GAP: a Global Adaptive Pruning Method for Large Language Models

Zhihua Ban[*†], **Haotian Ma**[*],
Siheng Zhang, **Shengyu Liu**, **Xichen Chen**, **Ming Yang**
CVTE Research
zhihua.ban@gmail.com, mahaotian@cvte.com,
{zhangsiheng,liushengyu,chenxichen,yangming}@cvte.com

Abstract

The deployment of Large Language Models (LLMs) faces significant challenges due to high computational costs, driving the demand for effective pruning techniques. Existing structured pruning methods employ uniform compression rates across network layers, neglecting the varying importance of different network depths. To address this limitation, we propose a novel optimization framework that directly minimizes global capability loss through layer-adaptive pruning rates. The framework formulates the pruning task as a combinatorial optimization problem constrained by a total parameter budget, and an efficient dynamic programming solution is derived to determine optimal layer-wise compression rates. Experiments demonstrate that, when tuning is not included, our approach achieves comparable performance with state-of-the-art methods at high pruning rates (37-50% reduction), and shows significant advantages at low pruning rates (13-25% reduction). When tuning is included, our method achieves the best performance among the compared methods.

1 Introduction

Large Language Models (LLMs) have achieved superior performance in a wide range of domains(Brown et al., 2020; Achiam et al., 2023; Zhao et al., 2023). However, the significant increase in both inference latency and memory cost still restricts its further applications, raising the demand for model compression techniques (quantization, pruning, distillation, etc. (Zhu et al., 2023; Tang et al., 2025)).

Structured pruning (Cheng et al., 2023) involves removing contiguous or complete structural units, such as filters, channels, or network layers, from the model. This facilitates straightforward implementation across diverse computing chips, which motivates our focus on structured pruning in this work.

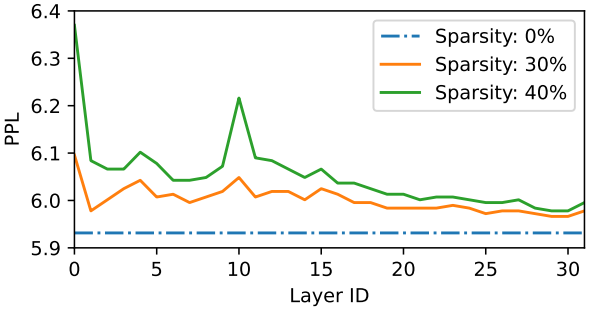

Figure 1: PPL comparison on WikiText-2 (Merity et al., 2017) test set for LLM-Pruner method with identical pruning rate applied to different layers while others remain unpruned, using the LLaMA2-7B (Touvron et al., 2023) model.

Although existing methods have achieved promising results, they still fall short in terms of accuracy loss after pruning. LLM-Pruner (Ma et al., 2023) achieves structured pruning by ranking the importance of coupled structures and employs LoRA fine-tuning to compensate for the accuracy loss caused by pruning. To avoid fine-tuning, SliceGPT (Ashkboos et al., 2024) applies orthogonal matrix transformations to each weight matrix and removes slices of the transformed matrices. However, both methods adopt a uniform pruning rate, which does not account for the varying abilities of the weights at different depths. As shown in Figure 1, layers at different depths contribute differently to the model's ability, as measured by PPL. This has also been empirically demonstrated in FLAP (An et al., 2023), where adaptive pruning rates are shown to be more likely to achieve better accuracy.

FLAP achieves adaptive pruning by formulating a structured importance metric that quantifies the recoverability of output features when removing weight columns, standardizing these scores

[*] Both authors contributed equally to this research.
[†] Corresponding author.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 20898–20903
November 4-9, 2025

across layers to dynamically determine pruning ratios. In LLM-Streamline (Chen et al., 2025), layers with high similarity are identified as redundant and pruned, while a lightweight network is trained via hidden state distillation to replace these pruned layers, ensuring minimal performance degradation. Although abilities in different depths are considered in both FLAP and LLM-Streamline, their pruning decisions fundamentally rely on localized layer-wise metrics. In this work, a novel framework is proposed to directly optimize the model's end-to-end performance after pruning, unlocking the room for further improvement in pruning accuracy.

As the main contributions of this work, we propose an optimization model that considers the end-to-end capability loss of the model after pruning. To solve this global optimization model, an efficient algorithm is derived by approximating the computation of the loss function. Experiments demonstrate that our method enhances pruning accuracy at various pruning rates, which is good news for LLM applications.

