TELL-TALE: Task Efficient LLMs with Task Aware Layer Elimination

Omar Naim[*]
IRIT
Université de Toulouse
omar.naim.docs@gmail.com

Krish Sharma[*]
IRIT
Université de Toulouse
krish.sharma@irit.fr

Niyar R Barman
IRIT
Université de Toulouse
niyar.barman@irit.fr

Nicholas Asher
IRIT
CNRS
nicholas.asher@irit.fr

Abstract

Large Language Models (LLMs) typically come with a fixed architecture, despite growing evidence that not all layers contribute equally to every downstream task. We introduce TALE (Task-Aware Layer Elimination), an inference-time method that improves task performance by selectively removing layers that are irrelevant or detrimental for a given task. TALE optimizes task-specific performance, yielding a task-optimized architecture without retraining. Across 9 tasks and 5 model families, under both zero-shot and few-shot settings, TALE consistently matches or surpasses baseline performance while simultaneously reducing computational costs. TALE also synergizes with fine-tuning, leading to further performance improvements. Computing TALE for a new task requires modest resources, making it a practical and deployable solution for task-specialized LLM inference.

1 Introduction

Substantial computational costs of Large Language Models (LLMs) can limit their use in resource-constrained settings and high-throughput applications. This has motivated a search for approaches that improve task performance while reducing computational cost. The growing use of multi-agent systems, where LLMs are specialized for particular roles, further amplifies this need. However, developing methods that address it remains challenging. Fine-tuning can improve task performance but does not reduce inference cost and requires substantial training data and compute. General pruning methods reduce computational cost but typically require retraining and often degrade performance on downstream tasks.

To address this need, we introduce TALE (Task-Aware Layer Elimination), a simple, greedy, and iterative method that removes layers based on their impact on task-specific validation accuracy. At each step, TALE evaluates all possible single-layer removals and selects the one that maximizes validation performance, repeating this process until performance falls below a user-defined threshold. The method is hardware-agnostic, requires no retraining, and operates entirely at inference time. Unlike existing pruning approaches, TALE directly optimizes task-specific accuracy at each pruning step, enabling it to improve performance over the original model while remaining complementary to fine-tuning.

Our study of state-of-the-art, mid-sized transformers extends prior work showing that not all layers contribute equally to downstream tasks (De Vries et al., 2020; Dalvi et al., 2020; Sajjad et al., 2023). We find that layer importance is highly task-dependent, and that removing task-misaligned layers can improve both efficiency and accuracy.

We showcase TALE on five modern LLMs (LLaMA 3.1 8B, Qwen 2.5 7B, Qwen 2.5 0.5B, Mistral 7B, and Lucie 7B) across nine diverse benchmarks. We compare against prior training-free pruning methods on LLaMA 2 (7B and 13B), showing that TALE achieves higher accuracy with lower computational cost. These improvements arise without retraining, architectural modification, or changes to the underlying model weights; they persist under alternative data splits, random seeds, and evaluation protocols. Overall, TALE provides a practical approach for building task-specialized LLMs: it can be applied *post hoc* to existing checkpoints and complements fine-tuning.

2 Related work

While a large body of work focuses on improving the efficiency of LLMs through pruning, sparsity, and model compression, most approaches prioritize reducing computational cost, often at the ex-

[*]Equal contribution

Findings of the Association for Computational Linguistics: ACL 2026, pages 22616–22638

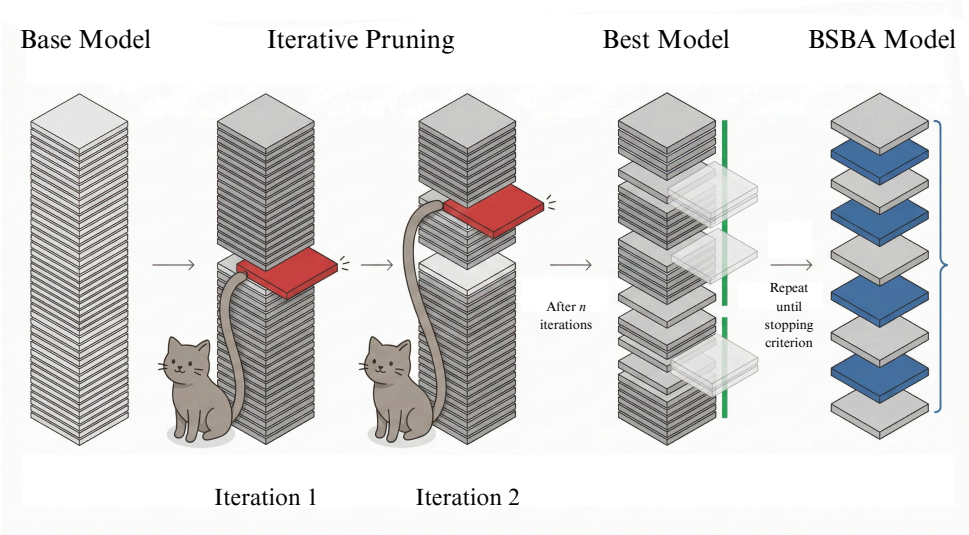


Figure 1: How TALE works to produce BEST and Best Speedup above Baseline (BSBA) models.

pense of downstream performance. Moreover, relatively little work explores using pruning to improve task-specific accuracy.

Structured pruning methods, which remove entire components such as neurons, attention heads, or layers (He et al., 2017; Voita et al., 2019; Lagunas et al., 2021), are the most closely related methods to TALE. Recent work has explored both training-free and analysis-driven approaches to understanding and reducing redundancy in pretrained transformers. For instance, Zhang et al. (2024) propose training-free fine-tuning of sparse LLMs, while Dalvi et al. (2020) analyze redundancy using layer-level similarity and task-specific probing. **SLEB** (Song et al., 2024) removes layers based on cosine similarity of their representations and validates pruning decisions using perplexity on a linguistic dataset. Similarly, BlockPruner (Zhong et al., 2025) segments Transformer layers into attention and MLP blocks and prunes them using perplexity-based heuristics. Because these methods rely on general-purpose metrics such as perplexity or representational similarity, they are largely task-agnostic.

Other approaches operate at different levels of granularity. **SliceGPT** (Ashkboos et al., 2024) reduces layer dimensionality via principal component analysis. Early-exit methods (Huang et al., 2024; Woczyk et al., 2021) dynamically determine stopping layers at inference time, typically using auxiliary predictors or retrieval mechanisms. At a finer granularity, **SparseGPT** (Frantar and Alistarh, 2023) and **Wanda** (Sun et al., 2023) perform unstructured pruning based on local reconstruction criteria or magnitude-activation products, achieving high sparsity but often degrading downstream performance. Methods such as OWL (Yin et al., 2024), FLAP (An et al., 2024), and DarwinLM (Tang et al., 2025) explore non-uniform sparsity or evolutionary search, primarily targeting efficiency gains.

Peer et al. (2022) propose greedy layer pruning (GLP), which iteratively removes layers to meet a target accuracy. While superficially similar, GLP differs from TALE in two key ways: it requires retraining on the downstream task and removes a predefined number of layers. In contrast, TALE requires no retraining and adaptively determines the number of layers to remove based on validation performance.

Overall, TALE differs from prior training-free approaches in both granularity and optimization objective. It operates at the layer level and directly optimizes task-specific validation accuracy, whereas existing methods typically rely on general-purpose criteria such as perplexity or reconstruction error. As shown in Section 5.1, this leads to consistent improvements in downstream accuracy while reducing computational cost. These results also suggest that commonly used proxies, such as representational similarity or perplexity, may be suboptimal for identifying task-relevant layers.

3 Basics and Intuitions

A transformer maps a sequence of input vectors $X^{(0)} = (x_1, \ldots, x_n)$ to a sequence of hidden representations through a stack of L layers. Each layer ℓ takes $X^{(\ell-1)}$ as input and produces

$$X^{(\ell)} = (x_1^{(\ell)}, \ldots, x_n^{(\ell)}),$$

by applying self-attention and feed-forward blocks with residual connections. Concretely, each layer updates the residual stream as

$$X^{(\ell)} = X^{(\ell-1)} + F_\ell(X^{(\ell-1)}), \quad \ell = 1, \ldots, L,$$

where F_ℓ denotes the transformation implemented by layer ℓ. From this perspective, removing a layer ℓ corresponds to setting $F_\ell \equiv 0$, omitting its residual contribution and reducing the update to the identity map:

$$X^{(\ell)} = X^{(\ell-1)}.$$

This provides a natural interpretation of layer-wise pruning as deleting components of the residual update while preserving the overall stream structure.

Intermediate representations. To better understand the role of individual layers, we examine intermediate hidden states. Instead of using only the final representation $X^{(L)}$, we project intermediate representations $X^{(k)}$ for $k < L$ directly into the vocabulary space using the output projection matrix W_{out}:

$$\hat{y}^{(k)} = \text{softmax}\Big(W_{\text{out}}\, x_n^{(k)}\Big).$$

While evaluating $\hat{y}^{(k)}$ across different values of k, we were surprised to find that for many tasks, intermediate layers ($k < L$) can achieve higher accuracy than the final layer (Figure 6). This suggests that additional layers do not always improve task-specific performance: some layers contribute marginally, while others may introduce representational noise. Not all layers in an LLM are equally useful, and selectively removing redundant layers can preserve or even improve downstream accuracy.

We formalize this idea in the next section as a greedy, task-aware layer pruning procedure.

4 TALE

TALE is a greedy layer-pruning algorithm for compressing pre-trained open-weight LLMs. It systematically removes layers while preserving, and sometimes improving, task performance (Algorithm 1). Starting from the full pre-trained model, TALE evaluates all possible single-layer removals at each step and computes the validation accuracy of each candidate architecture. The layer whose removal yields the highest validation accuracy is permanently eliminated, and the resulting compressed model becomes the starting point for the next step. This process continues until performance falls below a predefined threshold.

