GPTailor: Large Language Model Pruning Through Layer Cutting and Stitching

Guinan Su [1] **Li Shen** [2] **Lu Yin** [3] **Shiwei Liu** [4] **Yanwu Yang** [5] **Jonas Geiping** [1 6 7]

Abstract

Large language models (LLMs) have shown remarkable capabilities in language understanding and generation. However, such impressive capability typically comes with a substantial model size, which presents significant challenges in deployment and inference. While structured pruning of model parameters offers a promising way to reduce computational costs at deployment time, current methods primarily focus on single model pruning. In this work, we develop a novel strategy to compress models by strategically combining or merging layers from finetuned model variants, which preserves the original model's abilities by aggregating capabilities accentuated in different finetunes. We pose the optimal tailoring of these LLMs as a zero-order optimization problem, adopting a search space that supports three different operations: (1) Layer removal, (2) Layer selection from different candidate models, and (3) Layer merging. Our experiments demonstrate that this approach leads to competitive model pruning, for example, for the Llama2-13B model families, our compressed models maintain approximately 97.3% of the original performance while removing $\sim$ 25% of parameters, significantly outperforming previous state-of-the-art methods.

1. Introduction

The unique strengths of modern Large Language Models (LLMs) in language understanding, generation, and reasoning (Touvron et al., 2023; OpenAI et al., 2023; Chiang et al., 2023) are inextricably linked to their immense size. Research in this field has generally followed a trajectory of scaling model parameters and data to enhance performance, guided by two fundamental principles: scaling laws, which establish that performance improves predictably with increased parameters (Kaplan et al., 2020; Hoffmann et al., 2022; Wei et al., 2022), and over-parameterization theory, which demonstrates that models with excess parameters achieve better optimization and generalization (Allen-Zhu et al., 2019a;b; Li et al., 2020). These principles have led researchers to develop billion-parameter architectures delivering unprecedented performance across diverse tasks.

Despite these impressive capabilities, deploying LLMs presents significant challenges due to their substantial computational demands. Various post-training techniques have been proposed to address the issues faced when deploying models to consumer GPUs or local devices, or when reducing costs, including model pruning (Frantar & Alistarh, 2023; Dettmers et al., 2023b; Xia et al., 2023; Kim et al., 2024; Ma et al., 2023), knowledge distillation into smaller models (Chen et al., 2022; Hsieh et al., 2023; Shridhar et al., 2023; Tunstall et al., 2023), and quantization of weights (Yao et al., 2022; Gholami et al., 2022; Dettmers et al., 2023a). While quantization reduces parameter precision but requires specific hardware support, and knowledge distillation necessitates costly retraining, structured pruning offers a flexible and hardware-agnostic approach by eliminating redundant parameters to decrease computation costs.

Existing pruning methods typically focus on pruning individual models through manually designing metrics that assess the importance of specific structures or layers based on hidden state changes or gradient information (Kim et al., 2024; Men et al., 2024; Ma et al., 2023). However, these approaches inevitably cause performance degradation and require additional post-training to recover performance.

To address these limitations, we take a radically different perspective and re-formulate structured pruning as the problem of *pruning not individual models, but a family of task-specific finetuned versions of a given model*. These finetuned variants are surprisingly helpful for model pruning, as each variant accentuates a particular task, such as coding, math, or language understanding. Further, the variants are close enough that model merging can be employed to re-combine layers from multiple variants, if needed (Worts-

[1]Max Planck Institute for Intelligent Systems [2]Sun Yat-sen University [3]University of Surrey [4]University of Oxford [5]University of Tübingen [6]ELLIS Institute Tübingen [7]Tübingen AI Center. Correspondence to: Li Shen <shenli6@mail.sysu.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

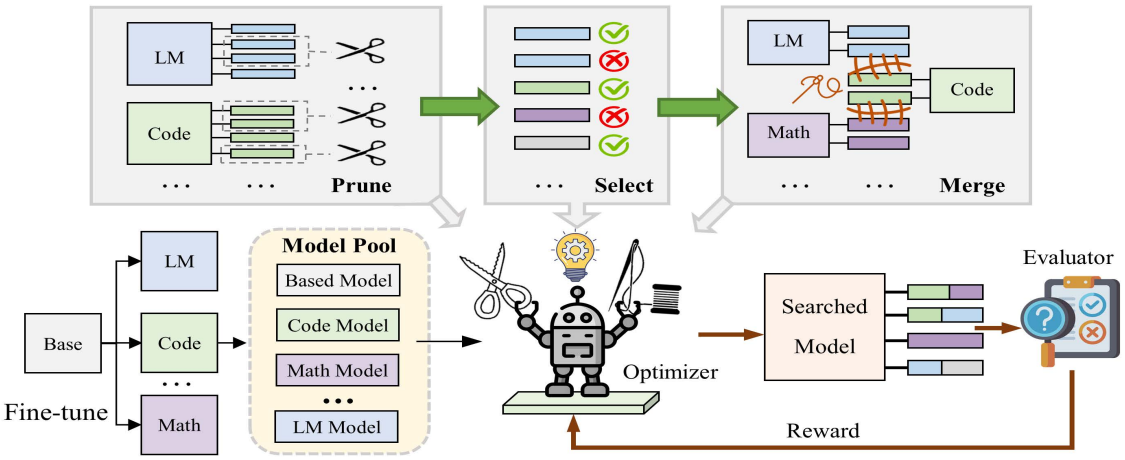


Figure 1. Our Approach: **Model Pruning through Cutting and Stitching**. We achieve competitive model pruning performance by running a zero-order search that tailors layers based on a shared pool of finetuned variants of the original model, selecting and stitching layers if necessary. The model finetunes accentuate task-specific skills, allowing us to merge key components into a smaller model, maintaining, for example, 97% of capabilities of Llama-13B, even after a 25% reduction in layers.

man et al., 2022). These observations lead us to our main question: **Can we develop better compressed models by strategically combining or merging layers from different models?** Motivated by this question, we propose a novel structured pruning method based on zero-order optimization that supports three different operations to combine layers from different models into a smaller, more efficient model: **(1) Layer removal, (2) Layer selection from related candidate models, (3) Layer merging**.

For the optimization, we define multiple objective functions that capture different aspects of model performance across different tasks to better preserve the original model's capabilities and run a fully data-driven zero-order optimization, instead of relying on expert-made heuristics for pruning. We employ SMAC (Lindauer et al., 2022), which strategically allocates computational resources by evaluating configurations at different calibration data sizes, thereby reducing computational costs while boosting the efficiency of finding superior solutions. We rigorously validate our method's effectiveness by evaluating it on Llama-7B and Llama-13B with four state-of-the-art structural pruning methods across comprehensive benchmarks. Our experimental results demonstrate that our approach maintains excellent performance while outperforming existing pruning methods.

In summary, the main contributions of this paper are:

- We propose a novel structured pruning method that formulates pruning as a zero-order optimization problem over a pool of candidate models, enabling automated discovery of efficient models that leverage capabilities from multiple models.
- We find that this approach allows for a cost-effective model pruning stage that is effective without the need for post-training to heal the pruned model.
- We validate our method's effectiveness through extensive experiments, comparing against modern LLM pruning methods on 14 benchmark tasks.

Our method maximally preserves the capabilities of the dense model: 92.2% for the 7B model and 97.3% for the 13B model. significantly outperforming previous methods.

2. Related Work

Compression of Language Models. Large language models (Touvron et al., 2023; OpenAI et al., 2023) require compression to reduce parameters and latency. We focus on structural pruning, with recent methods including LLM-Pruner (Ma et al., 2023), SliceGPT (Ashkboos et al., 2024), LaCo (Yang et al., 2024) and ShortGPT (Men et al., 2024). our work employs zero-order search to combine pruning and merging across model families.

Model Merging. Model merging enhances capabilities without additional training, evolving from weighted parameter averaging (Utans, 1996) to Task Arithmetic (Ilharco et al., 2022). Recent approaches leverage sparsity: TIES-Merging (Yadav et al., 2024) selects parameters by magnitude while resolving sign conflicts, and DARE (Yu et al., 2024) combines sparsification with rescaling. Evolutionary model merging (Akiba et al., 2024) optimizes coefficients through evolutionary search, while multi-fidelity approaches enable efficient fine-grained exploration (Su & Geiping, 2025). Our work builds upon multi-fidelity optimization for efficient compressed model search.

3. Methods

We reformulate model compression as a zero-order optimization problem selecting and merging layers across multiple candidate models. An overview is provided in Figure 1.

Problem Setup Given a base model M_{base} and candidate models $\mathcal{M} = \{M_1, M_2, ..., M_K\}$ fine-tuned from it, we aim to find an optimal pruned model maximizing performance under a target sparsity constraint $s \in [0, 1]$. The pruned model combines layers from candidate models through merging, selection, and removal operations, determined by hyperparameters $\omega \in \Omega$. Each configuration ω defines how to form a pruned model M_ω, with performance evaluated by function $f(M_\omega)$. Our optimization problem is: $\omega^* = \arg\min_{\omega\in\Omega} f(M_\omega)$ subject to $\mathrm{S}(M_\omega) \leq s$ where $\mathrm{S}(\cdot)$ calculates the fraction of pruned parameters compared to the base model.

Search Space Design For a base model with l layers and K candidate models, we design the search space through: **(1)** A binary vector $\mathbf{r} = [r_1, r_2, \ldots, r_l]$ where $r_i \in \{0, 1\}$ indicates layer retention ($r_i = 0$) or removal, with $\sum_{i=1}^{l} r_i = \lceil l \cdot s \rceil$ satisfying target sparsity. **(2)** For each retained layer i, a selection vector $\mathbf{c}_i = [c_{i,1}, c_{i,2}, \ldots, c_{i,K}]$ where $c_{i,j} \in \{0, 1\}$ indicates whether the j-th candidate model's layer is selected. If $\sum_{j=1}^{K} c_{i,j} = 0$, we use the base model's layer. **(3)** When multiple candidates contribute ($\sum_{j=1}^{K} c_{i,j} > 1$), we specify a merge method $m_i \in \{1, 2, \ldots, Z\}$ with hyperparameters $\mathbf{h}_i = [h_{i,1}, h_{i,2}, \ldots, h_{i,P_i}]$ controlling the combination mechanism.

Target Objective Function We define a multi-objective function measuring model effectiveness across calibration datasets $\mathcal{D}_{\text{calibration}}$. Given tasks $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$, we employ Pareto Efficient Global Optimization (ParEGO) (Knowles, 2006) to identify Pareto-optimal solutions through scalarization:

$$f_{multi}(M_\omega, \lambda) = \max_{i=1,\ldots,m} \{\lambda_i \cdot f_i(M_\omega)\} + \alpha \sum_{i=1}^{m} \lambda_i \cdot f_i(M_\omega) \tag{1}$$

where $f_i(M_\omega)$ is the i-th objective, λ_i are weights satisfying $\sum_{i=1}^{m} \lambda_i = 1$ and $\lambda_i \geq 0$. The optimizer outputs a Pareto front of configurations representing different task performance trade-offs.

