Multilingual Iterative Model Pruning: What Matters?

Anonymous ACL submission

Abstract

Pruning techniques have been studied to construct small models for efficiency, yet the effect of cross-lingual, which shows language performance transferability, is understudied in this field. In this work, we investigate cross-lingual effects in multilingual large language model compression using iterative pruning and recovery. We employ structured layer pruning with LoRA-based recovery and knowledge distillation, testing whether calibration languages different from target evaluation languages can preserve multilingual performance. Experiments on Qwen2.5-7B and Llama3.1-8B demonstrate that any recovery language consistently outperforms no-recovery baselines, with even low-resource languages like Swahili providing 5% improvements. In contrast to expectations, dominant pretraining languages do not always yield the best results, where Indonesian achieves the highest performance in Llama3.1-8B, while Japanese performs the best in Qwen2.5-7B. Our findings reveal that cross-lingual calibration effectively maintains multilingual capabilities in the iterative pruning.

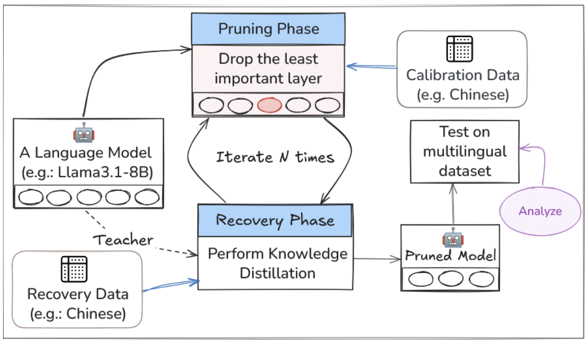

Figure 1: Example iterative pruning using zh (Chinese) as the calibration and recovery dataset where we test the crosslingual capability on different datasets

1 Introduction

Multilingual Large Language Models (LLMs) have proliferated rapidly, creating a need to compress them due to deployment costs. While recent works (Kim et al., 2024; Ushio et al., 2023; Choenni and Titov, 2025) have begun examining multilingual compression, the language in the data used to do the compression process needs further investigation. The impact of the language selection in aiding the process needs further investigation. Specifically, the cross-lingual effects, how the language choice impacts the performance of other languages, remain underexplored. Investigating this behavior would help in reducing both computational and data complexity for compression, for instance, by effectively selecting a language calibration dataset that has better preservation in multilingual performance transfer, particularly in cross-lingual transfer scenarios.

Recent works use an iterative approach to compress LLMs(Muralidharan et al., 2024; Zhang et al., 2024; Li et al., 2022). For instance, Muralidharan et al. (2024) successfully reduced a 15B model to smaller 8B and 4B versions while achieving competitive results compared to other LLMs of similar size. Yet the process remains data-intensive, and the impact of data size is unclear. Moreover, it is unexplored whether the cross-lingual setting can effectively guide recovery in multilingual tasks, specifically in cross-lingual effect.

This leads us to ask: **How is the cross-lingual capability preserved in iterative pruning and recovery phases under resource-constrained scenarios?** To investigate this, we adapt the iterative compression methodology to a more practical setting: we focus on structured pruning through layer removal and employ parameter-efficient recovery using LoRA (Hu et al., 2021) with knowledge distillation using a small data size. Specifically, we examine whether using a language different from the target task for calibration and recovery can retain performance in the tested language while inducing cross-lingual performance preservation.

This study finds that **cross-linguality exists in recovery that consistently outperforms the pruning process without recovery even language that has different script than the tested language**. For instance, using Swahili, a low-resource language, provides better results than without recovery. Interestingly, we observe that language-dominant recovery performs better in iterative pruning; for instance, Chinese (zh) shows superior performance even when the script differs significantly from the target language. Additionally, **different multilingual models exhibit distinct cross-lingual behaviors during compression**. Our experiments on Qwen2.5-7B (Yang et al., 2024a) and Llama3.1-8B (Grattafiori et al., 2024) reveal varying cross-lingual transfer patterns. Each iteration has a different language that performs the best, which is different for each task. In approximately 25% compression rate, we observed that Chinese and Japanese achieve the top-2 highest average performance in Qwen2.5-7B, while in Llama3.1-8B, these are achieved by Indonesian and Chinese, respectively. Surprisingly, in Llama3.1-8B, English ranks only sixth in our experimental results, challenging our assumptions about English's dominance in multilingual compression.

2 Background: Structured Prunning

The increasing scale of LLMs has driven efforts to reduce their computational and memory footprint for deployment. A common 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] Different from most works, in this paper we emphasize on recovery as an important aspect of model compression.

3 Methodology

We do 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), the approach in this paper does the simplified version: (1) **Pruning Phase:** a direct layer-wise pruning strategy and (2) **Recovery Phase:** knowledge distillation using LoRA . We make the iterative pruning more efficient to run with lower resource requirements. Our approach is illustrated in Figure 1.

3.1 Our 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

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

B_i is defined as:

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

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 highest 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 Our 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 $j+1$

4 Experiment Setup

Languages We choose 10 languages as calibration and recovery languages: `zh` (Mandarin Chinese), `ru` (Russian), `id` (Indonesian), `en` (English), `es` (Spanish), `ar` (Arabic), `hi` (Hindi), `ja` (Japanese), `vi` (Vietnamese), and `sw` (Swahili). We selected these languages as they represent diverse language families and writing systems, while covering both high-resource and lower-resource (`sw` and `vi`), allowing us to examine how linguistic similarity and resource availability affect cross-lingual compression performance.

Pruning Calibration For the pruning phase, we use 10 instances as the calibration dataset for each language sampled randomly uniform from wikipedia[3] following the Yang et al., 2024b calibration dataset used. We tried increasing the calibration datasets to 1,000 in §6 and found that it has minimal impact on increasing performance.

Recovery Dataset For the recovery dataset, we target the general data domain, where we used `wikitext-2-raw-v1`[4] for `en` and we created other languages' data by following the number of rows to approximately make the size close to the `en` dataset.

Models We used two widely used LLM families which have multilingual capability, Qwen2.5-7B (Yang et al., 2024a) and Llama3.1-8B (Grattafiori et al., 2024). We use these models to observe their differences in their multilingual behaviors, as they are pre-trained differently, especially in terms of data size.

Evaluation We use common multilingual benchmarks used widely: `pawsx`, `xnli`, `xcopa`, `xstorycloze`, `xwinograd`, and `xquad`.[5] To evaluate our model, we use off-the-shelf `lm-eval-harness` (Gao et al., 2024) library, using their pre-defined metric for each task (F1

[2]See Appendix A for more explanation

[3]We sample uniformly from https://huggingface.co/datasets/wikimedia/wikipedia/

[4]`https://huggingface.co/datasets/Salesforce/wikitext/`

[5]we also denote xstorycloze and xwinograd as `xstory` and `xwino` respectively.

lang	#-L	Llama3.1-8B pawsx	xnli	xcopa	xstory	xwino	xquad	avg	Qwen2.5-7B pawsx	xnli	xcopa	xstory	xwino	xquad	avg
-	0	63.16	45.46	61.69	63.58	81.41	38.78	59.02	59.81	43.44	61.64	62.07	81.52	66.78	62.54
ar	8	52.04	40.78	55.75	55.36	68.04	6.50	46.41	46.99	38.68	55.58	55.43	68.06	11.09	45.97
en	8	50.74	40.54	55.67	55.62	71.86	10.06	47.41	48.10	37.26	55.67	54.50	68.69	5.09	44.88
es	8	51.43	41.05	55.71	55.21	70.40	8.39	47.03	47.14	**39.06**	55.40	54.86	68.44	12.36	46.31
hi	8	53.46	**41.06**	56.35	55.40	70.47	8.08	47.47	47.28	38.55	55.58	54.50	67.59	10.12	45.60
id	8	53.10	40.36	55.44	55.18	**74.15**	**13.27**	**48.58**	47.17	38.54	55.02	55.27	68.08	12.62	46.12
ja	8	51.63	40.82	56.15	55.48	71.12	9.43	47.44	**48.78**	38.62	55.33	54.57	67.77	**13.36**	**46.40**
ru	8	**54.75**	40.82	55.31	55.14	72.33	11.54	48.31	47.48	38.62	55.60	54.71	68.55	12.16	46.19
sw	8	51.97	41.00	55.62	55.04	70.17	8.23	47.00	47.54	38.44	55.44	54.30	66.22	6.75	44.78
vi	8	51.38	39.67	54.91	53.84	71.21	5.42	46.07	48.19	38.50	55.13	54.22	68.35	12.28	46.11
zh	8	52.50	40.67	**56.55**	**55.66**	73.01	12.95	**48.56**	47.24	38.73	**55.85**	**55.48**	**68.98**	12.07	**46.39**
nr	8	48.91	37.33	54.44	51.57	66.22	3.19	43.61	47.49	37.10	55.25	53.53	65.90	5.01	44.05

Table 1: Results in prunning the model using iterative pruning approach. #-L denotes number of pruned layers. These scores for each task are the average across all available tested language in the benchmark. **Bold** denotes the highest performing score or close (less than 0.05% difference) for each task and average. nr denotes iterative pruning without recovery phase.

Figure 2: Best performance in each iteration heatmap in Llama3.1-8B (left) and Qwen2.5-7B across iterations. The language code in each box represents the language that achieves the highest score while below them show the performance score.

or accuracy score). We use the context length of 2,048 tokens and employ the zero-shot setting to obtain the output.

Pruning Setup We do the recovery by doing 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}. For efficient recovery training, we conduct a single epoch on the recovery dataset. The trainings were performed on 1xH200. In §5, we show the 8th pruned iteration as it is approximately 25% the size of Llama layer size, following other works in structure pruning commonly presents (Yang et al., 2024b; Men et al., 2024; Ashkboos et al., 2024; Lin et al., 2024) .