For this paper, we set the stopping threshold to 8% below the baseline task accuracy, allowing the search to explore slightly lower-performing intermediate architectures in case later pruning steps yield larger gains. In practice, however, we did not observe any cases in which performance recovered after falling below the baseline. Once the threshold is reached, the algorithm returns the most compressed model whose performance remains above the threshold.

Algorithm 1 TALE : 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 the 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 are below the threshold
return M*
```

4.1 Benchmarks

We evaluate TALE across a diverse suite of nine benchmarks spanning reasoning, language understanding, and commonsense reasoning tasks. For

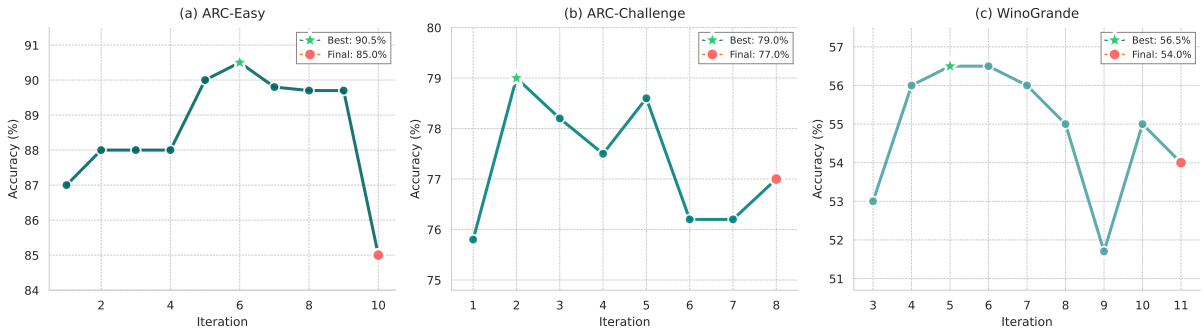


Figure 2: Accuracy progression of TALE across three benchmark datasets for LLaMA 3.1 8B. Each curve represents the accuracy at successive iterations. The ⋆ denotes the best-performing layer drop configuration, while the ● highlights the best speedup with at least baseline accuracy (BSBA) configuration. Plots for all tasks are provided in Appendix K.

mathematical reasoning, we use **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).

For language understanding, common sense reasoning, and multi-task generalization, we use **MMLU** (Hendrycks et al., 2021a) and **BoolQ** (Clark et al., 2019); **Winogrande** (Sakaguchi et al., 2021), **CommonsenseQA** (Talmor et al., 2019), and **BIG-Bench** (Srivastava et al., 2023), as well as **ARC-Easy** and **ARC-Challenge** (Clark et al., 2018).

4.2 Evaluation protocol

TALE is in effect an **end-to-end optimizer**; it optimizes a model for a particular task and a particular evaluation. So it is important to evaluate TALE with respect to different evaluations procedures.

We consider two complementary evaluation methods. First, we use the standard LM-Eval framework, which selects the answer with the highest probability among the provided options for multiple-choice tasks. While this approach is widely adopted, it does not always capture the quality of the models generated outputs.

To complement this, we introduce an additional evaluation procedure, **Decoder Eval**, which directly evaluates the models generated responses. In this setting, the model is prompted to produce its answer in a structured format. We then extract the final answer from the generated output and compare it to the ground truth to compute accuracy.

4.3 The dynamics of TALE 's iterations

To better understand how TALE behaves during optimization, we analyze how task performance evolves throughout the iterative pruning process.

Figure 2 illustrates representative optimization trajectories for LLaMA 3.1 8B on three benchmark tasks. Each curve shows the change in accuracy as layers are progressively removed. Across tasks, we observe a consistent pattern: the first pruning step often yields a noticeable improvement in accuracy, followed by smaller gains or mild fluctuations. After this initial phase, performance typically decreases monotonically as additional layers are removed, eventually falling below the baseline. The full set of trajectories is shown in Figure 10. These curves suggest structured, task-dependent redundancy patterns that merit further investigation.

From these dynamics, we identify three key properties of TALE :

(i) **Existence of high-performing compressed models.** TALE often identifies configurations that outperform the original model on a given task. We refer to these as **Best (BEST)** models. In addition, **Best Speedup with at least Baseline Accuracy (BSBA)** denotes the most compressed configurations that remain within the performance threshold defined by the baseline.

(ii) **Multi-step gains before degradation.** Performance improvements are not limited to the first pruning step; in many cases, accuracy improvements persist across multiple iterations before diminishing returns appear, indicating substantial redundancy in these pre-trained models.

(iii) **Task-dependent pruning dynamics.** The behavior of TALE varies across tasks. For ex-

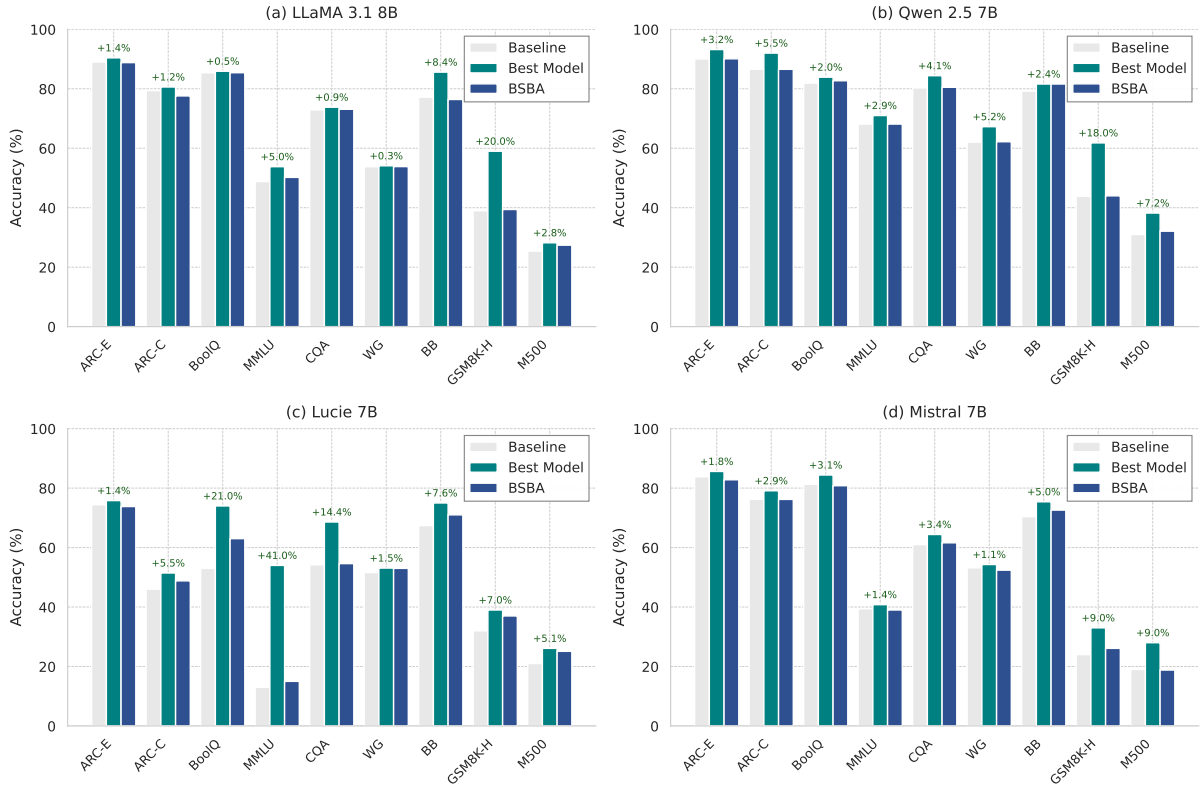


Figure 3: **Zero-shot performance comparison across four language models with layer dropping** Light blue bars show baseline performance, while dark blue bars represent best model performance after strategic layer dropping. Percentages on top of the columns represent the absolute accuracy gained with best models over base models. Results demonstrate that dropping 1 to 10 layers can improve or maintain performance across most benchmarks, with notable gains on mathematical reasoning tasks (e.g., LLaMA 3.1 8B: 39.0% 59.0% on GSM8K-HARD with only 1 layer dropped). This suggests significant redundancy in deeper language model architectures.

ample, ARC-Easy and MMLU tolerate deeper pruning while maintaining or improving performance, whereas reasoning-intensive tasks such as GSM8K-Hard reach peak performance earlier and degrade more quickly, reflecting differences in layer importance across domains.

5 Results

We evaluate TALE along five dimensions: (i) overall performance across models and tasks, (ii) robustness across random seeds and evaluation protocols, (iii) comparison with state-of-the-art pruning methods, (iv) computational efficiency, and (v) interaction with fine-tuning and few-shot learning.

5.1 Overall Performance

We first evaluate TALE in a zero-shot setting on four medium-scale models (LLaMA 3.1 8B, Mistral 7B, Lucie 7B, Qwen 2.5 7B) and one smaller model (Qwen 2.5 0.5B). Figure 3 and Table 17 compare baseline models with their pruned counterparts.

Across all models and benchmarks, we observe consistent improvements in accuracy after applying TALE , although the magnitude of gains varies across tasks and architectures.[1] On ARC-Challenge, improvements are modest for LLaMA (+1.6%) but more pronounced for Qwen 2.5 7B (+6.3%). Reasoning-intensive tasks benefit the most, with gains ranging from 23% to 51% on MATH500 and GSM8K across models. Results obtained under LM-Eval and Decoder Eval exhibit the same behavior (for details see Table 5 in Appendix E and Table 7 in Appendix H), indicating that improvements are not specific to a single evaluation protocol.

Beyond accuracy, TALE also reduces computational cost by removing unnecessary layers, highlighting substantial redundancy in modern language models. Notably, selectively removing task-misaligned layers can *improve* performance, rather than degrade it as commonly expected in pruning.

[1]Code available at https://github.com/omyokun/tale/

Taken together, these findings establish the central empirical result of this work: **task-aware layer pruning can simultaneously improve downstream performance and reduce model depth**.

Remark on Validation Set Size In our study, TALE requires only modestly-sized sets for task-specific optimization, ranging from 500 to 1500 examples. As seen in Table 4 (Appendix D), once the validation set size exceeds 500 examples, the set of layers dropped stabilizes across all tasks.

5.2 Robustness

We evaluate the robustness of TALE to random initialization by varying the pruning seed. Figure 4 shows that performance variations across seeds are minimal. Table 5 (Appendix E) reports the mean and variance over five seeds for the best configurations of LLaMA, Qwen, Lucie, and Mistral.

