LaCo: Large Language Model Pruning via Layer Collapse

Yifei Yang[1,2,3], **Zouying Cao**[1,2,3], **Hai Zhao**[1,2,3*]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering, Shanghai Jiao Tong University
[3]Shanghai Key Laboratory of Trusted Data Circulation and Governance in Web3
`{yifeiyang,zouyingcao}@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn`

Abstract

Large language models (LLMs) based on transformer are witnessing a notable trend of size expansion, which brings considerable costs to both model training and inference. However, existing methods such as model quantization, knowledge distillation, and model pruning are constrained by various issues, including hardware support limitations, the need for extensive training, and alterations to the model internal structure. In this paper, we propose a concise layer-wise structured pruner called *Layer Collapse (LaCo)*, in which rear model layers collapse into a prior layer, enabling a rapid reduction in model size while preserving the model structure. Comprehensive experiments show that our method maintains an average task performance of over 80% at pruning ratios of 25-30%, significantly outperforming existing state-of-the-art structured pruning methods. We also conduct post-training experiments to confirm that the *LaCo* effectively inherits the parameters of the original model. Additionally, we perform ablation studies on various settings of *LaCo*. Finally, we discuss our motivation from the perspective of layer-wise similarity and evaluate the performance of the pruned LLMs across various pruning ratios[1].

1 Introduction

Recently, large language models (LLMs) based on Transformer (Vaswani et al., 2017) have showcased impressive capabilities across diverse tasks. However, the prevailing trend in model development leans towards larger scales, placing substantial demands on computational resources.

To mitigate the above challenge, various approaches have been explored to reduce the inference and training costs of models or to derive a smaller model from an LLM, including model quantization (Dettmers et al., 2022; Yao et al., 2022; Xiao et al., 2023), knowledge distillation (Liu et al., 2022; Hsieh et al., 2023; Shridhar et al., 2023), and model pruning (Zhang et al., 2022; Frantar and Alistarh, 2023; Ma et al., 2023). However, existing solutions exhibit certain notable drawbacks. Model quantization typically necessitates specific hardware support and often impacts model performance. Knowledge distillation often requires retraining a smaller model, which is costly and task-specific. Model pruning, whether non-structured or structured, has its issues. Non-structured pruning often involves model sparsity, which generally leads to certain performance loss and also relies on hardware support. Structured pruning entails removing specific modules, often altering the model structure and diminishing the model portability.

Considering the above issues, we contemplate directly pruning the model with a new idea: to prune some layers directly from a well-trained LLM and substitute the parameters of one layer for multiple layers, enabling effective model pruning.

Specifically, we observe that merging the param-

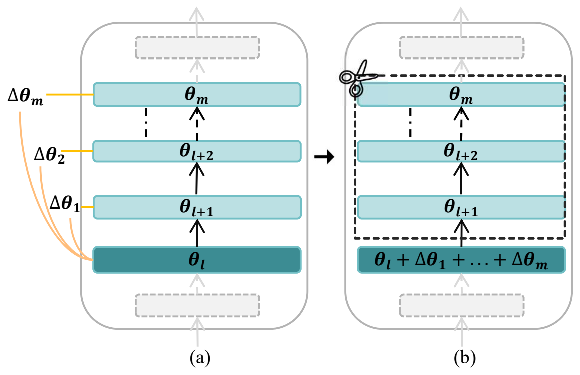


Figure 1: An example of *Reserving-Differences-while-Seeking-Common (RDSC) Layer Merge.* In (a), we perform parameter differencing, which we regard as Reserving-Differences. In (b), we conduct parameter merging, which we interpret as Seeking-Common.

[*] Corresponding author. This research was supported by the Joint Research Project of Yangtze River Delta Science and Technology Innovation Community (No. 2022CSJGG1400), the Joint Funds of the National Natural Science Foundation of China (Grant No. U21B2020).

[1]`https://github.com/yangyifei729/LaCo`

eter differentials of certain layers with their subsequent layers often does not significantly impact model performance, as illustrated in Figure 1. We term it the *Reserving-Differences-while-Seeking-Common (RDSC) Layer Merge*, as it incorporates parameter differencing and merging. Building upon this insight, we introduce a streamlined yet potent layer-wise pruner dubbed *Layer Collapse (LaCo)*, in which rear layers collapse into a prior layer, with the objective of preserving the model's output representation. In this paper:

• The *Layer Collapse* can directly remove 30%-50% of model layers without training while maintaining the model performance. Experiments on multiple benchmarks show that our approach outperforms state-of-the-art structured pruning methods under equivalent pruning ratios.

• The *Layer Collapse* preserves the internal structure of LLMs, such as maintaining intermediate dimensions. So, the pruned models can be seamlessly integrated into existing applications without any changes to the system's implementation.

• We conduct post-training to confirm that *Layer Collapse* can efficiently inherit parameters and requires only minimal training to restore the pruned model to the original model's loss convergence level. Additionally, we discuss our motivation and evaluate the performance of pruned models using *LaCo* across different pruning ratios. We also perform ablation studies on various settings of *LaCo*.

2 Method

2.1 Reserving-Differences-while-Seeking-Common Layer Merge

For the l-th layer of an LLM, we denote all its parameters, including those in self-attention (SAN) and MLP as $\boldsymbol{\theta}_l$. For the m consecutive layers following it, we merge the parameters of $\boldsymbol{\theta_{l+1}}, \boldsymbol{\theta_{l+2}}, \cdots, \boldsymbol{\theta_{l+m}}$ into $\boldsymbol{\theta_l}$ to form $\boldsymbol{\theta_l^*}$:

$$\begin{aligned}\boldsymbol{\theta}_l^* &= \boldsymbol{\theta}_l + (\boldsymbol{\theta}_{l+1} - \boldsymbol{\theta}_l) + \cdots + (\boldsymbol{\theta}_{l+m} - \boldsymbol{\theta}_l) \\ &= \boldsymbol{\theta}_l + \sum_{k=1}^{m} (\boldsymbol{\theta}_{l+k} - \boldsymbol{\theta}_l)\end{aligned} \tag{1}$$

where $(\boldsymbol{\theta_{l+k}} - \boldsymbol{\theta_l})$ is the layer-wise parameter difference. Given identical layer structures, we independently apply these processes to both SAN and MLP. Then, these m consecutive layers will be discarded. Subsequent model pruning will continuously involve RDSC Layer Merge which can be regarded as the continual collapse of layers onto specific layers, hence the name *Layer Collapse*.

