	17.31	6	1.4	7.8	**26.8**(+243.58% ↑)	2	1.1	8.99	5	1.2
Math500	20.50	**26.00**(+26.83% ↑)	3	1.2	26.00	3	1.2	18.00	**27.00**(+50.0% ↑)	2	1.1	21.00	4	1.2

Dataset	Lucie 7B (zero-shot)							Mistral 7B (zero-shot)						
	Baseline	Best Model			BSBA			Baseline	Best Model			BSBA		
	Perf.	Perf.	#D	Sp.	Perf.	#D	Sp.	Perf.	Perf.	#D	Sp.	Perf.	#D	Sp.
ARC-Easy	72.45	**76.55**(+5.66% ↑)	6	1.3	72.55	13	1.8	81.02	**83.45**(+4.23% ↑)	5	1.3	81.09	9	1.5
ARC-Challenge	48.00	**53.79**(+12.06% ↑)	7	1.4	51.38	11	1.8	72.20	**74.83**(+3.64% ↑)	6	1.4	72.41	8	1.4
BoolQ	53.70	**77.50**(+44.32% ↑)	5	1.3	60.60	19	2.8	80.36	**83.20**(+3.53% ↑)	6	1.2	80.60	10	1.5
MMLU	21.36	**42.98**(+101.2% ↑)	8	1.4	22.73	22	3.3	52.73	**57.81**(+9.63% ↑)	2	1.0	52.91	8	1.4
CommonQA	55.50	**69.70**(+25.59% ↑)	3	1.5	57.10	17	2.6	57.32	**61.40**(+7.12% ↑)	4	1.2	57.40	7	1.3
Winogrande	54.20	**57.80**(+6.64% ↑)	5	1.2	54.30	15	1.9	52.55	**58.80**(+11.53% ↑)	10	1.6	53.43	13	1.8
BIG-Bench	69.60	**77.20**(+9.84% ↑)	9	1.2	72.00	15	1.6	70.00	**76.40**(+9.14% ↑)	9	1.3	72.80	11	1.4
GSM8K-HARD	14.20	**17.80**(+25.35% ↑)	1	1.1	17.40	3	1.1	11.24	**19.10**(+69.92% ↑)	2	1.1	15.73	4	1.2
Math500	19.00	**27.00**(+42.11% ↑)	2	1.1	26.00	3	1.2	8.00	**16.00**(+100% ↑)	1	1.0	10.00	4	1.2

Dataset	Qwen 2.5 0.5B (zero-shot)						
	Baseline	Best Model			BSBA		
	Perf.	Perf.	#D	Sp.	Perf.	#D	Sp.
ARC-Easy	40.00	**60.91**(+48.49% ↑)	3	1.1	48.36	5	1.4
ARC-Challenge	35.52	**40.34**(+13.57% ↑)	1	1.1	37.24	4	1.5
BoolQ	62.30	**67.20**(+7.87% ↑)	5	1.4	66.20	6	1.5
MMLU	31.48	**39.97**(+26.96% ↑)	2	1.1	33.90	5	1.4
CommonQA	42.40	**49.10**(+15.80% ↑)	2	1.3	44.00	3	1.4
Winogrande	49.86	**51.88**(+4.51% ↑)	5	1.3	49.87	17	3.9
BIG-Bench	72.40	**73.60**(+1.66% ↑)	2	1.2	73.60	2	1.2
GSM8K-HARD	6.74	**11.24**(+66.77% ↑)	1	1.2	8.99	2	1.2
Math500	8.00	**12.00**(+50% ↑)	1	1.1	9	2	1.1

Table 1: Performance comparison across language models under 0-shot evaluation. We report accuracy (**Perf.**), number of dropped layers (**#D**), and relative inference speedup (**Sp.**). Percentage gain $= \frac{\text{Acc}_{\text{Best}} - \text{Acc}_{\text{Baseline}}}{\text{Acc}_{\text{Baseline}}} \times 100$. Best accuracy is highlighted in **bold**; BSBA shows balanced trade-offs.

prisingly, our experiments show the opposite: TALE not only preserves fine-tuning efficacy but in several cases improves both accuracy and efficiency.

We systematically explored four settings: (i) fine-tuning the base model (FT), (ii) applying TALE after fine-tuning (FT $\rightarrow$ TALE), (iii) pruning first and then fine-tuning (TALE $\rightarrow$ FT), and (iv) pruning first then finetuning and then again pruning (BASE $\rightarrow$ TALE $\rightarrow$ FT $\rightarrow$ TALE). Across reasoning and knowledge benchmarks, we consistently observed moderate but significant gains, especially on Winogrande (Table 2).