5 Experiment Results

Recovery with any language consistently outperforms non-recovery across all tasks. In Table 1, we observe that recovery using any language other than the target language maintains better performance than without recovery, even with the low-resource languages (e.g., sw and vi). The performance gap between the best recovery method and no-recovery is moderate (2-3%) for most tasks, with the most substantial improvements observed in xwinograd and xquad, where recovery provides 8-10% gains in Llama3.1-8B. These results suggest that recovery with any language, including

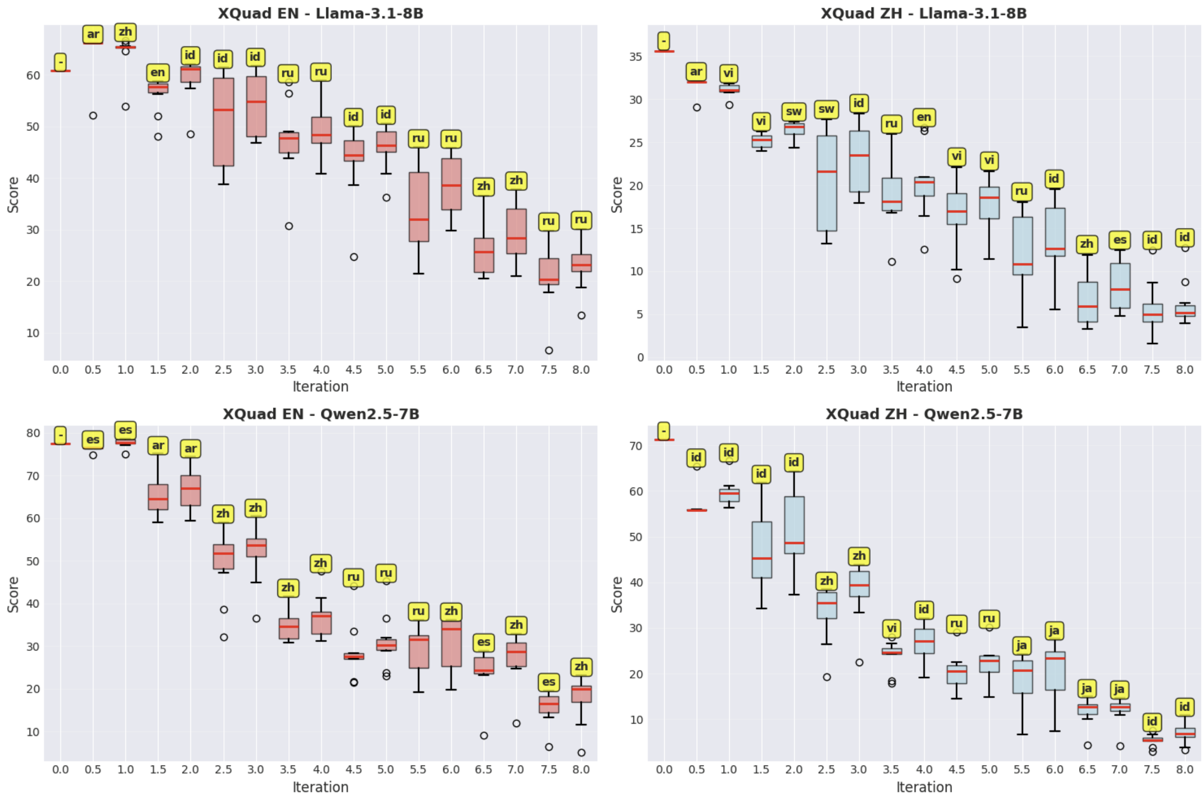

Figure 3: xquad performance on xquad using zh and en calibration and recovery dataset across iterations. The boxplot distribution is the performance across languages. x.5 and x.0 demonstrates the pruning phase and recovery phase performance respectively in each iteration

low-resource languages like sw, yields better results than discarding the recovery phase entirely.

Dominant pretraining languages do not guarantee optimal recovery performance. Contrary to our initial guess that dominant pretraining languages (en for Llama3.1-8B and zh for Qwen2.5-8B) would achieve superior cross-lingual recovery, Table 1 reveals an intriguing patterns. While Qwen2.5-8B shows zh achieving closely (~0.01) to the best average score as predicted, surprisingly, id achieves the best results in Llama3.1-8B, with English ranking only sixth. Notably, zh performs second-best in Llama3.1-8B despite its different script from en. Task-specific patterns further vary between models: ru performs best on pawsx in Llama3.1-8B, while ja excels in Qwen2.5-7B, suggesting model-dependent sensitivity to language-task combinations during pruning.

The best recovery languages vary across pruning iterations. Analysis of performance across the 8-layer pruning process reveals that the best-performing recovery language changes between iterations. Figure 2 illustrates this behavior. For pawsx in Llama, id consistently outperforms other languages in early iterations, while ru performs the best in later stages. Qwen exhibits even more variation, alternating between en, vi, ru, and ja across iterations. Interestingly, xquad shows more stable patterns: en dominates middle iterations (3-7) in Llama, while zh maintains superiority in Qwen, though this consistency does not extend to other tasks.

Language-specific performance patterns emerge across pruning iterations. Having examined aggregated results across languages, we now analyze individual language performance within a single dataset. We focus on xquad as it exhibits the highest variance across languages in both Llama3.1-8B and Qwen2.5-7B models.

In Figure 3, the observation of the performance across iterations for en and zh on xquad in both models shows consistent performance degradation during the pruning phase. The recovery phase demonstrates clear improvements, as shown by upward shifts in the box plot distributions after the pruning phase, indicating that recovery benefits most languages, though performance still declines

Figure 4: Results in xquad on different language using each of language as pruning and recovery dataset tested in language available in xquad for Llama3.1-8B and Qwen2.5-7B. **nr** denotes pruning without recovery. Red border cells depict performance that has less performance than non-recovery. '-' denotes performance of the unpruned models.

Recovery #tokens	#-L	Llama3.1-8B							Qwen2.5-7B						
		pawsx	xnli	xcopa	xstory	xwino	xquad	avg	pawsx	xnli	xcopa	xstory	xwino	xquad	avg
2.5M	4	59.51	43.33	58.71	59.85	78.98	33.80	55.70	52.00	41.38	59.05	58.78	77.55	31.11	53.31
8M	4	59.63	43.30	59.02	59.85	78.89	33.68	55.73	49.94	41.96	58.93	58.87	77.39	31.99	53.18
23.8M	4	59.32	43.43	58.87	59.79	78.92	33.54	55.65	50.31	41.73	59.15	58.75	77.55	29.83	52.88
2.5M	8	50.74	40.54	55.67	55.62	71.86	10.06	47.41	48.10	37.26	55.67	54.50	68.69	5.09	44.88
8M	8	50.21	40.43	55.56	55.53	71.79	9.90	47.24	46.86	37.62	55.44	54.28	68.35	5.86	44.73
23.8M	8	50.33	40.44	55.67	55.44	71.54	9.90	47.22	47.10	37.53	55.60	54.44	68.55	5.56	44.80

Table 2: Performance comparison across different recovery data sizes configurations for Llama3.1-8B and Qwen2.5-7B models, showing accuracy scores (%). #-L denotes number of pruned layers.

with subsequent pruning iterations.

The performance gap between languages widens during the pruning phase, particularly by the third iteration where en-zh performance differs by approximately 20% in Llama3.1-8B and 10% in Qwen on xquad. This suggests that layer importance rankings derived from calibration datasets are language-dependent, where the choice of calibration language influences both task performance and cross-lingual results, with some languages providing better preservation during performance degradation.

Cross-lingual recovery benefits vary significantly across target languages and models. We extend the analysis from Table 1 by examining individual language performance on xquad, as shown in Figure 4.

Most recovery languages outperform the non-recovery baseline, with several exceptions: in Llama, ar, ro, and th underperform when recovering vi performance, and ar fails when recovering Arabic performance. In Qwen, seven languages (de, el, en, es, ru, th, and tr) perform worse than non-recovery when recovering English performance. The fact that en recovery is detrimental for English tasks in Qwen presents an interesting pattern. We observe that **optimal recovery languages do not correspond to the target evaluation language**. For instance, id achieves the best results for xquad_ar rather than using ar for recovery. Additionally, zh effectively maintains English performance despite having a different script system. Consistent with Table 1, id and ja exhibit top performers across multiple target language benchmarks in the cross-lingual recovery setting.

6 Analysis in Calibration and Recovery Dataset Setup

To ascertain our experiment setup, we check the impact of the sizes of calibration and recovery datasets, with the addition of using all languages instead of a language in pruning and recovering the model in the general domain.

Calibration and recovery dataset size shows minimal impact on performance. We examine whether dataset size affects model performance dur-

#Calibration Rows	#-L	Llama3.1-8B							Qwen2.5-7B						
		pawsx	xnli	xcopa	xstory	xwino	xquad	avg	pawsx	xnli	xcopa	xstory	xwino	xquad	avg
10	4	59.04	43.32	58.75	59.85	78.92	33.34	55.53	49.83	41.98	59.13	58.88	77.55	32.45	53.30
100	4	59.60	43.35	58.82	59.82	78.89	33.80	55.71	50.09	41.98	59.05	58.89	77.61	32.45	53.35
1000	4	59.50	43.37	58.69	59.81	78.92	33.81	55.68	50.83	40.27	58.96	58.97	78.58	30.43	53.01
10	8	50.51	40.47	55.75	55.47	71.63	10.18	47.34	50.00	37.42	55.60	53.35	68.33	5.29	45.00
100	8	50.34	40.40	55.82	55.48	71.99	9.81	47.31	50.55	37.40	55.69	53.38	67.88	4.70	44.93
1000	8	50.36	40.46	55.75	55.58	71.63	9.98	47.29	47.29	37.44	55.98	54.37	67.81	2.88	44.30

Table 3: Performance comparison across different calibration pruning data sizes and number of layer pruning configurations for each model, showing the respective scores. Results are shown for pruning sizes of 10, 100, and 1000 with both 4 and 8 pruned layers. #-L denotes number of pruned layers.