Algorithm 1 Workflow of Layer Collapse

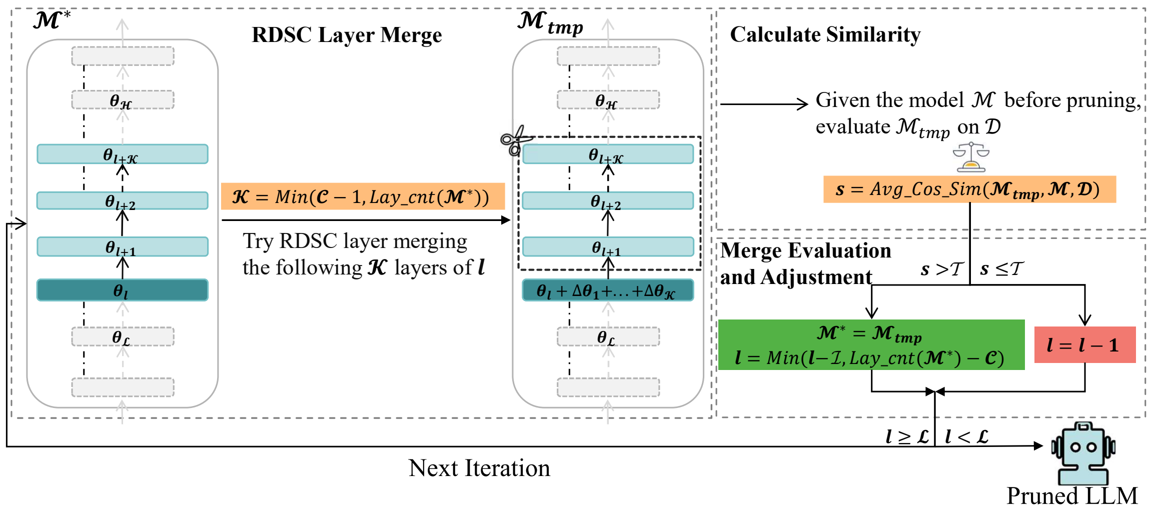
```
Require: LLM $\mathcal{M}$; Number of layers combined in
    each merge $\mathcal{C}$; Layer range $[\mathcal{L}, \mathcal{H}]$; Minimum
    interval between two adjacent merged layers $\mathcal{I}$;
    Few-shot Calibration Samples $\mathcal{D}$; Threshold
    for representation similarity $\mathcal{T}$
Ensure: Pruned LLM $\mathcal{M}^*$
 1: $\mathcal{M}^* \leftarrow \mathcal{M}$
 2: $l \leftarrow \mathcal{H} - \mathcal{C}$
 3: while $l >= \mathcal{L}$ do
 4:     $\mathcal{K} \leftarrow$ Min($\mathcal{C} - 1$, Layer_Count($\mathcal{M}^*$)$-l$)
 5:     $\mathcal{M}_{tmp} \leftarrow$ RDSC_Lay_Merge($\mathcal{M}^*$, $l$, $\mathcal{K}$)
 6:     $s \leftarrow$ Avg_Cos_Sim($\mathcal{M}_{tmp}$, $\mathcal{M}$, $\mathcal{D}$)
 7:     if $s > \mathcal{T}$ then
 8:         $\mathcal{M}^* \leftarrow \mathcal{M}_{tmp}$
 9:         $l \leftarrow l - \mathcal{I}$
10:         if $l >$ Layer_Count($\mathcal{M}^*$) then
11:             $l \leftarrow$Layer_Count($\mathcal{M}^*$)$-\mathcal{C}$
12:         end if
13:     else
14:         $l \leftarrow l - 1$
15:     end if
16: end while
17: return $\mathcal{M}^*$
```

2.2 Layer Collapse

We dynamically merge adjacent layers from the topmost layer down, ensuring the pruned model's output representation on few-shot calibration samples remains as similar as possible to the original model to minimize performance loss. Algorithm 1 summarizes the workflow of *Layer Collapse*:

(1) Preparation

For an LLM $\mathcal{M}$ to be pruned, we define the number of layers to be merged during each merging operation as $\mathcal{C}$. We configure the merging to operate within a certain range of layers, denoted as $[\mathcal{L}, \mathcal{H}]$. As the layer merging operation inevitably leads to a performance loss, to prevent consecutive layer merging from causing a sharp decline in the model performance, we set a minimum interval of layers between two merging operations as $\mathcal{I}$. Few-shot calibration samples $\mathcal{D}$, typically a few plain sentences, are used during the pruning process. We perform forward computations on $\mathcal{D}$ with both the pruned and original models to obtain the output representations and ensure that the similarity of representations is not less than the threshold $\mathcal{T}$.



$\mathcal{M}^*$
RDSC Layer Merge
$\mathcal{M}_{tmp}$
Calculate Similarity
$\theta_{\mathcal{H}}$
$\theta_{l+\mathcal{K}}$
θ_{l+2}
θ_{l+1}
θ_l
$\theta_{\mathcal{L}}$
$\mathcal{K} = Min(\mathcal{C}-1, Lay\_cnt(\mathcal{M}^*))$
Try RDSC layer merging the following $\mathcal{K}$ layers of l
$\theta_l + \Delta\theta_1 + \ldots + \Delta\theta_{\mathcal{K}}$
Given the model $\mathcal{M}$ before pruning, evaluate $\mathcal{M}_{tmp}$ on $\mathcal{D}$
$s = Avg\_Cos\_Sim(\mathcal{M}_{tmp}, \mathcal{M}, \mathcal{D})$
Merge Evaluation and Adjustment
$s > \mathcal{T}$
$s \leq \mathcal{T}$
$\mathcal{M}^* = \mathcal{M}_{tmp}$
$l = Min(l - \mathcal{I}, Lay\_cnt(\mathcal{M}^*) - \mathcal{C})$
$l = l - 1$
$l \geq \mathcal{L}$
$l < \mathcal{L}$
Next Iteration
Pruned LLM

Figure 2: An illustration of Layer Collapse.

(2) Pruning (line 1-17)

We present an illustration of *Layer Collapse* in Figure 2. We begin by initializing the model $\mathcal{M}^*$ with the model $\mathcal{M}$ and set a layer pointer l to start from $\mathcal{H} - \mathcal{C}$. Then, the iterative process begins:

RDSC Layer Merge (line 4-5) During each iteration, our approach involves merging the $\mathcal{K}$ layers following layer l into layer l itself and then discarding the redundant $\mathcal{K}$ layers, where $\mathcal{K}$ is the minimum of $\mathcal{C} - 1$ and the total layer count of $\mathcal{M}^* - l$, implying merging either the subsequent $\mathcal{C} - 1$ layers or all layers following l, thus to prune the model $\mathcal{M}^*$, resulting in the interim model $\mathcal{M}_{tmp}$.

Calculate Similarity (line 6) We process each sentence in $\mathcal{D}$ using forward computations with $\mathcal{M}_{tmp}$ and $\mathcal{M}$ to derive their representations which are the output hidden-states of the last layer of the model. For every sentence, we then calculate the cosine similarity between these representations from both models, averaging these values to obtain the overall similarity score s.

Merge Evaluation and Adjustment (line 7-15) Then, we evaluate s against the threshold $\mathcal{T}$. Should s exceed $\mathcal{T}$, the current merge is considered successful. Then, $\mathcal{M}_{tmp}$ is updated to $\mathcal{M}^*$ for the next iteration, and the pointer l is adjusted downwards by $\mathcal{I}$ layers. Conversely, l is simply reduced by a single layer. It is important to highlight that the instances may occur where l falls below the total layer count of $\mathcal{M}^*$ after a series of successive merges. Consequently, it is required to reset l to the layer count in $\mathcal{M}^* - \mathcal{C}$, as illustrated in line 11.

We iterate through the above process until l is less than $\mathcal{L}$ and output the pruned LLM.

2.3 Complexity Analysis

The complexity of *LaCo* primarily depends on model inference. In the worst-case scenario, with $\mathcal{L}$ set to 0 and $\mathcal{H}$ to the total number of layers, if in each iteration the similarity s is less than $\mathcal{T}$, all layers will be traversed. Thus, the worst-case time complexity is $O(\mathcal{H} \times ||\mathcal{D}||)$. For example, for Llama2-13B with 40 layers and $||\mathcal{D}||$ consisting of 10 sentences, the maximum number of inference steps would be only 400 sentences, which can be completed within minutes on a single GPU.

3 Experiments

3.1 Models

To assess the effectiveness of the proposed *LaCo*, we conduct experiments on popular English LLMs, Llama2-7B and 13B (Touvron et al., 2023). Additionally, we test the effectiveness on bilingual LLMs, specifically Baichuan2-7B and 13B (Yang et al., 2023), which support both Chinese and English. We leverage the base versions of these LLMs. All these models are decoder-only models based on the transformer architecture.

3.2 Benchmarks

To comprehensively evaluate the pruned model's capabilities, we utilized the OpenCompass evaluation framework (Contributors, 2023). Specifically, following OpenCompass categorization, we conduct evaluations in five aspects: Reasoning, Language, Knowledge, Examination and Understanding. We select several benchmarks from each category. **Reasoning:** CMNLI (Xu et al., 2020), HellaSwag (HeSw) (Zellers et al., 2019), PIQA

(Bisk et al., 2019). **Language:** CHID (Zheng et al., 2019), WSC (Levesque et al., 2012). **Knowledge:** CommonSenseQA (CSQA) (Talmor et al., 2018), BoolQ (Clark et al., 2019). **Examination:** MMLU (Hendrycks et al., 2021), CMMLU (Li et al., 2023). **Understanding:** Race-High/Middle (H/M) (Lai et al., 2017), XSum (Narayan et al., 2018), C3 (Sun et al., 2020).

