

Breaking Expert Knowledge Limits: Self-Pruning for Large Language Models

Haidong Kang¹ Lihong Lin¹ Enneng Yang² Hongning Dai³ Hao Wang^{4*}

¹Northeastern University, China ²Sun Yat-sen University, China

³Hong Kong Baptist University, China

⁴Xidian University, China

hdkang@stumail.neu.edu.cn, linlh@mails.neu.edu.cn, yangenn@mail.sysu.edu.cn

hndai@ieee.org, haow@ieee.org

Abstract

Large language models (LLMs) have achieved remarkable performance on a wide range of tasks, hindering real-world deployment due to their massive size. Existing pruning methods (e.g., Wanda) tailored for LLMs rely heavily on manual design pruning algorithms, thereby leading to huge labor costs and requires expert knowledge. Furthermore, we are the first to identify the serious outlier value issue behind dramatic performance degradation under high pruning ratios that are caused by uniform sparsity, raising an additional concern about how to design adaptive pruning sparsity ideal for LLMs. Can LLMs prune by themselves? In this work, we introduce an affirmative answer by proposing a novel pruning method called AutoPrune, which first overcomes expert knowledge limits by leveraging LLMs to design optimal pruning algorithms for themselves automatically without any expert knowledge. Specifically, to mitigate the black-box nature of LLMs, we propose a Graph-driven Chain-of-Thought (GCoT) to optimize prompts, significantly enhancing the reasoning process in learning the pruning algorithm and enabling us to generate pruning algorithms with superior performance and interpretability in the next generation. Finally, grounded in insights of outlier value issue, we introduce Skew-aware Dynamic Sparsity Allocation (SDSA) to overcome the outlier value issue, mitigating performance degradation under high pruning ratios. We conduct extensive experiments on mainstream LLMs benchmarks, demonstrating the superiority of AutoPrune, which consistently excels state-of-the-art competitors. The code is available at: <https://anonymous.4open.science/r/AutoPrune>.

1. Introduction

Large Language Models (LLMs) [1, 9, 14, 31] have attracted substantial attention from both academia and indus-

try, which has reshaped the study of diverse tasks due to superior performance, e.g., language translation [3], code generation [15]. However, the huge size and computational costs of LLMs (e.g., the GPT-175B [2] with 175 billion parameters, taking 350GB of memory with FP16 for inference) have become a key bottleneck for widespread real-world deployment in both edge and cloud scenarios [40]. This highlights the need to craft lightweight LLMs, facilitating deployment on resource-constrained devices.

So far, several pruning methods tailored for LLMs have been proposed [7, 30, 34], liberating the LLMs from the above bottleneck in a training-free manner from the lens of unstructured pruning. In short, the core idea is to measure weight importance using statistical or theoretical properties of model weights or activations. For instance, Magnitude [17] is a superior pruning method for DNNs, which is also adopted for LLMs pruning due to only relying on the magnitude of weights. However, SparseGPT [7] finds that “Magnitude” will result in dramatic performance degradation even under low levels of sparsity. To mitigate this problem, SparseGPT [7] adopts the Optimal Brain Surgeon (OBS) [10] to evaluate weight importance based on the statistical characteristics of the Hessian matrix, enabling selective pruning of insignificant weights. Moreover, SparseGPT suffers from huge computational budgets due to the expensive second-order optimization. To address this, Wanda [30] simplifies the computations by measuring the product of weight and activation magnitudes.

Challenges. Although these methods have taken the first step tailored for lightweight LLMs in an unstructured pruning way, they still suffer from critical limitations: (1) *rely heavily on extensive human expert knowledge through tens of thousands of trial-and-error (as shown in Fig. 1(b)), resulting in significant labor costs;* (2) *sensitivity to outlier weights, causing severe performance degradation under high pruning ratios (as depicted in Fig. 3).* Those issues highlight the need to rethink the design of pruning algorithms tailored for LLMs.