Across all models, we observe consistently low variance, indicating that TALE is stable with respect to seed choice and does not rely on favorable initialization. These results demonstrate that the performance gains of TALE are robust and reproducible.

Figure 4: **Model Stability Across Datasets.** Heatmap visualizing the standard deviation (σ) of the final accuracy scores for the Best Model variant of Llama, Qwen, Lucie and Mistral, four different LLMs in the 7-8b range across nine datasets. The results are aggregated from five independent runs (seeds). Lighter green indicates higher stability (lower σ); darker green, lower stability (higher σ). LLaMA 3.1 8B shows the overall lowest variance.

5.3 Comparisons

We compare TALE against recent training-free pruning methods on LLaMA-2-7B and LLaMA-2-13B, including SLEB, SparseGPT, Wanda, and SliceGPT. Results are reported in Figure 5.

Across all benchmarks, TALE consistently outperforms prior approaches at comparable sparsity levels and is the only method that reliably matches or exceeds the performance of the unpruned baseline across tasks. In contrast, existing pruning methods generally degrade downstream accuracy while reducing model depth.

A key distinction between TALE and prior work is the optimization objective. While most existing methods rely on general-purpose signals such as perplexity or representation similarity to identify redundant layers, TALE directly optimizes task-specific validation accuracy. This task-aware objective leads to more effective identification of layers that are detrimental for downstream performance.

To show this, we used a representational similarity technique (cosine similarity or cossim) to guide TALE. TALE registered drops similar to those seen with SLEB and other similarity driven optimization methods. On ARC-Easy, for instance, cossim led TALE to drop 2 layers, **dropping task accuracy** from Llama's baseline of **79.5** to a pruned model accuracy of **58.5** with a time speed up of 1.32. Table 11 in Appendix M shows that TALE suffers similar drops in performance when perplexity is used as an optimizing objective.

In addition, we evaluate task-aware variants of SLEB and BlockPruner, which are given access to the same validation data and evaluation metric as TALE. As shown in Table 1, these methods still do not perform as well as TALE across all tasks.[2]

Layer selection beyond fixed truncation. Our experiments indicate that for reasoning, intensive tasks (e.g., GSM8K, MATH500), removing early or intermediate layers can lead to performance gains, an effect that standard top-k layer truncation strategies fail to capture. Additionally, a key strength of TALE is that the number of pruned layers is not fixed *a priori*. Instead, it adaptively determines when to stop pruning. Empirically, we observe that a specific number of removed layers n may be optimal, while pruning further $(n + 1)$ can cause a sharp drop in performance. Moreover,

[2]We also note that TALE appears superior to early-exit methods we examined. From the limited information we have about RAEE outside of classification tasks, RAEE gives a score on ARC easy of 65%, while TALE substantially improves on the Llama baseline score of 76% and TALE on ARC Easy gives 79 % using LM eval.

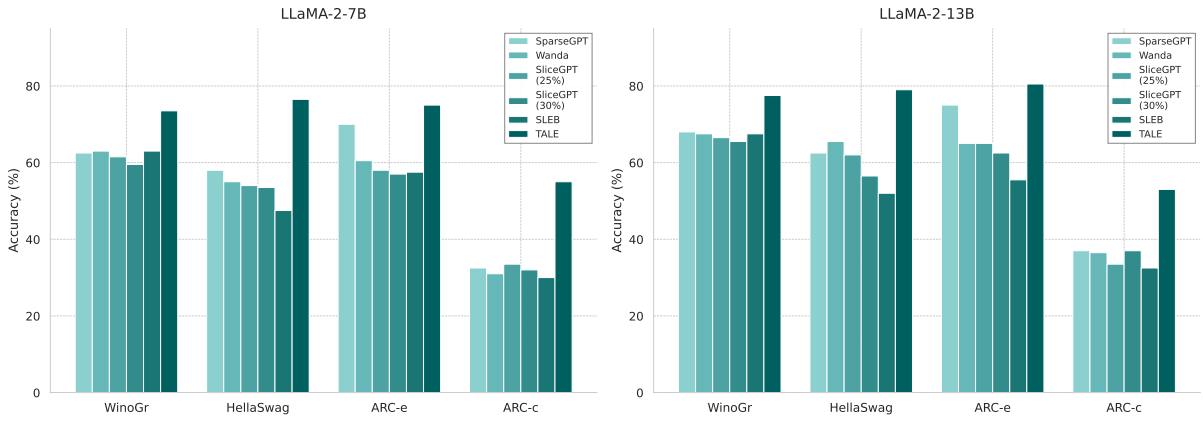


Figure 5: **Performance comparison of pruning methods on LLaMA-2 models.** Evaluation of seven pruning approaches using LM-Eval accuracy across four zero-shot benchmarks for LLaMA-2-7B (left) and LLaMA-2-13B (right). Methods are shown in a blue gradient from light (Baseline) to dark (TALE). TALE achieves the highest accuracy across all tasks while maintaining equivalent sparsity to SLEB, which performed second best to TALE. Method outperformed all structured pruning methods (SpareGPT, Wanda, SliceGPT) on both model sizes, demonstrating effective layer dropping without accuracy loss.

this optimal point varies across datasets, making any fixed, pre-defined pruning budget potentially suboptimal or even harmful. This highlights a key limitation of approaches that fix the number of layers to prune from the outset.

Eval	Method	ARC-e	ARC-c	WinoGr
dec	TALE	76.7	54.3	73.1
	SLEB-ta	61.0	38.0	66.5
	BlockPr-ta	64.6	39.6	65.59
LM	BlockPr-ta	65	41	66
	TALE	81	55	78

Table 1: Comparison of TALE to SLEB and BlockPruner using both decoder and LM eval evaluations

5.4 Computational Cost and Amortized Efficiency

TALE requires a modest, one-time computation to identify the optimal layer-set for a given task, which is then amortized over the models entire inference lifetime. For an L-layer model and a validation set of size V, the time of pruning process is proportional to $O(I \cdot L \cdot V \cdot T_{\text{layer}})$, where I is the number of pruning iterations. For LLaMA 3.1 8B across our benchmarks ($L = 32, V \approx 500-1500$), the pruning required approximately 1 to 2 GPU-hours on a single A100.

In Appendix F, we also report first-token latency and aggregate throughput for a subset of model/task pairs, measured on one A100-80GB NVIDIA GPU (with identical decoding settings for base and pruned models). In these experiments, the reported pruned model is the BEST variant, not the BSBA variant. TALE improves first-token latency in 9/9 settings (macro average: -14.3%) and throughput in 9/9 settings (macro average: +17.9%).

Takeaways. TALE consistently uncovers high accuracy and high accuracy/high efficiency models. By balancing task fidelity with computational savings, it enables both accuracy-focused and efficiency-focused deployment. Even strong, larger models like Qwen 7B see significant improvements, but so do small models (Qwen 0.5B).

5.5 Interactions between TALE, few-shot learning and fine-tuning

Few-shot setting. As few-shot prompting improves baselines on many tasks,[3] we tested on Lucie and LLaMA models whether TALE could synergize with few-shot prompting to bring higher gains (Tables 8 and 9). TALE-pruned variants still achieve higher accuracy in nearly all settings. This shows that TALE-induced improvements are complementary to gains from in-context learning.

Fine-tuning. We next investigate how layer pruning interacts with fine-tuning. Since pruning reduces model representational capacity, one might expect it to negatively impact fine-tuning performance compared to baseline instruct-tuned models. However, our results show the opposite: TALE **not only preserves fine-tuning results but can even improve accuracy and efficiency**.

[3] in particular reasoning tasks like GSM8K and Math500

Model	Dataset	Baseline		TALE		FT Only		TALE → FT		FT → TALE		(TALE → FT) → TALE	
		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
	GSM8K	15.07	0	37.08	3	42.70	0	53.96	1	50.86	2	54.02	2
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.

We explored four settings: (i) fine-tuning the base model (FT), (ii) applying TALE after fine-tuning (FT → TALE), (iii) pruning first and then fine-tuning (TALE → FT), and (iv) pruning first, then fine-tuning, and finally pruning again (TALE → FT → TALE). Across various benchmarks, we consistently observed mostly moderate and sometimes significant gains after iterating pruning and fine-tuning, especially on Winogrande and GSM8K (Table 2). This suggests that pruning can act as a regularizer, simplifying the optimization landscape by removing redundant layers.