We conduct evaluations using official scripts from OpenCompass, all zero-shot or few-shot, without additional training. Two evaluation modes are utilized: perplexity (PPL) and generation (GEN) [2]. For CHID and XSum, we use the GEN mode. For the WSC dataset, we use both PPL ($\mathrm{WSC_P}$) and GEN ($\mathrm{WSC_G}$) modes. The remaining benchmarks are evaluated using the PPL mode. The evaluation results on each benchmark are converted to a score by OpenCompass, where a higher score indicates better performance. OpenCompass provides official evaluation results for the Baichuan2 and Llama2 series. However, to avoid discrepancies resulting from hardware and software environments, as well as potential errors in official results, we reproduce all results to ensure fairness.

3.3 Baselines

Since *LaCo* involves structured pruning, which directly removes components from LLMs, we select two state-of-the-art (SOTA) structured pruning methods, LLM-Pruner (LLMPru.) (Ma et al., 2023) and SliceGPT (Ashkboos et al., 2024), as our baselines. These methods have surpassed the previous SOTA sparsity method, SparseGPT (Frantar and Alistarh, 2023). In our experiments, we set the pruning ratios of baselines to be equivalent to or slightly smaller than *LaCo* to ensure fairness.

3.4 Settings

Since previous work mostly set pruning ratios below 30%, we heuristically adjust the hyperparameters to bring the model pruning ratio close to 30%, as shown in Appendix A Table 8. We randomly select 5 sentences from both the English and Chinese Wikipedia datasets for Baichuan2 and 10 sentences from English Wikipedia for Llama2 as few-shot calibration samples. All experiments are conducted on a server with 8 Nvidia A100 80GB GPUs.

[2]opencompass.readthedocs.io/en/latest/get_started/faq.html

3.5 Main Results

In Table 1, we present the results of four LLMs under different pruning methods across various benchmarks. "Dense" represents the official results of the unpruned LLMs in OpenCompass leaderboards, while "Dense*" represents our reproduction of the "Dense" results. "LLMPru." and "SliceGPT" correspond to the two baselines, respectively. "Ratio" refers to the overall pruning ratio, namely the proportion of the total number of pruned parameters to that of the unpruned model. "Lay." denotes the total number of layers in the model.

Comparing Dense and Dense*, the results show not much difference, with most discrepancies within 5%. This indicates our experimental setup is error-free. To ensure fairness, we compare the results against Dense* in the subsequent analyses.

Upon comparing LaCo with the baselines, from Table 1, it can be observed that LaCo achieves the best results on most benchmarks, despite our pruning ratio being slightly higher than the baselines.

To provide a more intuitive presentation of the results in Table 1, we compute the average scores of each pruner across all benchmarks (Avg.), the average scores per category (Reas., Lan., Know., Exam., Unde.), and the average performance percentages relative to Dense* across all benchmarks (Per.) in Table 2. Overall, our average scores are significantly higher than the baselines. LaCo shows superior performance in four out of five categories. Though there is a slight dip in Reasoning, it remains comparable. Additionally, LaCo's average performance percentage across all datasets, relative to Dense*, is far superior to the baselines. The average percentage surpasses 80% in three out of four models, with the lowest being 73% on Baichuan2-7B. In contrast, none of the baselines exceed 70%.

To demonstrate the stability of the pruned models by LaCo, we compute the performance percentage relative to Dense* (Appendix D.4 Table 22). LaCo-pruned models maintain performance above 70% on most benchmarks and do not experience crashes, with no performance dropping below 30%.

Notably, on three benchmarks evaluated through GEN mode, CHID, XSUM, and WSC_G, the LLMs pruned by LaCo maintain relatively stable performance, while models pruned by baselines exhibit poorly, with even multiple results becoming 0.00. GEN mode scores are based on the model's generated sentences, and the models pruned by baselines are prone to producing meaningless repeti-

LLM	Pruner	Ratio/Lay.	Reasoning			Language			Knowledge		Examination		Understanding			
			CMNLI	HeSw	PIQA	CHID	WSC$_P$	WSC$_G$	CSQA	BoolQ	MMLU	CMMLU	Race$_H$	Race$_M$	XSum	C3
Llama2 -7B	Dense	0%/32	34.90	74.00	78.30	46.50	-	66.30	66.50	74.90	46.80	31.80	37.50	40.20	19.70	42.80
	Dense*	0%/32	32.98	71.35	78.18	46.04	37.50	38.46	66.67	70.67	45.92	31.86	35.51	33.15	19.68	43.78
	LLMPru.	27.0%/32	34.33	**56.46**	**71.22**	25.25	36.54	0.96	42.51	55.20	23.33	25.25	22.56	22.35	11.51	25.64
	SliceGPT	26.4%/32	31.70	50.27	66.21	20.79	36.54	19.23	41.36	38.32	**28.92**	**25.37**	21.07	21.66	4.89	**39.78**
	LaCo	**27.1%/23**	**34.43**	55.69	69.80	**36.14**	**40.38**	**25.00**	**45.70**	**64.07**	26.45	25.24	**22.61**	**23.61**	**15.64**	39.67
Llama2 -13B	Dense	0%/40	41.40	77.50	79.80	53.00	-	66.30	66.70	82.40	55.00	38.40	58.90	63.00	23.40	46.10
	Dense*	0%/40	32.99	74.83	79.71	52.97	50.96	63.46	66.91	71.50	55.63	38.74	58.03	60.24	23.56	47.51
	LLMPru.	24.4%/40	**33.03**	**67.76**	**76.66**	35.64	40.38	0.00	50.86	56.42	25.21	24.71	22.47	22.08	**19.17**	32.33
	SliceGPT	23.6%/40	29.82	55.71	69.04	19.31	36.54	**36.54**	47.26	37.86	37.14	25.79	23.41	24.03	5.27	41.92
	LaCo	**24.6%/30**	32.86	64.39	74.27	**40.10**	**52.88**	35.58	**52.66**	**63.98**	**45.93**	**32.62**	**54.49**	**56.55**	14.45	**44.93**
Baic2. -7B	Dense	0%/32	32.90	67.00	76.20	82.70	-	66.30	63.00	63.20	54.70	57.00	52.50	50.90	20.90	64.50
	Dense*	0%/32	33.37	67.56	76.17	82.67	41.35	63.46	63.14	63.30	54.25	56.95	52.63	51.04	20.84	64.55
	LLMPru.	24.2%/32	32.28	**53.66**	**71.82**	69.80	**53.85**	0.00	**47.83**	**61.19**	24.93	25.69	21.96	22.28	**15.98**	41.64
	SliceGPT	22.2%/32	32.07	25.29	50.33	14.85	36.54	0.00	19.57	39.30	25.18	25.25	23.53	22.49	0.00	26.58
	LaCo	**24.2%/23**	**33.00**	52.28	68.50	**76.24**	42.31	**26.92**	47.26	56.15	**31.53**	**31.24**	**28.99**	**27.72**	12.03	**50.85**
Baic2. -13B	Dense	0%/40	32.70	70.80	78.10	83.20	-	63.20	65.60	67.00	59.50	61.30	67.20	68.90	25.20	65.60
	Dense*	0%/40	33.21	71.10	78.07	83.17	41.35	63.46	65.60	67.00	58.81	61.27	67.27	68.94	24.95	65.64
	LLMPru.	24.3%/40	**33.80**	53.57	**71.82**	72.77	37.50	0.00	38.82	56.54	23.19	25.18	21.17	21.61	**13.67**	39.89
	SliceGPT	22.8%/40	32.07	25.85	51.03	10.40	36.54	0.00	18.02	37.83	22.95	25.26	21.56	21.52	0.00	24.99
	LaCo	**24.7%/30**	33.03	**60.71**	68.88	**76.73**	**44.23**	**60.58**	**55.45**	**62.35**	**51.35**	**53.65**	**56.92**	**57.80**	12.32	**61.10**

Table 1: The main results of our experiments. *Lay.* is the number of model layers. *Dense* is the official LLM results in OpenCompass and *Dense** is our reproduction. *LLMPru.* and *SliceGPT* are two baseline comparisons.

tive outputs. In Appendix D.5 Table 23, we showcase an example from the Xsum benchmark, where Llama2-7B, pruned by baselines, produces nonsensical repeated outputs, whereas our LaCo yields outputs resembling normal sentences.