Search Optimizer We employ SMAC (Lindauer et al., 2022) to efficiently navigate the search space, using calibration dataset size as fidelity represented by budgets b where $b_{\text{min}} \leq b \leq b_{\text{max}}$. Smaller budgets use fewer samples for faster evaluations, while larger budgets use more samples for greater reliability. We use Random Forest (Breiman, 2001) as a surrogate model to sample new configurations. The complete process is described in Algorithm 1.

4. Experiments

4.1. Experimental Settings

Benchmarks. We evaluate using OpenCompass (Contributors, 2023) across five aspects: Reasoning (CMNLI (Xu et al., 2020), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020)), Language (CHID (Zheng et al., 2019), WSC (Levesque et al., 2012)), Knowledge (CSQA (Talmor et al., 2018), BoolQ (Clark et al., 2019)), Examination (MMLU (Hendrycks et al., 2020), CMMLU (Li et al., 2023)), and Understanding (Race-H/M (Lai et al., 2017), XSum (Narayan et al., 2018), C3 (Sun et al., 2020)). We use generative evaluation for CHID and XSum, both cloze and generative for WSC, and cloze for the rest. Details in Appendix C.

Baselines. We compare with LLM-Pruner (Ma et al., 2023), SliceGPT (Ashkboos et al., 2024), LaCo (Yang et al., 2024), and ShortGPT (Men et al., 2024), ensuring comprehensive comparison through three scenarios: (1) pruning each candidate model separately, (2) pruning then merging, and (3) merging then pruning. All merging uses task-arithmetic (Ilharco et al., 2022) with factors in $[0.5, 1.0]$.

Model Selection. We evaluate on Llama2-7B and Llama2-13B (Touvron et al., 2023). For 7B, our base model is Llama-2-7B with three candidates: Llama-2-7B-Chat (Touvron et al., 2023), MAmmoTH-7B (Yue et al., 2023), and Llama-2-Coder-7B (Manuel Romero, 2023). For 13B, we use Llama-2-13B with WizardLM-13B (Xu et al., 2023), WizardMath-13B (Luo et al., 2023), and Llama-2-13B-Code-Alpaca (Chaudhary, 2023). We remove 28% (9/32) of layers for 7B and 25% (10/40) for 13B, matching the setting in ShortGPT and LaCo for fair comparison.

Calibration Data. Our calibration dataset includes 1000 examples from PIQA training set, 500 from WSC training set, 1000 from CSQA training set, and 1000 from MMLU validation set, ensuring diverse capability coverage.

Objective and Optimizer. We implement SMAC (Lindauer et al., 2022) with 500 search trials. To improve efficiency, we start with models with randomly removed middle layers (Su & Geiping, 2025). We set $b_{\text{min}} = 100$, $b_{\text{max}} = 1000$, and $\eta = 3$, resulting in budgets of {100, 300, 1000} for PIQA, CSQA, and MMLU, and {100, 200, 500} for WSC.

4.2. Main Results

To validate our method, we compared it with the four baselines: LLM-Pruner (LLMPru) (Ma et al., 2023), SliceGPT (Ashkboos et al., 2024), LaCo (Yang et al., 2024), and ShortGPT (Men et al., 2024). We reproduce results and evaluate on OpenCompass (Contributors, 2023). As mentioned, to validate that our proposed approach of *"pruning while merging"* is optimal, we also re-run each pruning method on (1) pruning each candidate model individually and picking the best, (2) "pruning-then-merging": First pruning each candidate model using the baseline method and then merging them, and (3) "merging-then-pruning": First merging the candidate models and then applying pruning. Table 1 reports the best single model pruning and best merge results of all baselines, with full results in Appendix E.

Table 1. Comparison of pruning methods on multiple natural language benchmarks. "Single" refers to the best performance achieved when pruning a single model directly, while "Merge" refers to the best performance achieved through either "pruning-then-merging" or "merging-then-pruning". For 7b model: Llama-2-7B-Chat (LM), MAmmoTH-7B (Math), Llama-2-Coder-7B (Code), and Llama-2-7B (Base), for 13b model: WizardLM-13B (LM), WizardMath-13B (Math), llama-2-13b- code-alpaca (Code), and Llama-2-13B (Base). The cells highlighted in blue show three selected Pareto-optimal solutions of our method.

LLM	Pruner (ratio)	Type	Reasoning			Language			Knowledge				Understanding				Avg	Avg*
			CNLI	HeSw	PIQA	CHID	WSC_P	WSC_G	CSQA	BoolQ	MMLU	CMLU	$Race_H$	$Race_M$	XSum	C3		
Llama -7B	Dense (0.0%)	Base	32.98	71.34	78.18	41.56	37.50	38.46	55.04	70.70	46.67	31.88	35.53	33.36	19.55	43.84	45.47	42.30
		Math	32.99	68.60	75.79	39.71	39.42	36.54	50.78	69.36	43.04	32.16	30.36	36.42	20.88	43.45	44.25	41.70
		LM	31.30	71.28	75.95	36.11	63.46	59.62	64.29	74.77	48.30	33.93	52.52	55.22	22.45	47.56	52.63	47.24
		Code	32.99	70.27	78.62	41.61	36.54	41.35	57.41	71.04	46.22	32.20	41.25	39.69	18.79	46.25	46.73	43.79
	LLMPru (25.3%)	Single	32.99	**59.57**	**73.34**	**30.32**	46.15	0.00	20.15	57.28	23.21	25.16	21.56	21.52	**15.19**	31.07	32.68	32.74
		Merge	34.71	**60.57**	**73.50**	26.62	40.38	5.77	19.90	52.14	24.01	25.30	23.07	22.98	**15.51**	32.49	32.64	32.60
	SliceGPT (26.3%)	Single	31.89	41.55	58.81	18.43	39.42	4.81	19.49	40.09	25.38	25.02	25.59	26.88	8.78	39.56	28.98	28.64
		Merge	32.85	37.61	57.56	17.33	53.85	2.88	19.41	42.66	25.22	24.68	25.21	24.72	12.78	40.22	29.78	28.67
	LaCo (27.1%)	Single	32.97	55.24	69.53	**31.47**	36.54	34.62	22.11	67.22	29.08	26.16	28.53	28.27	14.68	**43.51**	37.14	36.45
		Merge	31.89	56.26	**71.22**	**27.32**	39.42	22.12	23.42	72.66	29.30	26.00	25.19	26.81	**16.11**	**43.62**	36.52	36.21
	ShortGPT (27.1%)	Single	33.09	57.42	66.54	21.53	56.73	**48.08**	52.50	67.34	43.68	28.31	32.53	31.69	12.40	39.45	42.24	35.97
		Merge	34.10	54.18	64.42	16.83	61.54	36.54	55.61	**73.21**	36.84	25.61	42.94	45.89	10.12	35.73	42.40	37.62
	Ours (27.1%)		**35.46**	54.43	67.74	23.63	**63.46**	**43.27**	**62.90**	**75.08**	**48.75**	**33.86**	**55.35**	**58.64**	12.99	**44.16**	48.55	43.73
			34.94	**58.14**	69.48	21.53	**63.46**	41.35	**62.74**	66.24	**47.39**	**34.11**	**49.17**	**50.56**	3.46	41.53	46.01	39.96
			34.95	54.92	67.08	24.48	**63.46**	**46.15**	**62.00**	**75.90**	**48.73**	**34.13**	**54.03**	**57.45**	13.20	43.01	48.54	43.56
Llama -13B	Dense (0.0%)	Base	32.99	74.77	79.71	47.35	50.96	63.46	67.24	71.38	55.84	38.74	57.98	60.17	23.47	47.51	55.11	50.48
		LM	35.36	70.41	78.73	36.21	57.69	60.58	65.03	73.70	53.48	30.85	66.12	71.66	22.44	52.00	55.30	50.97
		Math	32.99	68.78	77.26	44.36	36.54	19.23	60.36	78.44	54.21	38.12	47.74	48.82	19.51	44.66	47.93	47.05
		Code	32.99	74.82	80.14	47.30	51.92	63.46	68.88	72.72	55.92	39.26	58.03	63.72	24.45	48.38	55.86	51.30
	LLMPru (21.2%)	Single	**33.49**	60.28	**75.57**	23.68	39.42	0.00	19.00	63.24	23.27	25.23	22.36	21.45	**17.13**	32.00	32.58	33.21
		Merge	**33.86**	64.11	**73.50**	22.18	**60.58**	0.00	21.46	61.96	23.84	25.62	22.16	21.59	14.98	32.11	34.14	33.17
	SliceGPT (23.6%)	Single	**33.19**	42.44	59.90	18.03	54.81	19.23	32.51	41.22	33.09	25.75	29.45	29.87	9.99	37.75	33.37	29.74
		Merge	30.98	46.83	62.57	19.33	51.92	49.04	37.76	38.38	33.55	25.22	23.53	23.05	9.95	39.67	35.13	28.55
	LaCo (24.6%)	Single	32.33	60.18	70.57	**32.67**	34.62	34.62	52.58	62.66	36.26	25.80	60.38	62.53	8.79	**49.21**	44.51	43.84
		Merge	**33.49**	62.50	74.37	**35.26**	**63.46**	**63.46**	18.84	64.65	41.83	24.87	26.10	25.97	15.93	39.51	42.16	34.71
	ShortGPT (24.6%)	Single	32.95	62.64	**73.50**	28.22	36.54	50.96	65.44	67.71	53.50	30.73	**65.52**	**71.38**	**19.12**	**48.60**	50.49	47.43
		Merge	31.07	63.24	68.61	27.17	49.04	43.27	65.68	**78.01**	51.26	36.88	57.38	62.67	**16.94**	44.05	49.66	46.38
	Ours (24.6%)		32.99	**66.81**	**75.03**	29.07	54.81	**62.50**	**69.37**	**74.28**	**55.90**	**39.71**	**65.52**	**71.03**	16.80	46.74	54.33	49.22
			31.80	**68.63**	72.52	30.97	**60.58**	**55.77**	**67.49**	**73.70**	**54.61**	**39.29**	61.92	**70.13**	16.19	48.11	53.69	48.97
			29.05	**69.76**	72.74	**34.22**	**58.65**	54.81	**68.06**	69.82	**53.99**	**38.36**	**62.32**	66.71	16.60	**51.01**	53.29	48.65

Our approach achieves the best results across multiple benchmarks compared to all tested LLM pruning methods. In terms of overall performance, our method maximally preserves the capabilities of the dense model: 92.2% (48.55/52.63) for the 7B model and 97.3% (54.33/55.86) for the 13B model. To ensure our results were not biased by our calibration data, we also calculate an avg* excluding the four benchmarks from which training data was selected for calibration (MMLU, CSQA, WSC, PIQA). As shown in the avg* column, our method still outperformed all baselines, further validating our approach. Notably, our method achieved comparable or even better results than dense models on many benchmarks. We attribute these gains to: 1) Pruning might mitigate "overthinking" (Kaya et al., 2019) effects, evident in benchmarks like CNLI and WSC where other baseline pruning methods also improved performance, and 2) our merging strategy is effectively compensating for information loss from pruning.