Training Data Type	#-L	Llama3.1-8B							Qwen2.5-7B						
		pawsx	xnli	xcopa	xstory	xwino	xquad	avg	pawsx	xnli	xcopa	xstory	xwinogr	xquad	avg
Mixed	4	59.26	44.06	59.78	60.44	77.64	20.94	53.68	52.27	41.33	58.60	58.55	77.55	27.51	52.63
En	4	59.51	43.33	58.71	59.85	78.98	33.80	55.70	52.00	41.38	59.05	58.78	77.55	31.11	53.31
Mixed	8	53.48	41.16	55.93	55.95	71.59	6.59	47.45	47.21	38.82	55.64	54.70	68.96	9.83	45.86
En	8	50.74	40.54	55.67	55.62	71.86	10.06	47.41	49.83	39.23	56.47	55.82	72.51	16.42	48.38

Table 4: Performance comparison across different training data types that mixes all language (mixed) and english only (en). Results are shown for both 4 and 8 pruned layers with different training data compositions. #-L denotes number of pruned layers.

ing compression. Table 2 shows that on average, different data sizes yield similar results across iterations, indicating that dataset size does not impact much under our experimental setups.

We also investigate calibration dataset size for the pruning phase, given models' sensitivity to layer removal decisions. Table 3 demonstrates minimal differences across dataset sizes, with the exception of xquad tasks in both Llama and Qwen at the 8th iteration, where slight performance degradation occurs. To conclude, larger pruning datasets do not consistently correspond to improved performance.

Mixed-language data shows model-dependent results but generally underperforms monolingual English on some xquad and xwinograd. Previous experiments used single languages for recovery and pruning. We investigate whether combining all languages into mixed datasets affects performance, maintaining dataset sizes comparable to the English monolingual condition. Results are presented in Table 4.

For Llama, mixed-language data shows slightly better average results than English on `pawsx`, `xnli`, and `xcopa` tasks. Qwen exhibits the opposite pattern on these same tasks. For `xwinograd` and `xquad`, both models show that English outperforms mixed-language data on average. Overall, results indicate that monolingual English is either comparable to or better than mixed-language datasets across most experimental setups.

7 Comparison to Non-Iterative Approaches

So far, we have shown the multilingual capability. However, to ascertain the iterative pruning method effectiveness, we need to compare it to other non-iterative methods. To do so, we compare it with two baseline layer pruning methods: LaCO (Yang et al., 2024b) and ShortGPT (Men et al., 2024). We check the performance only in English tasks using English calibration and recovery dataset.

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. 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`. We then used the experiment setup as defined in §4. To have a better assessment, we categorize the benchmark dataset into three categories: reasoning (`arc-challenge`,`arc-easy` (Clark et al., 2018), `hellaswag` (Zellers et al., 2019), `COPA` (Roemmele

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**
	Ours	24	**16.89**	**33.02**	**67.85**	**47.49**	80.0	**74.27**	**84.10**	**35.69**	**60.93**	**62.26**	23.80
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
	Ours	21	**16.40**	**35.58**	**71.13**	**45.59**	**77.0**	**74.32**	**83.48**	**36.08**	**57.70**	**53.73**	30.94

Table 5: 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, ours) 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.

et al., 2011), PIQA (Bisk et al., 2020)), language comprehension (BLiMP (Warstadt et al., 2020), RACE (Lai et al., 2017), and Winogrande (Sakaguchi et al., 2021)), and knowledge (BoolQ (Clark et al., 2019) and MMLU (Hendrycks et al., 2021)).

Iterative approach Outperforms Other Baselines Overall Table 5 presents the experimental results. The iterative pruning outperforms other methods (LaCO and ShortGPT) across all model scales. Specifically, it 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). The iterative pruning also achieves the highest performance in the reasoning domain.

In the language category, our approach maintains performance better than the other methods, particularly on BLIMP, where these models 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%). These results suggest that our method offers particular advantages for language comprehension in large models.

In the knowledge domain, iterative prunning achieves strong BoolQ performance. 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 by 1-2% for the others.

8 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 a 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 recovery, critical to our work, to study whether multilingual capabilities can be retained or not.

9 Conclusion

This work analyzes the cross-lingual performance in iterative pruning in a multilingual model. We found that iterative pruning induces cross-linguality even using a different language than the original compared to without recovery. Additionally, each iteration has different language that performs the best. Our findings demonstrate an intriguing aspect related to cross-linguality in iterative pruning.

Limitations

We acknowledge the limitations in our experimental setups that we only tested ten languages. More languages may have enriched the analysis performed in this research. Additionally, we only observe the Qwen2.5 and Llama3 models, where other models may exhibit different patterns, as we have pointed out in our results that each model exhibits different behavior. Finally, we only test the data in general data for each language. Having specific task-oriented data or language, while also additional sampling techniques, may be worth pursuing for future works.

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 Overleaf's AI) 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, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, pages 7432–7439.

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 and Ivan Titov. 2025. M-wanda: Improving one-shot pruning for multilingual llms. *Preprint*, arXiv:2505.21171.

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. *Preprint*, arXiv:1803.05457.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural architecture search: A survey. *Preprint*, arXiv:1808.05377.

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

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, and 5 others. 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, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

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. *Preprint*, arXiv:2309.09298.

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. *Preprint*, arXiv:1503.02531.

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. *Preprint*, arXiv:2106.09685.

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.

Hwichan Kim, Jun Suzuki, Tosho Hirasawa, and Mamoru Komachi. 2024. Pruning multilingual large language models for multilingual inference. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9921–9942, Miami, Florida, USA. 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. *Preprint*, arXiv:2105.08919.

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.

Yuchao Li, Fuli Luo, Chuanqi Tan, Mengdi Wang, Songfang Huang, Shen Li, and Junjie Bai. 2022. Parameter-efficient sparsity for large language models fine-tuning. *Preprint*, arXiv:2205.11005.

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. *Preprint*, arXiv:2408.09632.

Gui Ling, Ziyang Wang, Yuliang Yan, and Qingwen Liu. 2024. Slimgpt: Layer-wise structured pruning for large language models. *Preprint*, arXiv:2412.18110.

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. *Preprint*, arXiv:2407.00928.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *Preprint*, arXiv:2305.11627.

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. *Preprint*, arXiv:2403.03853.

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. *Preprint*, arXiv:2407.14679.

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. *Preprint*, arXiv:2306.11695.

Asahi Ushio, Yi Zhou, and Jose Camacho-Collados. 2023. Efficient multilingual language model compression through vocabulary trimming. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14725–14739, Singapore. Association for Computational Linguistics.

Tycho F. A. van der Ouderaa, Markus Nagel, Mart van Baalen, Yuki M. Asano, and Tijmen Blankevoort. 2024. The llm surgeon. *Preprint*, arXiv:2312.17244.

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. *Preprint*, arXiv:2310.06694.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 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. *Preprint*, arXiv:2310.05175.

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

Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. 2024. Loraprune: Structured pruning meets low-rank parameter-efficient fine-tuning. *Preprint*, arXiv:2305.18403.

A Layer Mapping in Recovery Phase

To define the mapping function $\text{map}(l)$ 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, $\text{map}(l)$ is adjusted by increasing it by the number of dropped layers with indices less than $\text{map}(l)$. For example, if the dropped layer indices are $[3, 4]$ and $l = 10$, then $\text{map}(10) = 12$, 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 $\text{map}(l)$ maps the student's layer index l to the teacher's original index m, where m is the unique solution to the equation $m = l + |\{d \in D \mid d < m\}|$.

B Additional Monolingual Performance Analysis

Iterative Prunning's recovery phase boosts performance, notably for larger models on reasoning and language tasks. We investigated the impact of each phase of Iterative Prunning. The results are shown in Figure 15. 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.

Language	#Tokens
ar	8.5m
en	2.5m
es	6.0m
hi	9.0m
id	6.2m
ja	8.3m
ru	7.2m
sw	3.3m
vi	6.2m
zh	7.6m

Table 6: Recovery dataset size in token size computed using Llama3.1 tokenizer.

C Iterative Prunning Preservation Analysis

Iterative Prunning effectively preserves language and reasoning abilities across iterations, though knowledge retention presents a challenge. Figure 17 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 Iterative Pruning's effectiveness in preserving language and reasoning abilities, though it suggests challenges in maintaining knowledge-based performance across iterations.

D Recovery Data Token Size

The number of tokens used in our experiments can be seen in Table 6.

E Performance Trend for Each Iteration

The fine-grained performance trend on monolingual performance can be seen in Figure 5.

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 17). 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.

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

Iterative Prunning 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 E 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, Iterative Pruning 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 16 provides the MMLU performance across MMLU groupings. [6] It shows that the pruning phase induces significant performance drops in some cases, notably in the early layer dropping of Llama3.1-8B (around 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.

Our approach exhibits task-specific layer sensitivities that vary between models. We investigated which layer drops correlate with significant performance declines, indicating layer importance. Figure 6 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 iterations 1, 6, and 7, and on `winogrande` in iterations 1, 6, and 8. MMLU on Llama3.1-8B shows steep declines in iterations 10 and 11 during early iterations, followed by improvement and stagnation. Qwen2.5-7B exhibits different trends, with notable (>5%) decreases on MMLU in iterations 3, 4, 6, and 7.