TALE also reduced computation costs for fine-tuning. For example, pruning LLaMA-3.1 8B prior to fine-tuning reduces training time by approximately 18.5% (2–2.5 GPU hours on an A100) while 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%).

Summary. Overall, these results reveal a consistent pattern: TALE **synergizes with few-shot learning and fine-tuning**. Rather than limiting model capacity in a harmful way, task-aware pruning removes detrimental components, leading to models that are both more efficient and, in many cases, more accurate.

6 Discussion

We summarize six key observations from our experiments below.

a. Task dependency of layer importance The literature offers differing views on layer importance. Some argue that early layers are essential (Dalvi et al., 2020), while others emphasize the importance of later layers (Tenney et al., 2019; Bansal et al., 2023; Song et al., 2025). Our findings show that layer importance is fundamentally task-specific; for example, removing early layers reduces accuracy to near zero on commonsense reasoning tasks (Figure 11), whereas removing LLaMAs layer 3 improves performance on GSM8K-Hard.

b. Related tasks often exhibit similar layer dependencies Common sense reasoning tasks (see Figure 11) show importance concentrated in comparable regions of the network. All models show sizable accuracy boosts in mathematical reasoning tasks after pruning between one and three early-to-middle layers (e.g., LLaMA layer 3, Mistral layers 6 and 22, Lucie layer 12) (Figures 13, 14, 15). By contrast, knowledge-intensive tasks (ARC, BoolQ, CommonsenseQA, Winogrande, and BIG-Bench) exhibit more modest improvements (although LLaMA shows an 11% gain on BIG-Bench) and benefit more from removing later layers. These results may aid model interpretability, as plotting performance degradation from layer ablations helps localize task-specific capabilities within the network.

Initial multilingual testing of TALE on Lucie, which is tuned for French conversational proficiency (Gouvert et al., 2025), using bilingual versions of the same dataset, shows that optimal pruning is task-specific rather than language-specific.

c. Layer redundancy persists even in single-task training. To isolate this effect, we trained a transformer from scratch on a single task (in-context learning of linear functions), hypothesizing that layer redundancy might arise primarily from multi-task pretraining. However, even in this controlled setting, several layers proved redundant, and some even degraded performance (Figure 7).

d. Generality In principle, TALE can combine tasks to produce more general optimized models. A LLaMA math model without layer 12 improves over the baseline LLaMA on Math500 and GSM8K tasks. A promising direction is to prune models jointly across multiple tasks using different mixtures of data to guide the pruning process.

e. Model-specific effects with TALE TALE affects different models differently. While LLaMA benefits the least from TALE , Lucie achieves large gains on MMLU and double-digit improvements on ARC-Challenge, CommonsenseQA, BoolQ, and GSM8K-hard. TALE confers more modest but still substantial gains on Qwen-7B and Mistral. Lucie also benefits from more aggressive pruning than the other models.

The fact that Lucie was trained on a much smaller dataset (3T tokens vs. 15T for LLaMA and 13T for Qwen) suggests intriguing interactions between pretraining and TALE improvements. We hypothesize that models trained close to their performance ceiling (via large-scale pretraining, instruction tuning, or RLHF) yield smaller gains from TALE , whereas models trained under more limited objectives may benefit more.

f. MI Analysis We use Mutual Information (MI) to investigate why selectively removing layers can improve accuracy, focusing on how information about the output evolves as it propagates through the network. Unlike correlation, MI captures nonlinear statistical dependencies and thus provides a more complete measure of dependence (Kinney and Atwal, 2014). We estimate MI using MINE (Belghazi et al., 2018), a widely used approximation method.

Our analysis reveals that many layers exhibit a pronounced drop in MI (Figure 8). TALE removes some of these layers, but not all; overall, it reduces the peaks and valleys in the MI profile across layers. However, removing all layers associated with decreases in MI leads to very poor performance. This suggests that some local decreases in MI across adjacent layers are necessary for proper model functioning (see Appendix J for details).

7 Conclusions

We introduced TALE, a simple and effective task-aware layer elimination strategy that removes layers irrelevant to a target task T. Across a wide range of models and benchmarks, TALE improves the performance of the base model while reducing computational cost, outperforming prior training-free pruning approaches.

Unlike existing methods that rely on task-agnostic heuristics, TALE directly optimizes task-specific validation accuracy, enabling it to identify compact, task-specialized architectures without retraining. We further show that TALE complements both few-shot prompting and fine-tuning, often yielding additional gains in both accuracy and efficiency.

Beyond performance improvements, TALE provides a practical and flexible approach to adapting large language models post hoc. It can be applied to pre-trained, instruction-tuned, and fine-tuned models, making it suitable for a wide range of deployment scenarios. In particular, TALE is well-suited for high-throughput and resource-constrained settings, such as multi-agent systems and interactive applications, where balancing model capability and computational efficiency is critical.

Overall, our results highlight the importance of task-aware model adaptation and suggest that selectively removing misaligned components can be as beneficial as adding capacity, opening new directions for efficient and specialized use of large language models.

Limitations

While TALE demonstrates consistent gains across a broad range of models and tasks, we note several limitations.

First, TALE operates at the level of *entire transformer layers*. This design choice prioritizes simplicity, transparency, and compatibility with existing checkpoints, but it does not exploit finer-grained structure such as attention heads, blocks, or token-level adaptivity. More granular structured pruning or adaptive computation methods may provide complementary benefits, particularly when retraining or architectural modification is

permissible.

Second, TALE performs *task-specific* layer selection using a held-out optimization split. As a result, the resulting pruned architectures are specialized to individual tasks rather than universally optimal across tasks. While this specialization aligns with deployment scenarios where the target task is known in advance, it may be less suitable in settings that require a single model to perform well across many heterogeneous tasks without reconfiguration.