2 Method

We argue that neural network modules at varying depths contribute differently to the capabilities of LLMs. Therefore, we assign each module a set of selectable retained parameter counts M_{i,t_i}, where $i \in \{1, \ldots, N\}$ denotes the i-th module of an LLM, and N is the total number of modules. For instance, in LLaMA2-7B, each decoder layer is considered as a module, then N corresponds to the number of decoder layers. The pruning rates available for the i-th module are denoted by $t_i \in \{r_{i,1}, r_{i,2}, \ldots, r_{i,K_i}\}$, where K_i is the number of different pruning rates for the i-th module. Specifically, we set $r_{i,1} = 0\%$, which means that $M_{i,1}$ is the original parameter count of the i-th module. The optimization objective is to identify the optimal combination of retained parameter counts, subject to a constraint on the total retained parameters, as formalized below.

$$
\begin{aligned}
M^* &= \{M_1^*, \ldots, M_N^*\} && (1)\\
&= \arg\min J(M_{1,t_1}, \ldots, M_{N,t_N}), && (2)\\
\text{s.t.} \sum_{i=1}^{N} M_{i,t_i} &\in [M^L, M^U]. && (3)
\end{aligned}
$$

Here, $\{M_1^*, \ldots, M_N^*\}$ denotes the optimal combination of the retained parameter counts, with M_i^* indicating the retained parameter count selected for the i-th module in this combination. M^L and M^U represent the lower and upper bounds of the desired total retained parameter range, respectively. $J(\cdot)$ denotes the loss function corresponding to any given combination of retained parameters. The loss is calculated by first determining the optimal logit path of the original model on a calibration set, and then subtracting the sum of logits of the pruned model on the same path from the sum of logits of the original model on that path.

If each module has K selectable retained parameter counts, directly enumerating all combinations that satisfy the constraints would result in a number of combinations on the order of $O(K^N)$. For instance, when $K = 10$ and $N = 32$, this number becomes prohibitively large, rendering direct solution infeasible.

In AWQ (Lin et al., 2024), the selection of quantization parameters for each module is independent and has achieved state-of-the-art results. Inspired by this, we assume that the loss function satisfies the following equation.

$$
J(M_{1,t_1}, \ldots, M_{N,t_N}) = \sum_{i=1}^{N} J_i(M_{i,t_i}) \,, \qquad (4)
$$

where

$$
\begin{aligned}
J_i(M_{i,t_i}) = J(&M_{1,1}, \ldots, M_{i-1,1},\\
&M_{i,t_i}, M_{i+1,1}, \ldots, M_{N,1}).
\end{aligned} \qquad (5)
$$

In other words, the loss of the retained parameter count combination $\{M_{1,t_1}, \ldots, M_{N,t_N}\}$ is equal to the sum of N independent losses. Specifically, the i-th independent loss corresponds to the scenario where only the i-th module is pruned to M_{i,t_i}, while all other modules remain unpruned (i.e., $M_{j,t_j} = M_{j,1}$ for $i \neq j$).

Let $C_i(R_i)$ denote the minimum loss when the first i modules are pruned and the remaining $N - i$ modules are not pruned, given that the total number of retained parameters in the first i modules is R_i. Then,

$$
\begin{aligned}
C_i(R_i) &= \min \left\{ \sum_{j=1}^{i} J_j(M_{j,t_j}) \middle| R_i = \sum_{j=1}^{i} M_{j,t_j} \right\}\\
&= \min \left\{ C_{i-1}(R_{i-1}) + J_i(M_{i,t_i}) \;\middle|\; R_i = \right.\\
&\qquad \left. R_{i-1} + M_{i,t_i} \right\},
\end{aligned} \qquad (6)
$$

Algorithm 1 The Proposed Adaptive Pruning Algorithm

Assign an appropriate set of retained parameter counts to each module.
for each M_{i,t_i} **do**
    Perform pruning on a validation set using LLM-Pruner (Ma et al., 2023) without tuning.
    Compute $J_i(M_{i,t_i})$ using Equation 5.
end for
Initialize $C_1(R_1)$ according to Equation 7.
for $i = 2$ to N **do**
    for each R_{i-1} **do**
        for each M_{i,t_i} **do**
            if $C_{i-1}(R_{i-1}) + J_i(M_{i,t_i}) < C_i(R_i)$ **then**
                Update $C_i(R_i) = C_{i-1}(R_{i-1}) + J_i(M_{i,t_i})$.
                Update the optimal combination for R_i to R_{i-1} and M_{i,t_i}.
            end if
        end for
    end for
end for
Determine R_N^* and M^* according to Equation 8.