F Scores across iterations

Table 7 and 8 to Table 21 and 22 show the performance of multilingual iterative pruning across tasks in Llama3.1-8B and Qwen2.5-7B, respectively. Additionally, each iteration performance across multilingual tasks can be seen in Figure 7,8,9,10,11,11,12,13, and 14.

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

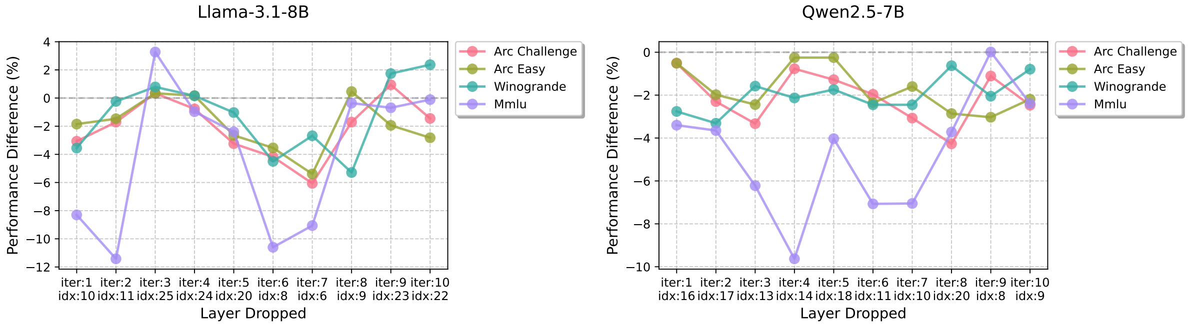

Figure 6: The performance differences between before and after two phases done for each iteration (iter) on LLAMA 3-1-8B and Qwen 2.5-7B. idx denoted the index of the dropped layer (starts from 0).

lang	pawsx	xnli	xcopa	xstorycloze	xwinograd	xquad	avg
ar	61.99	45.12	60.76	62.82	80.31	39.79	58.47
en	62.19	45.13	60.95	62.72	80.53	40.35	58.64
es	62.01	45.18	60.74	62.87	80.62	38.86	58.38
hi	61.30	45.45	61.38	62.58	79.79	33.16	57.28
id	62.39	45.17	60.96	63.06	80.69	39.64	58.65
ja	62.08	45.13	60.93	62.84	80.53	40.00	58.58
ru	61.79	45.07	60.76	62.98	80.56	38.84	58.33
sw	61.85	45.03	60.80	62.87	80.56	39.36	58.41
vi	61.79	45.12	60.85	62.95	80.62	40.03	58.56
zh	62.04	45.22	60.85	62.91	80.85	40.99	58.81

Table 7: Llama-3.1-8B results at iteration 1.

lang	pawsx	xnli	xcopa	xstorycloze	xwinograd	xquad	avg
ar	58.78	43.31	61.33	61.15	80.67	62.40	61.27
en	59.25	43.28	61.18	61.19	81.14	60.70	61.12
es	59.12	43.22	61.05	61.15	80.65	62.30	61.25
hi	58.56	43.21	61.18	61.16	80.60	62.45	61.19
id	58.03	43.52	61.40	61.46	81.43	61.06	61.15
ja	58.67	43.30	61.26	61.09	80.72	61.67	61.12
ru	58.62	43.15	61.27	61.10	80.72	61.89	61.13
sw	58.64	43.18	61.26	60.99	80.76	60.62	60.91
vi	58.73	43.24	60.91	61.03	80.81	62.62	61.22
zh	58.88	43.19	61.15	61.17	81.23	61.58	61.20

Table 8: Qwen2.5-7B results at iteration 1.

lang	pawsx	xnli	xcopa	xstorycloze	xwinograd	xquad	avg
ar	59.82	44.63	60.73	61.64	78.24	33.88	56.49
en	61.24	44.73	61.16	61.65	80.24	36.07	57.51
es	61.20	44.78	61.15	61.77	80.17	34.89	57.33
hi	59.73	44.23	60.76	61.31	77.95	28.73	55.45
id	61.56	44.77	61.20	61.85	80.04	35.44	57.48
ja	60.27	45.15	60.55	61.84	79.14	32.08	56.50
ru	61.21	44.59	61.15	61.77	79.88	34.42	57.17
sw	60.23	44.51	61.13	61.92	80.40	35.22	57.24
vi	61.42	44.64	60.98	61.97	80.06	34.67	57.29
zh	59.78	44.79	60.80	61.66	78.49	35.85	56.89

Table 9: Llama-3.1-8B results at iteration 2.

lang	pawsx	xnli	xcopa	xstorycloze	xwinograd	xquad	avg
ar	57.61	42.99	60.93	60.41	78.67	57.83	59.74
en	56.24	43.07	60.53	60.56	80.11	50.58	58.51
es	56.36	43.19	60.60	60.74	79.97	52.05	58.82
hi	58.14	42.38	60.76	60.57	79.61	50.52	58.66
id	57.36	43.28	60.40	60.54	80.27	57.90	59.96
ja	58.17	42.81	60.38	59.78	79.82	56.98	59.66
ru	58.03	42.01	60.74	60.38	79.79	46.82	57.96
sw	54.86	43.06	60.78	60.31	78.87	49.34	57.87
vi	58.56	43.01	60.18	60.02	80.06	57.97	59.97
zh	55.46	42.92	60.93	60.57	80.56	52.36	58.80

Table 10: Qwen2.5-7B results at iteration 2.

lang	pawsx	xnli	xcopa	xstorycloze	xwinograd	xquad	avg
ar	58.16	44.10	59.74	60.28	76.65	24.21	53.86
en	58.98	43.94	60.20	60.92	79.21	34.89	56.35
es	59.14	43.96	60.31	61.00	78.74	33.49	56.11
hi	58.27	44.20	60.34	60.67	76.78	24.97	54.21
id	62.56	44.48	60.13	60.78	79.59	35.09	57.11
ja	57.79	44.25	59.80	60.31	77.14	24.93	54.04
ru	60.98	43.99	60.04	60.65	79.77	33.23	56.44
sw	61.04	44.45	60.00	60.91	79.86	34.72	56.83
vi	59.46	44.27	60.34	60.76	77.72	23.86	54.40
zh	57.87	44.42	59.87	60.51	76.89	24.88	54.07

Table 11: Llama-3.1-8B results at iteration 3.

lang	pawsx	xnli	xcopa	xstorycloze	xwinograd	xquad	avg
ar	57.16	41.77	60.29	59.92	77.88	39.54	56.09
en	54.49	42.86	60.02	59.60	78.78	44.29	56.67
es	52.94	42.21	59.76	59.70	79.39	38.55	55.43
hi	56.57	41.02	59.89	59.60	78.44	41.07	56.10
id	54.82	42.25	59.66	59.73	79.43	45.09	56.83
ja	55.79	42.32	59.51	59.28	79.73	46.49	57.19
ru	57.49	41.60	59.58	59.05	78.74	41.56	56.34
sw	50.23	41.39	59.49	59.36	78.22	33.82	53.75
vi	56.79	42.23	59.45	59.23	79.48	47.27	57.41
zh	54.61	42.69	59.82	59.63	79.19	48.46	57.40

Table 12: Qwen2.5-7B results at iteration 3.

lang	pawsx	xnli	xcopa	xstorycloze	xwinograd	xquad	avg
ar	55.57	43.87	59.64	59.44	75.25	16.61	51.73
en	59.51	43.33	58.71	59.85	78.98	33.80	55.70
es	56.59	43.50	59.78	59.65	76.67	23.30	53.25
hi	58.21	44.13	59.05	59.75	76.40	23.99	53.59
id	59.89	44.10	59.74	59.88	77.41	25.02	54.34
ja	56.35	43.33	59.09	59.53	76.98	23.48	53.13
ru	61.02	43.05	58.07	59.47	78.78	31.93	55.39
sw	58.63	43.66	59.73	59.56	77.55	24.40	53.92
vi	57.43	43.55	59.31	59.70	77.25	21.39	53.10
zh	56.64	43.68	59.11	59.92	76.26	23.21	53.14

Table 13: Llama-3.1-8B results at iteration 4.

lang	pawsx	xnli	xcopa	xstorycloze	xwinograd	xquad	avg
ar	55.51	40.60	59.34	58.47	76.24	28.81	53.16
en	52.00	41.38	59.05	58.77	77.55	31.11	53.31
es	53.25	41.38	59.04	58.83	77.55	31.04	53.51
hi	54.94	40.74	58.78	58.72	77.14	31.87	53.70
id	54.69	41.16	58.82	58.53	78.62	35.52	54.56
ja	55.31	41.04	59.05	58.34	77.86	31.23	53.81
ru	55.69	40.71	58.82	58.28	77.64	32.07	53.87
sw	49.41	41.49	58.47	58.16	76.62	30.44	52.43
vi	55.94	40.80	58.58	58.30	77.32	31.61	53.76
zh	51.34	41.75	59.15	58.84	78.51	36.33	54.32

Table 14: Qwen2.5-7B results at iteration 4.

lang	pawsx	xnli	xcopa	xstorycloze	xwinograd	xquad	avg
ar	55.81	43.36	58.60	58.89	74.69	16.80	51.36
en	57.06	42.83	58.29	58.79	76.49	23.91	52.90
es	55.80	42.67	58.07	58.49	75.70	22.81	52.26
hi	55.01	43.73	58.67	58.70	73.97	15.82	50.99
id	59.41	43.46	58.45	58.61	76.87	23.52	53.39
ja	56.68	42.78	58.56	58.84	76.40	23.24	52.75
ru	59.12	42.77	57.67	58.76	76.58	22.27	52.86
sw	58.16	43.09	58.93	58.62	77.12	23.13	53.18
vi	56.84	41.69	58.09	58.23	76.69	18.93	51.75
zh	57.06	43.12	58.58	58.96	75.84	23.26	52.80

