IteRABRe: Iterative Recovery-Aided Block Reduction

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have grown increasingly expensive to deploy, driving the need for effective model compression techniques. While block pruning offers a straightforward approach to reducing model size, existing methods often struggle to maintain performance or require substantial computational resources for recovery. We present IteRABRe, a simple yet effective iterative pruning method that achieves superior compression results while requiring minimal computational resources. Using only 2.5M tokens for recovery, our method outperforms baseline approaches by 3% on average when compressing the Llama3.1-8B and Qwen2.5-7B models. IteRABRe demonstrates particular strength in the preservation of linguistic capabilities, showing an improvement 5% over the baselines in language-related tasks. Our analysis reveals distinct pruning characteristics between these models, while also demonstrating preservation of multilingual capabilities.

1 Introduction

We are in the era of the burgeoning of producing Large Language Models (LLMs), which has led to the necessity of making them smaller due to deployment costs. Several approaches focus on model reduction, such as model sparsification, reducing LLM hidden sizes, or removing presumably unimportant blocks. However, preserving performance while compressing models remains challenging.

Block pruning is a straightforward compression approach for reducing LLM size, motivated by the layer redundancy found in LLM architectures (Men et al., 2024; Dumitru et al., 2024; Chen et al., 2025). While detecting redundant or unimportant blocks can minimize performance degradation from pruning, some loss is inevitable. Although post-finetuning can help recover performance, simultaneous pruning of multiple blocks may still cause unrecoverable damage.

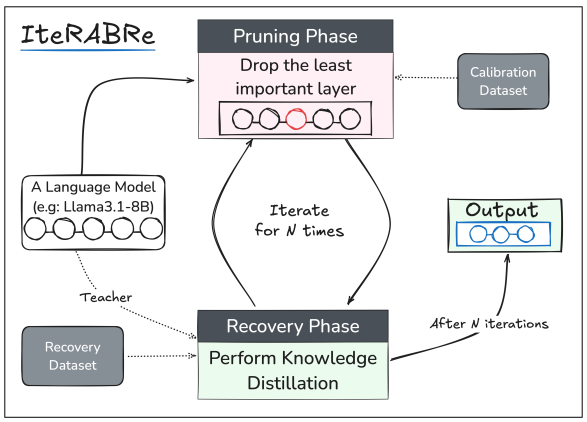

Figure 1: Overall methodology of IteRABRe. It consists of pruning and recovery phase that are done iteratively until the desired compression rate or minimum number of layer achieved.

One possible solution is to take an iterative approach. Muralidharan et al., 2024 proposes iterative pruning with knowledge distillation to recover from performance loss. This work successfully compresses a 15B LLM into smaller 8B and 4B versions, achieving competitive results compared to other LLMs of similar size. However, this approach may be impractical for those with limited computational resources, as it employs a compute-dependent method in the pruning process and requires 8T tokens for the recovery process.

This leads us to ask: ***Can we develop an exhustive, efficient and effective iterative block pruning method for model compression?*** We investigate this question by introducing IteRABRe, a straightforward iterative pruning approach. We choose layer pruning for its simplicity and enhanced interpretability in preservation. To test efficiency, we perform recovery using only 2.5M tokens of data. Our method outperforms other baselines by approximately 3% on average for Llama3.1-8B and Qwen2.5-7B models. Furthermore, our approach shows particular strength in preserving linguistic tasks, demonstrating 5% better performance than

baselines. Additionally, this approach also exhibits zero-shot cross-lingual capabilities, such as retaining the German language using solely English data recovery. Our key contributions are:

1. We introduce IteRABRe, a simple and efficient iterative pruning approach with recovery.
2. We provide detailed analysis of performance degradation, including recovery phase impact and patterns in layer index drop sequences, probing different behavior exhibited in each model family.
3. We present granular analysis of IteRABRe's impact on knowledge, linguistic, and multilingual zero-shot capabilities.

2 Background: Structured Prunning

The increasing scale of Large Language Models (LLMs) has driven efforts to reduce their computational and memory footprint for deployment. A prominent approach is **structured pruning** (Wang et al., 2020), where some components (e.g., layers, attention heads) are removed from a large model M_L to derive a smaller model M_S. However, pruning often causes performance degradation, making a careful selection of components and recovery strategies after pruning necessary (Sun et al., 2024; Yin et al., 2024; Ma et al., 2023). The following are the explanations of these phases:

Pruning Phase Formally, let M_L consist of N transformer component blocks $\{B_1, B_2, ..., B_N\}$. Pruning involves ranking blocks by importance and retaining the top-k blocks ($k < N$) to form M_S. The importance of a block B_i is determined by a scoring function $f(B_i)$, which can be defined as:

$$f(B_i) = \text{Importance}(B_i; \mathcal{D}_{\text{eval}})$$

Here, $\mathcal{D}_{\text{eval}}$ is a validation dataset (calibration dataset) used to compute metrics to determine the blocks' importance. Blocks are then sorted by $f(B_i)$, and the least important $N - k$ are pruned or dropped:

$$M_S = \text{Prune}(M_L, k)$$

Recovery Phase To alleviate performance degradation due to pruning, this phase fine-tunes M_S on a recovery dataset $\mathcal{D}_r$ on the respective tasks, such as Causal Language Modeling. The recovery process is useful to adapt to its new structure and reallocate its internal knowledge to its remaining capacity.

The recovery process optimizes:

$$\theta_S^* = \arg\min_{\theta_S} \mathcal{L}(M_S(\theta_S; \mathcal{D}_r), y)$$

where θ_S, y denotes the parameters of M_S and ground truth, respectively.[1]

3 Proposed Method: IteRABRe

We introduce IteRABRe, an iterative compression framework for large language models that alternates between pruning and recovery phases until a target model size, which is the number of layers, is reached. This process continues until the desired number of layers in M_s. Although iterative compression has been explored previously (Muralidharan et al., 2024), our approach makes two key distinctions: (1) a direct layerwise pruning strategy and (2) an efficient recovery process that achieves competitive performance with significantly reduced data requirements. The method introduced in Muralidharan et al., 2024 may achieve better results through extensive compute and data resources, however our approach demonstrates competitive performance to other baselines with substantially lower resource requirements. Our approach is illustrated in Figure 1.

The following subsections elaborate on the respective phases in IteRABRe defined in §2.

3.1 IteRABRe's Pruning Phase

We define B_i as transformer blocks, where each block consists of **self-attention and feed-forward components**. To minimize performance degradation during pruning, we evaluate the importance of each layer B_i by measuring its contribution to the model's output quality.