For example, pruning LLaMA-3.1 8B before fine-tuning reduced fine-tuning time by 2–2.5 GPU hours on an A100 (an 18.5% reduction) while simultaneously improving Winogrande performance by $+2.4\%$. Iteratively applying pruning and fine-tuning allowed us to prune up to 8 layers achieving still higher accuracy (87.37%) than the full fine-tuned model (85.00%). Similarly, pruning the fully fine-tuned model yielded a 7-layer reduction while maintaining strong accuracy (86.66%).

On knowledge-heavy tasks such as MMLU and CommonsenseQA, the gains were smaller but consistent: pruned fine-tuned models retained or slightly exceeded the accuracy of their full fine-tuned counterparts, while requiring fewer parameters to optimize. This suggests that pruning can act as a regularizer, simplifying the optimization landscape by removing redundant layers.

Model	Dataset	Baseline		Pruned Only		FT Only		Prune → FT		FT → Prune		(Prune → FT) → Prune	
		Perf.	#D	Perf.	#D	Perf.	#D	Perf.	#D	Perf.	#D	Perf.	#D
Llama 3.1 8B	Winogrande	53.83	0	56.67	4	85.00	0	87.06	4	86.74	7	87.37	8
	MMLU	54.87	0	59.90	1	63.62	0	63.49	1	64.21	2	64.01	2
	CommonQA	72.20	0	75.30	3	81.88	0	81.80	3	83.40	3	82.90	6
Qwen 0.5B	Winogrande	49.86	0	51.88	5	50.43	0	50.43	5	50.49	2	52.49	9
	MMLU	31.48	0	39.98	2	44.87	0	43.76	2	45.53	2	45.58	3

Table 2: Comparison of **Llama 3.1 8B** and **Qwen 0.5B** across Winogrande, MMLU, and CommonQA under different pruning and fine-tuning regimes. Columns denote: (i) Baseline = original model, (ii) Pruned Only = TALE without fine-tuning, (iii) FT Only = fine-tuned without pruning, (iv) Prune → FT = prune then fine-tune, (v) FT → Prune = fine-tune then prune, (vi) (Prune → FT) → Prune = best fine-tuned-pruned model further pruned. Perf. = performance score, #D = number of deleted layers.

Overall, these results highlight an unexpected but consistent trend: *pruning with* TALE *does not hinder fine-tuning but instead synergizes with it.* Pruned models fine-tune faster, require fewer parameters to adapt, and are close to or better in performance than their full counterparts. Pruning finetuned models almost always improves performance. This shows that TALE pruning can be effectively interleaved with fine-tuning to create models that are both more accurate and computationally efficient.

6 INFORMATION THEORY AND WHY PRUNED MODELS PERFORM BETTER.

Our experiments show that selectively dropping layers improves accuracy for a targeted task. This seems counterintuitive: why could removing parts of a carefully trained model improve it?

Alemi et al. (2016); Tishby & Zaslavsky (2015) have used information theory (Shannon, 1948) to analyze how neural networks learn and represent data. We can use also this tool to explain the effects of our pruning algorithm. Fano & Hawkins (1961) define $\mathrm{I(X;Y)}$, the mutual information between two random variables X and Y, with the equation:

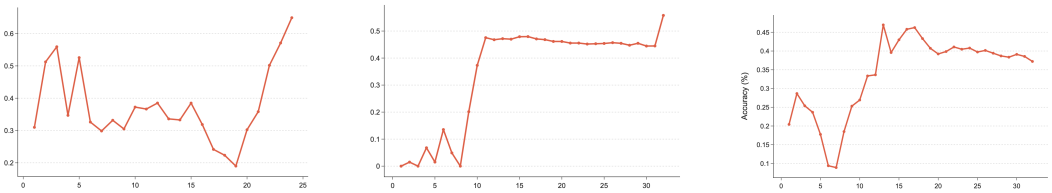
$$\mathrm{I(X;Y) = H(Y) - H(Y \mid X) = H(X) - H(X \mid Y)} = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log \frac{p(x,y)}{p(x)\,p(y)} \tag{1}$$

where $p(x, y)$ is the joint distribution of X and Y, and $p(x), p(y)$ are their marginals and where $\mathrm{H(X)} = -\sum_x p(x) \log p(x)$ is the Shannon (1948) entropy. $\mathrm{I(X;Y)}$ measures how much knowing X reduces uncertainty about Y Tishby & Zaslavsky (2015); Shwartz-Ziv & Tishby (2017).

To apply these notions in our case, we estimate $\mathrm{I}(\mathrm{X}^{(l)}, \mathrm{Y})$ at each layer using trained probes as mutual information approximators. While this approach yields $\mathrm{I}_{\text{probe}}(\mathrm{X}^{(l)}, \mathrm{Y})$, an approximation of the true mutual information, probe-based estimates capture the linearly accessible task-relevant information at each layer, which is meaningful since many downstream tasks employ linear classification heads Belinkov (2022).