where

$$C_1(R_1) = J_1(R_1)\ ,\quad R_1 \in \left\{M_{1,t_1} \mid t_1 \in \{r_{1,1}, \ldots, r_{1,K_1}\}\right\}. \tag{7}$$

Clearly, the combination of retained parameter counts corresponding to R_N^* is the optimal solution M^*, where

$$R_N^* = \arg\min\{C_N(R_N) \mid M^L \leq R_N \leq M^U\} \tag{8}$$

By reasonably partitioning an LLM, the same set of K retained parameter counts can be assigned to each module, excluding the LM head and word embeddings. The number of possible values for R_N does not exceed the number of combinations of placing N identical balls into K distinct bins, given by $\binom{N+K-1}{K-1}$. Therefore, even if we compute all possible R_N values, the time complexity will not exceed $O\left(\binom{N+K-1}{K-1}NK\right)$. This complexity is significantly lower than the aforementioned exponential time complexity. For example, when $N = 32$ and $K = 10$, running the solution program on a 2.5 GHz CPU yields results with only a few seconds of delay.

Setting each linear layer of an LLM as a module incurs a high computational cost, especially for calculating $J_i(M_{i,t_i})$. Consequently, we designate each decoder layer as a basic module. For each M_{i,t_i}, the retained parameters is determined by LLM-Pruner (Ma et al., 2023) without performing the tuning step. The overall procedure of our algorithm is depicted in Algorithm 1.

3 Experiments[1]

The effectiveness of our algorithm is demonstrated on LLaMA2-{7B,13B} (Touvron et al., 2023), LLaMA3.1-8B (Grattafiori et al., 2024) and Qwen2.5-{7B,14B} (Qwen et al., 2025), and is benchmarked against state-of-the-art open-source pruning algorithms. These include methods without tuning, DS (Dumitru et al., 2024), FLAP (An et al., 2023) and SliceGPT (Ashkboos et al., 2024), as well as methods with tuning, LLM-Streamline (Chen et al., 2025) and LLM-Pruner (Ma et al., 2023). For fair comparison, all methods use the same calibration set, which consists of 128 samples, each with a length of 2048 tokens, randomly selected from WikiText-2 (Merity et al., 2017). Methods that require tuning are fine-tuned on Alpaca (Taori et al., 2023).

The performance of all compared algorithms is evaluated on the WikiText-2 test set using perplexity (PPL) as the metric. Additionally, the algorithms are assessed on seven zero-shot tasks through the *lm-eval-harness* (Gao et al., 2024), including BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-easy (Clark et al., 2018), ARC-challenge (Clark et al., 2018), and OpenbookQA (Mihaylov et al.,

[1]All datasets, models, and other artifacts used in this study are publicly available and can be freely used for academic purposes. We have ensured that our use of these resources complies with their respective licenses and terms of use.

2018). The average accuracy across these tasks is reported for each method.

For each module $i \in \{1, \ldots, N\}$, we define a set of $K = 10$ pruning rate options, $\{r_{i,1}, \ldots, r_{i,K}\}$, as $\{0\%, 10\%, \ldots, 90\%\}$. As metioned in secion 2, the search complexity for finding all $C_N(R_N)$ is at most $O\left(\binom{N+K-1}{K-1} NK\right)$. By merging redundant search paths, the effective search space is made significantly smaller than this upper bound. Table 1 summarizes this effective space, along with the corresponding time and memory requirements, with all experiments run on a 2.45GHz CPU.

Model	Effective Search Space Size ($\times 10^3$)	Search Time (ms)	Search Memory (KiB)
LLaMA2-7B	45	20	200
LLaMA2-13B	71	32	250

Table 1: Resources required by the search procedure across models.