Specifically, we compute the cosine similarity between the last hidden state of the original model M_L and the last hidden state of the candidate pruned models $M_{cs}^{(i)}$, where $M_{cs}^{(i)}$ is obtained by removing one layer of self-attention from M_L. The importance score $f(B_i)$ for a block B_i is defined as:

$$f(B_i) = \frac{1}{|\mathcal{D}_{\text{eval}}|} \sum_{d=1}^{|\mathcal{D}_{\text{eval}}|} \text{sim}\left(h(M_L)_d, h(M_{cs}^{(i)})_d\right)$$

[1]These variables declared in this section will be used throughout this paper.

where $h(M_L)_d$ is the last hidden state of the original model M_L with N layers for the d-th input sequence in $\mathcal{D}_{\text{eval}}$, $h(M_{cs}^{(i)})_d$ is the last hidden state of the pruned model $M_{cs}^{(i)}$ with $N-1$ layers for the same input sequence. $\text{sim}(\cdot,\cdot)$ denotes the cosine similarity function. After computing $f(B_i)$ for all blocks, we sort the blocks by their similarity scores. The lowest similarity block will be selected for removal, as it indicates the least impact on model performance. This process yields our final pruned model M_{cs} with the selected blocks removed. M_{cs}, then will be processed in the Recovery Phase.

For better clarity in the following sections, we also denote $M_{cs}^{[j]}$ as the final pruned model chosen in iteration j.

3.2 IteRABRe's Recovery Phase

To further preserve the degradation quality of the model, we employ knowledge distillation, where we put the original model, M_L as the teacher T and the pruned model from the previous phase in the same iteration j as its student $M_{cs}^{[j]}$, which we denote here as S. We follow the TinyBERT design (Jiao et al., 2020), where we compute the mean square error (MSE) between all hidden states, attention, and output logits. We use MSE for the output logits as it shows better performance than KL Divergence (Kim et al., 2021). Formally, it is defined as follows:

$$\mathcal{L}_{KD} = \sum_{l=1}^{L} \Big(\text{MSE}(\mathbf{H}_T^{\text{map}(l)}, \mathbf{H}_S^{l}) + \text{MSE}(\mathbf{A}_T^{\text{map}(l)}, \mathbf{A}_S^{l})\Big) + \text{MSE}(\mathbf{z}_T, \mathbf{z}_S)$$

Here, $\mathbf{H}_T^{map(l)}$ and $\mathbf{H}_S^{l}$ represent the hidden states in layers l and $map(l)$ for the teacher and student models, respectively, while $\mathbf{A}_T^{map(l)}$ and $\mathbf{A}_S^{l}$ denote their corresponding attention matrices. The output logits of the teacher and student models are represented by $\mathbf{z}_T$ and $\mathbf{z}_S$, respectively. $map(l)$ is defined as the mapping of a student's layer to the teacher's layer which aligns the student's layer index l with the corresponding original index in the teacher model [2].

After this phase, we produce a recovered pruned model $M_{cs-rec}^{[j]}$ as the final chosen in iteration j. $M_{cs-rec}^{[j]}$ is then processed to the next iteration $j+1$

[2] See Appendix E for more explanation

4 Experimental Setup

To test our proposed algorithm and analyze it, we use the following setups:

Evaluation To have a better assessment of our approach, we categorize the benchmark dataset into three categories, **reasoning, language comprehension, and knowledge**. For reasoning, we leverage `arc-challenge` and `arc-easy dataset` (Clark et al., 2018), `hellaswag` (Zellers et al., 2019), `COPA` (Roemmele et al., 2011), `PIQA` (Bisk et al., 2020). For language comprehension, we test our data on `BLiMP` (Warstadt et al., 2020), `RACE` (Lai et al., 2017), and `Winogrande` (Sakaguchi et al., 2021). As for the knowledge category, we use `BoolQ` (Clark et al., 2019) and `MMLU` (Hendrycks et al., 2021). To evaluate our model, we use off-the-shelf `lm-eval-harness` (Gao et al., 2024) library. We use the context length of 1024 and employ the zero-shot setting to obtain the score. To measure the performance, we use test set of `wikitext-2-raw-v1` to measure perplexity and use accuracy for the rest.

Models We used two widely used LLM families, Qwen2.5 (Yang et al., 2024a) and Llama3 (Grattafiori et al., 2024). To observe the impact of model size, we use 8B, 3B, and 1B from Llama3 and 7B and 0.5B for Qwen2.5.

Pruning Phase For the pruning phase, we use 10 instances as the calibration dataset, sampled from the `wikitext-2-raw-v1` dataset on Hugging Face, following Yang et al. (2024). The sampled Wikitext data is provided in Appendix A. As demonstrated by Yang et al. (2024b), using different samples does not significantly affect the results.

Recovery Phase We employ Knowledge Distillation (Hinton et al., 2015) following the TinyBERT approach (Jiao et al., 2020). To accommodate our computational constraints, we implement LoRA (Hu et al., 2021) with a rank of 32. Our training configuration includes a batch size of 4 with gradient accumulation of 8 (effective batch size of 32), learning rate of 1×10^{-4}, and maximum sequence length of 512. For efficient recovery training, we conduct a single epoch on the `Wikitext-2-raw-v1` dataset, comprising approximately 2.5M tokens. The training was performed on 2 $\times$ A100 GPUs.

Baseline To evaluate IteRABRe, we compare it with two baseline layer pruning methods:

Model	Approach	#L	Wiki↓	Reasoning					Language Comprehension			Knowledge	
				ARC-C	ARC-E	HellaSwag	COPA	PIQA	BLiMP	RACE	Winogrande	BoolQ	MMLU
Llama3.1 8B	Not Pruned	32	8.65	51.28	81.48	60.03	87.0	80.14	81.93	39.14	73.56	82.08	63.59
	LaCO	24	23.55	30.29	63.01	43.22	**81.0**	71.76	79.34	30.91	55.72	61.99	23.96
	ShortGPT	24	6636.72	27.47	42.68	28.28	63.0	60.55	66.84	25.07	53.91	37.58	**32.21**
	IteRABRe	24	**16.89**	**33.02**	**67.85**	**47.49**	80.0	**74.27**	**84.10**	**35.69**	**60.93**	**62.26**	23.80
Llama3.2 3B	Not Pruned	28	11.06	42.24	74.54	55.27	86.0	76.61	82.15	40.19	69.77	73.24	54.38
	LaCO	**24***	**25.55**	26.96	56.65	**42.54**	**80.0**	**71.76**	80.15	**32.63**	55.72	60.03	24.47
	ShortGPT	21	235.23	**30.89**	49.71	37.34	71.0	64.15	72.53	30.53	**61.88**	45.02	**34.38**
	IteRABRe	21	26.92	30.12	**58.84**	41.53	76.0	70.08	**80.60**	32.44	58.33	**62.17**	26.15
Llama3.2 1B	Not Pruned	16	13.91	31.31	65.40	47.78	77.0	74.48	82.44	37.89	60.77	63.98	37.54
	LaCO	12	80.16	19.28	40.15	29.77	56.0	61.10	72.93	26.03	51.38	37.83	**23.05**
	ShortGPT	12	846.24	**25.51**	34.89	31.51	**70.0**	59.03	74.20	24.88	**54.14**	55.96	22.70
	IteRABRe	12	**31.85**	23.89	**51.14**	**33.39**	66.0	**65.61**	**82.71**	**28.04**	51.22	**62.14**	23.00
Qwen2.5-7B	Not Pruned	28	10.35	47.78	80.39	60.03	91.0	78.67	82.24	41.63	72.93	84.65	71.91
	LaCO	**22***	48.38	29.52	50.80	39.32	71.0	67.14	75.60	27.18	55.88	47.19	**31.83**
	ShortGPT	21	18.57	33.79	70.88	44.32	76.0	74.27	81.93	33.01	53.51	45.84	26.52
	IteRABRe	21	**16.40**	**35.58**	**71.13**	**45.59**	**77.0**	**74.32**	**83.48**	**36.08**	**57.70**	**53.73**	30.94
Qwen2.5-0.5B	Not Pruned	24	21.68	29.52	64.56	40.64	74.0	70.29	81.73	35.02	56.35	62.42	47.73
	LaCO	18	230.05	**24.06**	45.79	30.95	60.0	62.51	71.20	26.41	**51.22**	54.13	**25.54**
	ShortGPT	18	45.86	21.42	52.15	32.39	62.0	65.18	77.94	**28.80**	49.33	**58.96**	25.17
	IteRABRe	18	**38.17**	23.89	**54.71**	**32.41**	**68.0**	**65.40**	**78.61**	27.27	50.75	47.16	23.32