Figure 2 illustrates our best-performing 7B-pruned model and best-performing 13B-pruned models's structure (See Table 12 for architectural details). We observe that both models tend to remove middle-to-later layers, with the 13B model removing layers from layer 25 and the 7B model from layer 19. This suggests information redundancy in these layers, aligning with findings that later layers exhibit high similarity and redundancy (Men et al., 2024; Gromov et al., 2024). The 13B model shows a simpler structure dominated by a single LM model with concentrated layer removal, while the 7B model shows a more complex structure utilizing mixed and specialized models with scattered layer removal. This suggests that as model size decreases, more diverse mixing strategies may be needed to maintain performance. This architectural difference, coupled with the superior preservation rate of the 13B model compared to the 7B model, demonstrates that robustness (redundancy) scales with model size. More results in Appendix A.

5. Conclusions

We present a novel LLM compression approach combining layers from fine-tuned model variants rather than pruning single models. By formulating this as zero-order optimization with a search space supporting layer removal, selection and merging, we effectively preserve capabilities while reducing size. Experiments show our compressed Llama2-7B and Llama2-13B models retain 92.2% and 97.3% of original performance despite removing 25% of parameters, outperforming previous methods without expensive post-training. Our work demonstrates that cutting and stitching layers from multiple fine-tuned variants is more effective than traditional single-model pruning.

Acknowledgements

The authors thank Alexander Panfilov and Niccolò Ajroldi for their insightful comments and valuable feedback on this work. JG acknowledges the support of the Hector II foundation. GS acknowledges the support of the International Max Planck Research School for Intelligent Systems.

References

Ajibawa. Code-llama-3-8b, 2023. URL https://huggingface.co/ajibawa-2023/Code-Llama-3-8B.

Akiba, T., Shing, M., Tang, Y., Sun, Q., and Ha, D. Evolutionary optimization of model merging recipes. *arXiv preprint arXiv:2403.13187*, 2024.

Allen-Zhu, Z., Li, Y., and Liang, Y. Learning and generalization in overparameterized neural networks, going beyond two layers. *Advances in neural information processing systems*, 32, 2019a.

Allen-Zhu, Z., Li, Y., and Song, Z. A convergence theory for deep learning via over-parameterization. In *International conference on machine learning*, pp. 242–252. PMLR, 2019b.

Ashkboos, S., Croci, M. L., Nascimento, M. G. d., Hoefler, T., and Hensman, J. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*, 2024.

Bisk, Y., Zellers, R., Gao, J., Choi, Y., et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Breiman, L. Random forests. *Machine learning*, 45:5–32, 2001.

Chaudhary, S. Code alpaca: An instruction-following llama model for code generation. https://github.com/sahil280114/codealpaca, 2023.

Chen, Z., Gao, Q., Bosselut, A., Sabharwal, A., and Richardson, K. Disco: Distilling counterfactuals with large language models. *arXiv preprint arXiv:2212.10534*, 2022.

Chiang, W.-L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J. E., et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6, 2023.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

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

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023a.

Dettmers, T., Svirschevski, R., Egiazarian, V., Kuznedelev, D., Frantar, E., Ashkboos, S., Borzunov, A., Hoefler, T., and Alistarh, D. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023b.

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

Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., and Keutzer, K. A survey of quantization methods for efficient neural network inference. In *Low-power computer vision*, pp. 291–326. Chapman and Hall/CRC, 2022.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Gromov, A., Tirumala, K., Shapourian, H., Glorioso, P., and Roberts, D. A. The unreasonable ineffectiveness of the deeper layers. *arXiv preprint arXiv:2403.17887*, 2024.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Hsieh, C.-Y., Li, C.-L., Yeh, C.-K., Nakhost, H., Fujii, Y., Ratner, A., Krishna, R., Lee, C.-Y., and Pfister, T. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301*, 2023.

Ilharco, G., Ribeiro, M. T., Wortsman, M., Gururangan, S., Schmidt, L., Hajishirzi, H., and Farhadi, A. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and

Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kaya, Y., Hong, S., and Dumitras, T. Shallow-deep networks: Understanding and mitigating network overthinking. In *International conference on machine learning*, pp. 3301–3310. PMLR, 2019.

Kim, B.-K., Kim, G., Kim, T.-H., Castells, T., Choi, S., Shin, J., and Song, H.-K. Shortened llama: A simple depth pruning for large language models. *arXiv preprint arXiv:2402.02834*, 11, 2024.

Knowles, J. Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE transactions on evolutionary computation*, 10(1):50–66, 2006.

Lai, G., Xie, Q., Liu, H., Yang, Y., and Hovy, E. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*, 2017.

Levesque, H. J., Davis, E., and Morgenstern, L. The winograd schema challenge. *KR*, 2012:13th, 2012.

Li, H., Zhang, Y., Koto, F., Yang, Y., Zhao, H., Gong, Y., Duan, N., and Baldwin, T. Cmmlu: Measuring massive multitask language understanding in chinese. *arXiv preprint arXiv:2306.09212*, 2023.

Li, Z., Wallace, E., Shen, S., Lin, K., Keutzer, K., Klein, D., and Gonzalez, J. Train big, then compress: Rethinking model size for efficient training and inference of transformers. In *International Conference on machine learning*, pp. 5958–5968. PMLR, 2020.

Lindauer, M., Eggensperger, K., Feurer, M., Biedenkapp, A., Deng, D., Benjamins, C., Ruhkopf, T., Sass, R., and Hutter, F. Smac3: A versatile bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(54):1–9, 2022. URL http://jmlr.org/papers/v23/21-0888.html.

Lu, Z., Zhou, A., Lu, Z., Luo, S., Shi, W., Zhang, R., Song, L., Zhan, M., and Li, H. Mathcoder: Seamless code integration in LLMs for enhanced mathematical reasoning. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=z8TW0ttBPp.

Lu, Z., Zhou, A., Wang, K., Ren, H., Shi, W., Pan, J., Zhan, M., and Li, H. Mathcoder2: Better math reasoning from continued pretraining on model-translated mathematical code, 2024b. URL https://arxiv.org/abs/2410.08196.

Luo, H., Sun, Q., Xu, C., Zhao, P., Lou, J., Tao, C., Geng, X., Lin, Q., Chen, S., and Zhang, D. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023.

Ma, X., Fang, G., and Wang, X. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

Manuel Romero. llama-2-coder-7b (revision d30d193), 2023. URL https://huggingface.co/mrm8488/llama-2-coder-7b.

Men, X., Xu, M., Zhang, Q., Wang, B., Lin, H., Lu, Y., Han, X., and Chen, W. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*, 2024.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models, 2016.

Narayan, S., Cohen, S. B., and Lapata, M. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*, 2018.

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Rae, J. W., Potapenko, A., Jayakumar, S. M., and Lillicrap, T. P. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

Shridhar, K., Stolfo, A., and Sachan, M. Distilling reasoning capabilities into smaller language models. *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 7059–7073, 2023.

Su, G. and Geiping, J. Fine, i'll merge it myself: A multi-fidelity framework for automated model merging. *arXiv preprint arXiv:2502.04030*, 2025.

Sun, K., Yu, D., Yu, D., and Cardie, C. Investigating prior knowledge for challenging chinese machine reading comprehension. *Transactions of the Association for Computational Linguistics*, 8:141–155, 2020.

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

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E.,

Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Tunstall, L., Beeching, E., Lambert, N., Rajani, N., Rasul, K., Belkada, Y., Huang, S., Von Werra, L., Fourrier, C., Habib, N., et al. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*, 2023.

Utans, J. Weight averaging for neural networks and local resampling schemes. In *Proc. AAAI-96 Workshop on Integrating Multiple Learned Models. AAAI Press*, pp. 133–138. Citeseer, 1996.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. PMLR, 2022.

Xia, M., Gao, T., Zeng, Z., and Chen, D. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*, 2023.

Xu, C., Sun, Q., Zheng, K., Geng, X., Zhao, P., Feng, J., Tao, C., and Jiang, D. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*, 2023.

Xu, L., Hu, H., Zhang, X., Li, L., Cao, C., Li, Y., Xu, Y., Sun, K., Yu, D., Yu, C., et al. Clue: A chinese language understanding evaluation benchmark. *arXiv preprint arXiv:2004.05986*, 2020.

Yadav, P., Tam, D., Choshen, L., Raffel, C. A., and Bansal, M. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36, 2024.

Yang, Y., Cao, Z., and Zhao, H. Laco: Large language model pruning via layer collapse. *arXiv preprint arXiv:2402.11187*, 2024.

Yao, Z., Yazdani Aminabadi, R., Zhang, M., Wu, X., Li, C., and He, Y. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35: 27168–27183, 2022.

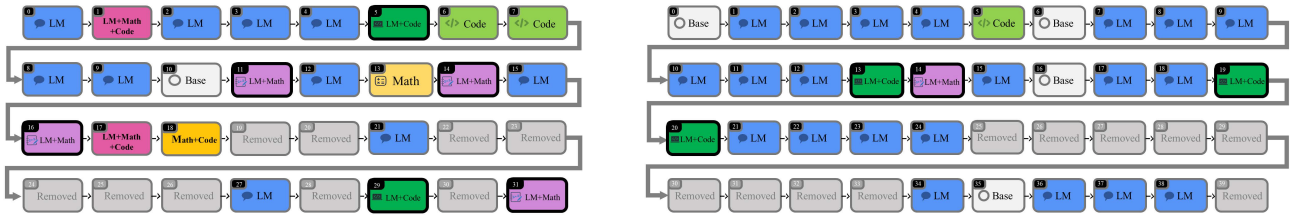
Yu, L., Yu, B., Yu, H., Huang, F., and Li, Y. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*, 2024.