Rate	Method	LLaMA2-7B		LLaMA2-13B		LLaMA3.1-8B	
		PPL ↓	Acc ↑	PPL ↓	Acc ↑	PPL ↓	Acc ↑
0%	Dense	5.47	64.71	4.88	67.74	6.22	69.12
13%	DS	7.34	53.98	6.41	58.74	10.28	50.66
	LLM-Pruner	6.99	56.85	5.49	63.62	8.37	60.04
	SliceGPT	7.25	54.02	6.38	60.68	11.92	49.30
	FLAP	6.39	59.78	5.75	62.05	**7.96**	58.70
	Ours	**6.36**	**62.63**	**5.32**	**66.12**	8.24	**65.12**
25%	DS	9.71	48.86	8.22	52.33	14.42	44.82
	LLM-Pruner	16.65	42.47	8.91	53.71	34.17	48.14
	SliceGPT	9.5	48.75	8.18	51.30	26.71	41.54
	FLAP	8.75	52.36	7.6	57.51	12.37	49.56
	Ours	**8.37**	**57.16**	**6.99**	**63.41**	**11.62**	**54.07**
37%	DS	14.65	43.69	11.87	47.50	21.94	41.08
	LLM-Pruner	45.26	39.11	31.11	40.65	148.41	42.32
	SliceGPT	15.72	42.84	12.35	45.49	39.78	39.31
	FLAP	13.77	**48.70**	10.5	52.59	**19.13**	45.03
	Ours	**13.17**	48.69	**9.48**	**55.87**	19.46	**48.90**
50%	DS	**25.46**	40.38	20.14	42.16	**35.15**	39.22
	LLM-Pruner	367.33	37.76	708.4	38.49	295.15	37.67
	SliceGPT	27.46	39.64	22.59	40.72	67.26	37.81
	FLAP	31.56	**44.44**	16.08	**49.52**	78.03	40.12
	ours	30.62	43.53	**14.69**	48.22	58.11	**41.64**

Table 2: Performance of LLaMA models pruned by different methods without tuning. Bold values highlight the best performance.

As shown in Table 2, our method achieves the best PPL on WikiText-2. On zero-shot tasks, our method exhibits comparable performance to the second-best FLAP method at high pruning rates but outperforms almost all others at low pruning rates. Specifically, at a 25% pruning rate for zero-short tasks, our method incurs a loss of approximately 11.7% for the 7B model and 6.4% for the 13B model. In contrast, FLAP experiences a loss of 19.1% for the 7B model and 15.1% for the 13B model. This indicates that our method's performance is nearly twice that of the second-best method. Evidently, if a smaller loss in accuracy is desired in exchange for a low pruning rate, our method is a better choice.

Rate	Method	LLaMA2-7B		LLaMA2-13B		LLaMA3.1-8B	
		PPL ↓	Acc ↑	PPL ↓	Acc ↑	PPL ↓	Acc ↑
0%	Dense	5.47	64.71	4.88	67.74	6.26	69.12
13%	LLM-Pruner*	6.57	61.37	5.40	65.28	8.24	64.22
	StreamLine*	7.24	63.88	5.93	**68.12**	9.94	**68.80**
	ours*	**6.36**	**64.00**	**5.36**	67.72	**8.02**	66.32
25%	LLM-Pruner*	8.5	55.80	7.21	61.37	15.16	53.86
	StreamLine*	11.99	58.35	7.74	64.22	14.47	58.02
	ours*	**7.99**	**62.40**	**6.62**	**65.76**	**10.75**	**60.75**
37%	LLM-Pruner*	16.91	54.24	11.62	61.15	22.75	47.94
	StreamLine*	11.09	51.56	17.18	51.97	24.22	**58.02**
	ours*	**9.94**	**55.97**	**7.69**	**62.26**	**13.80**	57.25
50%	LLM-Pruner*	173.51	39.89	277.27	39.87	34.17	44.11
	StreamLine*	40.57	40.57	19.46	51.80	41.86	45.20
	ours*	**13.8**	**49.47**	**10.09**	**55.40**	**20.40**	**52.18**

Table 3: Performance of LLaMA models pruned by different methods with tuning. Bold values highlight the best performance. The asterisk (*) denotes that the method needs tuning.

Our experiments show that LLM-Pruner with full-parameter tuning outperforms its LoRA-tuned version from the original paper. For fair comparison, both LLM-Pruner* and our method (ours* in Table 3) employ full-parameter tuning. As shown in Table 3, our method achieves the best performance on both WikiText-2 and zero-shot tasks across all tested pruning rates. Notably, the performance loss of our pruned model relative to the dense model, averaging 3.2% for the 7B and 13B models, is approximately half that of the second-best method, which averages 7.5% for the 7B and 13B models, at a 25% pruning rate for zero-shot tasks. This further confirms that our method achieves superior accuracy particularly at a lower pruning rate.