Table 1: Performance comparison across model scales and tasks, showing perplexity (Wiki↓, where lower is better) and accuracy scores (%). Bold indicates the best performance among other approaches (LaCO, ShortGPT, IteRABRe) for each metric. *: Due to the dependency on hyperparameter in LaCO, some of its results may have incomparable compression with others. #L denotes number of layers.

LaCO (Yang et al., 2024b) and ShortGPT (Men et al., 2024). While our approach adopts LaCO's layer importance assessment methodology, ShortGPT employs Block Influence (BI). Our method extends these approaches by incorporating recovery and iterative pruning. We target a compression rate of approximately 25%, following previous works. For ShortGPT, we implemented the method ourselves to obtain results, while for LaCO, we utilized their publicly available code. Since LaCO's compression rate varies with hyperparameters, we conducted a grid search and selected the model with the closest compression rate and highest perplexity score on `wikitext-v2-raw-v1`.

5 Results

IteRABRe Outperforms Other Baselines Overall Table 1 presents the experimental results. IteRABRe outperforms other methods (LaCO and ShortGPT) across all model scales. Specifically, IteRABRe maintains a lower perplexity on Wikitext compared to the baselines, avoiding the sharp increases observed with ShortGPT on Llama3.1-8B (6636.72) and LaCO on Qwen2.5-7B (48.38). IteRABRe also achieves the highest performance in the reasoning domain for the 7B and 8B models. However, for smaller models (0.5B, 1B, and 3B), IteRABRe exhibits a small performance gap compared to the baselines on `arc-challenge`, likely because this task's reliance on multi-hop reasoning demands greater model capacity.

In the language category, IteRABRe maintains performance better than the other methods, particularly on `BLIMP`, where the large models (7B and 8B) even outperform their non-pruned counterparts. We attribute this to the recovery phase, where training on `wikitext` helps preserve linguistic capabilities. On the other hand, `RACE` and `Winogrande` show moderate performance gaps (2-5%) across all model sizes. These results suggest that our method offers particular advantages for language comprehension in large models.

In the knowledge domain, IteRABRe achieves strong `BoolQ` performance, with the exception of Qwen2.5-0.5B. The improved accuracy for this model is likely due to the use of `wikitext` as a recovery training dataset. However, MMLU results lag behind the other methods, by approximately 9% compared to the highest performer on Llama3.1-8B and Llama3.2-3B, and by 1-2% on the other models. This difference may be due to the fact that our approach does not preserve or recover information crucial for maintaining MMLU performance.

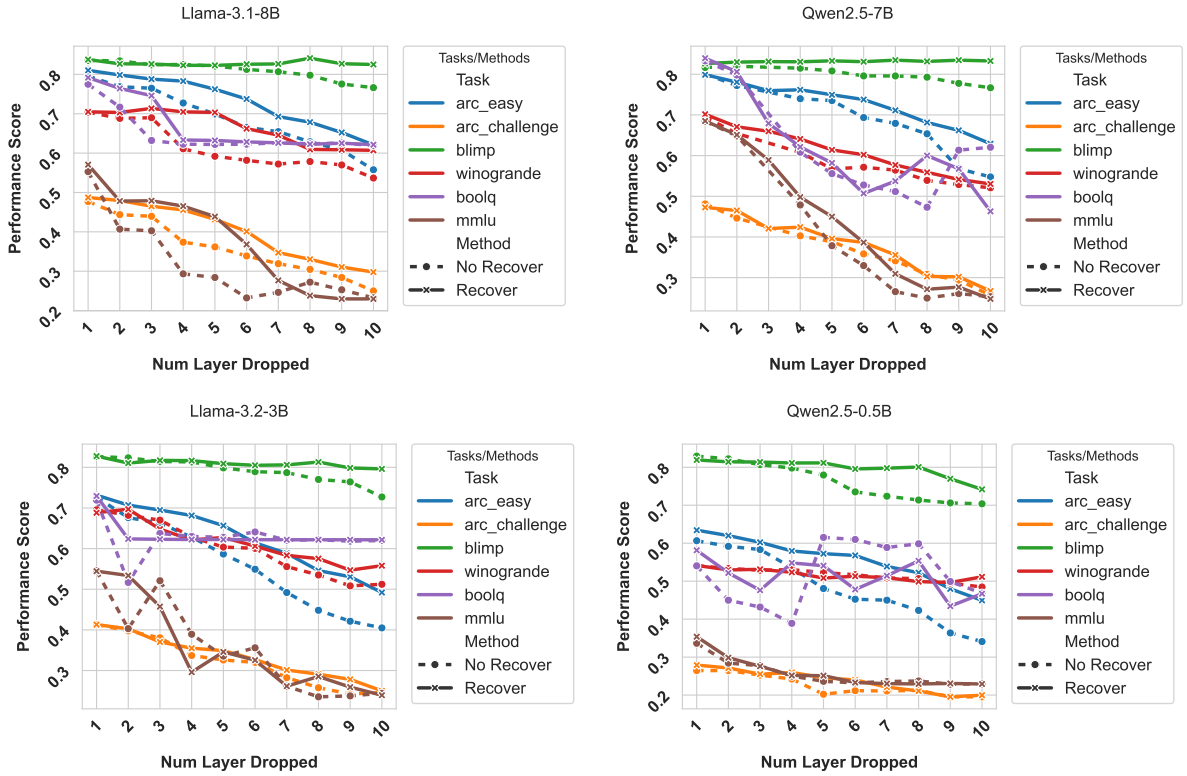

Figure 2: IteRABRe's performance on six different subtasks. dotted line denoted implementing IteRABRe without recovery phase while solid line denoted layer prunning and recovery phase are done in IteRABRe

IteRABRe's recovery phase boosts performance, notably for larger models on reasoning and language tasks. We investigated the impact of each phase of IteRABRe. The results are shown in Figure 2. In summary, the iterative recovery phase helps preserve performance on reasoning and language tasks, particularly in later iterations. For example, with Llama3.1-8B, the performance difference between the first and third iterations is approximately 1-3%, while it widens to 5-10% between the fourth and sixth iterations. This pattern is also observed on Winogrande. For BLIMP, the performance gap similarly increases in later iterations (6th-10th). QWEN exhibits the same trend, albeit with smaller gaps.