Yue, X., Qu, X., Zhang, G., Fu, Y., Huang, W., Sun, H., Su, Y., and Chen, W. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*, 2023.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Zheng, C., Huang, M., and Sun, A. Chid: A large-scale chinese idiom dataset for cloze test. *arXiv preprint arXiv:1906.01265*, 2019.

A. Additional results



a) Structure of the best 7B-pruned model. b) Structure of the best 13B-pruned model.

Figure 2. (a) Structure of our best-performing 7B-pruned model. The model integrates layers from multiple candidates: Llama-2-7B-Chat (LM), MAmmoTH-7B (Math), Llama-2-Coder-7B (Code), and Llama-2-7B (Base). The pruning ratio is 9/32, removing 9 layers out of 32 total layers. (b) Structure of our best-performing 13B-pruned model. The model integrates layers from multiple candidates: WizardLM-13B (LM), WizardMath-13B (Math), llama-2-13b-code-alpaca (Code), and Llama-2-13B (Base). The pruning ratio is 10/40, removing 10 layers out of 40 total layers.

Algorithm 1 The optimization process of pruning.

Require: Configuration space Ω, minimum budget $b_{\min}$, maximum budget $b_{\max}$, reduction factor η, maximum trials $T_{\max}$
Ensure: Optimized configuration ω^*

$s_{\max} = \lfloor \log_\eta \frac{b_{\max}}{b_{\min}} \rfloor$, $D \leftarrow \emptyset$, $T \leftarrow 0$ {Initialization}
for $s \in \{s_{\max}, s_{\max}-1, \ldots, 0\}$ and $T < T_{\max}$ **do**
  $n \leftarrow \lceil \frac{(s_{\max}+1)}{(s+1)} \cdot \eta^s \rceil$, $r \leftarrow b_{\min} \cdot \eta^s$ {Config count & budget}
  $\mathcal{C} \leftarrow$ Sample Configurations(n, D, Ω) {Sample configurations}
  for $i \in \{0, 1, \ldots, s\}$ and $T < T_{\max}$ **do**
    $n_i \leftarrow \lfloor n \cdot \eta^{-i} \rfloor$, $r_i \leftarrow r \cdot \eta^i$ {Stage parameters}
    for each $w \in \mathcal{C}$ and $T < T_{\max}$ **do**
      Evaluate $y_w \leftarrow f_{\text{multi}}(M_w, \lambda, r_i)$, $D \leftarrow D \cup \{(w, r_i, y_w)\}$, $T \leftarrow T + 1$
    end for
    Sort $\mathcal{C}$ by performance, keep the top $\lfloor n_i/\eta \rfloor$ configurations in $\mathcal{C}$
  end for
end for
return the best-performing configuration ω^* evaluated at highest budget =0

A.1. Efficiency Analysis

Our optimizer dynamically adjusts the budget allocation during the search process, where the budget is defined as the calibration dataset size used for search. As the allocation of search trials directly determines the overall search duration. Here, we analyze the budget distribution during the search process, as shown in Table 2. Our analysis reveals that only 22% of the search trials utilize the full budget, while over 41.4% of the evaluations were conducted with the minimum budget, which is 5-10 times smaller. This efficient allocation enables our pruning to significantly increase the chance of discovering superior configurations under the same computational budget.

A.2. Which Parts of the Search Space are Critical ?

To determine where the benefits of our approach come from, we designed ablation experiments to evaluate the contribution of different components in our search space. As our framework supports: (1) selectively choosing layers from different candidate models, (2) layer merging, and (3) layer removal, we conducted several experiments to isolate the impact of each component. Table 3 summarizes the performance comparison across various benchmarks (full results available in the Table 9)

Layer Removal Only (LR-only). We restricted the search space to only allow layer removal operations on a single 7B model. We ran experiments on all 7B models and report the best performer. This ablation shows a performance drop from

Search Budget	Percentage	Trials	Dataset	Size
Low	41.4	207	PIQA	100
			WSC	100
			CSQA	100
			MMLU	100
Medium	36.6	183	PIQA	300
			WSC	200
			CSQA	300
			MMLU	300
High	22.0	110	PIQA	1000
			WSC	500
			CSQA	1000
			MMLU	1000

Table 2. Budget allocation to search trials for pruning. 41% of trials require only the smallest budget size, significantly increasing efficiency.

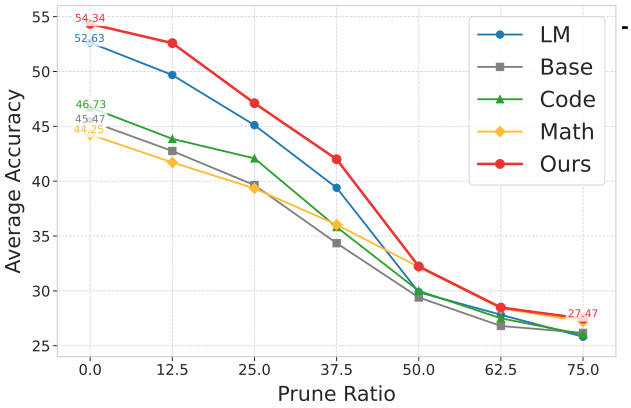


Figure 3. Performance Comparison Across Different Pruning Ratios.

our full approach (48.55 → 44.83 on average), confirming that merely pruning layers from a single model is insufficient for optimal performance. The performance degradation is particularly notable on language tasks (WSC_P: 63.46 → 49.04) and understanding benchmarks (Race_H: 55.35 → 42.51, Race_M: 58.64 → 43.04). It is worth highlighting that even our layer-removal-only for single model still outperforms the strongest baseline method, ShortGPT (44.83 vs. 42.24). This demonstrates that our approach can enhance performance even in this simplified setting.

Layer Selection and Removal (LS+LR). In this setting, we enabled both layer selection from different candidate models and layer removal operations but disabled the layer merging functionality. The results show even greater performance degradation (48.55 → 43.20 average) compared to the layer-removal-only setting. We observe a dramatic drop on WSC_G (43.27 → 26.92), indicating that merging operations play a critical role for certain grammatical reasoning tasks. The superior performance of LR-only (44.83) compared to LS+LR (43.20) demonstrates that simply combining layers from different models without proper integration through merging is suboptimal.

Table 3. Comparison of different searching settings across various benchmarks. Settings: LR-only: Layer-remove only, LS+LR: Layer-selection + layer-remove, FL-merge: Folding Layers Merging, Single-obj: Single-objective, PPL-obj: PPL as search objective.

Setting	Reasoning			Language			Knowledge				Understanding				Avg
	CNLI	HeSw	PIQA	CHID	WSC_P	WSC_G	CSQA	BoolQ	MMLU	CMLU	Race_H	Race_M	XSum	C3	
Ours	35.46	54.43	67.74	23.63	63.46	43.27	62.90	75.08	48.75	33.86	55.35	58.64	12.99	44.16	48.55
LR-only	34.96	53.80	66.70	18.58	49.04	58.65	60.61	68.87	47.85	33.54	42.51	43.04	8.05	41.42	44.83
LS+LR	32.92	55.84	65.07	17.98	63.46	26.92	58.97	51.22	48.97	34.61	48.68	49.44	8.33	42.41	43.20
FL-merge	32.99	52.90	63.66	19.28	46.15	62.50	60.52	75.20	48.30	34.33	50.77	55.29	6.39	39.40	46.26
Single-obj	32.15	56.02	67.46	19.08	39.42	48.08	62.33	74.43	47.40	34.14	50.94	52.86	12.35	41.97	45.62
PPL-obj	33.39	23.89	52.07	14.84	45.19	7.69	19.33	39.51	24.25	24.69	22.81	21.17	0.06	26.36	25.38

A.3. Additional Analysis

A.3.1. Varying the Pruning Ratio

To evaluate the benefits of evolving from "pruning single model" to "pruning from model variants" under varying pruning ratios, we compared our approach with each candidate model using the layer-remove configuration, as it achieves the strongest performance among single-model pruning methods, even surpassing the best-performing baseline, ShortGPT. Figure 3 visualizes the average accuracy among benchmark performances at different pruning ratios, with detailed results in Table 10. The accuracy of all models decreases as the pruning ratio increases. Our model achieves the best performance at almost all pruning ratios, especially in the low pruning ratio range of 0 - 37.5%. When pruning reaches 50%, the performance gap narrows across all models as they all experience performance collapse, including ours. We believe this is due to excessive parameter removal, after which effective model function cannot be maintained without additional post-training.

A.3.2. Pruning through Layer Folding

LaCo (Yang et al., 2024) is another pruning approach based on merging, but it differs from our method by merging only the later layers of a single model into adjacent earlier layers, based on activation similarity heuristics. To validate the effectiveness and potential of this type of within-model merge operation, we use our hyperparameter optimization framework

with a specially designed search space consisting of: (1) A binary selection vector $\mathbf{s} = [s_1, s_2, \ldots, s_k]$ indicating which layers to remove, and (2) An importance weight vector $\mathbf{w} = [w_1, w_2, \ldots, w_k]$ representing each layer's importance value.

Retained layer L'_i performs a depth-wise linear combination with itself and adjacent removed layers:

$$L'_i = \beta_i \cdot L_i + \sum_{j \in \mathcal{N}(i)} \beta_j \cdot L_j \cdot \mathbb{1}_{s_j=1}$$

where $\mathcal{N}(i)$ represents adjacent layers to L_i, $\mathbb{1}_{s_j=1}$ indicates layer j is removed, and β_j are normalized weights derived from $\mathbf{w}$ such that $\beta_i + \sum_{j \in \mathcal{N}(i)} \beta_j \cdot \mathbb{1}(s_j = 1) = 1$. This ensures retained layers incorporate information from nearby removed layers, preserving network functionality. This configuration parametrizes the options proposed in LACO and achieves better overall performance (46.26) compared to previous ablations but still falls short of our full approach. The performance gap is most pronounced on XSum (12.99 → 6.39) and PIQA (67.74 → 63.66), highlighting the importance of our optimized merging strategy for generative and reasoning tasks.

A.3.3. Different Calibration Datasets and Metrics

In our method, we use multiple-choice datasets as calibration data with accuracy as the metric in a multi-objective optimization approach. This results in a pruned model with broad capabilities. To further analyze this design choice, we conducted the following experiments:

Single Objective (Single-obj). We used the MMLU validation dataset for calibration with accuracy as the optimization objective. We evaluate the resulting pruned models across our benchmark suite. As shown in Table 3, while these models still perform adequately (45.62 average), the single-objective optimization led to a noticeable decline from our full approach (48.55 → 45.62). Importantly, the single-objective models demonstrated stronger performance on MMLU-related tasks but showed performance degradation on certain other tasks due to their narrow optimization focus. This confirms our hypothesis that broad, multi-objective optimization is necessary to preserve the broad functionality of modern LLMs, rather than overfitting to a single task domain.