In addition, we evaluate two Qwen variants. Table 4 shows that our method attains the highest accuracy under low sparsity and matches the best-reported results under high pruning ratios. As shown in Table 5, our method outperforms nearly all tuning-based competitors.

4 Conclusion

This work addresses the critical challenge of preserving model capabilities during LLM pruning through layer-adaptive pruning rates. We formulate structured pruning as a combinatorial optimization problem that explicitly minimizes global capability loss under parameter budget constraints. By establishing a dynamic programming framework, our method efficiently determines the optimal layer-wise pruning rates. Experiments on

Rate	Method	Qwen2.5-7B		Qwen2.5-14B	
		PPL ↓	Acc ↑	PPL ↓	Acc ↑
0%	Dense	6.83	69.90	5.28	72.46
13%	LLM-Pruner	7.87	66.35	6.62	70.05
	FLAP	7.79	**68.98**	6.84	66.54
	Ours	**7.74**	66.93	**6.36**	**71.88**
25%	LLM-Pruner	9.20	60.63	8.77	64.35
	FLAP	9.93	54.88	9.22	57.73
	Ours	**9.05**	**60.95**	**8.63**	**65.05**
37%	LLM-Pruner	**12.77**	49.90	**13.51**	**54.27**
	FLAP	14.37	49.05	21.16	43.82
	Ours	15.88	**53.35**	14.69	54.07
50%	LLM-Pruner	**23.85**	**43.40**	34.70	42.24
	FLAP	28.92	14.94	4942.58	37.95
	Ours	24.61	42.45	**33.11**	**43.34**

Table 4: Performance of Qwen models pruned by different methods without tuning. Bold values highlight the best performance.

Rate	Method	Qwen2.5-7B		Qwen2.5-14B	
		PPL ↓	Acc ↑	PPL ↓	Acc ↑
0%	Dense	6.83	69.90	5.28	72.46
13%	LLM-Pruner*	7.86	68.44	6.57	71.69
	StreamLine*	8.37	65.79	7.5	69.19
	Ours*	**7.74**	**70.52**	**6.22**	**72.21**
25%	LLM-Pruner*	9.05	62.77	8.77	64.35
	StreamLine*	10.92	58.47	9.78	62.36
	Ours*	**8.91**	**65.64**	**8.11**	**66.54**
37%	LLM-Pruner*	**11.09**	57.77	10.75	60.21
	StreamLine*	13.17	55.69	14.92	54.62
	Ours*	12.37	**59.66**	**10.58**	**60.93**
50%	LLM-Pruner*	14.69	51.72	16.91	51.71
	StreamLine*	34.17	46.06	30.62	47.5
	Ours*	**14.29**	**51.74**	**15.88**	**52.78**

Table 5: Performance of Qwen models pruned by different methods with tuning. Bold values highlight the best performance. The asterisk (*) denotes that the method needs tuning.

LLaMA and Qwen models demonstrate that our method achieves comparable performance with state-of-the-art approaches at high pruning rates (37-50% reduction), and significantly outperforms existing methods at low pruning rates (13-25% reduction), particularly for zero-shot tasks compared to the second-best approach. These results highlight the effectiveness of end-to-end consideration of accuracy loss and adaptive pruning rate assignment for different modules in LLM compression.

Limitations

Our method inherits the strength of structured pruning methods and hence is friendly for implementation on computing chips. Despite achieving promising results in structured pruning, our method still has room for improvement in terms of accuracy compared to other model compression techniques, such as unstructured pruning and quantization methods. We plan to explore distillation methods to transfer the knowledge from the original model to the pruned model, thereby further improving the accuracy of the pruned model.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. 2023. Fluctuation-based adaptive structured pruning for large language models. *Preprint*, arXiv:2312.11983.

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

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Xiaodong Chen, Yuxuan Hu, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. 2025. Streamlining redundant layers to compress large language models. *Preprint*, arXiv:2403.19135.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. 2023. A survey on deep neural network pruning:taxonomy, comparison, analysis, and recommendations. *arXiv preprint arXiv:2308.06767*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind

Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Razvan-Gabriel Dumitru, Paul-Ioan Clotan, Vikas Yadav, Darius Peteleaza, and Mihai Surdeanu. 2024. Change is the only constant: Dynamic llm slicing based on layer redundancy. *Preprint*, arXiv:2411.03513.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. A framework for few-shot language model evaluation.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for llm compression and acceleration. In *MLSys*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

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

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2).

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2023. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 12:1556–1577.