This estimate together with these three observations yields an information-theoretic explanation for why architectural pruning can improve model performance. (i) Certain layers in large pre-trained transformers decrease mutual information between the layer's representation and the target task (Figure 3 and Table 7). (ii) From TALE we select the layer ℓ removed at the first iteration, and compare $\mathrm{I}\big(X^{(\ell+1)}, Y\big)$ before deletion of layer ℓ to its value after deletion. Deleting this layer always increases the mutual information at the subsequent layer on the tasks, effectively preventing the previous layer, which acts as noise, from obstructing the flow of information through the network.

Although our findings depend on the approximative nature of probe-based MI estimation, they provide evidence that certain layers in over-parameterized transformers act as information bottlenecks, which degrade rather than refine task-relevant representations. By removing layers that decrease mutual information with the target task, we enable representations with higher task-relevant information to flow directly to subsequent layers using the residual connection. This improves the information flow through the remaining architecture, results in representations that are more predictive of the target task, and yields improved accuracy at lower computational cost.



(a) ARC-Easy (Qwen 0.5B) (b) BoolQ (Lucie 7B) (c) BigBench (Llama 8B)

Figure 3: Evolution of mutual information (MI) across transformer layers for different benchmark datasets and different models. Each subplot shows how information is processed and transformed as it flows through the network layers, demonstrating distinct patterns of information propagation for (a) ARC-Easy on Qwen 0.5B, (b) BoolQ on Lucie 7B, and (c) BigBench on LLama 8B.

7 Discussion

We summarize five key observations below from our experiments.

1. Deleting later layers frequently improves performance on various tasks. This challenges prior claims that early layers are largely redundant and more amenable to removal, instead highlighting their essential role in preserving core task-relevant representations. Even deleting many late layers does not reduce accuracy below baseline, whereas removing even a single early layer is often catastrophic (see Figure 4). All models exhibit similar behavior.

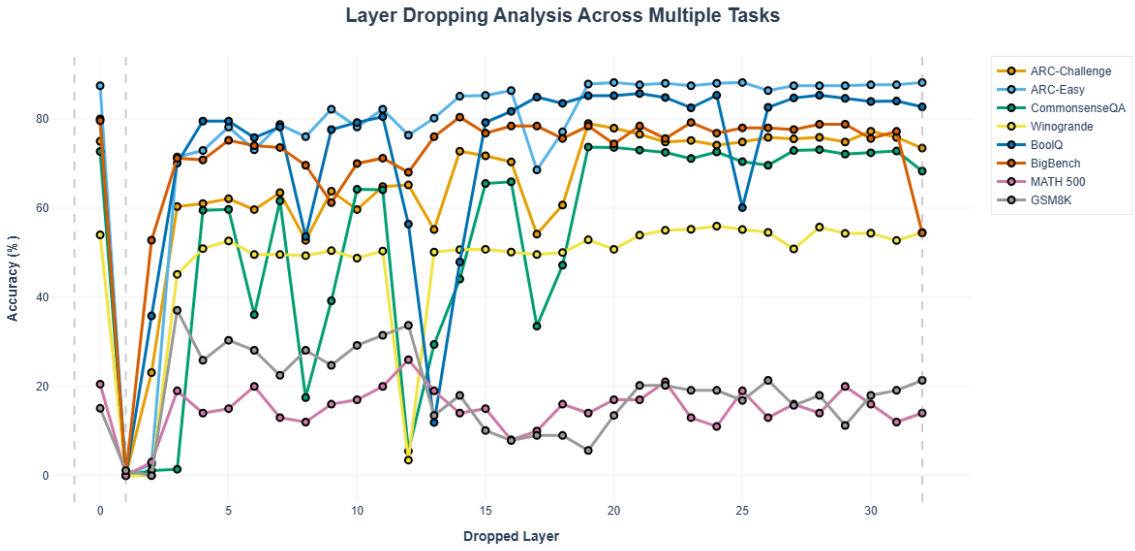


Figure 4: Nine benchmark tasks indicating performance after one layer is dropped from different positions in Llama3-8B.

On the other hand, early layers appear crucial for task-solving, even though probing outputs at those layers does not yield interpretable responses. These results are potentially important for model

interpretability. Plotting the performance degradation from ablating individual or consecutive layers, helps localize where specific task-solving abilities reside in the network.