Perplexity Objective (PPL-obj). We also experiment with perplexity (PPL) on WikiText (Merity et al., 2016) as a search metric, taking 1500 examples as our calibration dataset. The performance of resulting pruned models across benchmarks in Table 3 demonstrates a dramatic performance gap compared to all other configurations, with an average score of only 25.38 across benchmarks. Even when compared to the single-objective MMLU optimization (which uses a similarly sized dataset), the PPL-optimized models showed considerably weaker performance across most tasks. The catastrophic degradation on tasks like XSum (12.99 → 0.06) and reasoning benchmarks like HeSw (54.43 → 23.89) underscores the limitations of perplexity-guided optimization. This evidence reveals that while perplexity is a common metric for language model evaluation, it fails to serve as an effective signal for preserving model capabilities during pruning, particularly for tasks requiring reasoning and knowledge application rather than just fluent text generation.

A.3.4. Extending to Llama-3

We further extend our validation to Meta's Llama 3 8B model(Grattafiori et al., 2024). Llama 3 training on 15 trillion tokens—7× more data than Llama 2, incorporates architectural improvements including universal Grouped Query Attention (GQA), an optimized 128K vocabulary tokenizer and longer context window. These modifications further boost the performance on reasoning, code generation, and multilingual tasks. Despite similar model size, Llama-3 8B achieves better performance compared to Llama-2 7B(Touvron et al., 2023), which may carry different semantic densities that provide new challenges for maintaining model performance under compression. Validating on this next-generation model is crucial for establishing the practical applicability of our method in rapidly evolving LLM landscapes.

We use Meta-Llama-3-8B(Grattafiori et al., 2024) as base model, with three candidate models: Meta-Llama-3-8B-Instruct (LM)(Grattafiori et al., 2024), Code-Llama-3-8B (Code)(Ajibawa, 2023), and MathCoder2-Llama-3-8B (Math)(Lu et al., 2024a;b). We target removing 9 of the 32 layers and use the same experimental settings as Llama2-7B. We compare our method with the best-performing baseline, ShortGPT. As results shown in Table 4(full results available in the Table 11), our method retains 84.55% of the original performance after removing 9 layers, outperforming ShortGPT's 62.79% retention under same compression ratios. while both show lower retention than Llama2-7B results (92.2%) with similar model size. This decline indicates Llama3's reduced compressibility, We attribute this decline to (1) higher parameter utilization efficiency and (2) denser knowledge distribution from large-scale training, which eliminates layer redundancy. Despite this challenge, our method consistently outperforms the baseline, validating its effectiveness across model generations.

Table 4. Comparison of pruning methods on multiple natural language benchmarks. "Single" refers to the best performance achieved when pruning a single model directly, while "Merge" refers to the best performance achieved through either "pruning-then-merging" or "merging-then-pruning". For 8b model: Meta-Llama-3-8B-Instruct (LM), MathCoder2-Llama-3-8B (Math), Code-Llama-3-8B (Code), and Meta-Llama-3-8B (Base). The cells highlighted in blue show three selected Pareto-optimal solutions of our method.

LLM	Pruner	Type	Reasoning			Language			Knowledge				Understanding				Avg
	ratio/layer		CMNLI	HeSw	PIQA	CHID	WSC_P	WSC_G	CSQA	BoolQ	MMLU	CMLU	$Race_H$	$Race_M$	XSum	C3	
Llama3-8B	Dense	Base	32.98	74.67	80.96	73.78	56.73	36.54	73.79	69.97	64.74	50.79	63.21	70.54	3.28	55.18	57.65
		LM	33.00	71.08	80.69	65.53	55.77	69.23	76.66	78.87	65.97	53.64	76.44	81.75	17.97	63.95	63.61
		Math	32.99	71.66	77.97	57.09	37.50	58.65	68.22	69.08	62.08	45.85	64.75	69.08	8.68	53.86	55.53
		Code	32.98	65.56	74.70	78.42	61.54	61.54	63.47	78.35	48.03	34.55	52.40	58.43	19.36	46.41	55.41
	ShortGPT	Single	32.83	45.06	65.78	23.38	41.35	53.85	39.56	63.73	32.37	28.69	40.14	45.19	**3.68**	43.51	39.94
		Merge	32.95	48.58	64.96	18.43	36.54	35.58	42.83	**67.22**	33.05	28.71	30.16	32.45	**3.66**	44.27	37.10
	Ours		**33.46**	54.81	**69.53**	**32.27**	41.35	58.65	72.81	**65.29**	**63.36**	**50.14**	71.41	75.97	3.22	**47.12**	52.81
			33.07	**55.15**	69.37	31.77	**46.15**	**64.42**	**73.55**	64.43	62.35	49.06	**74.36**	**78.27**	2.95	46.03	53.64
			33.42	**54.83**	**69.75**	**34.02**	**47.12**	**62.50**	**73.79**	64.34	**63.13**	**50.04**	**72.81**	**77.65**	3.00	**46.52**	53.78

B. Baseline

To ensure fair comparison, we applied various baseline pruning methods including LLM-Pruner(LLMPru) (Ma et al., 2023), SliceGPT (Ashkboos et al., 2024), LaCo (Yang et al., 2024) and ShortGPT (Men et al., 2024):

LLM-Pruner adopts structural pruning that selectively removes non-critical coupled structures based on gradient information, maximally preserving the majority of the LLM's functionality. It applies post-training to the pruned model, for fair comparison, we do not apply post training to it.

SliceGPT is a post-training sparsification scheme that replaces each weight matrix with a smaller matrix, reducing the embedding dimension of the network. Specifically, they applied PCA to the hidden representation from shallow to deep layers, and incorporated the dimension reduction matrix into existing network parameters.

LaCo is a pruning method for large language models based on reducing layers. LaCo gradually merges similar layers from deep to shallow and sets a threshold to avoid continuously merging too many layers.

ShortGPT introduced the Block Influence (BI) metric, which uses the similarity between layer's input and output to measure the importance of each layer.

C. Evaluation Benchmarks

To thoroughly assess the capabilities of large language models before and after pruning, we employed a comprehensive suite of benchmark datasets spanning various aspects of language understanding and reasoning:

CMNLI (Chinese Multi-Genre Natural Language Inference) (CNLI) consists of two parts: XNLI and MNLI. It contains text from various domains, including fiction, telephone conversations, travel, and government sources. XNLI is a cross-lingual extension of the MultiNLI corpus, professionally translated into multiple languages, including Chinese, providing a robust framework for assessing language understanding across linguistic boundaries. Models must determine whether pairs of sentences exhibit entailment, contradiction, or neutrality.

HellaSwag (HeSw) tests commonsense reasoning about physical situations. The dataset uses a "Goldilocks" zone of complexity where examples are obviously nonsensical to humans but challenging for state-of-the-art models. Despite being trivial for humans (>95% accuracy), even advanced models struggled with this benchmark upon its release, making it effective for measuring progress in commonsense inference.

PIQA (Physical Interaction Question Answering) Developed by Bisk et al. (2020), this multi-choice question and answer dataset focuses on everyday scenarios, exploring models' understanding of real-world physical laws through daily situations.

CHID (Chinese IDiom) is an idiom cloze test focusing on the representation and selection of Chinese idioms, requiring cultural and linguistic knowledge specific to Chinese.

WSC (Winograd Schema Challenge) serves as a prominent benchmark for evaluating machine understanding through pronouns resolution problems that are trivial for humans but require commonsense reasoning for machines to solve correctly.

The dataset consists of pairs of sentences differing in one or two words with ambiguous pronouns resolved differently in the two sentences, designed to test a system's commonsense reasoning abilities.

CommonSenseQA (CSQA) is a multiple-choice question answering dataset containing 12,102 questions with one correct answer and four distractor answers, requiring different types of commonsense knowledge to predict the correct answers. The dataset was constructed using ConceptNet relations and crowd-sourced questions to test commonsense reasoning.

BoolQ provides 15,942 yes/no questions that occur naturally in unconstrained environments, testing models' binary decision-making abilities.

MMLU (Massive Multitask Language Understanding) evaluates models across 57 diverse subjects covering STEM, humanities, and social sciences. The benchmark tests knowledge and problem-solving ability with content ranging from elementary to professional levels. This benchmark has become a standard evaluation metric in the field, with scores prominently reported for virtually all language models, and uses multiple-choice questions that allow for simple accuracy calculations.

CMMLU (Chinese Massive Multitask Language Understanding) (CMLU) Developed to address the gap in evaluating knowledge and reasoning capabilities in Chinese, CMMLU is a comprehensive benchmark covering 67 subjects from elementary to advanced professional levels across natural sciences, social sciences, engineering, and humanities. The benchmark includes topics with Chinese-specific answers that may not be universally applicable in other regions or languages, making it a fully Chinese-oriented evaluation tool.

RACE (Reading Comprehension from Examinations) is collected from English examinations in China designed for middle and high school students, providing a culturally diverse reading assessment.

XSum evaluates abstract single document summarization systems, focusing on the ability to create concise one-sentence summaries capturing the essence of articles.

C3 (Chinese Multiple-Choice Machine Reading Comprehension) consists of multiple-choice questions from Chinese proficiency exams and ethnic Chinese exams.

D. Task Arithmetic Merging

Task Arithmetic (Ilharco et al., 2022) enhances model capabilities through vector operations by leveraging weighted combinations of task-specific knowledge. Given a base model with weights θ_{pre} and task-specific fine-tuned weights $\{\theta_t^{\text{ft}}\}_{t=1}^{n}$, task vectors are defined as:

$$\tau_t = \theta_t^{\text{ft}} - \theta_{\text{pre}} \tag{2}$$

The merged weights are then computed through:

$$\theta_{\text{Merge}} = \theta_{\text{pre}} + \lambda \sum_{t=1}^{n} \tau_t \tag{3}$$

where λ controls the magnitude of task-specific adaptations.

E. Full Baseline Results

To validate the efficiency of our proposed method, we conducted comparative experiments against established baseline techniques. For fair comparison with other baseline methods, we selected the same pruning ratios matching those used in LaCo (Yang et al., 2024) and ShortGPT (Men et al., 2024) while being lower than those of other approaches. In order to make a fairer comparison, we reproduced all the results and evaluated them on OpenCompass (Contributors, 2023) as in LaCo.All experiments run on NVIDIA Tesla A100 GPUs. For each baseline method, we explored three scenarios: (1) applying each baseline pruning method individually to all candidate models, (2) first pruning each candidate model using existing methods and then merging them, and (3) first merging the candidate models and then applying pruning techniques.