For knowledge tasks, MMLU shows a clear performance difference in both the 7B and 8B models. However, BoolQ exhibits an irregular trend with Qwen2.5-7B, with fluctuating performance (sometimes higher, sometimes lower) and ~1% differences in the Llama3 model. This behavior is also observed in smaller models (0.5B and 3B) for both tasks. Overall, the recovery phase provides a considerable performance improvement, except in the knowledge domain, especially for smaller models.

Iterative pruning generally outperforms direct pruning on most tasks, but its effectiveness varies on some tasks on each model. To evaluate the effectiveness of the iterative process, we compared iterative pruning with direct pruning with recovery. Direct pruning involves consecutively pruning all layers until the desired compression level is reached before the recovery phase, rather than iteratively pruning and recovering.

Table 2 shows that iterative pruning generally preserves performance better than direct pruning on most language and reasoning tasks, such as `arc-easy` and `blimp`, while also maintaining `wikitext` perplexity even with increased layer dropping. However, `winogrande` and `mmlu` do not benefit from iterative pruning, showing comparable or slightly reduced performance. `arc-challenge` and `race` show comparable performance between the two methods. We hypothesize that our experimental setup, specifically training on `wikitext`, may lead to fitting on narrow knowledge, which is less suitable for these particular tasks. Furthermore, `boolq` exhibits different behavior between Llama and Qwen, likely due to their distinct pre-training configurations. This suggests that the effectiveness of IteRABRe varies across tasks.

Approach	#Dropped Layers	Wiki	BOOLQ	ARC-E	COPA	BLiMP	H-SWAG	PIQA	ARC-C	RACE	WG	MMLU	Approach Avg Diff
Llama 3.1-8B													
Direct+R	4	12.22	65.05	76.60	87.00	82.98	54.69	77.04	44.54	39.23	71.03	43.31	
Iterative+R	4	12.12	63.33	78.28	85.00	82.30	55.18	77.15	45.56	39.14	70.48	46.53	**0.15**
Diff		*0.10*	*-1.72*	*1.68*	*-2.00*	*-0.68*	*0.49*	*0.11*	*1.02*	*-0.09*	*-0.55*	*3.22*	
Direct+R	8	20.64	51.52	66.37	79.00	81.85	46.49	72.31	33.53	35.79	66.30	37.52	
Iterative+R	8	16.89	62.26	67.85	80.00	84.10	47.49	74.27	33.02	35.69	60.93	23.80	-0.10
Diff		*3.75*	*11.01*	*1.48*	*1.00*	*2.25*	*1.00*	*1.96*	*-0.51*	*-0.10*	*-5.37*	*-13.72*	
Direct+R	12	35.74	38.13	55.43	70.00	81.32	39.32	67.79	27.73	32.92	57.06	24.67	
Iterative+R	12	31.63	62.17	58.04	75.00	82.29	38.56	67.36	27.22	32.06	54.78	22.96	**2.61**
Diff		*4.11*	*24.04*	*2.61*	*5.00*	*0.97*	*-0.76*	*-0.44*	*-0.51*	*-0.86*	*-2.29*	*-1.71*	
QWEN2.5-7B													
Direct+R	4	12.99	54.95	76.01	83.00	83.61	51.82	77.58	41.72	36.27	64.88	53.14	
Iterative+R	4	13.23	62.14	76.18	86.00	83.07	52.43	77.80	42.41	36.84	64.09	49.79	**0.77**
Diff		*-0.24*	*7.19*	*0.17*	*3.00*	*-0.54*	*0.61*	*0.22*	*0.68*	*0.57*	*-0.79*	*-3.35*	
Direct+R	8	18.96	48.20	67.89	78.00	83.34	42.43	72.47	32.08	32.73	54.70	26.17	
Iterative+R	8	18.79	60.00	68.14	75.00	83.16	42.43	73.23	30.29	33.88	55.88	27.15	**1.12**
Diff		*0.17*	*11.80*	*0.25*	*-3.00*	*-0.18*	*0.00*	*0.76*	*-1.79*	*1.15*	*1.18*	*0.98*	
Direct+R	12	46.15	57.09	53.75	63.00	78.97	33.38	66.05	22.95	29.00	52.88	24.38	
Iterative+R	12	35.22	42.69	57.15	66.00	81.47	35.50	66.05	24.66	28.71	53.04	24.18	-0.20
Diff		*10.93*	*-14.40*	*3.41*	*3.00*	*2.50*	*2.12*	*0.00*	*1.71*	*-0.29*	*0.16*	*-0.20*	
Tasks Avg Diff	-	**3.13**	**6.32**	**1.60**	**1.17**	**0.72**	**0.57**	**0.44**	**0.10**	**0.06**	-1.28	-2.46	-

Table 2: The comparison between direct and iterative approach. Diff denote the difference between direct with respect to iterative approach.

6 IteRABRe Preservation Analysis

IteRABRe effectively preserves language and reasoning abilities across iterations, though knowledge retention presents a challenge. Figure 3 shows the average performance trend across iterations for each task category. While Qwen2.5-7B exhibits a slight, steady decrease (averaging ~1% per iteration) in reasoning and language task performance, Llama3.1-8B plateaus in language but shows a steady decline in reasoning. Both models experience sharp performance drops in specific iterations (e.g., $M_{cs}^{[2]}$ for Llama and $M_{cs}^{[3]}$ for Qwen). This affirms IteRABRe's effectiveness in preserving language and reasoning abilities, though it suggests challenges in maintaining knowledge-based performance across iterations.

The recovery phase generally improves performance, though its impact is task and model dependent. The recovery phase generally improves performance by approximately 1% for both models (Figure 3). However, its impact varies; for example, $M_{cs-rec}^{[5]}$ on Llama3.1-8B shows a slight decrease in reasoning performance after recovery, while language task performance increases. This indicates that the recovery process's effectiveness depends on the model family and the specific task.

IteRABRe Preserves and May Improves Linguistic Capabilites We evaluated the preservation of linguistic capacity across iterations using BLIMP, a benchmark consisting of 67 fine-grained linguistic problems. We tested on Llama-3.1-8B and Qwen2.5-7B, categorizing the BLIMP subtasks into 13 groups for clearer visualization (see Appendix B for the groupings).

Overall, both models maintain or even improve scores across most categories in later iterations, surpassing the performance of the non-compressed models. Furthermore, IteRABRe with recovery consistently outperforms the pruned model without recovery, with the exception of the "binding theory" category. In this category, we observe a slight performance decay (~2%) starting from the seventh iteration for Llama3.1-8B and the eighth iteration for Qwen2.5-7B. The "coordinate structure" and "wh-that" categories exhibit differing trends between these family models. Llama3.1-8B shows an opposing trend at iteration 7 and beyond, with one subcategory plateauing while the other increases in performance.