Third, TALE relies on a greedy elimination procedure and a stopping tolerance hyperparameter to determine when further layer removal becomes detrimental. Although we observe stable behavior across random seeds and alternative data splits, more principled global optimization strategies or adaptive stopping criteria could further improve robustness and are left for future work.

Finally, our empirical comparisons focus on training-free, layer-level pruning methods that preserve the original model architecture. Approaches that rely on retraining, block-level restructuring, or task-adaptive control flow address a different point in the design space and are therefore not directly comparable within our experimental scope.

Overall, these limitations reflect deliberate design choices rather than deficiencies of the approach. We view TALE as complementary to retraining-based and fine-grained pruning methods, and believe that combining task-aware layer selection with such approaches is a promising direction for future research.

Acknowledgments

We gratefuly acknowledge the support of the grants SARER, Summ-RE (ANR-20-CE23-0017), and the AI Cluster ANITI (ANR-19-PI3A-0004). This work used the HPC resources from CALMIP (Grant 2016-P23060 and M24047).

References

Guillaume Alain and Yoshua Bengio. 2016. Understanding intermediate layers using linear classifier probes. In *International Conference on Learning Representations (ICLR) Workshop*.

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

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

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*.

Hritik Bansal, Karthik Gopalakrishnan, Saket Dingliwal, Sravan Bodapati, Katrin Kirchhoff, and Dan Roth. 2023. Rethinking the role of scale for in-context learning: An interpretability-based case study at 66 billion scale. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11833–11856.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. 2018. Mutual information neural estimation. In *International conference on machine learning*, pages 531–540. PMLR.

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

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. 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)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

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. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. ArXiv:1803.05457.

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

Fahim Dalvi, Hassan Sajjad, Nadir Durrani, and Yonatan Belinkov. 2020. Analyzing redundancy in pretrained transformer models. *arXiv preprint arXiv:2004.04010*.

Wietse De Vries, Andreas Van Cranenburgh, and Malvina Nissim. 2020. What's so special about bert's layers? a closer look at the nlp pipeline in monolingual and multilingual models. *arXiv preprint arXiv:2004.06499*.

Robert M. Fano. 1961. *Transmission of Information: A Statistical Theory of Communications*. M.I.T. Press.

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

Marylou Gabrié, Andre Manoel, Clément Luneau, Jean Barbier, Nicolas Macris, Florent Krzakala, and Lenka Zdeborová. 2019. Entropy and mutual information in models of deep neural networks. *Advances in Neural Information Processing Systems*, 32.

Shuyang Gao, Greg Ver Steeg, and Aram Galstyan. 2015. Efficient estimation of mutual information for strongly dependent variables. *arXiv preprint arXiv:1411.2003*.

Olivier Gouvert, Julie Hunter, Jérôme Louradour, Christophe Cerisara, Evan Dufraisse, Yaya Sy, Laura Rivière, Jean-Pierre Lorré, and 1 others. 2025. The lucie-7b llm and the lucie training dataset: Open resources for multilingual language generation. *arXiv preprint arXiv:2503.12294*.

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

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

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

Lianming Huang, Shangyu Wu, Yufei Cui, Ying Xiong, Haibo Hu, Xue Liu, Tei-Wei Kuo, Nan Guan, and Chun Jason Xue. 2024. Raee: A robust retrieval-augmented early exit framework for efficient inference. *Preprint*, arXiv:2405.15198.

Justin B Kinney and Gurinder S Atwal. 2014. Equitability, mutual information, and the maximal information coefficient. *Proceedings of the National Academy of Sciences*, 111(9):3354–3359.

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

Omar Naim, Jérôme Bolte, and Nicholas Asher. 2025a. Analyzing limits for in-context learning. *arXiv preprint arXiv:2502.03503*.

Omar Naim, Guilhem Fouilhé, and Nicholas Asher. 2025b. Re-examining learning linear functions in context. In *German Conference on Artificial Intelligence (Künstliche Intelligenz)*, pages 104–117. Springer.

David Peer, Sebastian Stabinger, Stefan Engl, and Antonio Rodríguez-Sánchez. 2022. Greedy-layer pruning: Speeding up transformer models for natural language processing. *Pattern Recognition Letters*, 157:76–82.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2023. On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 77:101429.

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.

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

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*.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam 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, and 432 others. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*. Preprint / TMLR.

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

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. 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, pages 4149–4158. Association for Computational Linguistics.

Shengkun Tang, Oliver Sieberling, Eldar Kurtic, Zhiqiang Shen, and Dan Alistarh. 2025. Darwinlm: Evolutionary structured pruning of large language models. *arXiv preprint arXiv:2502.07780*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*.

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

Jinyong Wen. 2024. Gaussian mutual information maximization for efficient graph self-supervised learning: Bridging contrastive-based to decorrelation-based. MM '24, New York, NY, USA. Association for Computing Machinery.

Maciej Woczyk, Bartosz Wójcik, Klaudia Baazy, Igor Podolak, Jacek Tabor, Marek mieja, and Tomasz Trzciski. 2021. Zero time waste: Recycling predictions in early exit neural networks. *Preprint*, arXiv:2106.05409.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Jaiswal, Mykola Pechenizkiy, Yi Liang, Michael Bendersky, Zhangyang Wang, and Shiwei Liu. 2024. Outlier weighed layerwise sparsity (owl) a missing secret sauce for pruning llms to high sparsity. In *Proceedings of the 41st International Conference on Machine Learning*, pages 57101–57115.

Yuxin Zhang, Lirui Zhao, Mingbao Lin, Sun Yunyun, Yiwu Yao, Xingjia Han, Jared Tanner, Shiwei Liu, and Rongrong Ji. 2024. Dynamic sparse no training: Training-free fine-tuning for sparse llms. In *The Twelfth International Conference on Learning Representations*.

Longguang Zhong, Fanqi Wan, Ruijun Chen, Xiaojun Quan, and Liangzhi Li. 2025. Blockpruner: Fine-grained pruning for large language models. *Preprint*, arXiv:2406.10594.