We use the official implement of LLM-pruner and LaCo, It's worth noting that when reproducing the LaCo method,

we referenced the hyperparameter settings from the original paper. Due to differences in hardware, we couldn't fully reproduce the paper's results: we couldn't obtain models with pruning ratios consistent with the paper using the provided hyperparameters. We maintained consistency in all other parameters while gradually adjusting the threshold from 0.75 until achieving the desired pruning ratio. The specific parameters are detailed in the Table 5.

For the reproduction of ShortGPT, we implemented the algorithm based on the original paper and similarly sampled 10,000 instances from the PG19 (Rae et al., 2019) dataset as calibration data, following the methodology described in the paper. The resulting removed layers are shown in the Table. The removed layers for the base model align with those reported in the ShortGPT paper, albeit in a different sequence. We attribute this variation to slight differences in calculated layer importance scores. The specific configuration of removed layers for each model is detailed in the Table 6.

For the merging process, we employed task arithmetic with weighting parameters in the range of [0.5, 1.0]. The full results of the baseline methods on the 7B model and the 13B model are presented in Table 7 and Table 8, respectively.

Table 5. Hyperparameter settings for LaCo results. $\mathcal{C}$: Number of layers combined in each merge; $\mathcal{L}$,$\mathcal{H}$: Layer range [$\mathcal{L}$, $\mathcal{H}$]; $\mathcal{I}$: Minimum interval between two adjacent merged layers; $\mathcal{T}$: Threshold for representation similarity.

Size	Model	$\mathcal{C}$	$\mathcal{L}$	$\mathcal{H}$	$\mathcal{I}$	$\mathcal{T}$
Llama2-13B	Llama-2-13B	6	1	40	2	0.7
	WizardLM-13B	6	1	40	2	0.65
	WizardMath-13B	6	1	40	2	0.7
	llama-2-13b-code-alpaca	6	1	40	2	0.7
	Merge-then-prune	6	1	40	2	0.65
	Prune-then-merge	6	1	40	2	0.65
Llama2-7B	Llama-2-7B	6	1	40	2	0.7
	Llama-2-7B-Chat	6	1	40	2	0.65
	MAmmoTH-7B	6	1	40	2	0.7
	Llama-2-Coder-7B	6	1	40	2	0.7
	Merge-then-prune	6	1	40	2	0.65
	Prune-then-merge	6	1	40	2	0.65

Table 6. Setup of Removed Layers for Candidate Models in ShortGPT.

Model	Removed Layers
Llama-2-7B	25, 27, 24, 28, 26, 29, 23, 22, 21
Llama-2-7B-Chat	27, 25, 24, 28, 29, 26, 23, 22, 21
MAmmoTH-7B	27, 25, 24, 28, 29, 23, 26, 22, 21
Llama-2-Coder-7B	27, 25, 24, 28, 29, 26, 23, 21, 22
Llama-2-13B	33, 32, 31, 30, 34, 35, 29, 28, 27, 26
WizardLM-13B	33, 32, 31, 30, 34, 35, 29, 28, 27, 36
WizardMath-13B	33, 31, 32, 30, 34, 35, 29, 28, 27, 36
llama-2-13b-code-alpaca	33, 31, 32, 30, 34, 35, 29, 28, 27, 26

Table 7. The main results of baseline methods on the 7B model across multiple natural language benchmarks using candidate models: Llama-2-7B-Chat (LM), MAmmoTH-7B (MAth), Llama-2-Coder-7B (Code), and Llama-2-7B (base). "PTM" (Pruning-then-Merging) refers to first pruning each candidate model using current pruner and then merging them. "MTP" (Merging-then-Pruning) refers to first merging the candidate models and then applying pruning. For LLMPruner and SliceGPT, alignment challenges exist after pruning. LLMPruner removes different model blocks, while SliceGPT calculates orthogonal transformation matrices that are highly dependent on each model's specific weight distributions and activation patterns, resulting in incompatible transformation spaces. Therefore, we only implemented "merge then prune".

LLM	Pruner (ratio/layer)	Type	Reasoning			Language			Knowledge				Understanding				Avg
			CMNLI	HeSw	PIQA	CHID	WSC_P	WSC_G	CSQA	BoolQ	MMLU	CMLU	$Race_H$	$Race_M$	XSum	C3	
Llama -7B	Dense	Base	32.98	71.34	78.18	41.56	37.50	38.46	55.04	70.70	46.67	31.88	35.53	33.36	19.55	43.84	45.47
		Math	32.99	68.60	75.79	39.71	39.42	36.54	50.78	69.36	43.04	32.16	30.36	36.42	20.88	43.45	44.25
		LM	31.30	71.28	75.95	36.11	63.46	59.62	64.29	74.77	48.30	33.93	52.52	55.22	22.45	47.56	52.63
		Code	32.99	70.27	78.62	41.61	36.54	41.35	57.41	71.04	46.22	32.20	41.25	39.69	18.79	46.25	46.73
	LLMPruner (25.32%)	Base	33.00	58.72	72.25	29.52	41.35	0.00	19.74	57.25	23.69	25.49	22.07	21.10	14.67	28.11	31.93
		LM	34.94	59.25	72.85	22.28	43.27	9.62	19.41	57.61	23.77	24.51	21.78	22.42	16.32	28.66	32.62
		MATH	32.99	55.74	70.84	25.82	37.50	21.15	18.84	54.31	24.77	25.20	22.87	23.89	10.91	28.00	32.35
		Code	32.99	59.57	73.34	30.32	46.15	0.00	20.15	57.28	23.21	25.16	21.56	21.52	15.19	31.07	32.68
		MTP	34.71	60.57	73.50	26.62	40.38	5.77	19.90	52.14	24.01	25.30	23.07	22.98	15.51	32.49	32.64
	SliceGPT (26.33%)	Base	31.08	42.90	61.43	19.53	36.54	0.00	20.88	37.95	24.78	24.78	21.24	21.73	6.58	37.42	27.63
		LM	31.70	43.50	61.37	18.28	40.38	0.96	21.21	38.96	25.56	25.28	21.93	22.42	13.13	38.36	28.79
		MATH	31.89	41.55	58.81	18.43	39.42	4.81	19.49	40.09	25.38	25.02	25.59	26.88	8.78	39.56	28.98
		Code	31.81	44.02	63.17	18.48	36.54	13.46	19.74	37.92	24.71	25.22	21.41	21.66	2.59	38.19	28.49
		MTP	32.85	37.61	57.56	17.33	53.85	2.88	19.41	42.66	25.22	24.68	25.21	24.72	12.78	40.22	29.78
	LACO	Base	32.85	53.33	68.23	31.62	36.54	4.81	20.39	62.02	26.60	25.27	24.70	23.61	9.38	42.47	32.99
		LM	32.97	55.24	69.53	31.47	36.54	34.62	22.11	67.22	29.08	26.16	28.53	28.27	14.68	43.51	37.14
		Math	32.97	55.24	69.53	31.47	50.00	34.62	22.11	67.22	29.44	26.16	22.53	23.68	14.68	39.34	37.07
		Code	32.28	53.68	69.15	32.22	36.54	1.92	20.56	61.99	26.31	25.43	27.10	22.70	11.14	43.07	33.15
		MTP	32.43	57.80	71.82	28.97	41.35	16.35	27.52	71.28	30.49	26.88	25.76	27.09	8.27	44.33	36.45
		PTM	31.89	56.26	71.22	27.32	39.42	22.12	23.42	72.66	29.30	26.00	25.19	26.81	16.11	43.62	36.52
	ShortGPT (27.1%)	Base	33.09	57.42	66.54	21.53	56.73	48.08	52.5	67.34	43.68	28.31	32.53	31.69	12.40	39.45	42.24
		LM	33.85	53.93	63.82	14.59	39.42	22.12	58.48	67.95	35.85	26.60	48.03	51.18	6.93	37.21	40.00
		MATH	33.97	56.69	63.38	17.78	54.81	44.23	37.26	69.82	30.68	25.26	28.24	30.29	8.26	31.67	38.02
		Code	32.74	56.69	65.07	17.78	58.65	35.58	53.24	67.52	44.82	28.92	35.62	37.53	14.32	40.66	42.08
		MTP	34.10	54.18	64.42	16.83	61.54	36.54	55.61	73.21	36.84	25.61	42.94	45.89	10.12	35.73	42.40
		PTM	34.10	54.18	64.42	16.83	61.54	36.54	55.61	73.21	36.84	25.61	42.94	45.89	10.12	35.73	42.40

Table 8. The main results of baseline methods on the 13B model across multiple natural language benchmarks using candidate models: WizardLM-13B (LM), WizardMath-13B (Math), llama-2-13b-code-alpaca (Code), and Llama-2-13B (Base). "PTM" (Pruning-then-Merging) refers to first pruning each candidate model using the current pruner and then merging them. "MTP" (Merging-then-Pruning) refers to first merging the candidate models and then applying pruning. For LLMPruner and SliceGPT, alignment challenges exist after pruning. LLMPruner removes different model blocks, while SliceGPT calculates orthogonal transformation matrices that are highly dependent on each model's specific weight distributions and activation patterns, resulting in incompatible transformation spaces. Therefore, we only implemented "merge then prune"