2. Task dependence of layer importance. Which layers improve or harm performance when removed is highly task dependent. Sometimes a single layer is critical: for instance, removing layer 25 of Llama-8B on CommonsenseQA causes a 50-point accuracy drop. Removing Llama's layer 3 improves performance on GSM8K-hard but hurts MATH500; the revers happens when removing layer 11. Removing early layers (1–3) reduces accuracy to near zero on commonsense reasoning tasks (Figure 4), suggesting that certain early layers localize critical task-relevant information.

3. Structured task-specific patterns. Although pruning is task-specific, related tasks often exhibit similar layer dependencies. Commonsense reasoning tasks (see Figure 4) show importance concentrated in comparable regions of the network. Mathematical reasoning tasks benefit from pruning one to three early layers (e.g., LLaMA layer 3, Mistral layers 6 and 22, Lucie layer 12), but not more (Figures 7, 8, 9). Commonsense and language tasks (ARC, BoolQ, CommonsenseQA, Winogrande, and BIG-Bench) benefit from deleting later layers (Tables 7, 9, 8). This suggests that later layers often play a decoding role for predictions into natural language, which reinforces point 1—pruning them doesn't harm predictive capability.

We observe stronger pruning gains in reasoning-heavy tasks under zero-shot evaluation. All models showed notable accuracy boosts from one or two layer deletions on mathematical reasoning (e.g., Llama's large gain on GSM8K-hard, and double-digit gains on MATH500 for Qwen, Mistral, and Lucie). By contrast, knowledge-intensive tasks exhibit more modest improvements (e.g., an 11% gain for Llama on BIG-Bench).

4. Model-specific pruning effects. Different models display distinct pruning behavior. For example, pruned Lucie achieved a 101% gain on MMLU and double-digit gains on ARC-Challenge, CommonsenseQA, GSM8K-hard, and BoolQ. While Qwen-7B, Llama-8B and Mistral share similar architecture and scale, they had modest gains on these data sets. Lucie also benefitted from more substantial pruning than the other models. Interestingly, Lucie was trained on a much smaller dataset (3T tokens vs. 15T for Llama and 13T for Qwen) and also tuned for French conversational proficiency Gouvert et al. (2025), suggesting that some layers for one of our tasks may encode multilingual-specific functionality.

This highlights intriguing interactions between pretraining and pruning efficiency. One hypothesis is that models trained close to their performance ceiling (via large-scale pretraining, instruction tuning or RLHF) yield smaller pruning gains, whereas models trained under limited objectives may benefit more. Notably, even the large corpus trained Qwen-0.5B showed strong pruning efficiency gains.

We experimented with producing pruned models for several tasks, and we saw that a single, general pruning improved speed up without much loss in accuracy across multiple tasks (Appendix E). This shows that judicious pruning can preserve generalization ability across several tasks.

5. TALE is versatile. TALE can prune: base pre-trained models, instruction-tuned models (as we mainly do here), fine-tuned, and post-trained models with RLHF. Our results show that interleaving pruning and fine-tuning yields compounded benefits that may extend to other training regimes.

8 CONCLUSIONS

TALE is a generic toolkit for removing layers not needed for a certain task that yields increased performance and reduced computational costs. We have shown that it can profitably interact with further training or fine tuning, allowing users to further increase their models' task specific performance. We have also shown that TALE can improve models both at small and larger scales. TALE can benefit high-throughput applications with time constraints–e.g. in multi-agent systems with task-specific agents, real-time recommendation engines, or interactive AI assistants. TALE can also help smaller companies or organizations that face critical trade-offs between model capability and computational efficiency use large language models at scale.

REFERENCES

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.

Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *International conference on machine learning*, pp. 2285–2294. PMLR, 2015.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018. arXiv:1803.05457.

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

Chun Fan et al. Layer-wise neuron pruning using mutual information. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Robert M Fano and David Hawkins. Transmission of information: A statistical theory of communications. *American Journal of Physics*, 29(11):793–794, 1961.

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

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. 2023b. Referenced as Frantar and Alistarh (2023) in the survey.

Praveen Ganesh et al. Mint: Mutual information-based neuron trimming for dnn compression. *arXiv preprint arXiv:2003.08472*, 2020.

Olivier Gouvert, Julie Hunter, Jérôme Louradour, Christophe Cerisara, Evan Dufraisse, Yaya Sy, Laura Rivière, Jean-Pierre Lorré, et al. The lucie-7b llm and the lucie training dataset: Open resources for multilingual language generation. *arXiv preprint arXiv:2503.12294*, 2025.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015a.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015b.

Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pp. 293–299. IEEE, 1993.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 1389–1397, 2017.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021a. arXiv:2009.03300.

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

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R Devon Hjelm. Mine: mutual information neural estimation. *arXiv e-prints*, pp. arXiv–1801, 2018.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. Shortened llama: A simple depth pruning for large language models. *arXiv preprint arXiv:2402.02834*, 11:1, 2024.