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 -8% 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. The LM-Eval methodology presents a significant limitation: it selects the answer with the highest probability among the provided options rather than assessing what the model would actually generate. This approach ignores hallucination behavior and systematically inflates scores; for example, in a two-choice setting, a hallucinated answer still has a 50% chance of being counted as correct. Furthermore, LM-Eval often assigns relatively high scores to weak models, compressing performance differences and making stronger approaches appear only marginally better despite substantial real-world gains. This produces a misleading picture of model capability, as high LM-Eval results do not guarantee that a model will produce correct, coherent outputs in practice. For these reasons, we relied primarily on Decoder Eval that measures actual accuracy based on the models generated outputs, which we implemented for each task.

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 elseno 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)."

MATH500 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.

A.1 Layer Removal Implementation

Layer pruning was implemented through a custom model wrapper that reconstructs the architecture excluding specified layers. Given delete indices $\mathcal{D}$, we create a `ModifiedModel` that:

1. Preserves the embedding layer, final normalization, and language modeling head from the original model
2. Constructs a new layer sequence $\mathcal{L}' = \{l_i \mid i \notin \mathcal{D}\}$ by filtering out deleted layers while maintaining the order of retained layers
3. Updates the model configuration to reflect the new layer count

The forward pass implements standard transformer computation: input embeddings are passed through the retained layers sequentially with causal attention masking, then normalized and projected to vocabulary logits. Position embeddings are generated automatically if not provided. This architecture is fully compatible with the Hugging Face training pipeline and can be directly used with LoRA fine-tuning without requiring custom training loops.

A.2 Fine-tuning Training Details

All fine-tuning experiments were conducted using Parameter-Efficient Fine-Tuning (PEFT) with Low-Rank Adaptation (LoRA). We employed 4-bit quantization using the BitsAndBytes library to reduce memory footprint during training. The quantization configuration used NF4 (4-bit NormalFloat) quantization type with float16 compute dtype, without nested quantization.

LoRA Configuration: We applied LoRA to all linear layers in the model with the following hyperparameters: rank $r = 64$, alpha $\alpha = 16$, and dropout rate of 0.1. These parameters were kept consistent across both full and pruned models to ensure fair comparison.

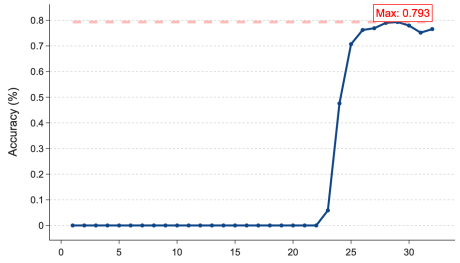
Optimization Settings: Training was performed for 10 epochs using the paged AdamW optimizer (32-bit) with a learning rate of 2×10^{-4} and weight decay of 0.001. We used a cosine learning rate schedule with a warmup ratio of 0.03. Gradient clipping was applied with a maximum gradient norm of 0.3. The effective batch size was 60 (per-device batch size of 2 with gradient accumulation steps of 30). Gradient

checkpointing was enabled to reduce memory consumption during training.

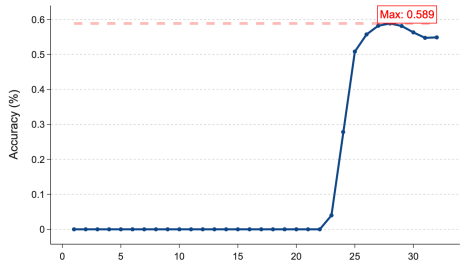
Data Processing: All sequences were truncated or padded to a maximum length of 300 tokens. We used right-side padding with a special padding token (ID: 128004). Packing was disabled to maintain sequence boundaries, and we removed unused columns from the dataset. The dataloader used 4 workers with the last incomplete batch dropped to ensure consistent batch sizes.

Hardware and Implementation: All experiments were conducted on NVIDIA A100 GPUs. We used mixed-precision training without fp16 or bf16 enabled at the trainer level, relying instead on the 4-bit quantization for memory efficiency. Training logs were reported to TensorBoard every 25 steps.

B Intuition behind TALE



(a) ARC-Challenge



(b) MMLU

Figure 6: Layer-wise output performance for LLaMA models: results when generating predictions from intermediate layers 1 through 32 on two different datasets.

C Number of parameters per layer for each model

Model	Params/Layer	Layers
LLaMA 3.1 8B	218,112,000	32
Qwen 2.5 7B	233,057,792	28
Mistral 7B	218,112,000	32
Lucie 7B	192,946,176	32
Qwen 2.5 0.5B	14,912,384	24

Table 3: Model parameter counts comparison showing parameters per layer and total number of layers.

D Ablation study on validation Set of Pruning

We analyze the effect of validation set size on TALE's layer selection. Table 4 reports the specific layers dropped for different validation set sizes across three tasks (ARC-Easy, MMLU, GSM8K) and two models (Llama 3.1 8B, Qwen 2.5 7B).

Model	Val Size	Task	Dropped Layers
Llama 3.1 8B	200	ARC-E	{19, 20, 22, 29, 32 }
		MMLU	{ 21 }
		GSM8K	{ 3 }
	500	ARC-E	{ 19, 20, 21, 29, 32 }
		MMLU	{ 21 }
		GSM8K	{ 3 }
	1000	ARC-E	{ 19, 20, 21, 29, 32 }
		MMLU	{ 21 }
		GSM8K	{ 3 }
Qwen 2.5 7B	100	ARC-E	{ 22 , 27 , 28 }
		MMLU	{ 18 , 22 , 24 , 27 , 28 }
		GSM8K	{ 19 }
	500	ARC-E	{ 19, 22 , 28 }
		MMLU	{ 22 , 23 , 26 , 27 , 28 }
		GSM8K	{ 19 }
	1000	ARC-E	{ 19 , 22 , 28 }
		MMLU	{ 22 , 23 , 26 , 27 , 28 }
		GSM8K	{ 19 }

Table 4: Layers removed by TALE for different validation-set sizes across three tasks. This reveals the stability of pruning decisions directly.

E Robustness Study on TALE

See Table 5.

Dataset	LLaMA 3.1 8B (0-shot)							Qwen 2.5 7B (0-shot)						
	Baseline	Best Model			BSBA			Baseline	Best Model			BSBA		
	Perf.	Perf. ± Std	#D	Sp.	Perf.	#D	Sp.	Perf.	Perf. ± Std	#D	Sp.	Perf.	#D	Sp.
ARC-Easy	89	90.4 ± 0.14	5	-14.6%	88.8	8	-23.5%	90.04	93.2 ± 0.16	3	-10.0%	90.08	7	-30.3%
ARC-Challenge	79.4	80.6 ± 0.18	4	-11.7%	77.6	7	-20.5%	86.55	92.00 ± 0.15	2	-6.7%	86.55	6	-19.9%
BoolQ	85.4	85.9 ± 0.12	3	-8.8%	85.4	7	-17.6%	81.90	83.90 ± 0.14	4	-13.3%	82.70	5	-23.2%
MMLU	48.8	53.8 ± 0.22	1	-2.9%	50.2	9	-26.4%	68.10	71.00 ± 0.19	5	-16.6%	68.13	6	-19.9%
CommonQA	72.9	73.8 ± 0.15	3	-8.8%	73.10	6	-17.6%	80.30	84.40 ± 0.17	2	-6.6%	80.50	6	-19.9%
Winogrande	53.8	54.1 ± 0.20	4	-11.7%	53.83	12	-32.2%	62.04	67.25 ± 0.21	3	-10.0%	62.19	6	-19.9%
BIG-Bench	77.2	85.6 ± 0.25	5	-14.4%	76.4	11	-32.2%	79.20	81.60 ± 0.14	6	-19.9%	81.60	6	-19.9%
GSM8K-HARD	39.0	59.0 ± 0.30	1	-2.9%	39.4	4	-11.7%	43.80	61.80 ± 0.28	2	-43.6%	43.99	5	-17.6%
Math500	25.4	28.2 ± 0.18	2	-6.0%	27.4	3	-9.1%	31.00	38.20 ± 0.22	2	-6.6%	32.10	4	-13.3%

Dataset	Lucie 7B (0-shot)							Mistral 7B (0-shot)						
	Baseline	Best Model			BSBA			Baseline	Best Model			BSBA		
	Perf.	Perf. ± Std	#D	Sp.	Perf.	#D	Sp.	Perf.	Perf. ± Std	#D	Sp.	Perf.	#D	Sp.
ARC-Easy	74.4	75.8 ± 0.15	6	-18.1%	73.8	8	-23.5%	83.8	85.6 ± 0.14	5	-15.4%	82.8	9	-27.7%
ARC-Challenge	46.0	51.45 ± 0.22	7	-22.1%	48.8	11	-33.1%	76.2	79.1 ± 0.18	6	-18.5%	76.2	8	-24.6%
BoolQ	53.0	74.0 ± 0.30	5	-17.2%	63.0	19	-54.2%	81.3	84.4 ± 0.16	4	-18.5%	80.8	5	-27.7%
MMLU	13.0	54.0 ± 0.35	8	-24.1%	15	22	-60.2%	39.4	40.8 ± 0.20	2	-6.2%	39.0	8	-24.6%
CommonQA	54.2	68.6 ± 0.27	3	-9.1%	54.6	17	-48.2%	61.0	64.4 ± 0.19	4	-12.3%	61.6	7	-21.5%
Winogrande	51.6	53.1 ± 0.21	5	-27.1%	53.0	15	-45.2%	53.2	54.3 ± 0.16	10	-30.7%	52.4	13	-40.0%
BIG-Bench	67.4	75.0 ± 0.25	9	-27.1%	71	15	-45.1%	70.4	75.4 ± 0.22	9	-28.0%	72.6	11	-33.8%
GSM8K-HARD	32	39.0 ± 0.28	1	-3.1%	37	3	-9.1%	24	33 ± 0.24	2	-6.2%	26.1	4	-12.3%