We also conduct experiments with Llama2-70B on several benchmarks. The results in Appendix D.1 Table 19 show that LaCo still outperforms the baseline on larger-scale model.

In summary, LaCo is a superior pruner that preserves model performance and demonstrates exceptional stability across various benchmarks. It relies solely on parameter differences and additions, without altering the model's internal structure, resulting in a concise and efficient pruning solution.

LLM	Pruner	Avg.	Per.	Reas.	Lan.	Know.	Exam.	Unde.
Llama2 -7B	Dense*	46.55	100%	60.83	40.67	68.67	38.89	33.03
	LLMPru.	32.36	67.79%	**54.00**	20.92	48.86	24.29	20.52
	SliceGPT	31.87	67.37%	49.39	25.52	39.84	**27.15**	21.85
	LaCo	**37.46**	**80.28%**	53.30	**33.84**	**54.89**	25.85	**25.38**
Llama2 -13B	Dense*	55.50	100%	62.51	55.80	69.20	47.18	47.34
	LLMPru.	36.19	65.87%	**59.15**	25.34	53.64	24.96	24.01
	SliceGPT	34.97	61.78%	51.52	30.80	42.56	31.46	23.66
	LaCo	**47.55**	**85.21%**	57.17	**42.85**	**58.32**	**39.28**	**42.60**
Baic2. -7B	Dense*	56.52	100%	59.03	62.49	63.22	55.60	47.26
	LLMPru.	38.78	69.65%	**52.59**	41.22	**54.51**	25.31	25.46
	SliceGPT	24.36	44.27%	35.90	17.13	29.44	25.22	18.15
	LaCo	**41.79**	**73.26%**	51.26	**48.49**	51.70	**31.38**	**29.90**
Baic2. -13B	Dense*	60.70	100%	60.79	62.66	66.30	60.04	56.70
	LLMPru.	36.40	60.70%	53.06	36.76	47.68	24.18	24.08
	SliceGPT	23.43	40.33%	36.32	15.65	27.92	24.10	17.02
	LaCo	**53.94**	**87.94%**	**54.21**	**60.51**	**58.90**	**52.50**	**47.04**

Table 2: The average scores and the percentages comparison with the Dense*.

3.6 Comparison of Perplexity

Since PPL is also a commonly used metric for evaluating model performance, we want to compare how LaCo differs from the baseline in maintaining the model's PPL. We evaluate the PPL of the Llama2-7B model with 27% sparsity using different pruners. The evaluation is performed on 500 sentences selected from Wikipedia, each with a fixed length of 512 tokens.

The results in Table 3 show that the model pruned by LaCo also achieves a lower PPL com-

Pruner	Dense	LaCo	LLM-Pruner	SliceGPT
PPL	4.46	**13.93**	17.30	14.51

Table 3: PPL for different pruners.

pared to other baselines, further highlighting the advantage of our LaCo.

3.7 Pruning Time

Pruner	LaCo	LLM-Pruner	SliceGPT
Pruning Time	**14.7s**	15.9s	313s

Table 4: Pruning time for different pruners.

To verify that LaCo has lower time complexity and faster pruning speed than the baselines, we compare LaCo with them for 27% sparsity pruning of Llama2-7B on a single A100 GPU. For fairness, we only measure the main pruning process, excluding the time for loading models, loading data, and storing models. The results in Table 4 show LaCo pruning is more efficient compared to the baselines.

3.8 Memory Usage and Inference Speed

We also aim to investigate whether the model pruned by LaCo offers advantages in memory usage and inference speed compared to the models pruned by the baselines. In Table 5, we present the average memory consumption and inference speed of the Llama2-13B pruned models from Table 1 on the English Wiki dataset (The results for all models are in Appendix D.2 Table 20). All models are loaded in Bf16 on a single A100 GPU.

Pruner	LaCo	Dense	LLMPru.	SliceGPT
Memory	**19422**	25902	19874	22506
Infer.	**38.65**	29.98	27.15 (↓)	35.16

Table 5: Memory usage (MB) and inference speed (tokens/s) of the Llama2-13B pruned by different pruners. ↓ indicates performance worse than the Dense model.

The results indicate that the LaCo-pruned models consume less memory and achieve faster inference speeds. Moreover, while existing baselines may decrease inference speeds compared to the dense model, LaCo does not have this issue.

4 Further Analysis

4.1 Post-training and Re-pruning

4.1.1 Post-training

Due to the inevitable performance loss caused by pruning, we investigate whether models pruned using our LaCo can effectively inherit parameters from the original model and quickly recover performance through post-training on the full parameters. Specifically, we select the pruned Llama2-7B and Baichuan2-7B models obtained through LaCo in the main experiments and post-train them. For training pruned Llama2-7B, we utilize approximately 1.0 billion tokens from the English dataset, while for pruned Baichuan2-7B, we employ approximately 1.25 billion tokens, with a 50% from English and Chinese. The detailed implementation can be found in the Appendix C.

In Figure 3, we present the loss curves. It can be observed that both models converge rapidly during training, with the loss showing a sharp decline after about 250 steps, then stabilizing. The pruned Llama2-7B and Baichuan2-7B models, both approximately 5 billion parameters, exhibit final convergence losses around 1.6 and 2.0, which are quite comparable to the reported values of 1.75 for Llama2-7B and 1.9 for Baichuan2-7B in their technical reports. The post-training of pruned Llama2-7B and Baichuan2-7B on 4 Nvidia A100 80GB GPUs takes approximately 28 hours and 35 hours, respectively. Training a 5B LLM from scratch requires at least 500 billion tokens on hundreds of A100 GPUs for several months. However, we achieve a loss-converged model of similar size with only one-thousandth of their cost. This indicates that the pruned models have effectively inherited the parameters of the original models, enabling them to rapidly recover performance with minimal post-training and achieve convergence.

We also evaluate the post-trained models on multiple benchmarks with detailed results in Appendix E Table 24. The average scores for each category and the overall average are in Table 6.

From the tables, it is evident that the post-training of pruned Llama2-7B significantly improves its performance across various benchmarks. However, the performance of pruned Baichuan2-7B after post-training shows mixed results, with some benchmarks showing improvement while others exhibit a decrease and there is also a slight decrease in the overall score. We speculate that

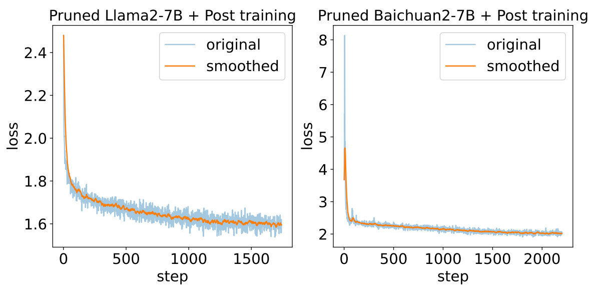


Figure 3: Loss curves for post-training.

LLM	Method	Avg.	Reas.	Lan.	Know.	Exam.	Unde.
Llama2-7B	Dense*	46.55	60.83	40.67	68.67	38.89	33.03
	LaCo	37.46	53.30	33.84	54.89	25.85	25.38
	LaCo +post train	**40.33**	**56.66**	**36.43**	**61.85**	**27.40**	**26.70**
	LaCo +post train +re prune	32.40	48.07	20.26	49.46	25.72	24.56
Baic2.-7B	Dense*	56.52	59.03	62.49	63.22	55.60	47.26
	LaCo	**41.79**	51.26	**48.49**	51.70	**31.38**	29.90
	LaCo +post train	40.46	**51.67**	40.82	**53.97**	27.98	**31.28**

Table 6: Average scores across all categories and the overall average score of pruned models, post-trained models, post-trained models followed by re-pruning.

the pre-training data of Baichuan2-7B includes a variety of sources, resulting in a data distribution different from that of our post-training data, hindering the effectiveness of post-training. However, the consistent score improvement on pruned Llama2-7B indicates that models pruned using our LaCo indeed effectively inherit the parameters and can regain performance through low-cost post-training.

LaCo achieves excellent performance through post-training, prompting us to compare its effectiveness with the SOTA LLM-Pruner on the same training data. Our results, shown in the Appendix D.3 Table 21, indicate that the model pruned by LaCo outperforms the one pruned by LLM-Pruner after post-training. Meanwhile, LaCo also significantly reduces training resource consumption.