François Lagunas, Ella Charlaix, Victor Sanh, and Alexander M Rush. Block pruning for faster transformers. *arXiv preprint arXiv:2109.04838*, 2021.

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

Yun Li et al. E-sparse: Boosting the large language model inference through entropy-based n:m sparsity. 2023. Referenced as Li et al. (2023b) in the survey.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*, 2024.

Sharan Narang, Erich Elsen, Gregory Diamos, and Shubho Sengupta. Exploring sparsity in recurrent neural networks. *arXiv preprint arXiv:1704.05119*, 2017.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021. doi: 10.1145/3452469.

Abigail See, Minh-Thang Luong, and Christopher D Manning. Compression of neural machine translation models via pruning. *arXiv preprint arXiv:1606.09274*, 2016.

Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

Suraj Srinivas and R Venkatesh Babu. Data-free parameter pruning for deep neural networks. arxiv 2015. *arXiv preprint arXiv:1507.06149*, 2015.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam R. Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ambrose Slone, Ameet Rahane, Anantharaman S. Iyer, Anders Andreassen, Andrea Madotto, Andrea Santilli, Andreas Stuhlmüller, Andrew Dai, Andrew La, Andrew Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, and ... others. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=uyTL5Bvosj. Preprint / TMLR.

Mingjie Sun et al. A simple and effective pruning approach for large language models. 2024. Referenced as Sun et al. (2024) in the survey.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pp. 4149–4158. Association for Computational Linguistics, June 2019. doi: 10.18653/v1/N19-1421.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pp. 1–5. Ieee, 2015.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*, 2019.

Daniel Westphal et al. Mutual information preserving pruning (mipp). *arXiv preprint arXiv:2411.00147*, 2024.

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

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared llama: Accelerating language model pre-training via structured pruning. 2024.

Stephen Zhang and Vardan Papyan. Oats: Outlier-aware pruning through sparse and low rank decomposition. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yuxin Zhang, Lirui Zhao, Mingbao Lin, Sun Yunyun, Yiwu Yao, Xingjia Han, Jared Tanner, Shiwei Liu, and Rongrong Ji. Dynamic sparse no training: Training-free fine-tuning for sparse llms. In *The Twelfth International Conference on Learning Representations*.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 12: 1556–1577, 2024.

A Implementation Details

Hardware. All experiments were conducted on 1 NVIDIA A100 GPU with 80GB memory.

Models. We applied TALE to five open-weights LLMs of varying scales: **Qwen2.5-0.5B-Instruct**, **Qwen2.5-7B-Instruct**, **Lucie-7B-Instruct**, **Mistral-7B-Instruct**, and **Llama-3.1-8B-Instruct**.

Datasets for TALE pruning. The greedy layer-pruning algorithm was evaluated across nine widely used benchmarks covering reasoning, commonsense, and knowledge-intensive tasks: **ARC-Challenge**, **ARC-Easy**, **MMLU**, **Winogrande**, **GSM8K**, **MATH500**, **CommonQA**, **BIG-Bench**, and **BoolQ**.

Pruning setup. At each iteration, TALE evaluates all candidate single-layer deletions with respect to validation accuracy. The pruning threshold was defined as the baseline accuracy of the full model, ensuring that pruning never reduces performance relative to the original unpruned model. The iterative procedure terminates once no further layer removals satisfy this criterion.

Fine-tuning setup. For fine-tuning experiments, we focused on **Winogrande** and **MMLU**. We employed LoRA with rank 64, a batch size of 4, and the optimizer `paged_adamw_32bit`. A cosine learning rate scheduler was used, and models were trained for 10 epochs.

Evaluation. We measured performance in terms of task accuracy on the test portions of datasets through automatic evaluation, with no human assessment involved, while inference efficiency was reported as relative speedup over the full baseline model. To evaluate MATH500 and GSM8K automatically without any human verification, we force the model to predict the final answer after the reasoning steps in a particular format and then calculate the accuracy. This could lower accuracy from what is expected; but since we used the same evaluation criteria for all the techniques and models and are interested in relative improvements or decreases in performance, the increase/decrease due to our methods from what might have been expected is justified.

Prompting. For zero-shot and few-shot evaluation, we used task-specific prompts. Below we show the prompt used for datasets, consisting of a system instruction :

ARC-E & ARC-C System Prompt

You are a Science expert assistant. Your task is to answer multiple-choice science questions at grade-school level. Each question has four answer choices, labeled A, B, C, and D.
For each question: - Carefully read the question and all answer choices. - Select the single best answer from the options (A, B, C, or D). - Respond only with the letter of the correct answer, and nothing else—no explanation or extra words.
Be precise and consistent: Only the answer letter.