MMLU performance is sensitive to pruning, with recovery offering moderate gains across MMLU task categories Figure 5 provides the MMLU performance across MMLU groupings. [3] It shows that the pruning phase induces significant performance drops in some cases, notably in the early layer dropping of Llama3.1-8B (around

[3] using groupings defined in `lm-eval-harness`

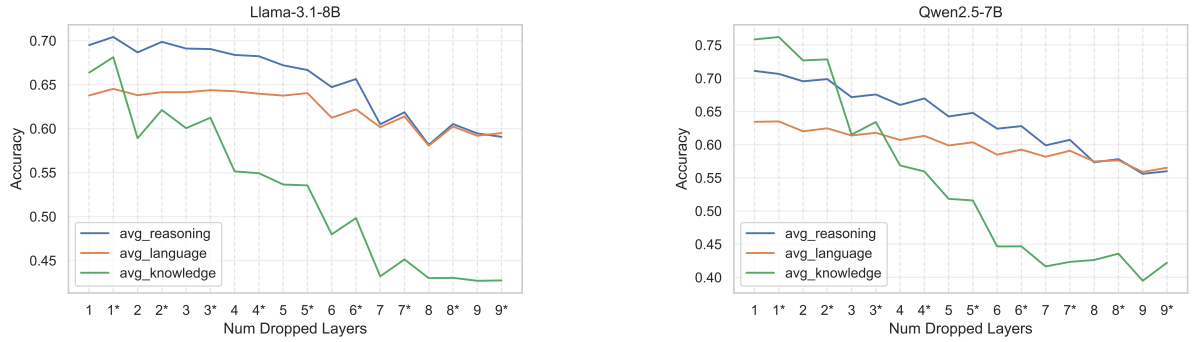

Figure 3: The average performance across pruning and recovery phase for 10 iterations on Llama 3.1-8B and Qwen2.5-7B on an average aggregation of reasoning, language, and knowledge tasks.

10%) and from the third layer onward in Qwen2.5-7B. This suggests greater sensitivity of knowledge-based tasks to pruning. The subsequent recovery phase provides moderate improvements (about 2-3%) for both models. Interestingly, Llama3.1-8B at M^2_{cs-rec} shows a moderate performance gain, sustained across the next four iterations. This sustained improvement is not exhibited in Qwen2.5-7B, which instead exhibits a steady performance decline. Performance trends across iterations are similar across MMLU categories within the same model, yet differ between models. These differences highlight model-specific variations in knowledge retention, potentially due to the distinct pretraining strategies of Llama3.1-8B and Qwen2.5-7B.

Model	Approach	L	xW	xSc	xNLI
Llama3.1-8B	Non-pruned	32	81.43	63.61	45.65
	LaCO	24	67.39	52.05	37.78
	S-GPT	24	56.37	48.80	34.25
	Ours P	24	66.40	51.65	37.45
	Ours P+R	24	**71.68**	**55.53**	**39.77**
Qwen2.5-7B	Non-pruned	28	81.48	62.04	43.37
	LaCO	22	64.71	51.66	36.49
	S-GPT	21	66.33	55.28	37.32
	Ours-P	21	65.54	53.48	36.99
	Ours-P+R	21	**72.26**	**55.76**	**39.46**

Table 3: Performance Comparison in Multilingual Data. XW denotes `XWinograd` and XSc denotes `XStoryCloze`. Ours denotes IteRABRe, with **R** and **P** denotes running it with recovery and pruning phases, respectively.

IteRABRe recovery improves multilingual performance, but the effect of improvement varies significantly across languages and tasks. Given that our recovery phase uses English data, and both models possess multilingual capabilities, we investigated how much multilingual capacity is retained and whether IteRABRe induces zero-shot cross-lingual generalization during recovery. We evaluated our approach on three multilingual benchmarks: `XWinograd`, `XStoryCloze`, and `XNLI`.

Table 3 compares the baseline models with IteRABRe (without recovery). The results demonstrate that our recovery method improves performance by 5-6% on `XWinograd` across both models, with 2-5% improvements on `XStoryCloze` and `XNLI`. These findings suggest effective generalization to multilingual data from English-based recovery.

Figure 7 presents performance per language within each dataset. Both Llama3.1-8B and Qwen2.5-7B show similar patterns, though Llama3.1-8B exhibits larger performance gaps. For Llama3.1-8B, English shows the strongest performance preservation, with approximately 8% improvements on `XWinograd`, `XStoryCloze`, and `XNLI` compared to the non-recovery baseline. Other languages show more modest gains, averaging around 2% post-recovery. Notably, some languages in Llama 3.1-8B's `XStoryCloze`, particularly `es` and `id`, show improvements comparable to `en` (6-8%). In `xnli`, `de` and `en`, performances approach those of the unpruned models. These findings align with Choenni et al. (2023), suggesting varying cross-lingual influence. Some languages, such as `hi` and `ur`, show minimal preservation (less than 1% improvement). `zh` notably performs worse with recovery compared to layer pruning alone. These results indicate that while English-based recovery can facilitate zero-shot cross-lingual generalization, its effectiveness varies considerably across languages and tasks.

More analysis can be seen in Appendix F

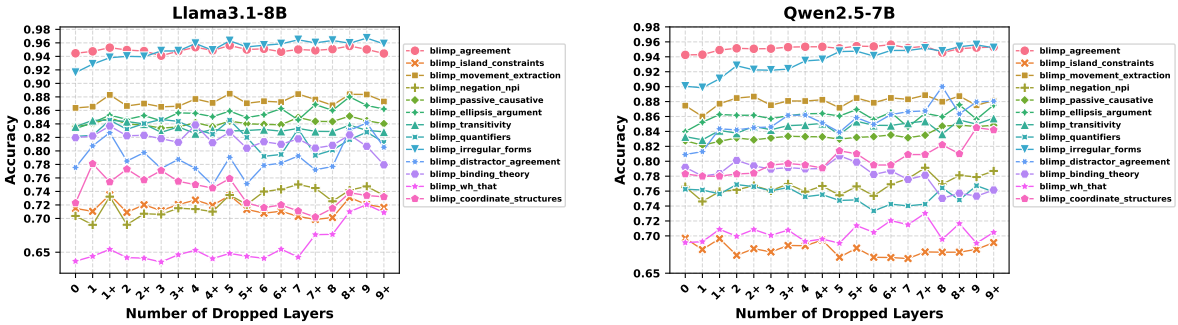

Figure 4: Line charts showing BLIMP performance across 13 groupings for Llama-3.1-8B and Qwen-2.5-7B over 10 iterations. "+" markers indicate the recovery phase; all other markers represent the pruning phase.