4.1.2 Re-pruning

Since it is possible to partially restore performance using post-training on an LLM with approximately 25%-30% of its parameters pruned, it raises the question of whether we can further prune the post-trained model to obtain one with only around 50% parameters while still maintaining relatively good performance. Thus, we further prune the previously post-trained pruned Llama2-7B model using LaCo, resulting in a model with 17 layers, retaining 55% of the parameters of the original Llama2-7B model. We evaluate the re-pruned model. The detailed results are shown in Appendix E Table 24 and the average results are in Table 6.

The tables show that even with only 55% parameters, the model still retains about 70% of the original 7B model performance. However, our training data quality and scale are limited. With more and better training data, LaCo should demonstrate even greater utility in the pruning+post-training+re-pruning pipeline on larger models.

4.2 Layer-wise Similarity

This section discusses our motivation for merging adjacent layers. Our primary motivation comes from observing that the changes in parameters and output representations between adjacent layers in the LLMs are not particularly significant.

In Figure 4, we show the L2 similarities between the SAN q, k, v matrices of each layer and their counterparts in the subsequent layer, as well as the upscaling and downscaling matrices of the MLP for both Llama2-7B and Baichuan2-7B. The results indicate that the maximum L2 values between corresponding matrices in adjacent layers are generally no more than 200. Given the large sizes of the MLP upscaling (11008x4096) and SAN q, k, v (4096x4096) matrices, the change in each element between adjacent layers is minimal.

In Figure 5 (a), we randomly select 20 sentences from Wikipedia and calculate the cosine similarity between the hidden-states of adjacent layers outputs. The results show that for both Baichuan2-7B and Llama2-7B, the representation similarity between adjacent layers from layers 3 to 28 is typically very close to 1. The high similarity in parameters and representations between adjacent layers leads us to consider that a single layer might replace multiple subsequent layers.

Moreover, the similarity in parameters suggests focusing on the differences between layers. Inspired by previous model merging work (Yu et al., 2023; Matena and Raffel, 2022), we come up with collecting parameter differences between layers and merging them into a single layer. To verify that RDSC Layer Merge can replace multiple layers with one, we conduct the experiment: we merge every four consecutive layers into one within lay-

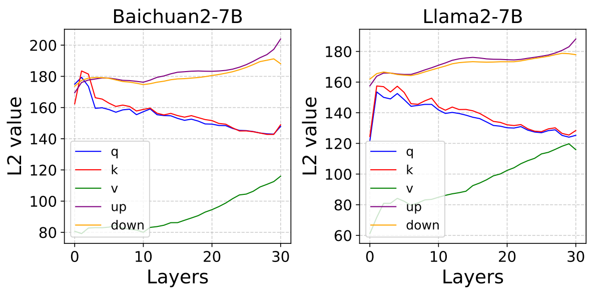
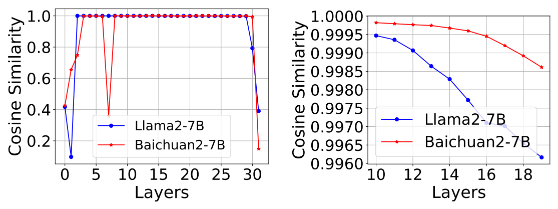


Figure 4: The L2 similarity of corresponding matrices between adjacent layers.



(a) The cosine similarity of output representations between adjacent layers.

(b) The similarity of output representations before and after RDSC Layer Merge.

Figure 5: The cosine similarity of layer representations.

LLM	Ratio/Lay.	Avg.	Reas.	Lan.	Know.	Exam.	Unde.
Llama2 -7B	0%/32	46.55	60.83	40.67	68.67	38.89	33.03
	12.0%/28	36.13	44.46	**36.31**	**56.35**	**26.34**	24.54
	27.1%/23	**37.46**	**53.30**	33.84	54.89	25.85	**25.38**
	45.0%/17	30.00	43.66	19.27	48.06	24.78	21.44
Llama2 -13B	0%/40	55.50	62.51	55.80	69.20	47.18	47.34
	14.6%/34	**53.89**	**60.56**	**54.51**	**63.58**	**46.10**	**47.46**
	24.7%/30	47.55	57.17	42.85	58.32	39.28	42.60
	49.7%/20	38.27	48.20	26.89	49.26	32.82	36.58

Table 7: Model performance at different pruning ratios.

ers 10 to 19 and evaluate the cosine similarity between the merged layer's output and the original last layer's output, as in Figure 5 (b), where the lowest cosine similarity on the 4096-dimensional vectors is above 0.996, confirming the effectiveness of RDSC Layer Merge in preserving representations.

4.3 Varying Pruning Ratio

In this section, we explore the performance of *LaCo* at different pruning ratios. We conduct experiments on Llama2-7B and Llama2-13B, controlling the pruning ratios at approximately 10%, 25% (our main experiments), and around 50% by setting different hyperparameters (as shown in Appendix A Table 9)[3]. We evaluate pruned models accordingly. The average results are shown in Table 7 and the detailed results are shown in Appendix E Table 25.

As the pruning ratio increases, overall model performance decreases. However, from a pruning ratio of around 10-15% to about 25%, the performance does not significantly decline, indicating our method's stability within this range. Furthermore, at a pruning ratio close to 50%, the model still maintains approximately 70% performance, demonstrating that our method prevents model crashes even with about half of the parameters removed.

5 Related Work

Model Quantization reduces model size by converting weights from high-precision floating points to lower-precision floating points or integers. SmoothQuant (Xiao et al., 2023) quantizes both weights and activations while smoothing activation outliers. GPTQ (Frantar et al., 2022) uses approximate second-order information for quantization. QLoRA (Dettmers et al., 2023a) backpropagates gradients through a frozen, 4-bit quantized model into Low Rank Adapters. OmniQuant (Shao et al., 2023) optimizes various quantization parameters.

Knowledge Distillation transfers knowledge from a large model to a smaller one. Distilling step-by-step (Hsieh et al., 2023) trains smaller models that outperform LLMs. DISCO (Chen et al., 2023) distills counterfactual knowledge from LLMs. SOCRATIC COT (Shridhar et al., 2023) distills the ability of Chain-of-Thought from LLMs. ZEPHYR (Tunstall et al., 2023) applies distilled direct preference optimization to learn a chat model.

Model Pruning refers to techniques for improving model efficiency by sparsification or parameter removal. Non-structured pruning often involves model sparsity. SparseGPT (Frantar and Alistarh, 2023) reduces the pruning problem to large-scale instances of sparse regression, while SpQR (Dettmers et al., 2023b) identifies and isolates outlier weights during LLM sparsification. Structured pruning primarily removes parts of model modules. LLM-Pruner (Ma et al., 2023) selectively eliminates non-critical structures based on gradient information. DirectShare (Cao et al., 2024) uses activation sharing methods to improve LLM inference efficiency. SheardL-

[3]Further ablation study on the hyperparameters $\mathcal{C}$, $\mathcal{I}$, $\mathcal{D}$, $\mathcal{T}$, different similarity metrics, different merging strategies, different calibration datasets can be found in Appendix B.

LaMA (Xia et al., 2023) uses targeted structured pruning and dynamic batch loading to prune Llama2. FLAP (An et al., 2024) assesses the recoverability of the output feature map after weight removal using the fluctuation pruning metric and standardizes importance scores to adaptively define the global compressed model structure. Shortened LLaMA (Kim et al., 2024) demonstrates that depth pruning can efficiently compress LLMs while matching or surpassing the performance of recent width pruning methods.

However, model quantization and sparsification typically require special hardware and usually impact performance. Knowledge distillation is costly and task-specific. Existing structured pruning methods often disrupt the model inherent structure. In contrast, *LaCo* maintains the model structure, which is more concise and preserves excellent performance. Although some existing works (Din et al., 2023; Fan et al., 2019; Belrose et al., 2023) have utilized layer-skipping/dropping to accelerate inference, LaCo is fundamentally different. It is the first pruner based on layer collapse, resulting in a smaller, faster, more memory-efficient model with strong performance. Furthermore, those methods typically require training new parameters to determine which layers to skip/drop during inference, whereas LaCo does not require any training.