Math500	21.0	26.1 ± 0.20	2	-6.0%	25.1	3	-9.1%	19	28 ± 0.23	1	-3.1%	18.8	4	-12.3%

Table 5: Robustness study of the proposed layer-dropping method across multiple language models under zero-shot evaluation. For each dataset and model, results are reported over five random seeds to account for variability in decoding and sampling. We present the baseline model accuracy and the accuracy of the best pruned configuration, along with their corresponding standard deviations computed across the 5 seeds. The table also includes the number of transformer layers removed in the best-performing configuration (**#D**) and the resulting inference speedup (**Sp.**) expressed as the percentage of total TFlops saved during evaluation. Bold values indicate the highest mean accuracy for each dataset.

F Practical computing savings and scaling

We quantify TALE's inference-cost reduction by measuring TFLOPs (tera-FLOPs) drop per removed layer. Across models and tasks, removing a single transformer layer yields a mean TFLOPs reduction of $3.00\% \pm 0.20\%$. Because TALE removes entire layers sequentially, the total TFLOPs reduction scales essentially linearly with the number of iterations (layers removed). In practice this means only a few iterations are required to reach common sparsity targets: e.g., three iterations remove roughly $\approx 9\%$ TFLOPs, sufficient to realize $\approx 10\%$ sparsity in our settings.

Wall-clock runtime and end-to-end cost. In addition to FLOPs-based estimates, we report wall-clock runtime measurements under the exact decoding settings used for evaluation. TALE incurs a one-time optimization cost per task, after which the pruned model is used for standard inference with reduced depth.

For all non-reasoning benchmarks (e.g., ARC, BoolQ, MMLU, CommonsenseQA), we use greedy decoding with `max_new_tokens = 1`, reflecting single-step answer generation. Under this setting, the total wall-clock time required to complete a full TALE optimization run is approximately 1 hour per dataset on a single NVIDIA A100 GPU.

For reasoning-heavy benchmarks (GSM8K-Hard and Math500), we use `max_new_tokens = 200` to allow full chain-of-thought generation. Due to longer decoding sequences, the average wall-clock time for TALE on these datasets is approximately 3 hours, measured over the evaluated sub-

Model	Dataset	Latency (s) ↓	Throughput (tok/s) ↑	Speedup
Llama-3.1-8B-Instruct	ARC-Easy	0.0402 → 0.0343 (-14.8%)	24.83 → 29.13 (+17.3%)	1.17×
	ARC-Challenge	0.0403 → 0.0355 (-11.9%)	24.78 → 28.13 (+13.5%)	1.14×
	BoolQ	0.0409 → 0.0373 (-8.8%)	24.41 → 26.76 (+9.6%)	1.10×
	MMLU	0.0408 → 0.0396 (-2.9%)	24.50 → 25.24 (+3.0%)	1.03×
	CommonQA	0.0403 → 0.0367 (-8.9%)	24.78 → 27.21 (+9.8%)	1.10×
	Winogrande	0.0404 → 0.0356 (-11.9%)	24.71 → 28.04 (+13.5%)	1.13×
	BIG-Bench	0.0404 → 0.0344 (-15.0%)	24.72 → 29.08 (+17.6%)	1.18×
	GSM8K-Hard	0.0406 → 0.0394 (-3.0%)	23.87 → 24.61 (+3.1%)	1.03×
	Math500	0.0407 → 0.0383 (-6.0%)	23.85 → 25.35 (+6.3%)	1.06×
Qwen-2.5-7B-Instruct	ARC-Easy	0.0356 → 0.0320 (-10.2%)	28.03 → 31.21 (+11.3%)	1.11×
	ARC-Challenge	0.0357 → 0.0333 (-6.8%)	27.99 → 30.02 (+7.3%)	1.07×
	BoolQ	0.0366 → 0.0316 (-13.5%)	27.31 → 31.57 (+15.6%)	1.16×
	MMLU	0.0359 → 0.0298 (-17.0%)	27.80 → 33.46 (+20.4%)	1.20×
	CommonQA	0.0354 → 0.0330 (-6.7%)	28.24 → 30.25 (+7.1%)	1.07×
	Winogrande	0.0354 → 0.0318 (-10.2%)	28.21 → 31.39 (+11.3%)	1.11×
	BIG-Bench	0.0357 → 0.0284 (-20.4%)	27.94 → 35.40 (+26.7%)	1.27×
	GSM8K-Hard	0.0361 → 0.0337 (-6.6%)	25.88 → 27.73 (+7.1%)	1.07×
	Math500	0.0361 → 0.0337 (-6.7%)	26.32 → 28.20 (+7.1%)	1.07×
Lucie-7B-Instruct-v1.1	ARC-Easy	0.0399 → 0.0327 (-18.1%)	25.03 → 30.54 (+22.0%)	1.22×
	ARC-Challenge	0.0398 → 0.0314 (-21.0%)	25.11 → 31.79 (+26.6%)	1.27×
	BoolQ	0.0402 → 0.0341 (-15.1%)	24.87 → 29.27 (+17.7%)	1.18×
	MMLU	0.0401 → 0.0304 (-24.1%)	24.89 → 32.79 (+31.7%)	1.32×
	CommonQA	0.0400 → 0.0364 (-9.1%)	24.96 → 27.46 (+10.0%)	1.10×
	Winogrande	0.0400 → 0.0339 (-15.1%)	25.00 → 29.43 (+17.7%)	1.18×
	BIG-Bench	0.0402 → 0.0292 (-27.3%)	24.86 → 34.17 (+37.4%)	1.37×
	GSM8K-Hard	0.0399 → 0.0388 (-2.9%)	24.60 → 25.33 (+3.0%)	1.03×
	Math500	0.0402 → 0.0378 (-6.0%)	24.46 → 26.00 (+6.3%)	1.06×
Mistral-7B-Instruct-v0.3	ARC-Easy	0.0403 → 0.0342 (-15.1%)	24.78 → 29.19 (+17.8%)	1.18×
	ARC-Challenge	0.0403 → 0.0330 (-18.2%)	24.77 → 30.29 (+22.2%)	1.22×
	BoolQ	0.0407 → 0.0334 (-18.1%)	24.51 → 29.91 (+22.0%)	1.22×
	MMLU	0.0407 → 0.0382 (-6.0%)	24.56 → 26.12 (+6.3%)	1.06×
	CommonQA	0.0405 → 0.0356 (-12.1%)	24.68 → 28.07 (+13.8%)	1.14×
	Winogrande	0.0404 → 0.0282 (-30.2%)	24.71 → 35.37 (+43.1%)	1.43×
	BIG-Bench	0.0400 → 0.0289 (-27.6%)	24.99 → 34.69 (+38.8%)	1.39×
	GSM8K-Hard	0.0401 → 0.0377 (-6.0%)	24.60 → 26.14 (+6.2%)	1.06×
	Math500	0.0405 → 0.0392 (-3.2%)	24.36 → 25.17 (+3.3%)	1.03×

Table 6: Latency and throughput comparison between baseline and TALE-pruned (BEST) models across tasks. TALE consistently reduces first-token latency and improves throughput.

set.

Importantly, this optimization cost is incurred only once per task. During deployment, inference is performed using the pruned model, yielding reduced per-example latency and throughput improvements proportional to the number of eliminated layers. These gains are amortized over all subsequent inference calls.

G Layer Redundancy Is Not a Multi-Task Artifact

G.1 Training details

We train a transformer model to perform in-context learning over the class of linear functions. Given a prompt consisting of a small set of input–output examples $(x_1, g(x_1), \ldots, x_p, g(x_p), x)$, the model is tasked with predicting the value $g(x)$ for the final input x. We follow the training and evaluation setup of (Naim et al., 2025a,b)

G.2 Plots

Figure 7

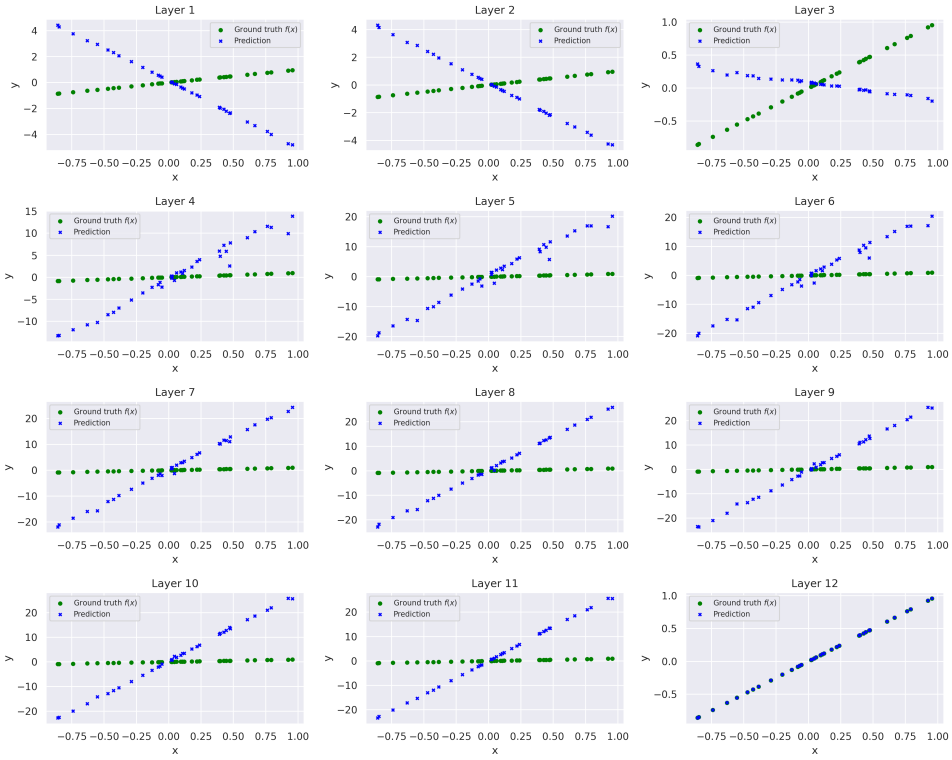


Figure 7: Plots showing evolution of the predictions over layers for $f(x) = x$ for a model trained on degree 1.

H LM-Eval results

Dataset	LLaMA 3.1 8B 0-shot						
	Baseline	Best Model			BSBA		
	Perf.	Perf.	#D	Sp.	Perf.	#D	Sp.
BoolQ	82.4	87.63	3	-8.8%	85.62	7	-20.5%
Hellaswag	52.5	55.5	3	-8.8%	54.5	5	-14.6%
COMMONQA	77.2	81.61	6	-17.6%	80.27	7	-20.5%
WINOGRANDE	75.92	78.93	4	-11.7%	76.59	5	-14.6%
GSM8k	43	58.5	2	-6.0%	58.5	2	-6.0%

Table 7: Results of **LLaMA 3.1 8B** across benchmarks. All tested on 0-shot and evaluated with LM Eval.

I Few-shot Learning Results

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 8: Results of **Lucie 7B** across nine benchmarks. All tested on 5-shots, except gms8k on 8-shots.

Dataset	LLaMA 3.1 8B few-shots						
	Baseline	Best Model			BSBA		
	Perf.	Perf.	#D	Sp.	Perf.	#D	Sp.
ARC-Easy	90.36	**92.18**	4	1.14	90.91	8	1.37
ARC-Challenge	78.2	**83.10**	3	1.17	78.62	9	1.42
BoolQ	82.7	85.3	4	1.11	83.0	6	1.22
MMLU	59.2	62.38	4	1.14	59.57	7	1.26
COMMONQA	73.30	75.30	6	1.22	73.80	7	1.32
WINOGRANDE	57.01	60.1	3	1.1	57.02	8	1.3
BIG-Bench	70.0	83.60	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	1	1.02	45.00	2	1.03

Table 9: Results of **LLaMA 3.1 8B** across nine benchmarks. All tested on 5-shots, except gms8k and MATH500 on 8-shots

J Information theory: Why pruned models might perform better.

Our results pose a puzzle: the increase in accuracy with TALE is counterintuitive: why would removing parts of a carefully trained model lead to better performance? One way to explore this question is mutual information.