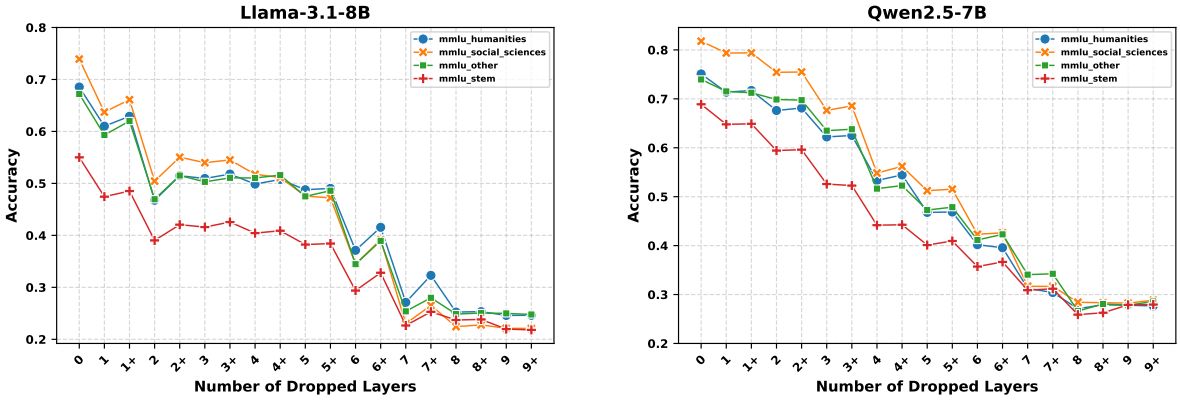

Figure 5: Line charts depicts MMLU groupings performance on Llama-3.1-8B and Qwen2.5-7B in 10 iterations. "+" markers indicate the recovery phase; all other markers represent the pruning phase.

7 Related Works

Model pruning has gained significant attention recently due to the emergence of Large Language Models (LLMs). One of the approaches is to do unit size reduction, where several approaches leverage dimensionality reduction techniques (Lin et al., 2024; Ashkboos et al., 2024) to compress weight matrices, thereby reducing hidden unit dimensions. Various metrics have been explored to identify prunable weights, including Hessian information (Frantar and Alistarh, 2023; Ling et al., 2024), Kronecker-factored curvature (van der Ouderaa et al., 2024), and magnitude information (Sun et al., 2024; Guo et al., 2024). On the other hand, block pruning is done by employing some metrics, such as Hessian information (Ma et al., 2023), output similarity (Yang et al., 2024b; Men et al., 2024), and learnable parameters to determine block significance (Liu et al., 2024; Xia et al., 2024). Some approaches opt to merge blocks instead of removing them (Yang et al., 2024b; Chen et al., 2024). Muralidharan et al., 2024 combines iterative pruning with Neural Architecture Search (Elsken et al., 2019), utilizing multiple metrics for model compression. Many of these techniques incorporate recovery phase (Ling et al., 2024; Sun et al., 2024; Yin et al., 2024; Ma et al., 2023; Muralidharan et al., 2024). In our work, we adopt an iterative approach based on output similarity, followed by a recovery process, prioritizing simplicity and minimal computational requirements.

8 Conclusion

This work introduced IteRABRe, a simple yet effective iterative block pruning method designed for simple and efficient LLM compression. IteRABRe outperforms other baselines requiring only 2.5M tokens. Furthermore, our analysis reveals distinct pruning patterns on the observed tasks across different model architectures and also preserves the multilingual capabilities, even with English-only recovery data.

Limitations

We only observe the Qwen2.5 and Llama3 family models. Additionally, we only observe Wikitext, to perform the pruning phase (following Yang et al., 2024b) and recovery phase. We leave the possibil-

ity of using a variety of dataset sizes or sampling techniques for this technique for future work.

Ethics Statement

This work has no ethical issues, as we propose to perform a compression technique. The data used do not contain personally identifiable information or offensive content. The artifacts we utilize are consistent with intended use and adhere to the license usage (research purpose). We use AI Assistants (LLMs, Grammarly, and Writefull) to assist our writing in correcting grammatical errors.

References

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

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

Xiaodong Chen, Yuxuan Hu, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. 2025. Streamlining redundant layers to compress large language models.

Yilong Chen, Junyuan Shang, Zhenyu Zhang, Shiyao Cui, Tingwen Liu, Shuohuan Wang, Yu Sun, and Hua Wu. 2024. LEMON: Reviving stronger and smaller LMs from larger LMs with linear parameter fusion. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8005–8019, Bangkok, Thailand. Association for Computational Linguistics.

Rochelle Choenni, Dan Garrette, and Ekaterina Shutova. 2023. How do languages influence each other? studying cross-lingual data sharing during LM fine-tuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13244–13257, Singapore. Association for Computational Linguistics.

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.

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.

Razvan-Gabriel Dumitru, Paul Ioan Clotan, Vikas Yadav, Darius Peteleaza, and Mihai Surdeanu. 2024. Change is the only constant: Dynamic LLM slicing based on layer redundancy. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9912–9920, Miami, Florida, USA. Association for Computational Linguistics.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural architecture search: A survey.

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

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. A framework for few-shot language model evaluation.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline

Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wen-

wen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models.

Hongcheng Guo, Jian Yang, Jiaheng Liu, Liqun Yang, Linzheng Chai, Jiaqi Bai, Junran Peng, Xiaorong Hu, Chao Chen, Dongfeng Zhang, Xu Shi, Tieqiao Zheng, Liangfan Zheng, Bo Zhang, Ke Xu, and Zhoujun Li. 2024. Owl: A large language model for it operations.

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

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Taehyeon Kim, Jaehoon Oh, NakYil Kim, Sangwook Cho, and Se-Young Yun. 2021. Comparing kullback-leibler divergence and mean squared error loss in knowledge distillation.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Chi-Heng Lin, Shangqian Gao, James Seale Smith, Abhishek Patel, Shikhar Tuli, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. 2024. Modegpt: Modular decomposition for large language model compression.

Gui Ling, Ziyang Wang, Yuliang Yan, and Qingwen Liu. 2024. Slimgpt: Layer-wise structured pruning for large language models.

Songwei Liu, Chao Zeng, Lianqiang Li, Chenqian Yan, Lean Fu, Xing Mei, and Fangmin Chen. 2024. Foldgpt: Simple and effective large language model compression scheme.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models.

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

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. 2024. Compact language models via pruning and knowledge distillation.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI spring symposium series*.

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.

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

Tycho F. A. van der Ouderaa, Markus Nagel, Mart van Baalen, Yuki M. Asano, and Tijmen Blankevoort. 2024. The llm surgeon.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2020. Structured pruning of large language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6151–6162, Online. Association for Computational Linguistics.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R Bowman. 2020. Blimp: The benchmark of linguistic minimal pairs for english. *Transactions of the Association for Computational Linguistics*, 8:377–392.

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

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024a. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Yifei Yang, Zouying Cao, and Hai Zhao. 2024b. LaCo: Large language model pruning via layer collapse. In

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 6401–6417, Miami, Florida, USA. Association for Computational Linguistics.

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.

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

A Calibration Dataset

Here is the calibration dataset sampled from Wikitext used to run the pruning phase. These are sampled uniformly.