Table 15: Llama-3.1-8B results at iteration 5.

lang	pawsx	xnli	xcopa	xstorycloze	xwinograd	xquad	avg
ar	49.20	40.27	58.76	57.90	74.80	16.57	49.58
en	51.41	40.05	58.44	57.88	76.17	24.07	51.34
es	51.66	41.05	58.24	57.65	75.93	26.28	51.80
hi	49.50	40.83	58.20	57.85	75.45	26.02	51.31
id	50.51	40.90	57.95	57.72	75.77	28.11	51.83
ja	50.67	41.13	58.09	57.32	75.95	27.44	51.77
ru	54.44	41.02	57.56	57.27	75.12	33.24	53.11
sw	48.47	40.38	57.71	56.85	74.17	21.73	49.89
vi	51.36	40.71	58.27	57.34	76.02	25.95	51.61
zh	50.74	40.97	57.80	57.52	76.29	26.91	51.70

Table 16: Qwen2.5-7B results at iteration 5.

lang	pawsx	xnli	xcopa	xstorycloze	xwinograd	xquad	avg
ar	54.16	42.30	57.36	57.20	71.66	10.80	48.91
en	55.75	41.83	57.09	57.48	75.79	23.41	51.89
es	52.48	42.46	57.66	57.56	73.68	16.19	50.00
hi	55.23	43.64	57.76	57.72	73.88	14.81	50.51
id	56.17	42.37	57.18	57.53	76.24	20.75	51.71
ja	53.16	42.01	58.04	57.90	74.85	18.61	50.76
ru	59.16	41.92	56.40	57.42	76.13	20.37	51.90
sw	56.25	41.95	57.31	57.19	74.31	13.73	50.12
vi	53.40	41.74	57.60	57.30	74.69	12.42	49.53
zh	55.77	42.37	57.67	57.76	75.12	19.73	51.40

Table 17: Llama-3.1-8B results at iteration 6.

lang	pawsx	xnli	xcopa	xstorycloze	xwinograd	xquad	avg
ar	48.53	40.26	57.84	56.91	72.38	16.62	48.75
en	51.64	39.37	57.33	56.57	74.67	16.91	49.41
es	51.77	39.75	57.51	56.72	74.56	18.88	49.87
hi	49.16	40.59	57.02	56.68	73.05	24.98	50.25
id	49.53	40.68	57.24	57.01	73.90	27.05	50.90
ja	49.83	40.66	57.22	56.65	73.64	27.90	50.98
ru	53.02	40.22	56.13	56.11	72.33	19.50	49.55
sw	48.94	39.48	56.89	55.98	70.60	12.64	47.42
vi	50.89	40.20	56.74	56.58	73.90	24.90	50.54
zh	49.53	40.82	56.91	56.73	74.67	26.63	50.88

Table 18: Qwen2.5-7B results at iteration 6.

lang	pawsx	xnli	xcopa	xstorycloze	xwinograd	xquad	avg
ar	54.39	41.68	56.47	56.19	71.00	10.22	48.32
en	52.99	41.05	56.05	56.37	74.26	15.33	49.34
es	52.50	41.83	56.74	56.35	73.19	14.94	49.26
hi	53.20	42.11	56.56	56.44	70.80	8.93	48.01
id	54.05	41.38	55.66	56.36	74.58	13.89	49.32
ja	52.22	41.27	57.07	56.43	71.79	11.44	48.37
ru	55.31	41.82	55.89	56.49	73.81	14.07	49.57
sw	51.33	42.19	56.16	55.95	70.69	8.96	47.55
vi	52.04	40.64	56.66	55.83	71.79	6.79	47.29
zh	55.28	41.49	56.82	56.42	74.56	18.46	50.50

Table 19: Llama-3.1-8B results at iteration 7.

lang	pawsx	xnli	xcopa	xstorycloze	xwinograd	xquad	avg
ar	47.47	39.11	56.53	55.81	71.00	14.22	47.36
en	49.83	39.23	56.47	55.82	72.51	16.42	48.38
es	49.52	39.76	56.47	55.95	71.84	17.32	48.48
hi	48.76	39.54	56.45	55.91	71.54	15.52	47.95
id	48.71	39.37	56.47	56.29	72.20	17.56	48.43
ja	50.71	39.52	56.49	55.70	71.70	18.91	48.84
ru	48.86	39.39	55.98	55.50	69.59	16.86	47.70
sw	47.73	38.24	56.00	55.15	68.91	7.47	45.59
vi	49.99	39.12	55.66	55.16	72.58	15.76	48.04
zh	49.59	39.44	56.18	55.99	72.78	19.91	48.98

Table 20: Qwen2.5-7B results at iteration 7.

lang	pawsx	xnli	xcopa	xstorycloze	xwinograd	xquad	avg
ar	52.04	40.78	55.74	55.36	68.04	6.50	46.41
en	50.74	40.54	55.67	55.62	71.86	10.06	47.41
es	51.43	41.05	55.71	55.21	70.40	8.39	47.03
hi	53.46	41.06	56.34	55.40	70.47	8.08	47.47
id	53.10	40.36	55.44	55.18	74.15	13.27	48.58
ja	51.63	40.82	56.15	55.48	71.12	9.43	47.44
ru	54.75	40.82	55.31	55.13	72.33	11.54	48.31
sw	51.97	41.00	55.62	55.04	70.17	8.23	47.00
vi	51.38	39.67	54.91	53.84	71.21	5.42	46.07
zh	52.50	40.67	56.55	55.66	73.00	12.95	48.56

Table 21: Llama-3.1-8B results at iteration 8.

lang	pawsx	xnli	xcopa	xstorycloze	xwinograd	xquad	avg
ar	46.99	38.68	55.58	55.43	68.06	11.09	45.97
en	48.10	37.26	55.67	54.50	68.69	5.09	44.88
en-nr	47.49	37.10	55.26	53.53	65.90	5.01	44.05
es	47.74	39.06	55.40	54.86	68.44	12.36	46.31
hi	47.28	38.55	55.58	54.50	67.59	10.12	45.60
id	47.17	38.53	55.02	55.27	68.08	12.62	46.12
ja	48.78	38.62	55.33	54.57	67.77	13.36	46.40
ru	47.48	38.62	55.60	54.71	68.56	12.16	46.19
sw	47.54	38.44	55.44	54.30	66.22	6.75	44.78
vi	48.19	38.50	55.13	54.22	68.35	12.28	46.11
zh	47.24	38.73	55.85	55.48	68.98	12.07	46.39

Table 22: Qwen2.5-7B results at iteration 8.

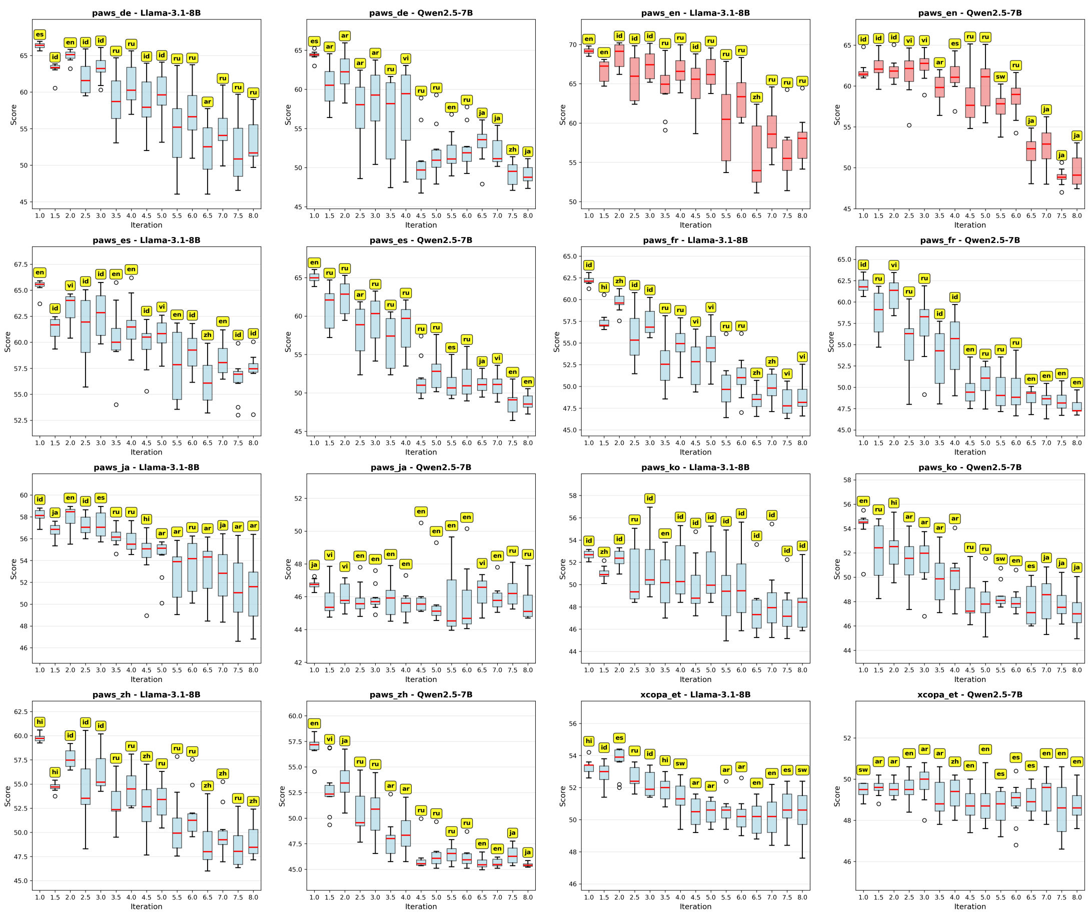

Figure 7: Boxplots showing the distribution of scores across iterations for different models and tasks. Each boxplot represents the score distribution for a specific task and model combination, with the best language annotated for each iteration. Note that the min and max values in y-axis are adjusted and different for each task.