Bigbench System Prompt

"You are a boolean expression evaluator. You must respond with exactly one word: either 'True' or 'False'. Do not provide explanations, steps, or any other text. Only respond with 'True' or 'False'."

BOOLQ System Prompt

"You are a helpful assistant that answers True/False questions based on given passages. Read the passage carefully and determine if the question can be answered as True or False based on the information in the passage. "Respond with only 'A' for True or 'B' for False."

CommonQA System Prompt

"You are a helpful assistant that answers multiple-choice questions requiring commonsense knowledge and reasoning. Read each question carefully and select the most logical answer from the given options based on common knowledge and reasoning. Respond with only the letter of your chosen answer (A, B, C, D, or E)."

GSM8K System Prompt

"You are a math problem solver. Solve the given math problem step by step. " "Show your complete reasoning and calculations. " "At the end, write your final answer after '####' like this: #### [your final numerical answer]""

MMLU System Prompt

"You are a helpful assistant that answers multiple-choice questions across various academic subjects including humanities, social sciences, STEM, and professional fields. Read each question carefully and select the best answer from the given options. Respond with only the letter of your chosen answer (A, B, C, or D)."

Winogrande System Prompt

You are a careful math problem solver. Show complete step-by-step reasoning and all calculations needed to arrive at the answer. Use clear, numbered or labeled steps so the reasoning is easy to follow.
IMPORTANT (formatting):

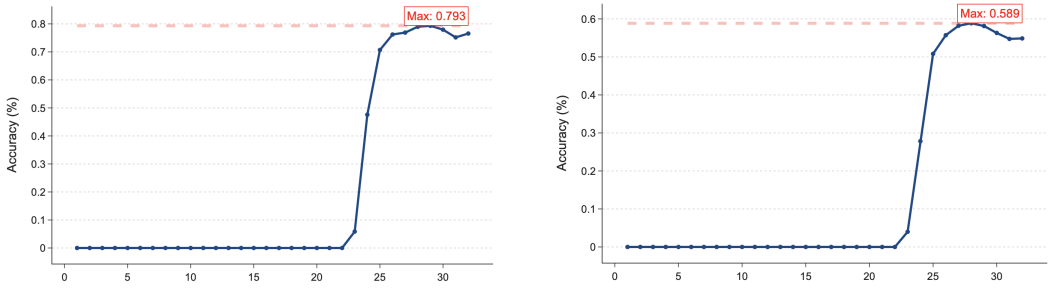
- After the full reasoning, write the **final answer on a new line by itself** in exactly this format:
  ####
  integer
- `<integer>` must be digits only, optionally with a leading "-" for negatives (e.g., `-7`).
- Do **not** add words, punctuation, units, or commentary on the same line as the #### line.
- The #### line must be the **final line of the output** (nothing may follow it).
- Assume all problems expect integer answers; ensure the final line contains a single integer.

B NUMBER OF PARAMETERS PER LAYER FOR EACH MODEL

Model	LLaMA 3.1 8B	Qwen 2.5 7B	Mistral 7B	Lucie 7B	Qwen 2.5 0.5B
Parameters	218,112,000	233,057,792	218,112,000	192,946,176	14,912,384

Table 3: Model parameter counts comparison. LLaMA 3.1 8B, Mistral 7B and Lucie 7B has 32 layers, Qwen 2.5 7B has 28 layers and Qwen 2.5 0.5B has 24 layers.

C INTUTION BEHIND TALE



(a) ARC-Challenge (b) MMLU

Figure 5: Layer-wise output performance for LLaMA models: results when generating predictions from intermediate layers 1 through 32 on three different datasets.

D RESULTS ON FEW-SHOT LEARNING ON LARGER MODELS

Dataset	Lucie 7B few-shots						
	Baseline	Best Model			BSBA		
	Perf.	Perf.	#D	Sp.	Perf.	#D	Sp.
ARC-Easy	69.2	72.36	9	1.41	71.27	12	1.68
ARC-Challenge	49.31	55,17	9	1.39	51.72	13	1.67
BoolQ	77.6	79.10	6	1.22	78.5	10	1.27
MMLU	41.02	43.44	7	1.26	41.48	11	1.55
COMMONQA	55.4	69.7	3	1.22	57.10	17	2.02
WINOGRANDE	52.8	56.90	12	1.58	53.30	17	1.74
BIG-Bench	68.8	77.20	9	1.61	72	15	2.23
GSM8K-HARD	26.97	29.21	1	1.03	26.97	2	1.1

Table 4: Results of **Lucie 7B** across nine benchmarks. All tested on 5-shots, except gms8k on 8-shots Performance (%) cells are color-coded: green = gain, red = decline, and gray = near-neutral change compared to baseline.