1. " = There 's Got to Be a Way = "
2. " Cullen is the namesake of the John Cullen Award , previously given to key IHL players."
3. "Ancient Egyptian deities are the gods and goddesses worshipped in ancient Egypt"
4. "Nationally important deities gave rise to local manifestations"
5. "The first aerodrome in the UK was established by the Aero Club at Muswell Manor on the Isle of Sheppey"
6. "Competitive gliding in the UK takes place between May and September"
7. "Most aerodromes used for public transport operations are required to be licensed by the CAA"
8. "Within this framework certain sectors of GA are governed on a devolved basis"
9. "James Pollock , in his final report as Mint Director in 1873"
10. "Stability of the ylide with higher stability similarly leading to greater reversibility."

B Performance Trend for Each Iteration

The fine-grained performance trend can be seen in Figure 6.

C Blimp Categories

Blimp categories that we define can be seen in Table 4,5,6,7.

D Granular Multilingual Results

The Granular multilingual across `xwinograd`, `xstorycloze`, `xnli` analysis can be viewed in Figure 7

E Layer Mapping in Recovery Phase

To define the mapping function map(l) in iteration j, we aim to align the student's layer index l with the corresponding original index in the teacher model. However, if any layers in the teacher model with indices lower than l were dropped before iteration j, the mapping must account for these dropped layers. Specifically, map(l) is adjusted by increasing it by the number of dropped layers with indices less than map(l). For example, if the dropped layer indices are $[3, 4]$ and $l = 10$, then map$(10) = 12$, as the two dropped layers shift the mapping while $map(1) = 1$. Formally, let D be the set of dropped layer indices in the teacher model before iteration j, sorted in ascending order. The function map(l) maps the student's layer index l to the teacher's original index m, where m is the unique solution to the equation $m = l + |\{d \in D \mid d < m\}|$.

F More Analysis

IteRABRe exhibits task-specific layer sensitivities that vary between models. We investigated which layer drops correlate with significant performance declines, indicating layer importance. Figure 8 shows performance differences across tasks and categories for Qwen2.5-7B and Llama-3.1-8B, revealing distinct drop patterns for each model. Llama3.1-8B's performance drops tend to occur in the lower half of its layers, while Qwen's are concentrated in the upper half. Specifically, Llama3.1-8B shows significant drops on `arc-easy` and `arc-challenge` in iteration 1, 6, and 7, and on `winogrande` in iteration 1, 6, and 8. MMLU on Llama3.1-8B shows steep declines in iteration 10 and 11 during early iterations, followed by improvement and stagnation. Qwen2.5-7B exhibits different trends, with notable (>5%) decreases on MMLU in iteration 3, 4, 6, and 7.

Group	Tests
blimp_agreement	• blimp_regular_plural_subject_verb_agreement_1 • blimp_regular_plural_subject_verb_agreement_2 • blimp_irregular_plural_subject_verb_agreement_1 • blimp_irregular_plural_subject_verb_agreement_2 • blimp_determiner_noun_agreement_1 • blimp_determiner_noun_agreement_2 • blimp_determiner_noun_agreement_irregular_1 • blimp_determiner_noun_agreement_irregular_2 • blimp_determiner_noun_agreement_with_adj_2 • blimp_determiner_noun_agreement_with_adj_irregular_1 • blimp_determiner_noun_agreement_with_adj_irregular_2 • blimp_determiner_noun_agreement_with_adjective_1 • blimp_anaphor_gender_agreement • blimp_anaphor_number_agreement
blimp_distractor_agreement	• blimp_distractor_agreement_relational_noun • blimp_distractor_agreement_relative_clause

Table 4: BLiMP Agreement Tests

Group	Tests
blimp_island_constraints	• blimp_wh_island • blimp_complex_NP_island • blimp_adjunct_island • blimp_sentential_subject_island • blimp_left_branch_island_echo_question • blimp_left_branch_island_simple_question
blimp_movement_extraction	• blimp_wh_questions_object_gap • blimp_wh_questions_subject_gap • blimp_wh_questions_subject_gap_long_distance • blimp_coordinate_structure_constraint_object_extraction • blimp_existential_there_subject_raising • blimp_existential_there_object_raising • blimp_expletive_it_object_raising
blimp_wh_that	• blimp_wh_vs_that_no_gap • blimp_wh_vs_that_no_gap_long_distance • blimp_wh_vs_that_with_gap • blimp_wh_vs_that_with_gap_long_distance

Table 5: BLiMP Syntax and Movement Tests

Group	Tests
blimp_passive_causative	• blimp_passive_1 • blimp_passive_2 • blimp_animate_subject_passive • blimp_causative
blimp_transitivity	• blimp_transitive • blimp_intransitive • blimp_inchoative • blimp_animate_subject_trans
blimp_irregular_forms	• blimp_irregular_past_participle_adjectives • blimp_irregular_past_participle_verbs

Table 6: BLiMP Argument Structure and Form Tests

Group	Tests
blimp_negation_npi	• blimp_npi_present_1 • blimp_npi_present_2 • blimp_only_npi_licensor_present • blimp_only_npi_scope • blimp_sentential_negation_npi_licensor_present • blimp_sentential_negation_npi_scope • blimp_matrix_question_npi_licensor_present
blimp_quantifiers	• blimp_superlative_quantifiers_1 • blimp_superlative_quantifiers_2 • blimp_existential_there_quantifiers_1 • blimp_existential_there_quantifiers_2
blimp_binding_theory	• blimp_principle_A_c_command • blimp_principle_A_case_1 • blimp_principle_A_case_2 • blimp_principle_A_domain_1 • blimp_principle_A_domain_2 • blimp_principle_A_domain_3 • blimp_principle_A_reconstruction
blimp_ellipsis_argument	• blimp_ellipsis_n_bar_1 • blimp_ellipsis_n_bar_2 • blimp_drop_argument
blimp_coordinate_structures	• blimp_coordinate_structure_constraint_complex_left_branch