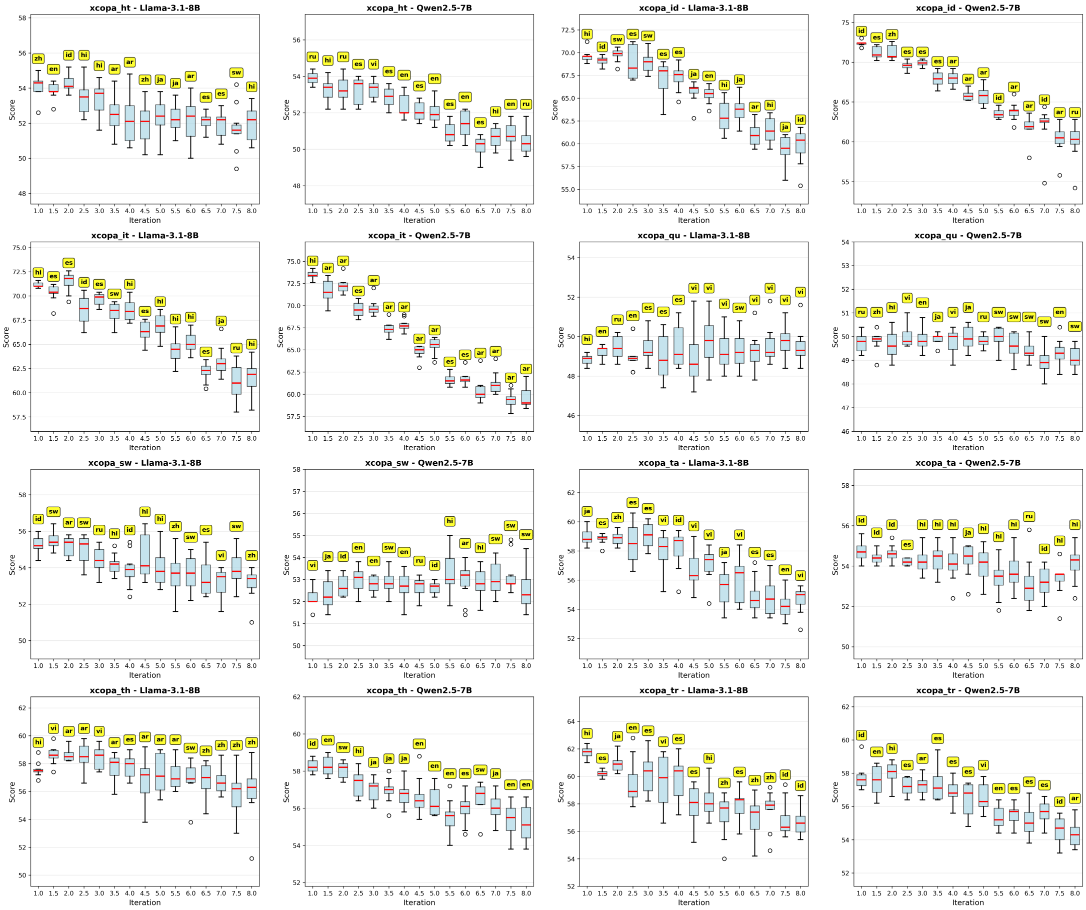

Figure 8: Boxplots showing the distribution of scores across iterations for different models and tasks. Each boxplot represents the score distribution for a specific task and model combination, with the best language annotated for each iteration. Note that the min and max values in y-axis are adjusted and different for each task.

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 23: BLiMP Agreement Tests

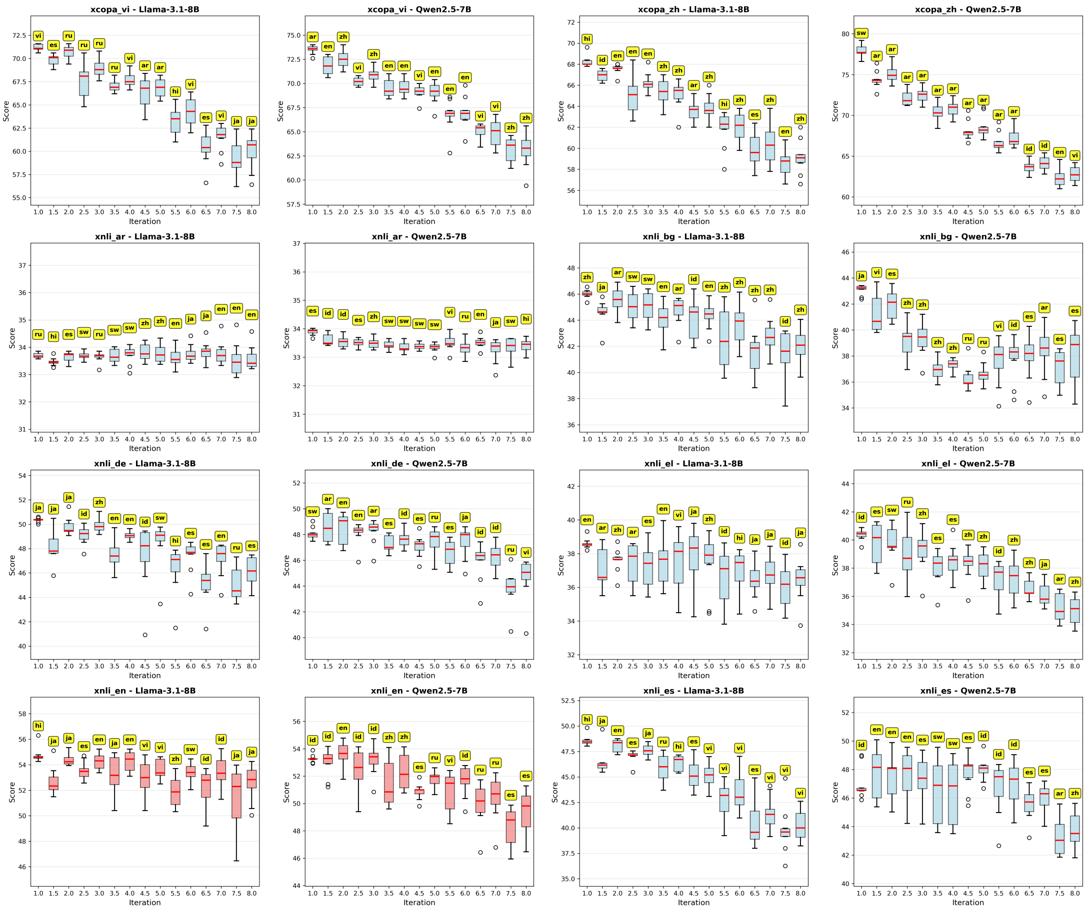

Figure 9: Boxplots showing the distribution of scores across iterations for different models and tasks. Each boxplot represents the score distribution for a specific task and model combination, with the best language annotated for each iteration. Note that the min and max values in y-axis are adjusted and different for each task.

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 24: BLiMP Syntax and Movement Tests

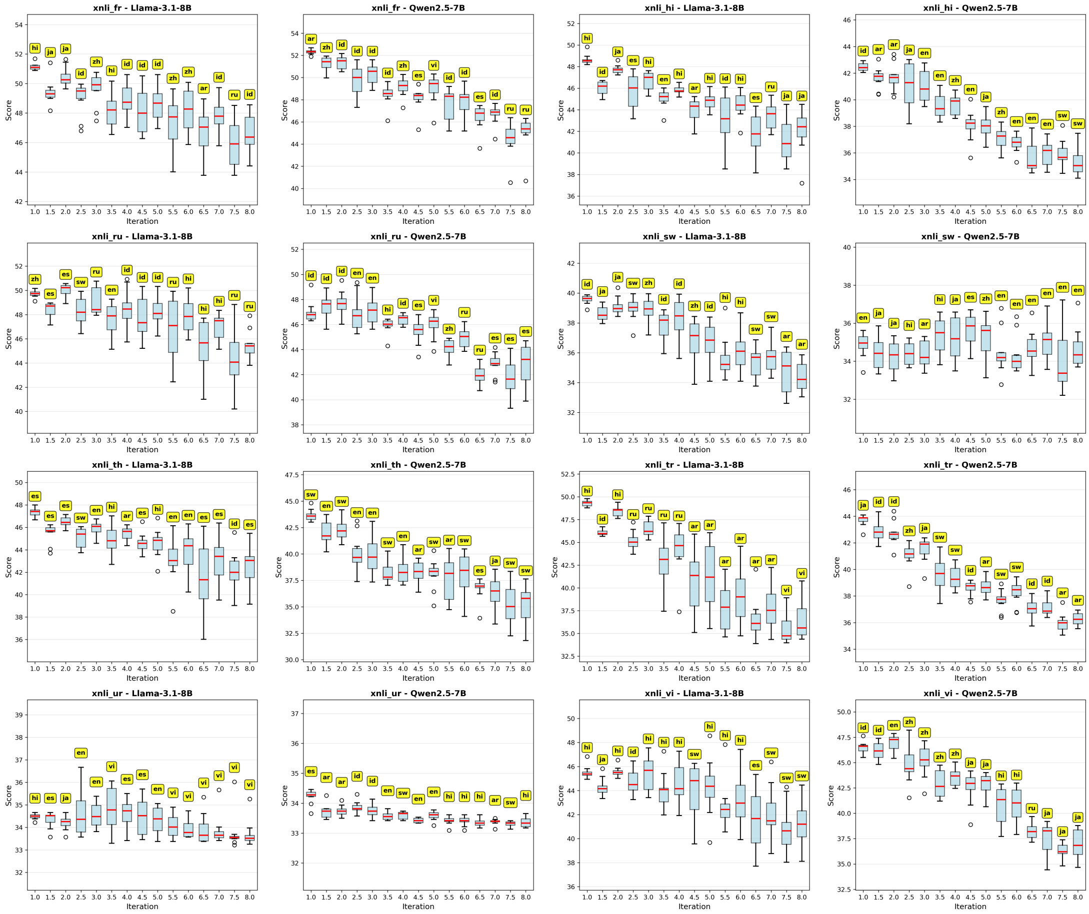

Figure 10: Boxplots showing the distribution of scores across iterations for different models and tasks. Each boxplot represents the score distribution for a specific task and model combination, with the best language annotated for each iteration. Note that the min and max values in y-axis are adjusted and different for each task.