Dataset	LLaMA 3.1 8B few-shots						
	Baseline	Best Model			BSBA		
	Perf.	Perf.	#D	Sp.	Perf.	#D	Sp.
ARC-Easy	90.36	**92.18** 2.01% ↑	4	1.14	90.91	8	1.37
ARC-Challenge	78.2	**83.10** 6.27% ↑	3	1.17	78.62	9	1.42
BoolQ	82.7	85.3 3.1% ↑	4	1.11	83.0	6	1.22
MMLU	59.2	62.38 5.37% ↑	4	1.14	59.57	7	1.26
COMMONQA	73.30	75.30 2.72% ↑	6	1.22	73.80	7	1.32
WINOGRANDE	57.01	60.15,26% ↑	3	1.1	57.02	8	1.3
BIG-Bench	70.0	83.60 19,43% ↑	5	1.2	81.20	15	1.83
GSM8K-HARD	60.67	60.67	0	1	60.67	0	1
MATH500	44.00	49.00 11.36% ↑	1	1.02	45.00	2	1.03

Table 5: Results of **LLaMA 3.1 8B** across nine benchmarks. All tested on 5-shots, except gms8k and MATH500 on 8-shots

E A TUNABLE METRIC FOR FINDING ACCURACY VS. SPEED UP OPTIMIZATION

To systematically select among these candidates according to user priorities, we propose the Accuracy–Efficiency Harmonic Mean (AE-HM):

$$r_A = \frac{\text{Acc(Model)}}{\text{Acc(Baseline)}}, \qquad \text{AE-HM(Model)} = \frac{(1+\lambda^2) r_A \, S}{\lambda^2 S + r_A} = \frac{1+\lambda^2}{\frac{\lambda^2}{r_A} + \frac{1}{S}} \tag{2}$$

where S denotes the relative inference speedup and λ controls the relative importance of accuracy versus efficiency. The user can set AE-HE's parameter λ to desired specifications: if $\lambda > 1$, we prioritize r_A; if $\lambda < 1$ we prioritize Speedup.

By computing AE-HM for candidate models, we can automatically identify the model with the highest score for a given task or a set of tasks given a particular AE-HM parameter setting:

$$M_{\text{best-compromise}} = \arg\max_i \text{AE-HM}(M_i) \tag{3}$$

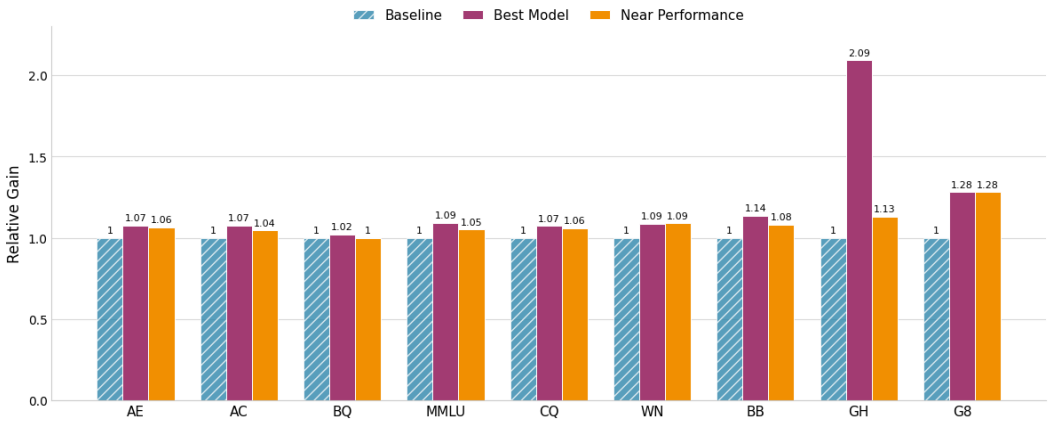


Figure 6: Relative Gain comparison across datasets. LLaMA $\beta = 3$

F Deleted Layers in each Model and Benchmark

Dataset	Best Model	BSBA
ARC-Easy	19 25 27 28	19 20 21 24 25 26 27 28
ARC-Challenge	19 22 27	19 20 21 22 23 24 26 27 28
BoolQ	19 25 26 32	15 19 21 22 25 26 30 32
MMLU	20 21 27 28	20 21 22 24 27 28 32
CommonQA	21 22 27 28 31 32	21 22 23 27 28 31 32
Winogrande	20 22 24	17 19 20 22 24 26 29 32
BIG-Bench	11 16 20 21 26	10 11 16 20 21 22 23 24 26 27 28 29 30 31 32
MATH500	28	24 28

Table 6: Deleted layers represented as color-coded inline numbers. Blue = Best Model, Orange = BSBA for LlaMA 3.1 8B with few-shots.