6 Conclusion

In this paper, we propose a concise layer-wise structured pruning method called *Layer Collapse (LaCo)*, which merges rear model layers into preceding layers for rapid model size reduction. *LaCo* does not require special hardware support and preserves the model's intrinsic structure. Experimental results show that *LaCo* significantly outperforms current SOTA structured pruning methods, also revealing potential parameter redundancy in existing LLMs. We conduct ablation studies on various settings of *LaCo*. We also post-train the pruned models, confirming that *LaCo* effectively inherits the original model parameters. Additionally, we discuss our motivation from the perspective of layer-wise similarity and explore the performance of *LaCo*-pruned models at different pruning ratios.

Limitations

Due to *LaCo*'s pruning process primarily relying on layer-wise iterations, it cannot directly control the pruning ratio like previous methods. Instead, it requires tuning hyperparameters such as the representation similarity threshold $\mathcal{T}$ for control. In future work, we will summarize additional experimental patterns regarding how to set hyperparameters to achieve a specific pruning ratio.

Our motivation comes from current model merging techniques, but like existing baselines (LLM-Pruner (Ma et al., 2023) and SliceGPT (Ashkboos et al., 2024)), our method lacks a complete theoretical proof. We consider this as future work.

Additionally, there may be better merging methods, even though our experimental results demonstrate that LaCo's current merging approach is effective. We will continue to search for improved layer merging methods in the future.

References

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

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

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language.

Zouying Cao, Yifei Yang, and Hai Zhao. 2024. Head-wise shareable attention for large language models. *arXiv preprint arXiv:2402.11819*.

Zeming Chen, Qiyue Gao, Antoine Bosselut, Ashish Sabharwal, and Kyle Richardson. 2023. Disco: distilling counterfactuals with large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5514–5528.

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. *arXiv preprint arXiv:1905.10044*.

OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. `https://github.com/open-compass/opencompass`.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023a. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. 2023b. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*.

Alexander Yom Din, Taelin Karidi, Leshem Choshen, and Mor Geva. 2023. Jump to conclusions: Short-cutting transformers with linear transformations. *arXiv preprint arXiv:2303.09435*.

Angela Fan, Edouard Grave, and Armand Joulin. 2019. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*.

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

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

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)*.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301*.

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

Sosuke Kobayashi. 2018. Homemade bookcorpus. https://github.com/soskek/bookcorpus.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2023. Cmmlu: Measuring massive multitask language understanding in chinese.

Chang Liu, Chongyang Tao, Jiazhan Feng, and Dongyan Zhao. 2022. Multi-granularity structural knowledge distillation for language model compression. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1001–1011.

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

Michael S Matena and Colin A Raffel. 2022. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716.

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

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. 2023. Omniquant: Omnidirectionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137*.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2023. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7059–7073.

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2020. Investigating prior knowledge for challenging chinese machine reading comprehension. *Transactions of the Association for Computational Linguistics*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, et al. 2023. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

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

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR.

Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, et al. 2020. Clue: A chinese language understanding evaluation benchmark. *arXiv preprint arXiv:2004.05986*.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, et al. 2023. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305*.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2023. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *arXiv preprint arXiv:2311.03099*.

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

Qingru Zhang, Simiao Zuo, Chen Liang, Alexander Bukharin, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2022. Platon: Pruning large transformer models with upper confidence bound of weight importance. In *International Conference on Machine Learning*, pages 26809–26823. PMLR.

Chujie Zheng, Minlie Huang, and Aixin Sun. 2019. ChID: A large-scale Chinese IDiom dataset for cloze test. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 778–787, Florence, Italy. Association for Computational Linguistics.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*.

A Hyperparameter Settings

LLM	$\mathcal{C}$	$\mathcal{L}$	$\mathcal{H}$	$\mathcal{I}$	$\mathcal{T}$
Llama2-7B	4	1	32	2	0.65
Llama2-13B	6	1	40	2	0.75
Baichuan2-7B	4	1	32	2	0.70
Llama2-13B	6	1	40	2	0.70

Table 8: Hyperparameter settings for main results.

LLM (Ratio/Lay.)	$\mathcal{C}$	$\mathcal{L}$	$\mathcal{H}$	$\mathcal{I}$	$\mathcal{T}$
Llama2-7B (12.0%/28)	5	1	32	2	0.85
Llama2-7B (27.1%/23)	4	1	32	2	0.65
Llama2-7B (45.0%/17)	6	1	32	2	0.45
Llama2-13B (14.6%/34)	7	1	40	2	0.85
Llama2-13B (24.7%/30)	6	1	40	2	0.75
Llama2-13B (49.7%/20)	7	1	40	2	0.45

Table 9: Hyperparameters for varying pruning ratios.

B Ablation Study

B.1 Impact of Number of Layers Merged per Operation $\mathcal{C}$

We conduct an ablation study on the number of layers to be merged during each merging operation $\mathcal{C}$, as it is one of the key parameters that control the compression rate. The results are shown in Table 10, where the experiment is based on the Llama2-7b with a 27% compression rate.

	BoolQ	PIQA	HeSw
$\mathcal{C} = 1$ **(Result in Table 15)**	52.57	68.17	48.61
$\mathcal{C} = 2$	63.12	68.93	54.98
$\mathcal{C} = 3$	63.98	69.71	55.34
$\mathcal{C} = 4$ **(Result in Table 1)**	64.07	69.80	55.69
$\mathcal{C} = 5$	64.15	70.02	55.76
$\mathcal{C} = 6$	64.00	69.89	55.49

Table 10: Ablation study on $\mathcal{C}$.

When $\mathcal{C}$ is too small, the model's performance degrades. When $\mathcal{C}$ is between 3 and 6, the performance remains relatively good. If $\mathcal{C}$ is too large, it causes the cosine similarity of the hidden-states to drop quickly below the threshold, stopping the pruning loop prematurely and resulting in a lower compression rate. Therefore, we recommend setting $\mathcal{C}$ between 4 and 7.

B.2 Impact of Minimum Interval $\mathcal{I}$

To explore the impact of different values of $\mathcal{I}$, we also conduct an ablation study. The experiment is based on the Llama2-7b model with a 27% sparsity. The results are shown in Table 11.

	BoolQ	PIQA	HeSw
$\mathcal{I} = 1$	63.99	68.90	55.79
$\mathcal{I} = 2$ **(Result in Table 1)**	64.07	69.80	55.69
$\mathcal{I} = 3$	64.05	69.57	55.48
$\mathcal{I} = 4$	62.29	68.92	54.87

Table 11: Ablation study on $\mathcal{I}$.

The results indicate that the model performs well when $\mathcal{I}$ is between 1 and 3. However, when $\mathcal{I}$ is set to 4, the model's performance declines. This observation aligns with our empirical findings: modifying the parameters of layers closer to the input end (such as layers 1-12) can lead to performance drops. Setting $\mathcal{I}$ to 4 causes the merge operations to occur closer to the input end. Therefore, we choose $\mathcal{I} = 2$, as it relatively maintains better model performance.

B.3 Impact of Dataset $\mathcal{D}$

To understand the impact of different datasets $\mathcal{D}$ on LaCo's effectiveness, we conduct an ablation study using Llama2-7B. We perform three rounds of pruning, each time selecting different sets of 10 sentences as $\mathcal{D}$, with a 27% compression ratio to match Table 1. The results are shown in Table 12.

	BoolQ	PIQA	HeSw
Result in Table 1	64.07	69.80	55.69
Round1	64.50	70.28	56.14
Round2	63.70	69.14	54.92
Round3	63.90	69.66	55.18

Table 12: Ablation study on $\mathcal{D}$, where the model is Llama2-7B with compression ratio of 27%.

We can find that the results are all on nearly the same scale, indicating that the random selection of sentences in $\mathcal{D}$ has no notable impact on the results.

B.4 Impact of Threshold $\mathcal{T}$

We also aim to understand the impact of threshold $\mathcal{T}$ on the model pruned by LaCo. For the results of Llama2-7B in Table 1, we keep other parameters unchanged and set $\mathcal{T}$ to 0.85, 0.45, and 0.25 ($\mathcal{T} = 0.65$ corresponds to the results in Table 1). We evaluate the pruned models on several datasets. The results, shown in Table 13, indicate that a smaller threshold leads to a larger compression ratio and poorer performance, which aligns with intuition.