(Alemi et al., 2016) use information theory to analyze how neural networks learn and represent data. Fano (1961) defines $\mathrm{I(X;Y)}$, the mutual information between two random variables X and Y, with the equation:

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

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. To attempt to explain why accuracy increases through task pruning we also use MI.

A major challenge of this approach is that it requires information about true distributions, which are infeasible to compute. As a result, researchers typically assume a Gaussian distribution (Gabrié et al., 2019; Gao et al., 2015; Wen, 2024) or approximate the probe using a classifier (Belinkov, 2022; Alain and Bengio, 2016) or an MLP (Belghazi et al., 2018). These approximations can yield useful insights. In our case, the Gaussian assumption did not fit our datasets. Since we evaluate on QA tasks, we used a trainable classifier to approximate the probes and estimate $I(X^\ell, \mathrm{Y})$ at each layer, where X^ℓ denotes the contextualized representations at layer ℓ and Y denotes the target answer. This approximates how much information the layer ℓ representations contain about the answer. The goal is then to examine whether some layers exhibit a sharp drop in information and whether those layers coincide with the ones whose removal leads to improved performance.

Our findings, summarized in Figure 9 and Table 13, reveal two key patterns: (i) several layers in large pre-trained transformers exhibit a pronounced drop in mutual information; (ii) removing layers dictated by TALE consistently increases the mutual information at the subsequent layer across tasks. Together, these results suggest that certain layers act more as bottlenecks than as contributors to task-relevant representations, providing a rationale for why pruning can lead to improved accuracy.

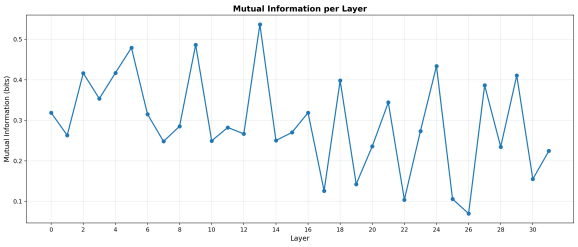
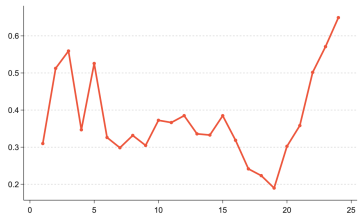
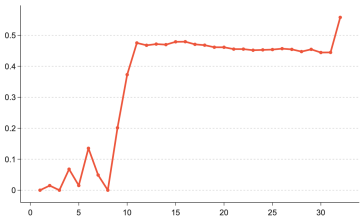


Figure 8: Evolution of Mutual Information about the output through layers for Llama3-8B.



(a) ARC-Easy (Qwen 0.5B)



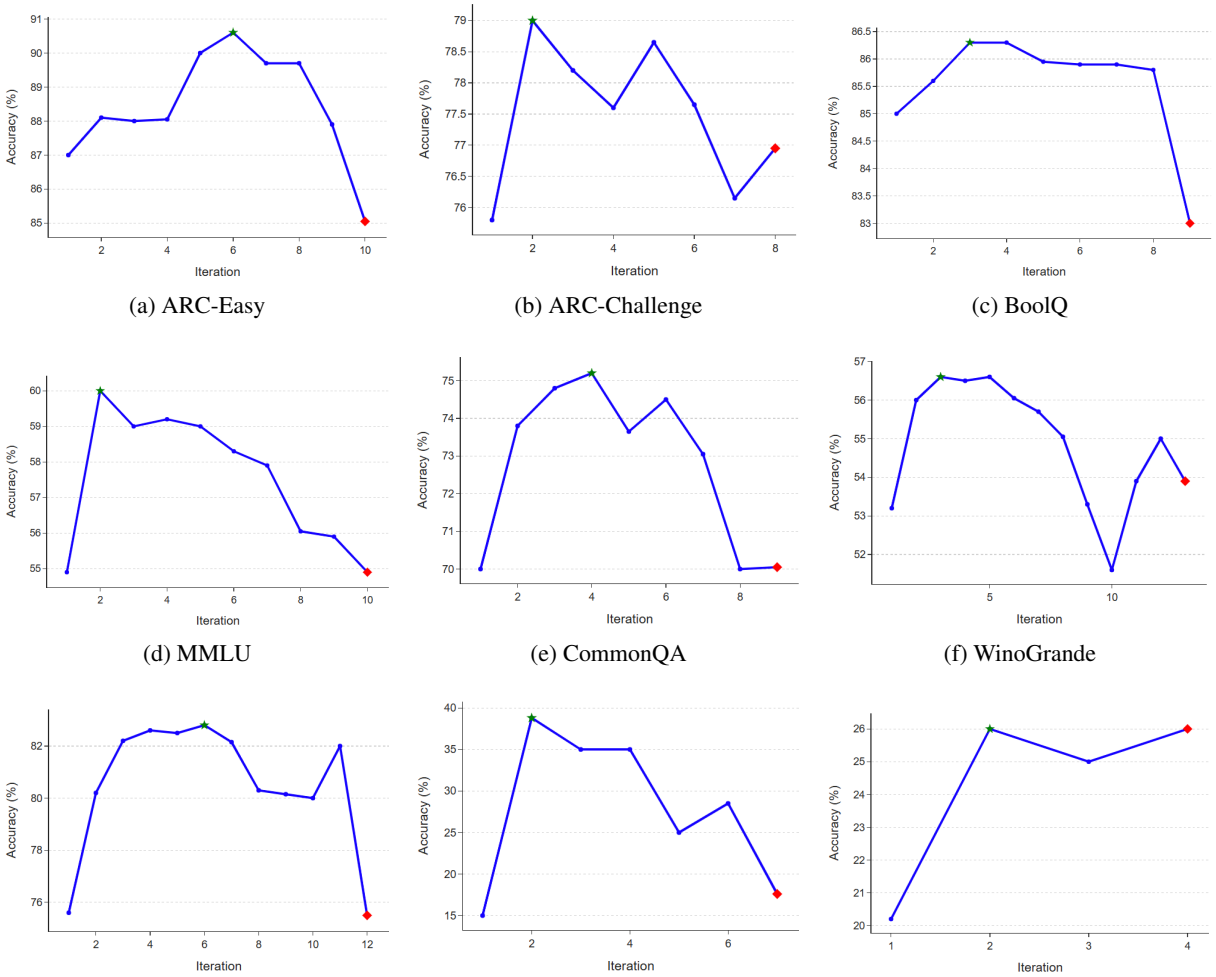
(b) BoolQ (Lucie 7B)

(c) BigBench (Llama 8B)

Figure 9: 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.

K TALE Dynamics

See Figure 10.



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

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

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

Figure 10: 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.

L General Pruning results

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 10: 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.

M TALE evaluation with perplexity

Model	**WikiText2**		**LAMBADA**	
	Vanilla	**TALE**	**Vanilla**	**TALE**
LLaMA 3.1 8B	24.6	24.9	28.1	28.9
Lucie 7B	46.1	36.4	52.5	43.8

Table 11: Perplexity scores for two models across WikiText2 and LAMBADA with Vanilla and TALE (sparisty 10%) configurations.

N More on pruning and a common pruned layers model

See Figure 11

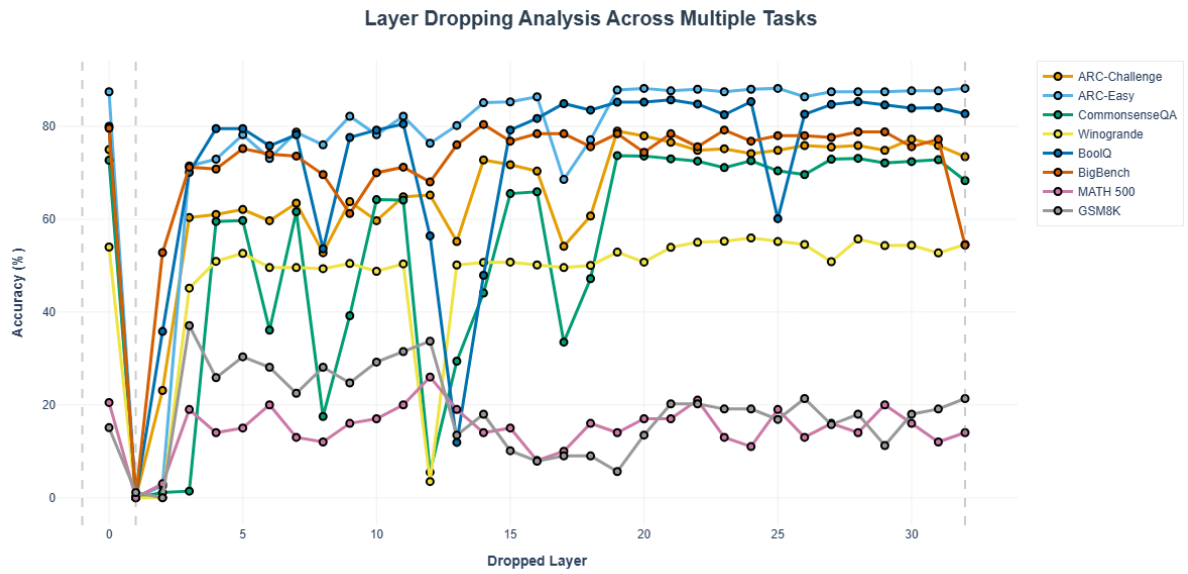


Figure 11: Nine benchmark tasks indicating performance after one layer is dropped from different positions in Llama3-8B.

O Comparison with Training-Free Pruning Technics

Model	Method	Sparsity	WinoGr	ARC-e	ARC-c
LLaMA-2-7B	Baseline	0%	41.2	51.7	40
	SLEB	10%	18 (-56.3% ↓)	29 (-43.9% ↓)	28.8 (-28.0% ↓)
	TALE	10%	**56** (+35.9% ↑)	**62.3** (+20.5% ↑)	**50** (+25.0% ↑)
	TALE	25%	**51** (+23.8% ↑)	**64.8** (+25.3% ↑)	**47.6** (+19.0% ↑)
LLaMA-2-13B	Baseline	0%	42	73.0	54.9
	SLEB	10%	24.2 (-42.3% ↓)	43.5 (-40.4% ↓)	29.8 (-47.3% ↓)
	Blockpruner	10%	24.2 (-42.3% ↓)	43.5 (-40.4% ↓)	29.8 (-47.3% ↓)
	TALE	10%	**56.4** (+34.3% ↑)	**77.3** (+5.9% ↑)	**64.4** (+17.1% ↑)
	TALE	25%	**55.2** (+31.4% ↑)	**75.3** (+3.2% ↑)	**64.1** (+16.4% ↑)

Table 12: Accuracies (%) with Decoder Eval on 0-shot tasks for LLaMA-2-7B and LLaMA-2-13B

P Deleted Layers in each Model and Benchmark

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
MATH500	11 22	11 22 26

Table 13: 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
MATH500	17	5 6 17 18

Table 14: Deleted layers represented as color-coded inline numbers. Blue = Best Model, Orange = BSBA for Qwen 2.5 7B 0-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
MATH500	12	12 21 23

Table 15: Deleted layers represented as color-coded inline numbers. Blue = Best Model, Orange = BSBA for Lucie 7B 0-shot.

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 16: Deleted layers represented as color-ccdinline numbers. Blue = Best Model, Orange = BSBA for Mistral 0-shot.

Q Scores with relative increases in accuracy

Dataset	LLaMA 3.1 8B (0-shot)							Qwen 2.5 7B (0-shot)						
	Baseline	Best Model			BSBA			Baseline	Best Model			BSBA		
	Perf.	Perf.	#D	Sp.	Perf.	#D	Sp.	Perf.	Perf.	#D	Sp. saved	Perf.	#D	Sp.
ARC-Easy	87.00	**90.55**(+4.08% ↑)	5	-14.6%	87.82	8	-23.5%	91.01	**91.82**(+0.89% ↑)	2	-10.0%	90.91	5	-30.3%
ARC-Challenge	75.86	**78.62**(+3.63% ↑)	4	-11.7%	76.90	7	-20.5%	86.55	**92.00**(+6.45% ↑)	2	-6.7%	86.55	6	-19.9%
BoolQ	85.00	**86.20**(+1.40% ↑)	3	-8.8%	85.70	7	-17.6%	84.10	**86.90**(+3.22% ↑)	4	-13.3%	82.70	5	-23.2%
MMLU	54.87	**59.90**(+9.17% ↑)	1	-2.9%	54.87	9	-26.4%	68.10	**71.00**(+4.26% ↑)	5	-16.6%	68.13	6	-19.9%
CommonQA	72.20	**75.30**(+4.29% ↑)	3	-8.8%	73.10	6	-17.6%	80.30	**84.40**(+5.11% ↑)	2	-6.6%	80.50	6	-19.9%
Winogrande	53.83	**56.67**(+5.28% ↑)	4	-11.7%	53.83	12	-32.2%	62.04	**67.25**(+8.40% ↑)	3	-10.0%	62.19	6	-19.9%
BIG-Bench	75.20	**83.60**(+11.17% ↑)	5	-14.4%	75.20	11	-32.2%	79.20	**81.60**(+3.03% ↑)	6	-19.9%	81.60	6	-19.9%
GSM8K-HARD	15.07	**37.08**(+146.05% ↑)	1	-2.9%	35.0	4	-11.7%	7.9	**27.0**(+243.58% ↑)	2	-6.6%	19.1	4	-13.3%
Math500	20.50	**26.00**(+26.83% ↑)	1	-2.9%	26.00	3	-8.8%	18.00	**27.00**(+50.0% ↑)	2	-6.6%	21.00	4	-13.3%

Dataset	Lucie 7B (0-shot)							Mistral 7B (0-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	-18.1%	72.55	13	-39.2%	81.02	**83.45**(+4.23% ↑)	5	-15.4%	81.09	9	-27.7%
ARC-Challenge	48.00	**53.79**(+12.06% ↑)	7	-21.1%	51.38	11	-33.1%	72.20	**74.83**(+3.64% ↑)	6	-18.5%	72.41	8	-24.6%
BoolQ	53.70	**77.50**(+44.32% ↑)	5	-17.2%	60.60	19	-54.2%	80.36	**83.20**(+3.53% ↑)	6	-18.5%	80.60	10	-27.7%
MMLU	21.36	**42.98**(+101.2% ↑)	8	-24.1%	39.39	15	-45.2%	52.73	**57.81**(+9.63% ↑)	2	-6.2%	52.91	8	-24.6%
CommonQA	55.50	**69.70**(+25.59% ↑)	3	-9.1%	57.10	17	-48.2%	57.32	**61.40**(+7.12% ↑)	4	-12.3%	57.40	7	-21.5%
Winogrande	54.20	**57.80**(+6.64% ↑)	5	-27.1%	54.30	15	-45.2%	52.55	**58.80**(+11.53% ↑)	10	-30.7%	53.43	13	-40.0%
BIG-Bench	69.60	**77.20**(+9.84% ↑)	9	-27.1%	72.00	15	-45.1%	70.00	**76.40**(+9.14% ↑)	9	-28.0%	72.80	11	-33.8%
GSM8K-HARD	14.20	**17.80**(+25.35% ↑)	1	-3.1%	17.40	3	-9.1%	11.24	**19.10**(+69.92% ↑)	2	-6.2%	15.73	4	-12.3%
Math500	19.00	**27.00**(+42.11% ↑)	2	-6.0%	26.00	3	-9.1%	8.00	**16.00**(+100% ↑)	1	-3.1%	10.00	4	-12.3%

Dataset	Qwen 2.5 0.5B (0-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**(+50.00% ↑)	1	1.1	9.00	2	1.1

Table 17: Performance comparison for LLaMA 3.1 8B, Qwen 2.5 7B, Lucie 7b, Mistral 7b and Qwen 2.5 0.5B under 0-shot evaluation on the 9 benchmarks with a different training test split from that used for Table 5). We report accuracy (%), number of layers dropped, and relative inference speed in time.