Table 7 shows how using AE-HM allows us to bring model size down effectively on our BSBA Llama model with 0 shot performance on our nine data sets. The BSBA LLama model had speed up gains between 27 and 46% on our various benchmarks and maintained performance at or above original model levels (See Table 7).

Dataset	Best Model	BSBA
ARC-Easy	19 20 21 29 32	19 20 21 22 25 27 29 32
ARC-Challenge	19 20 23 27	19 20 21 23 25 27 28
BoolQ	21 23 28	18 21 22 27 28 32
MMLU	21	19 21 22 24 25 26 27 28 31
CommonQA	19 23 28	19 22 23 26 27 28
Winogrande	23 24 26 32	20 21 22 23 24 25 26 27 29 31 32
BIG-Bench	14 20 22 28 29	14 18 20 21 22 23 24 28 29 31 32
GSM8K-Hard	3	3 21 22 25 26 27 29

Table 7: Deleted layers represented as color-coded inline numbers. Blue = Best Model, Orange = BSBA for LlaMA 3.1 8B 0 shot.

Dataset	Best Model	BSBA
ARC-Easy	19 22 28	6 19 22 24 26 27 28
ARC-Challenge	27 28	7 22 23 26 27 28
BoolQ	18 21 27 28	12 19 21 22 26 27 28
MMLU	22 23 26 27 28	18 22 23 26 27 28
CommonQA	22 28	6 21 22 23 27 28
Winogrande	22 26 27	6 20 22 25 26 27
BIG-Bench	10 19 23 25 26 27	10 19 23 25 26 27

Table 8: Deleted layers represented as color-coded inline numbers. Blue = Best Model, Orange = BSBA for **Qwen 2.5 7B** zero-shot.

Dataset	Best Model	BSBA
ARC-Easy	15 16 23 24 27 28	13 15 16 18 19 20 21 22 23 24 25 27 28
ARC-Challenge	16 18 20 21 23 25 26	15 16 18 19 20 21 22 23 25 26 28
BoolQ	8 17 25 28 29	5 8 11 12 13 14 15 16 17 19 20 23 25 26 27 28 29 31
MMLU	11 12 15 16 20 21 22 28	5 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 28 30 31
CommonQA	11 12 27	11 12 13 15 16 17 18 19 20 21 22 23 24 25 27 28
BIG-Bench	6 7 15 17 20 21 25 26 27	6 7 13 15 17 19 20 21 22 24 25 26 27 28 29
GSM8K-Hard	12	12 21 23

Table 9: Deleted layers represented as color-coded inline numbers. Blue = Best Model, Orange = BSBA for Lucie 7B 0 shots.

Dataset	Best Model	BSBA
ARC-Easy	21 22 24 26 29	21 22 23 24 25 26 29 30 32
ARC-Challenge	22 24 25 27 28 30	21 22 24 25 26 27 28 30
BoolQ	17 22 23 24 27 32	12 17 21 23 24 25 27 28 32
MMLU	24 30	22 23 24 25 26 27 30 32
CommonQA	19 22 25 28	19 21 22 24 25 28 32
Winogrande	18 19 20 22 23 24 26 27 31 32	4 13 18 19 20 22 23 24 26 27 29 31 32
BIG-Bench	3 5 15 22 23 24 26 27 28	3 5 14 15 18 22 23 24 26 27 28
GSM8K-Hard	6 22	6 11 22 28

Table 10: Deleted layers represented as color-ccdinline numbers. Blue = Best Model, Orange = BSBA for **Mistral** zero-shot.

G COMMON PRUNED LAYERS MODEL

Group	Dataset	**Baseline**	**Pruned Model**	speedup
Common-sense	ARC-Easy	87.0	87.82	1.2
	ARC-Challenge	75.86	75.00	1.21
	CommonQA	72.20	64.70	1.1
	Winogrande	54.20	50.57	1.13
Reading	BoolQ	85.0	85.5	1.17
	BIG-Bench	75.2	67.2	1.1

Table 11: Accuracy of LLaMA-3.1-8B (baseline) versus a pruned variant obtained by dropping layers selected through BSBA. For each task, BSBA identified removable layers, and we retained the intersection of layers that appeared in at least 75% of tasks within the Common-sense group (layers 19, 22, 23, 27) and (layers 18, 21, 22, 28, 32) for Reading Comprehension tasks. These layers were then pruned globally from the model, and performance was re-evaluated across tasks. Speedup is reported relative to the baseline.