	Ratio/Layer	BoolQ	PIQA	HeSw
$\mathcal{T} = 0.65$	27.1%/23	64.07	69.80	55.69
$\mathcal{T} = 0.85$	9.0%/29	70.92	76.01	68.15
$\mathcal{T} = 0.45$	48.0%/16	59.82	60.34	36.09
$\mathcal{T} = 0.25$	60.1%/12	41.68	51.74	26.37

Table 13: Ablation study on $\mathcal{T}$. The model is Llama2-7B. $\mathcal{T} = 0.65$ corresponds to the results in Table 1.

B.5 Different Similarity Metrics

we use the cosine similarity of the representations from the final layer outputs as a metric for LaCo. We also aim to explore the feasibility of using common distribution distances as metrics, such as KL divergence and kernel/linear CKA (Centered Kernel Alignment (Kornblith et al., 2019)). Specifically, we replace cosine similarity with distribution distances to measure the difference in representations. However, we find that the distribution distances almost remain constant, as in Table 14.

	KL Divergence	Kernel CKA	Linear CKA
Constant	0.00	1.00	1.00

Table 14: The distribution distances of model output representations.

KL Divergence being nearly 0 and CKA being nearly 1 indicate minimal differences in the distributions of the final layer outputs. This suggests that the LaCo merging process does not significantly alter the model output. Additionally, due to the high dimensionality of the representations, the distribution distances tend to be constant, resulting in a lack of discrimination. Therefore, we chose the more discriminative cosine similarity.

B.6 Drop or Merge

As shown in Eq. 1, LaCo merges multiple adjacent layers into one. This leads us to consider an extreme case: if we set $m = 1$, LaCo will no longer merge layers but simply drop a layer. We also aim to explore the performance differences between the drop and merge operations in LaCo. Thus, we conduct an experiment with $m = 1$ on Llama2-7B, setting the compression rate to the same 27% as in Table 1. The results on some benchmarks are shown in Table 15.

	BoolQ	PIQA	HeSw
Drop	52.57	68.17	48.61
Mege (Result in Table 1)	64.07	69.80	55.69

Table 15: Results of the drop or merge operation.

The results indicate that the drop operation is not as effective as the merge operation and $m = 1$ is not a good hyperparameter setting. The merge operation aligns better with our intention.

B.7 Iterative-based or Rule-based Merge

LLM	Strategy	BoolQ	PIQA	HeSw
Llama2-7B	Rule	63.49	68.72	53.16
	LaCo	**64.07**	**69.80**	**55.69**
Baichuan2-7B	Rule	52.94	67.28	48.78
	LaCo	**56.15**	**68.50**	**52.28**

Table 16: The results of using rule-based merging and LaCo iterative merging.

We want to determine if our iterative search-based merging strategy is superior to rule-based merging. To test this, we perform rule-based merging on Llama2-7B and Baichuan2-7B, both with 32 layers. We merge layers in groups of four, starting from the top, specifically merging layers (29, 30, 31, 32), (21, 22, 23, 24), and (13, 14, 15, 16). We avoid merging before the 16th layer due to significant performance drops observed in those cases. The resulting models achieved compression rates equivalent to those in Table 1. The results in Table 16 indicate that LaCo performs better than the rule-based approach. Notably, even simple rule-based merging can outperform baselines across multiple datasets, demonstrating the potential of merging for model compression.

B.8 Effect of Calibration Dataset from Different Sources

We explore whether calibration datasets from different sources will affect the performance.

Specifically, we select 10 sentences from the BookCorpus (Kobayashi, 2018) dataset used by LLM-Pruner as the calibration dataset. We prune Llama2-7B at a 27% compression rate and evaluate the results on several datasets. We repeat the experiments three times.

	BoolQ	PIQA	HeSw
Wikipedia	64.07	69.80	55.69
BookCorpus-Round1	64.15	69.60	55.87
BookCorpus-Round2	63.80	69.45	55.61
BookCorpus-Round3	64.30	70.08	55.98

Table 17: Results of calibration datasets from different sources.

The results in Table 17 are at the same level as those in our paper. This demonstrates that, given high-quality corpora, different calibration datasets have little impact on the performance of the models pruned with LaCo at the same compression rate.

B.9 Effect of Calibration Dataset Size

We want to explore how the size of the calibration dataset affects the performance of LaCo. Specifically, we take 5, 10, 50, 100, and 200 samples from Wikipedia and prune Llama2-7B at a 27% compression rate. The results are shown in the Table 18, and it can be seen that the size of the calibration dataset has little impact on the results.

	BoolQ	PIQA	HeSw
Result in Table 1 (10 samples)	64.07	69.80	55.69
5 samples	63.89	69.74	55.54
50 samples	64.10	69.69	55.81
100 samples	64.25	69.81	55.59
200 samples	63.98	69.92	55.62

Table 18: Results of different calibration dataset size.

Therefore, we chose a small calibration dataset to ensure both the pruning effect and the pruning speed. The size of our calibration dataset is the same as that used by the LLM-Pruner, which is 10 samples.

C Post-Training Implementation Details

We use the LLaMA-Factory (Zheng et al., 2024) framework along with DeepSpeed ZeRO-2. The sequence length is set to 4096, following the default settings for Llama2-7B and Baichuan2-7B. We use the Adam optimizer with a learning rate of 2e-4, setting $\beta_1 = 0.9$ and $\beta_2 = 0.95$. The batch size is 8 per GPU, resulting in a total batch size of 32, with gradient accumulation steps set to 4. We employ a cosine learning rate scheduler, apply a weight decay of 0.1, and set the maximum gradient normalization to 1.0.

D Supplementary Results (Part 1)

D.1 Results on Llama2-70B

We supplement the results using Llama2-70B with a compression rate set to 30%. We conduct experiments on CMNLI, HeSw, PIQA, and BoolQ, and Table 19 presents the average results.

Pruner	Dense	LaCo	LLM-Pruner	SliceGPT
Average	65.92	57.91	53.78	51.36

Table 19: The average scores on several benchmarks using different pruners with a 30% compression rate on Llama2-70B.

The results demonstrate that our method can also scale up effectively. It performs better than the baselines even on the 70B model.

D.2 Memory Consumption and Inference Speed

LLM	Pruner	Memory (MB)	Infer. (tokens/s)
Llama2 -7B	Dense	13410	38.53
	LLMPru.	10434	33.22 (↓)
	SliceGPT	11770	44.88
	LaCo	**9894**	**50.80**
Llama2 -13B	Dense	25902	29.98
	LLMPru.	19874	27.15 (↓)
	SliceGPT	22506	35.16
	LaCo	**19422**	**38.65**
Baic2. -7B	Dense	14810	37.13
	LLMPru.	11898	38.95 (↓)
	SliceGPT	13586	36.67
	LaCo	**11716**	**49.15**
Baic2. -13B	Dense	27410	36.93
	LLMPru.	22390	31.61 (↓)
	SliceGPT	23956	29.35 (↓)
	LaCo	**21010**	**47.46**

Table 20: The memory consumption and average inference speed on English Wikipedia dataset for different pruned models. ↓ means the performance worse than the Dense model.

D.3 Comparison of Post-trained Pruned Models

We prune Llama2-7B to 55% sparsity using LaCo and LLM-Pruner, then conduct post-training with

	GPU*hour	BoolQ	PIQA	HeSw
LaCo	**88**	**60.26**	**65.01**	**45.49**
LLM-Pru.	216	58.75	61.26	43.53

Table 21: Results on different datasets for models pruned to 55% sparsity using LaCo and LLM-Pruner on Llama2-7B, followed by post-training on the same data.

the same data in Table 6. The results in Table 21 show that LaCo performs better. Additionally, training the model pruned by LLM-Pruner requires 216 GPU*hours (27 hours on 8 A100 GPUs), while only 88 GPU*hours (22 hours on 4 A100 GPUs) for LaCo-pruned model. Thus, LaCo saves more computational resources for post-training.