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 25: BLiMP Argument Structure and Form Tests

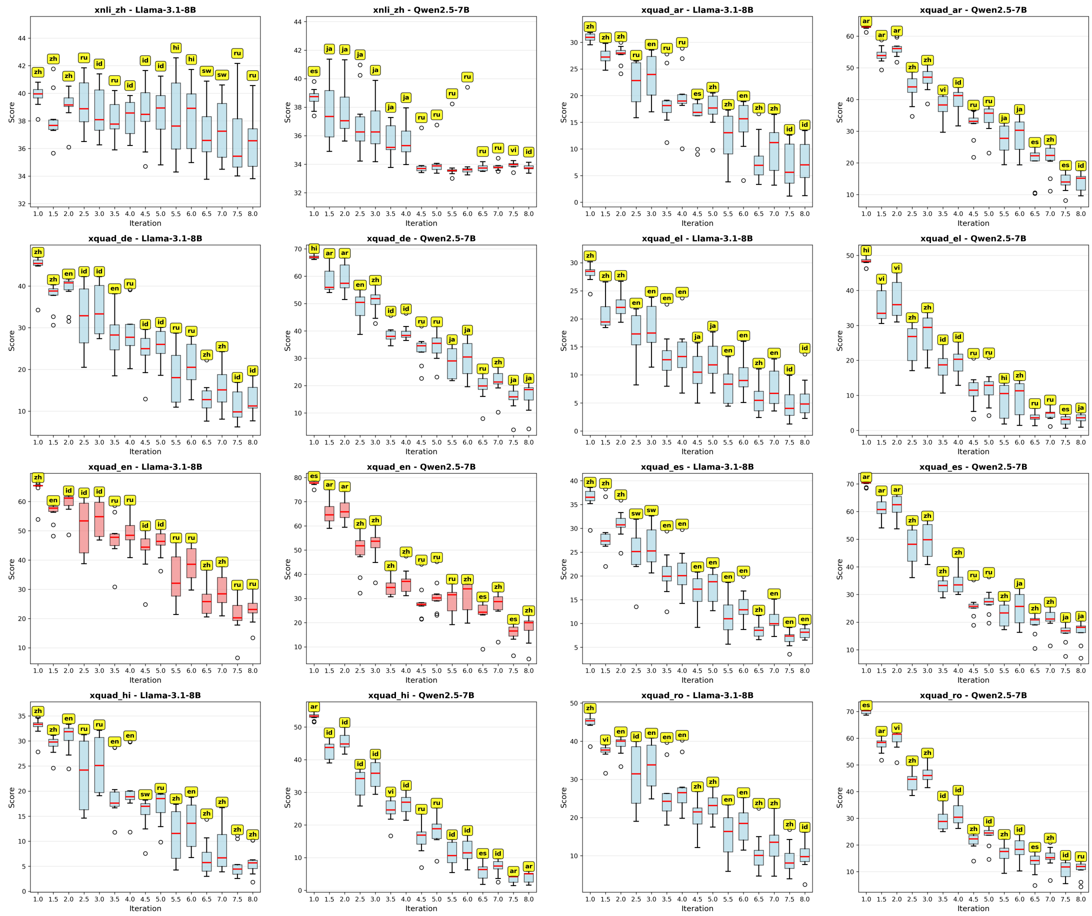

Figure 11: Boxplots showing the distribution of scores across iterations for different models and tasks. Each boxplot represents the score distribution for a specific task and model combination, with the best language annotated for each iteration. Note that the min and max values in y-axis are adjusted and different for each task.

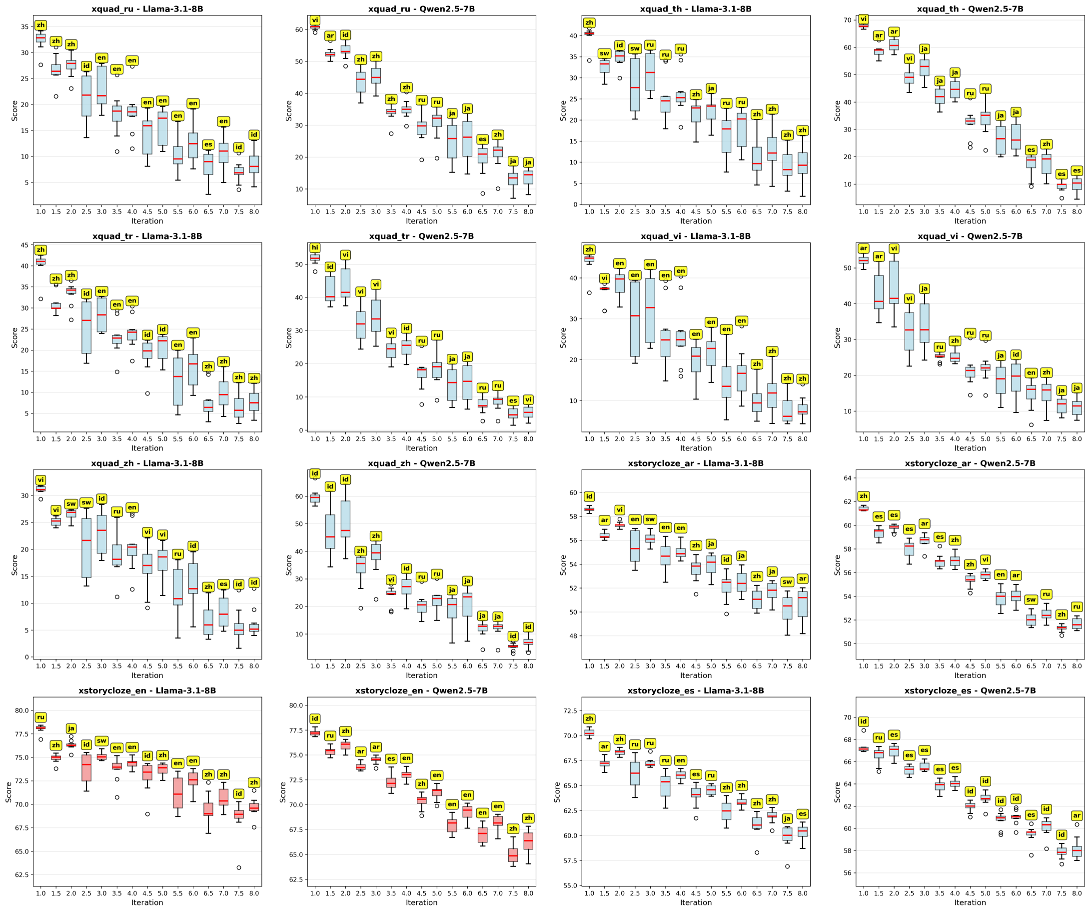

Figure 12: Boxplots showing the distribution of scores across iterations for different models and tasks. Each boxplot represents the score distribution for a specific task and model combination, with the best language annotated for each iteration. Note that the min and max values in y-axis are adjusted and different for each task.

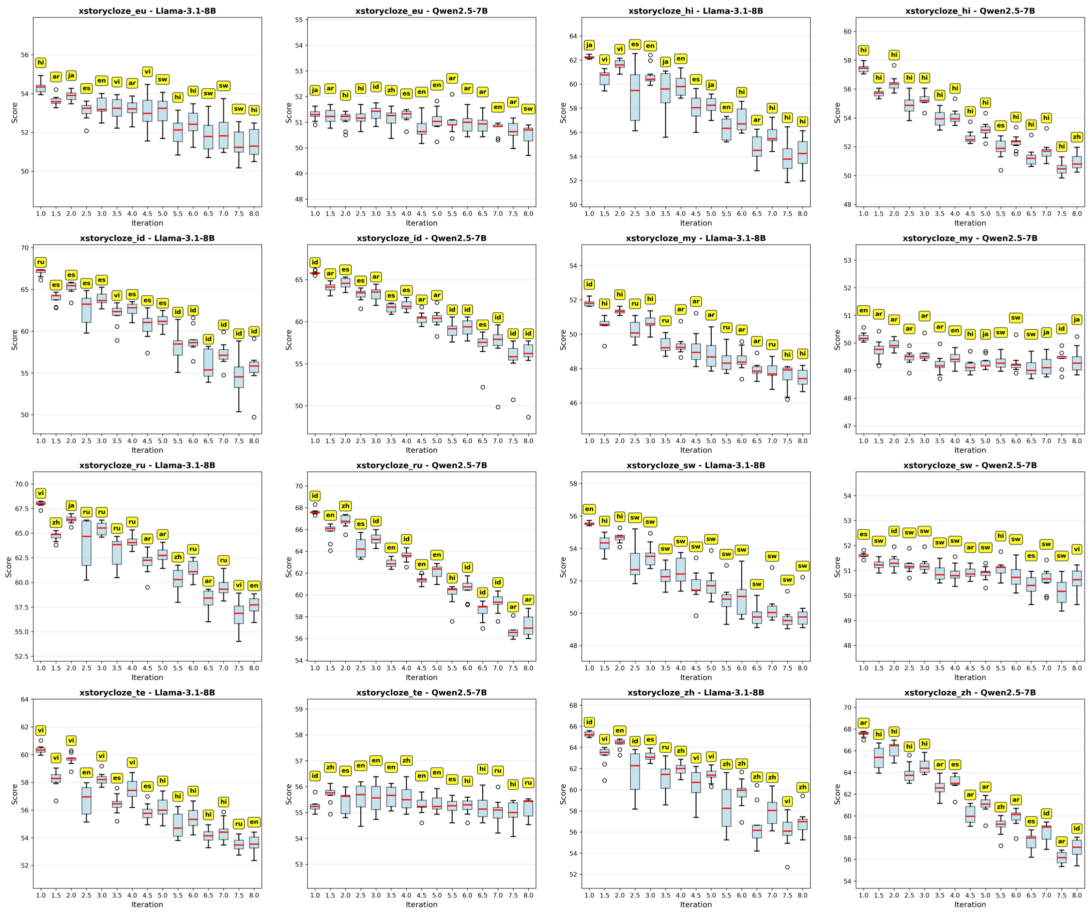

Figure 13: Boxplots showing the distribution of scores across iterations for different models and tasks. Each boxplot represents the score distribution for a specific task and model combination, with the best language annotated for each iteration. Note that the min and max values in y-axis are adjusted and different for each task.