LLM	Pruner ratio/layer	Type	Reasoning			Language			Knowledge				Understanding				Avg
			CMNLI	HeSw	PIQA	CHID	WSC_P	WSC_G	CSQA	BoolQ	MMLU	CMLU	$Race_H$	$Race_M$	XSum	C3	
Llama -13B	Dense	Base	32.99	74.77	79.71	47.35	50.96	63.46	67.24	71.38	55.84	38.74	57.98	60.17	23.47	47.51	55.11
		LM	35.36	70.41	78.73	36.21	57.69	60.58	65.03	73.70	53.48	30.85	66.12	71.66	22.44	52.00	55.30
		MATH	32.99	68.78	77.26	44.36	36.54	19.23	60.36	78.44	54.21	38.12	47.74	48.82	19.51	44.66	47.93
		Code	32.99	74.82	80.14	47.30	51.92	63.46	68.88	72.72	55.92	39.26	58.03	63.72	24.45	48.38	55.86
	LLMPruner (21.2%)	Base	33.27	63.57	75.41	34.17	37.50	0.00	19.57	45.35	23.08	25.36	21.61	21.80	14.41	29.64	31.77
		LM	33.49	60.28	75.57	23.68	39.42	0.00	19.00	63.24	23.27	25.23	22.36	21.45	17.13	32.00	32.58
		MATH	32.99	55.49	72.91	30.02	41.35	0.00	19.08	53.18	23.06	25.53	21.36	21.31	12.25	29.10	31.26
		Code	33.18	64.21	75.52	34.17	43.27	0.00	19.90	47.80	23.19	25.52	21.61	22.08	16.08	29.59	32.58
		MTP	33.86	64.11	73.50	22.18	60.58	0.00	21.46	61.96	23.84	25.62	22.16	21.59	14.98	32.11	34.14
	SliceGPT (23.6%)	Base	30.39	46.69	63.22	18.78	42.31	25.96	25.23	37.83	30.43	25.14	23.47	24.65	8.78	39.56	31.60
		LM	33.19	42.44	59.90	18.03	54.81	19.23	32.51	41.22	33.09	25.75	29.45	29.87	9.99	37.75	33.37
		MATH	32.73	36.27	59.30	17.38	42.31	0.00	21.62	37.83	30.33	25.16	23.84	24.16	1.54	40.82	28.09
		Code	30.82	46.69	63.00	19.18	42.31	27.88	24.82	37.83	31.38	25.20	23.47	24.65	8.83	40.00	31.86
		MTP	30.98	46.83	62.57	19.33	51.92	49.04	37.76	38.38	33.55	25.22	23.53	23.05	9.95	39.67	35.13
	LaCo (24.6%)	Base	32.97	59.38	73.45	36.26	37.50	37.50	19.41	57.31	25.03	24.41	22.47	23.19	16.39	37.92	35.94
		LM	32.33	60.18	70.57	32.67	34.62	34.62	52.58	62.66	36.26	25.80	60.38	62.53	8.79	49.21	44.51
		Math	33.97	56.51	72.25	33.52	44.23	44.23	21.38	64.19	25.35	24.55	21.98	21.94	12.77	37.48	36.74
		Code	32.99	59.53	75.03	38.41	51.92	0.00	19.49	53.18	24.48	24.72	22.87	22.28	17.70	37.53	34.30
		MTP	33.49	62.50	74.37	35.26	63.46	63.46	18.84	64.65	41.83	24.87	26.10	25.97	15.93	39.51	42.16
		PTM	31.85	29.80	51.31	12.74	36.54	36.54	19.57	62.08	24.37	25.19	22.10	22.77	0.40	35.12	29.31
	ShortGPT (24.6%)	Base	32.99	67.07	73.45	36.46	42.31	45.19	66.99	58.56	54.74	38.39	56.89	54.06	18.58	46.19	49.42
		LM	32.95	62.64	73.50	28.22	36.54	50.96	65.44	67.71	53.50	30.73	65.52	71.38	19.12	48.60	50.49
		MATH	32.99	59.63	70.40	31.12	40.38	1.92	59.71	70.00	52.70	36.94	43.51	44.29	7.73	43.84	42.51
		Code	32.92	67.03	74.37	36.41	55.77	46.15	68.96	60.55	54.94	38.30	53.60	58.57	8.41	47.18	50.23
		MTP	31.07	63.24	68.61	27.17	49.04	43.27	65.68	78.01	51.26	36.88	57.38	62.67	16.94	44.05	49.66
		PTM	31.08	63.32	68.66	27.12	49.04	43.27	65.68	77.98	51.23	36.82	57.40	62.47	17.01	43.95	49.65

Table 9. Performance comparison of various model pruning strategies across multiple benchmark categories. The settings include LR-only (Layer Removal only), LS+LR (combined Layer Selection and Layer Removal), FL-merge (Folding Layers Merging), Single-obj (Single-objective optimization), and PPL-obj (Perplexity-based objective). For multi-objective optimization approaches, three representative Pareto-optimal solutions (numbered 1-3) are presented.

setting	Reasoning			Language			Knowledge				Understanding				Avg
	CNLI	HeSw	PIQA	CHID	WSC_P	WSC_G	CSQA	BoolQ	MMLU	CMLU	$Race_H$	$Race_M$	XSum	C3	
LR-only-LM-1	33.93	57.51	65.49	18.18	62.46	48.03	58.79	62.18	45.76	30.95	49.54	53.36	1.45	38.60	44.73
LR-only-LM-2	33.58	52.10	64.25	19.53	50.00	62.50	63.64	41.80	48.33	32.84	51.03	51.46	5.47	39.56	44.01
LR-only-LM-3	34.96	53.80	66.70	18.58	49.04	58.65	60.61	68.87	47.85	33.54	42.51	43.04	8.05	41.42	44.83
LR-only-Math-1	33.77	54.49	68.23	21.93	62.50	37.50	27.85	57.52	37.08	28.73	31.42	34.05	7.51	37.92	38.61
LR-only-Math-2	31.69	56.56	68.77	27.07	63.46	30.77	36.69	62.35	39.17	29.15	33.39	38.65	4.41	43.34	40.39
LR-only-Math-3	32.94	58.43	69.64	25.97	54.81	25.96	29.89	62.84	33.46	26.92	31.39	32.10	8.06	40.16	38.04
LR-only-Code-1	30.13	57.60	70.35	27.07	63.46	11.54	50.94	65.96	42.64	30.96	36.39	36.77	3.15	43.78	40.77
LR-only-Code-2	34.94	57.37	68.55	28.67	42.31	41.35	54.46	63.00	42.49	27.39	34.88	35.31	4.08	43.78	41.33
LR-only-Code-3	34.93	56.71	69.42	25.92	59.62	31.65	52.83	62.20	43.03	28.80	38.51	39.07	2.87	41.70	41.95
LR-only-Base-1	32.67	54.21	66.00	26.07	36.54	1.92	49.47	64.19	44.47	28.84	38.99	38.86	0.25	41.59	37.43
LR-only-Base-2	32.22	56.48	67.46	26.32	61.54	50.00	41.44	66.91	40.54	28.01	37.94	39.35	0.96	41.92	42.22
LR-only-Base-3	31.13	52.90	67.95	27.97	36.54	0.00	54.63	64.13	43.01	30.03	35.56	37.05	6.79	41.70	37.81
FL-merge-1	32.99	52.90	63.66	19.28	46.15	62.50	60.52	75.20	48.30	34.33	50.77	55.29	6.39	39.40	46.26
FL-merge-2	32.99	51.99	63.44	18.33	46.15	63.46	61.26	74.77	48.80	33.84	51.11	56.34	5.75	37.86	46.15
FL-merge-3	33.89	51.15	62.62	18.63	50.00	61.54	60.44	75.78	48.61	33.96	50.74	55.85	5.72	38.03	46.15
LS+LR-1	34.75	53.65	66.32	17.83	63.46	22.12	59.71	70.61	47.32	33.77	36.62	33.91	8.54	42.35	42.21
LS+LR-2	31.74	55.25	68.39	26.77	63.46	10.58	58.72	66.27	47.40	33.15	40.02	45.26	2.62	44.16	42.41
LS+LR-3	32.92	55.84	65.07	17.98	63.46	26.92	58.97	51.22	48.97	34.61	48.68	49.44	8.33	42.41	43.20
Single-obj	32.15	56.02	67.46	19.08	39.42	48.08	62.33	74.43	47.40	34.14	50.94	52.86	12.35	41.97	45.62
PPL-obj	33.39	23.89	52.07	14.84	45.19	7.69	19.33	39.51	24.25	24.69	22.81	21.17	0.06	26.36	25.38

Table 11. The main results of the Llama3-8B model across multiple natural language benchmarks using candidate models: Meta-Llama-3-8B-Instruct (LM), MathCoder2-Llama-3-8B (Math), Code-Llama-3-8B (Code), and Meta-Llama-3-8B (Base). "PTM" (Pruning-then-Merging) refers to first pruning each candidate model using the current pruner and then merging them. "MTP" (Merging-then-Pruning) refers to first merging the candidate models and then applying pruning.

LLM	Pruner ratio/layer	Type	Reasoning			Language			Knowledge				Understanding				Avg
			CMNLI	HeSw	PIQA	CHID	WSC_P	WSC_G	CSQA	BoolQ	MMLU	CMLU	$Race_H$	$Race_M$	XSum	C3	
Llama3-8B	Dense	Base	32.98	74.67	80.96	73.78	56.73	36.54	73.79	69.97	64.74	50.79	63.21	70.54	3.28	55.18	57.65
		LM	33.00	71.08	80.69	65.53	55.77	69.23	76.66	78.87	65.97	53.64	76.44	81.75	17.97	63.95	63.61
		Math	32.99	71.66	77.97	57.09	37.50	58.65	68.22	69.08	62.08	45.85	64.75	69.08	8.68	53.86	55.53
		Code	32.98	65.56	74.70	78.42	61.54	61.54	63.47	78.35	48.03	34.55	52.40	58.43	19.36	46.41	55.41
	ShortGPT (24.6%)	Base	36.00	31.36	62.84	25.77	36.54	63.46	53.97	50.61	36.05	33.83	30.73	32.38	1.17	38.96	38.12
		LM	32.83	45.06	65.78	23.38	41.35	53.85	39.56	63.73	32.37	28.69	40.14	45.19	3.68	43.51	39.94
		Math	32.98	42.89	63.00	17.18	36.54	36.54	45.37	46.30	33.95	29.71	28.87	30.22	1.45	40.49	34.68
		Code	32.26	45.99	64.96	17.03	36.54	36.54	36.20	63.98	28.78	26.25	27.27	29.46	3.57	39.01	34.85
		MTP	32.98	48.51	64.85	18.33	36.54	35.58	42.83	67.06	33.05	28.73	30.07	32.66	3.64	44.33	37.08
		PTM	32.95	48.58	64.96	18.43	36.54	35.58	42.83	67.22	33.05	28.71	30.16	32.45	3.66	44.27	37.10