D.4 Performance Percentage for Main Results

LLM	Pruner	Ratio/Lay.	Reasoning(%)			Language(%)			Knowledge(%)		Examination(%)		Understanding(%)			
			CMNLI	HeSw	PIQA	CHID	WSC$_P$	WSC$_G$	CSQA	BoolQ	MMLU	CMMLU	Race$_H$	Race$_M$	XSum	C3
Llama2 -7B	Dense*	0%/32	100	100	100	100	100	100	100	100	100	100	100	100	100	100
	LLMPru.	27.0%/32	104.09	**79.13**	**91.10**	54.84	97.44	2.50	63.76	78.11	50.81	79.25	63.53	67.42	58.49	58.57
	SliceGPT	26.4%/32	96.12	70.46	84.69	45.16	97.44	50.00	62.04	54.22	**62.98**	**79.63**	59.34	65.34	24.85	**90.86**
	LaCo	**27.1%/23**	**104.40**	78.05	89.28	**78.50**	**107.68**	**65.00**	**68.55**	**90.66**	57.60	79.22	**63.67**	**71.22**	**79.47**	90.61
Llama2 -13B	Dense*	0%/40	100	100	100	100	100	100	100	100	100	100	100	100	100	100
	LLMPru.	24.4%/40	**100.12**	**90.55**	**96.17**	67.28	79.24	0.00	76.01	78.91	45.32	63.78	38.72	36.65	**81.37**	68.05
	SliceGPT	23.6%/40	90.39	74.45	86.61	36.45	71.70	**57.58**	70.63	52.95	66.76	66.57	40.34	39.89	22.37	88.23
	LaCo	**24.6%/30**	99.61	86.05	93.18	**75.70**	**103.77**	56.07	**78.70**	**89.48**	**82.56**	**84.20**	**93.90**	**93.87**	61.33	**94.57**
Baic2. -7B	Dense*	0%/32	100	100	100	100	100	100	100	100	100	100	100	100	100	100
	LLMPru.	24.2%/32	96.73	**79.43**	**94.29**	84.43	**130.23**	0.00	**75.75**	**96.67**	45.95	45.11	41.73	43.65	**76.68**	64.51
	SliceGPT	22.2%/32	96.10	37.43	66.08	17.96	88.37	0.00	30.99	62.09	46.41	44.34	44.71	44.06	0.00	41.18
	LaCo	**24.2%/23**	**98.89**	77.38	89.93	**92.22**	102.32	**42.42**	74.85	88.70	**58.12**	**54.86**	**55.08**	**54.31**	57.73	**78.78**
Baic2. -13B	Dense*	0%/40	100	100	100	100	100	100	100	100	100	100	100	100	100	100
	LLMPru.	24.3%/40	**101.78**	75.34	**91.99**	87.50	90.69	0.00	59.18	84.39	39.43	41.10	31.47	31.35	**54.79**	60.77
	SliceGPT	22.8%/40	96.57	36.36	65.36	12.50	88.37	0.00	27.47	56.46	39.02	41.23	32.05	31.22	0.00	38.07
	LaCo	**24.7%/30**	99.46	**85.39**	88.23	**92.26**	**106.96**	**95.46**	**84.53**	**93.06**	**87.32**	**87.56**	**84.61**	**83.84**	49.38	**93.08**

Table 22: The percentage of each model's score on each benchmark relative to the score of Dense* in the main results. Models pruned by LaCo maintain performance above 70% on most benchmarks and avoid crashes, with no performance falling below 30%.

D.5 Examples of Responses

Prompt	Document: The 18-year-old scored 88.40 to make history in what was the fifth and the final stop of the World Cup season.\nShe came ahead of Sweden's Emma Dahlstrom and Swiss Mathilde Gremaud.\nBoston-born Atkin, who initially competed for the US before switching to Great Britain aged 15, was making her 15th appearance at a World Cup event.\nAtkin will be competing at the Freestyle World Championships in Sierra Nevada, Spain (9-19 March). The event will be live on the BBC Sport website, app, connected TV and red button.\nBased on the previous text, provide a brief single summary:
Pruner	**Generated Responses**
LLMPru.	\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n
SliceGPT	of the 19900s of the 1900s of the 1900s of the 1900s.
LaCo	Boston-born Atkin, who initially competed for the US before switching to Britain aged 15, was making her 15th appearance at a World Cup event.\nThe 18-year-old scored 88.40 to make history in what was the fifth and the final stop of the World Cup season.

Table 23: A response on the Xsum benchmark from Llama2-7B after pruning with different pruners. In this case, the models pruned by the baseline pruners generate repetitive and meaningless text, while only LaCo is able to smoothly respond with meaningful text according to the instructions.

E Supplementary Results (Part 2)

LLM	Method	Reasoning			Language			Knowledge		Examination		Understanding			
		CMNLI	HeSw	PIQA	CHID	WSC$_P$	WSC$_G$	CSQA	BoolQ	MMLU	CMMLU	Race$_H$	Race$_M$	XSum	C3
Llama2 -7B	Dense*	32.98	71.35	78.18	46.04	37.50	38.46	66.67	70.67	45.92	31.86	35.51	33.15	19.68	43.78
	LaCo	34.43	55.69	69.80	36.14	**40.38**	25.00	45.70	64.07	26.45	25.24	22.61	23.61	**15.64**	**39.67**
	LaCo +post train	**34.92**	**61.88**	**73.18**	**38.12**	36.54	**34.62**	**57.49**	**66.21**	**29.47**	**25.33**	**28.33**	**29.87**	10.02	38.58
	LaCo +post train +re prune	33.80	45.35	65.07	23.27	36.54	0.96	38.49	60.43	26.07	25.37	23.07	22.98	15.48	36.71
Baichuan2 -7B	Dense*	33.37	67.56	76.17	82.67	41.35	63.46	63.14	63.30	54.25	56.95	52.63	51.04	20.84	64.55
	LaCo	**33.00**	52.28	68.50	76.24	**42.31**	**26.92**	47.26	56.15	**31.53**	**31.24**	**28.99**	**27.72**	12.03	50.85
	LaCo +post train	32.92	**52.67**	**69.42**	**78.22**	40.38	3.85	**52.01**	**55.93**	28.72	27.25	25.01	26.25	**15.82**	**58.03**

Table 24: The detailed scores across all benchmarks of pruned models, post-trained models, as well as post-trained models followed by re-pruning.

LLM	Ratio/Lay.	Reasoning			Language			Knowledge		Examination		Understanding			
		CMNLI	HeSw	PIQA	CHID	WSC$_P$	WSC$_G$	CSQA	BoolQ	MMLU	CMMLU	Race$_H$	Race$_M$	XSum	C3
Llama2 -7B	0%/32	32.98	71.35	78.18	46.04	37.50	38.46	66.67	70.67	45.92	31.86	35.51	33.15	19.68	43.78
	12.0%/28	32.99	**55.91**	**74.48**	**42.57**	36.54	**29.81**	**52.58**	60.12	25.59	**27.10**	22.01	21.73	**17.97**	36.44
	27.1%/23	**34.43**	55.69	69.80	36.14	**40.38**	25.00	45.70	**64.07**	**26.45**	25.24	**22.61**	**23.61**	15.64	**39.67**
	45.0%/17	32.58	38.33	60.07	20.30	36.54	0.96	34.73	61.38	23.98	25.59	22.38	23.26	1.28	38.85
Llama2 -13B	0%/40	32.99	74.83	79.71	52.97	50.96	63.46	66.91	71.50	55.63	38.74	58.03	60.24	23.56	47.51
	14.6%/34	32.99	**71.88**	**76.82**	**51.98**	**63.46**	**48.08**	**63.72**	63.43	**53.97**	**38.23**	**59.35**	**61.49**	**21.32**	**47.67**
	24.7%/30	32.86	64.39	74.27	40.10	52.88	35.58	52.66	**63.98**	45.93	32.62	54.49	56.55	14.45	44.93
	49.7%/20	**34.22**	46.55	63.82	13.37	56.73	10.58	36.28	62.23	38.41	27.24	51.97	56.41	1.56	36.38

Table 25: The detailed results of models pruned at different pruning ratios using LaCo across all benchmarks. As the pruning ratio increases, overall model performance decreases. However, performance remains stable from 10-25% pruning. Even at 50% pruning, the model can maintain about 70% performance.