Table 7: BLiMP Specialized Construction Tests

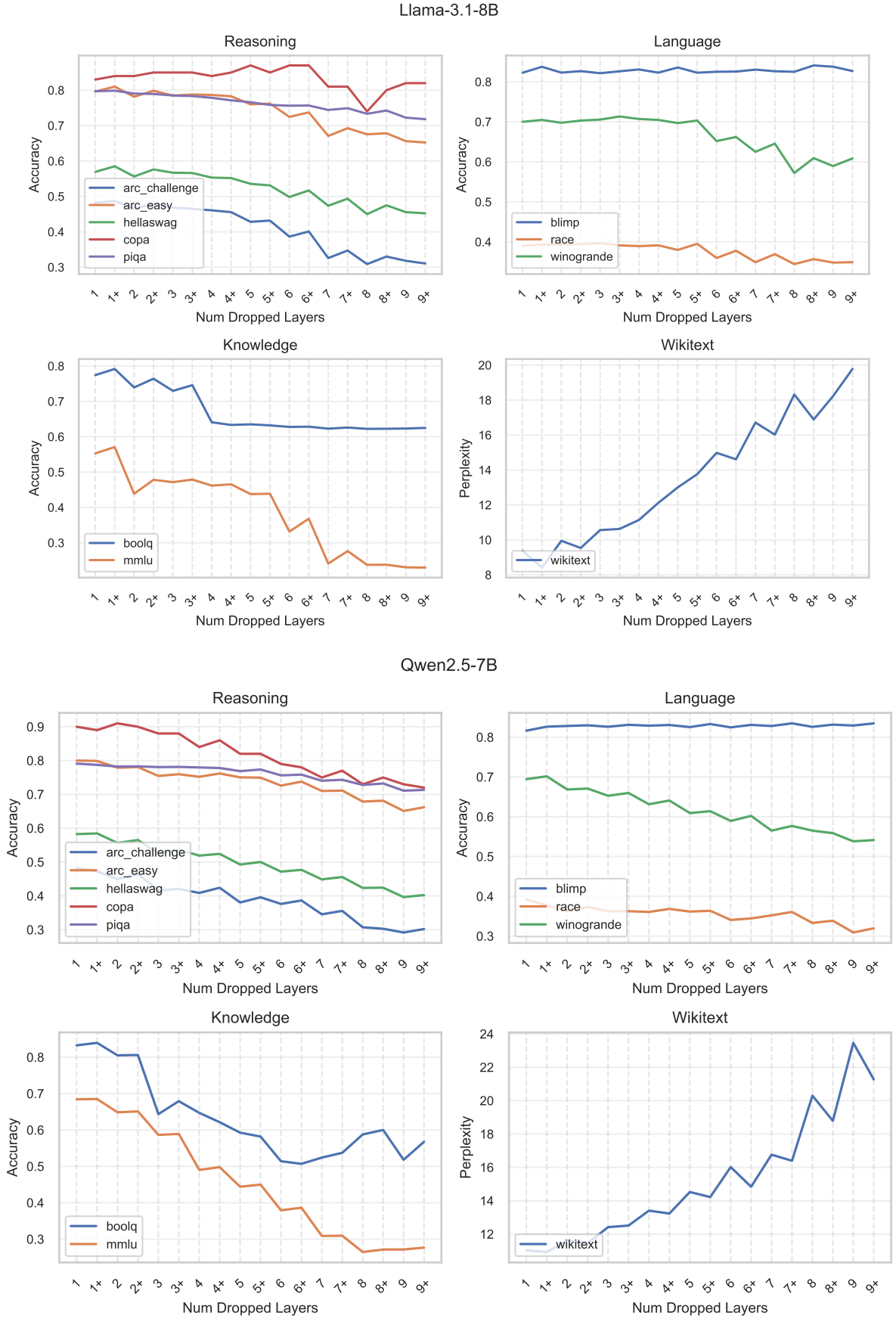

Figure 6: The performance across pruning and recovery phase for 10 iterations in Qwen2.5-7B and Llama3.1-8B.

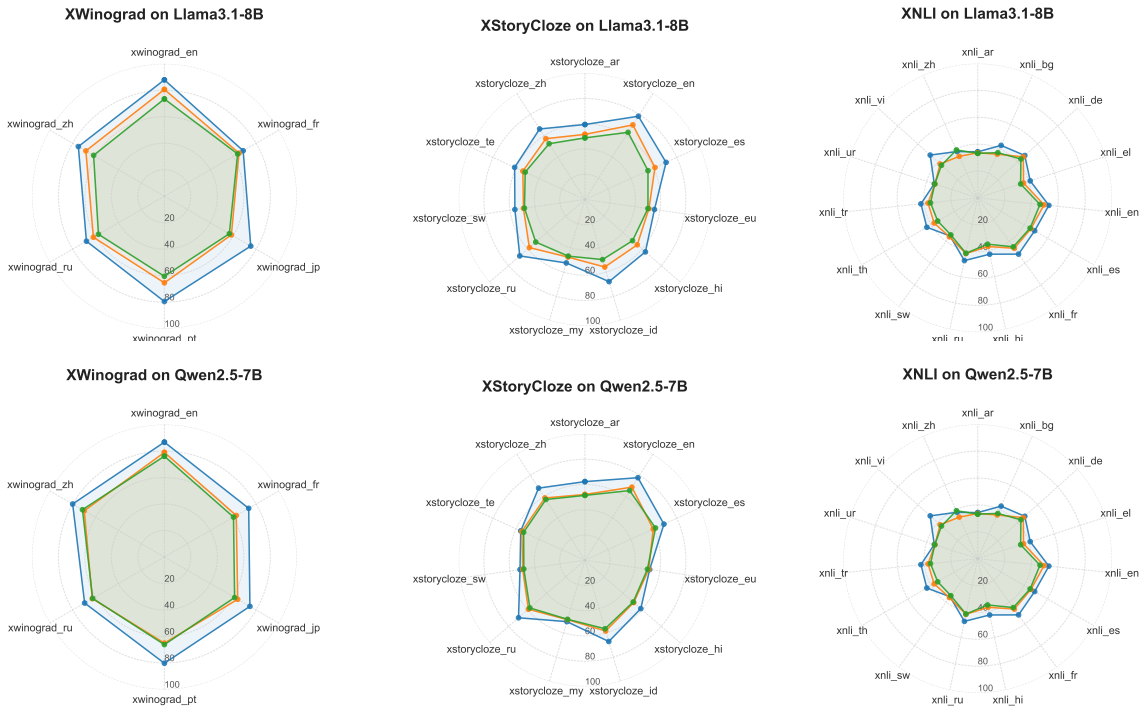

Figure 7: Granular multilingual performance (Accuracy) across XWinograd, XStoryCloze and XNLI on Qwen2.5-7B. , , and denotes result of non-pruned model, IteRABRe with recovery phase, and IteRABRe without recovery phase, respectively.

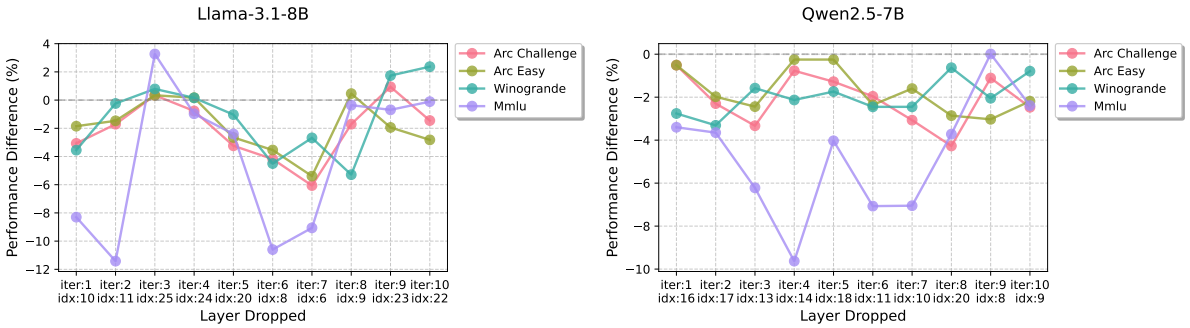

Figure 8: The performance differences between before and after two phases done for each iteration (`iter`) in IteRABRe on LLAMA 3-1-8B and Qwen 2.5-7B. idx denoted the index of the dropped layer (starts from 0).