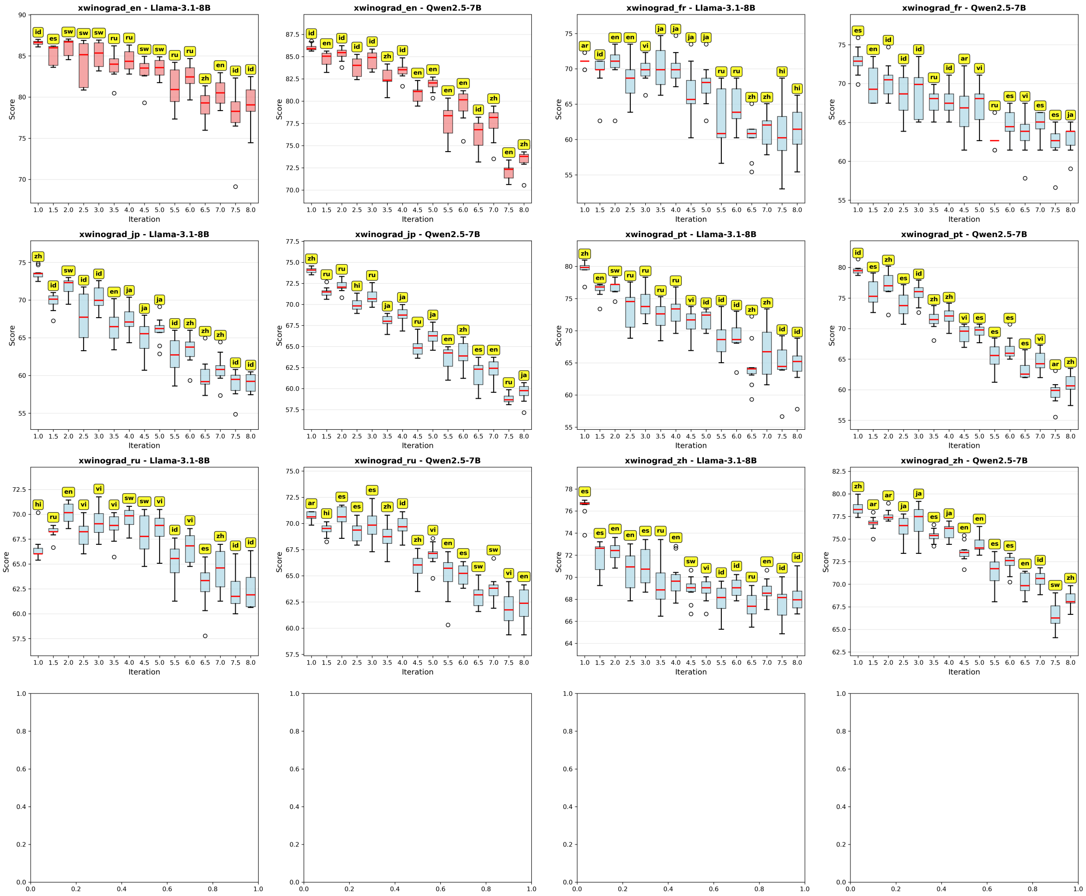

Figure 14: Boxplots showing the distribution of scores across iterations for different models and tasks. Each boxplot represents the score distribution for a specific task and model combination, with the best language annotated for each iteration. Note that the min and max values in y-axis are adjusted and different for each task.

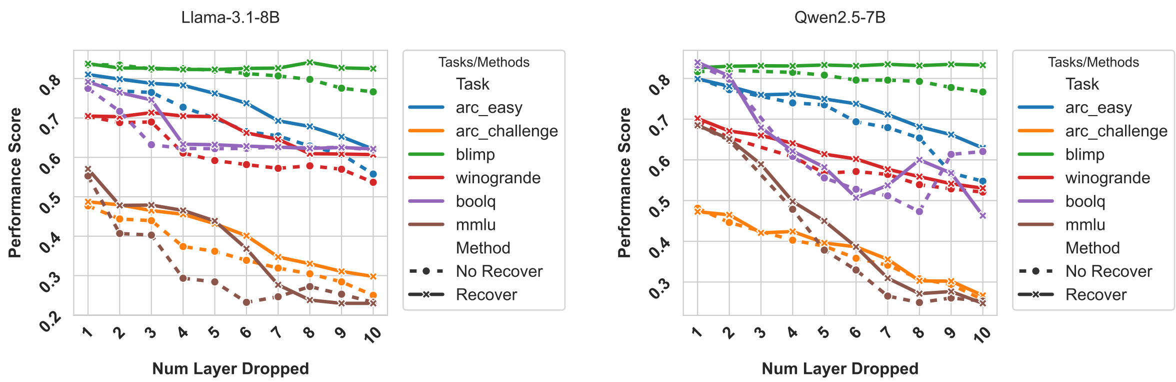

Figure 15: performance on six different subtasks. dotted line denoted implementing Iterative Prunning without recovery phase while solid line denoted layer prunning and recovery phase are done in Iterative Prunning

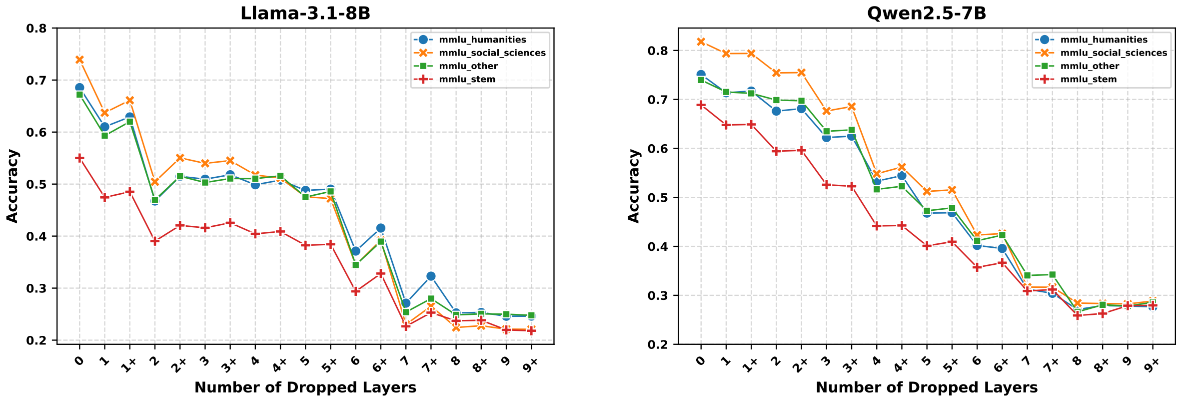

Figure 16: Line charts depicts MMLU groupings performance on Llama-3.1-8B and Qwen2.5-7B in 10 iterations. "+" markers indicate the recovery phase; all other markers represent the pruning phase.

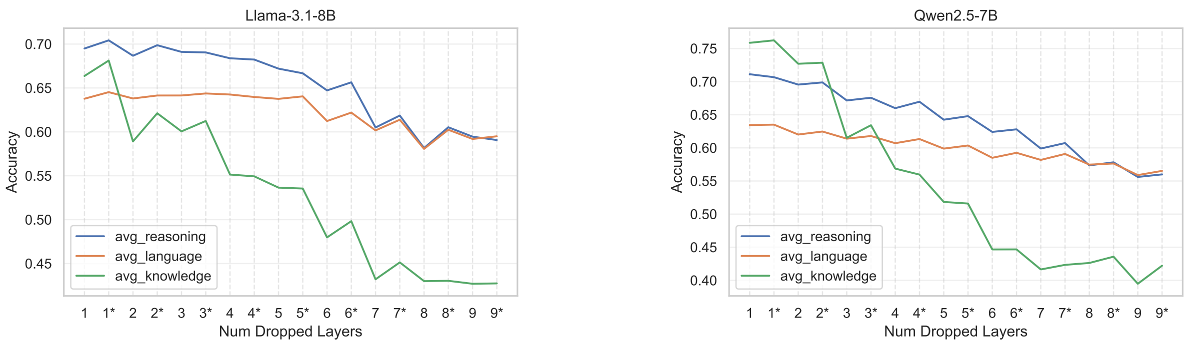

Figure 17: The average performance across pruning and recovery phase for 10 iterations on Llama 3.1-8B and Qwen2.5-7B on an average aggregation of reasoning, language, and knowledge tasks.

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 26: BLiMP Specialized Construction Tests