Table 10. Model Performance Comparison Across Pruning Ratios

Model	Prune Ratio	Reasoning			Language			Knowledge				Understanding				Avg
		CNLI	HeSw	PIQA	CHID	WSC_P	WSC_G	CSQA	BoolQ	MMLU	CMLU	$Race_H$	$Race_M$	XSum	C3	
Base	0	32.98	71.34	78.18	41.56	37.50	38.46	55.04	70.70	46.67	31.88	35.53	33.36	19.55	43.84	45.47
Base	12.5	32.99	67.06	74.92	39.61	36.53	1.92	57.41	69.36	47.15	31.61	39.11	38.65	17.59	44.60	42.75
Base	25	32.98	63.80	69.21	35.37	36.54	0.00	50.78	64.74	40.80	30.31	35.19	35.62	16.11	43.51	39.64
Base	37.5	32.58	45.04	61.53	20.68	36.54	2.88	42.18	64.43	39.87	29.42	31.90	29.74	2.77	41.37	34.35
Base	50	34.51	34.89	55.33	17.08	36.54	11.54	19.82	62.29	28.72	25.10	23.41	26.04	1.21	35.07	29.40
Base	62.5	35.14	29.71	52.83	14.94	39.42	1.92	21.46	50.06	24.55	25.16	26.76	25.42	0.09	27.62	26.80
Base	75	34.94	26.71	51.03	13.59	36.54	8.65	20.56	52.60	24.23	24.47	23.18	22.63	0.08	27.29	26.17
LM	0	31.30	71.28	75.95	36.11	63.46	59.62	64.29	74.77	48.30	33.93	52.52	55.22	22.45	47.56	52.63
LM	12.5	32.42	67.58	72.72	28.91	50.92	60.50	60.92	72.88	46.69	32.02	51.34	54.45	18.26	45.94	49.68
LM	25	30.10	60.63	66.82	20.53	48.96	42.31	65.88	70.82	42.09	32.40	48.23	50.43	15.75	43.62	45.11
LM	37.5	33.29	45.13	60.66	20.03	36.54	11.73	59.38	68.07	39.18	29.64	39.71	42.20	6.36	41.04	39.40
LM	50	34.93	34.67	56.20	16.18	36.54	8.65	22.28	62.14	32.01	26.44	25.39	25.49	2.34	35.01	29.88
LM	62.5	34.11	30.50	53.21	14.34	51.92	2.88	20.56	57.95	24.58	25.21	23.13	23.75	0.18	27.12	27.82
LM	75	34.87	27.03	52.19	14.54	39.42	0.00	20.23	53.87	24.45	24.83	21.41	22.14	0.02	26.69	25.82
Math	0	32.99	68.60	75.79	39.71	39.42	36.54	50.78	69.36	43.04	32.16	30.36	36.42	20.88	43.45	44.25
Math	12.5	32.97	64.72	73.06	37.50	23.08	23.07	51.43	71.16	42.91	31.90	32.99	36.07	19.30	43.83	41.71
Math	25	34.92	46.24	61.92	19.38	36.54	56.73	45.45	72.81	35.07	29.78	31.45	34.33	6.24	39.89	39.34
Math	37.5	32.99	55.42	62.81	23.82	38.38	4.81	37.87	68.68	36.46	27.19	28.02	33.79	13.88	39.37	36.04
Math	50	32.73	35.93	55.06	16.73	39.42	39.42	20.15	64.34	29.94	25.52	26.82	26.60	2.31	35.56	32.15
Math	62.5	34.93	31.06	54.08	13.79	58.65	4.81	20.56	46.24	26.70	25.05	26.56	26.53	0.57	28.33	28.42
Math	75	34.94	27.35	52.07	14.39	43.27	2.88	20.88	56.51	24.25	23.14	24.76	24.79	0.15	27.45	27.20
Code	0	32.99	70.27	78.62	41.61	36.54	41.35	57.41	71.04	46.22	32.20	41.25	39.69	18.79	46.25	46.73
Code	12.5	32.97	65.79	75.78	39.06	36.54	0.96	56.67	71.13	47.09	32.00	44.73	44.84	19.21	47.29	43.86
Code	25	32.99	63.06	72.02	35.67	36.54	0.00	50.59	68.87	40.50	28.87	36.64	38.59	17.59	45.64	40.51
Code	37.5	33.21	44.12	62.13	20.78	36.54	2.88	48.81	63.91	40.29	29.56	36.25	35.52	5.35	42.14	35.82
Code	50	34.93	34.15	54.95	16.73	36.54	17.31	22.03	62.54	28.46	25.16	24.13	24.44	2.03	36.62	30.00
Code	62.5	34.72	29.67	52.99	14.39	40.38	8.65	22.52	50.70	24.78	25.15	27.16	28.04	0.12	27.78	27.50
Code	75	34.94	26.79	50.82	13.99	38.46	5.77	24.08	48.38	24.08	24.52	22.73	22.49	0.13	27.29	26.03
Ours	0	36.88	73.16	78.67	39.46	64.46	45.19	65.37	78.43	49.75	35.08	58.78	61.65	24.50	49.33	54.34
Ours	12.5	33.00	66.78	75.19	34.92	64.42	63.46	63.98	75.87	48.79	34.13	53.89	56.20	20.21	45.37	52.59
Ours	25	32.99	57.31	68.34	22.38	63.46	63.46	57.58	62.17	45.92	30.96	52.20	56.06	7.12	39.67	47.11
Ours	37.5	35.67	51.02	63.44	20.68	62.50	22.00	57.99	67.52	47.09	34.11	44.00	46.38	2.96	39.34	42.00
Ours	50	33.97	41.99	58.16	21.08	38.54	24.12	26.52	46.03	32.32	28.30	28.99	28.88	6.30	36.11	32.23
Ours	62.5	33.30	28.34	51.96	18.09	46.15	6.88	23.88	45.81	26.41	26.95	28.73	28.72	5.09	28.47	28.48
Ours	75	34.93	30.45	49.18	20.48	39.54	10.81	21.98	45.29	25.28	24.68	26.30	26.93	0.46	28.38	27.47

Table 12. Architecture Parameters of pruned 13B models

Layer	Model-1			Model-2			Model-3		
	Type	Merge Factor	Output Scale	Type	Merge Factor	Output Scale	Type	Merge Factor	Output Scale
0	Base	-	1.00	LM	-	1.00	LM	-	1.00
1	LM	-	1.00	LM+Math	0.64	1.00	Base	-	1.00
2	LM	-	1.00	LM+Code	0.60	1.05	LM+Code	0.60	1.05
3	LM	-	1.00	LM	-	1.00	LM+Code	0.60	1.00
4	LM	-	1.00	LM	-	1.00	LM	-	1.00
5	Code	-	1.00	LM+Math	0.59	1.00	LM+Math	0.58	1.00
6	Base	-	1.00	LM	-	1.00	LM	-	1.00
7	LM	-	1.00	LM+Math	0.60	1.00	LM+Math	0.60	1.00
8	LM	-	1.00	LM	-	1.00	LM+Code	0.59	1.00
9	LM	-	1.00	LM	-	0.84	LM	-	0.93
10	LM	-	1.00	LM	-	1.02	LM	-	1.22
11	LM	-	1.00	LM+Code	0.66	0.77	LM+Math	0.66	1.00
12	LM	-	0.91	LM+Code	0.60	1.00	LM+Code	0.60	1.13
13	LM+Code	0.70	1.00	LM+Math	0.60	1.00	LM+Math +Code	0.60	1.11
14	LM+Math	0.70	1.00	LM+Math	0.60	1.00	LM	-	1.00
15	LM	-	1.00	LM+Math	0.70	1.00	LM+Math	0.66	1.00
16	Base	-	1.00	LM+Math	0.60	1.00	LM+Math	0.60	1.00
17	LM	-	1.00	LM	-	1.00	LM	-	1.00
18	LM	-	1.00	REMOVED			REMOVED		
19	LM+Code	0.70	1.00	LM+Code	0.60	1.00	LM+Code	0.60	1.01
20	LM+Code	0.70	1.00	LM	-	1.00	REMOVED		
21	LM	-	1.00	Base	-	1.07	Base	-	1.07
22	LM	-	1.00	Math	-	1.00	LM+Math	0.60	1.09
23	LM	-	1.00	REMOVED			REMOVED		
24	LM	-	1.00	Base	-	1.01	Base	-	1.01
25	REMOVED			REMOVED			REMOVED		
26	REMOVED			LM	-	1.04	LM	-	1.04
27	REMOVED			REMOVED			REMOVED		
28	REMOVED			REMOVED			REMOVED		
29	REMOVED			REMOVED			REMOVED		
30	REMOVED			Base	-	1.00	Base	-	1.00
31	REMOVED			REMOVED			REMOVED		
32	REMOVED			REMOVED			LM	-	1.00
33	REMOVED			REMOVED			REMOVED		
34	LM	-	1.00	Base	-	1.00	Code	-	1.00
35	Base	-	1.00	LM	-	1.13	LM	-	1.28
36	LM	-	1.00	REMOVED			REMOVED		
37	LM	-	1.00	LM	-	1.00	LM	-	1.00
38	LM	-	0.75	LM	-	1.00	Math	-	1.00
39	REMOVED			Math	-	1.00	Math	-	1.00

Table 13. Architecture Parameters of pruned 7B models

Layer	Model-1			Model-2			Model-1		
	Type	Merge Factor	Output Scale	Type	Merge Factor	Output Scale	Type	Merge Factor	Output Scale
0	LM	-	1.00	Math+Code	0.48	1.00	LM+Math	0.48	0.92
1	LM+Math+Code	0.50	1.00	LM	-	1.00	LM	-	1.00
2	LM	-	1.03	LM+Code	0.52	1.06	LM	-	1.03
3	LM	-	1.00	Base	-	0.98	Math	-	1.05
4	LM	-	1.04	LM	-	1.11	LM	-	1.11
5	LM+Code	0.59	1.08	LM+Math	0.38	1.12	LM	-	1.13
6	Code	-	1.19	Math	-	1.25	Code	-	1.11
7	Code	-	0.88	LM+Code	0.50	0.77	LM+Code	0.50	0.77
8	LM	-	1.28	LM	-	1.34	LM	-	1.19
9	LM	-	0.86	LM	-	0.93	LM+Code	0.51	0.56
10	Base	-	1.00	LM	-	1.00	LM	-	1.00
11	LM+Math	0.50	1.00	Math	-	1.02	LM	-	1.05
12	LM	-	1.00	LM+Math	0.41	0.99	LM+Math	0.41	1.00
13	Math	-	1.00	LM+Math	0.50	1.20	LM+Math	0.58	1.20
14	LM+Math	0.60	1.00	LM	-	1.00	LM+Math	0.54	1.00
15	LM	-	1.18	Code	-	0.97	Code	-	1.05
16	LM+Math	0.50	1.00	LM+Math	0.50	1.00	LM+Math	0.45	1.00
17	LM+Math+Code	0.50	1.00	Code	-	1.00	Math+Code	0.50	1.00
18	Math+Code	0.50	1.00	Base	-	1.00	Base	-	1.01
19	REMOVED			REMOVED			REMOVED		
20	REMOVED			REMOVED			REMOVED		
21	LM	-	1.00	REMOVED			LM	-	1.00
22	REMOVED			REMOVED			REMOVED		
23	REMOVED			REMOVED			REMOVED		
24	REMOVED			LM	-	1.00	REMOVED		
25	REMOVED			REMOVED			REMOVED		
26	REMOVED			REMOVED			REMOVED		
27	LM	-	1.00	Base	-	0.99	LM	-	0.99
28	REMOVED			LM	-	1.00	REMOVED		
29	LM+Code	0.50	1.00	LM	-	1.00	LM+Code	0.50	1.00
30	REMOVED			REMOVED			REMOVED		
31	LM+Math	0.50	1.00	REMOVED			LM+Math	0.50	1.00