Our New Observation. Inspired by the advanced capaci-

*Corresponding author

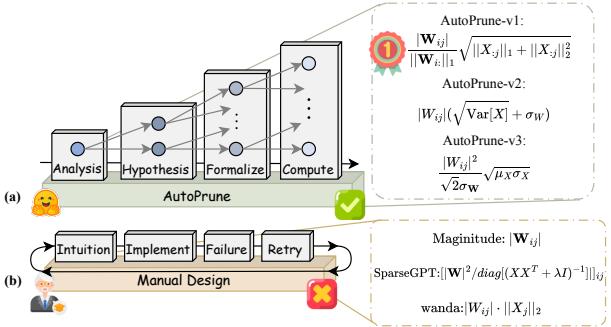


Figure 1. (a) AutoPrune v.s. (b) Manual Design. Manual design requires expert knowledge, resulting in huge labor costs. In contrast, our AutoPrune can efficiently design several specialized pruning algorithms by leveraging LLMs.

ties of LLMs (e.g., code/text generation, contextual reasoning), we raise a new question: **Can LLMs prune by themselves?** In this paper, we propose a novel paradigm to automatically design pruning algorithms via LLMs (as depicted in Fig. 1(a)), which revolutionizes the design paradigm of pruning tailored for LLMs. Compared with previous methods crafted by human experts, our method can effectively and efficiently design pruning algorithms with optimal performance (as shown in Fig. 2) without any expert knowledge. Furthermore, we observe a fatal outlier value issue in LLMs by skewness [29] (as shown in Fig. 3), which is sensitive to the pruning ratios of specific layers and dramatically decreases the performance of pruning methods. To reveal the root cause, we conduct an in-depth analysis of the underlying mechanism of the outlier value issue.

Contributions. In short, this paper aims to understand and address the aforementioned critical limitations caused by relying on extensive human expert knowledge and the outlier value issue. To tackle the limitations of relying on extensive human expert knowledge, we first revisit the mechanism of LLMs pruning (as shown in Table 2), and find that previous methods (i.e., SparseGPT, Wanda) are heuristics, relying on statistical or theoretical properties of weights or activations in LLMs. Motivated by the powerful knowledge generation capabilities of LLMs, we propose a novel pruning method (dubbed **AutoPrune**), which first revolutionizes the paradigm of LLMs pruning via automatically designing pruning algorithms by itself (as represented in Fig. 1). Before introducing our reasoning module, we first conduct a controlled study to understand whether *reasoning at all* helps an LLM design better pruning rules. We compare three variants under the same evaluation protocol: (i) a naive single-shot prompting baseline (no CoT), (ii) a linear step-by-step CoT that follows one reasoning trajectory, and (iii) a graph-driven variant that explores multiple trajectories. As summarized in Table 1, the naive baseline performs worst (MMLU [12] 27.25, WikiText-2 [21] 16.55), a linear

CoT improves but remains limited (29.15, 7.17), while the graph-driven variant attains the best results (29.69, 6.28). These observations indicate that merely appending a single chain-of-thought is insufficient; exploring and selecting among multiple reasoning paths is crucial for robustly discovering stronger pruning rules. Notably, to obtain positive feedback of LLMs-driven pruning algorithms designed for the target task, we propose Graph-driven Chain-of-Thought (GCoT) to enhance reasoning by measuring positive gain from the target task, enabling LLMs to craft better pruning algorithms in the next generation.

To tackle the outlier value issue, we perform an experimentation by skewness analysis on LLaMA-1-7B with Magnitude (Fig. 3), validating the statement that existing methods suffer from the outlier value issue, some layers in LLMs are sensitive to the outlier value issue, which can lead to significant performance degradation under high pruning ratios. To reveal the root cause, we conduct an in-depth analysis of the outlier value issue (as shown in Fig. 4), and observe that uniform sparsity could be the potential reason for the outlier value issue, where the importance of the layer is ignored. Grounded in empirical observations via skewness analysis, we propose Skew-aware Dynamic Sparsity Allocation (SDSA), which perfectly overcomes the outlier value issue by preventing the over-pruning of certain layers and the over-preservation of others, leading to a more balanced and stable adaptation trajectory.

To sum up, the main contributions of this paper are:

- **New LLMs Pruning Paradigm.** To the best of our knowledge, this paper is the pioneering effort to propose a novel LLM-pruning paradigm by leveraging LLMs to design optimal pruning algorithms for themselves automatically without any expert knowledge, breaking expert knowledge limits.
- **Outlier Value Issue.** We thoroughly scrutinize the mechanisms of LLMs pruning through which high pruning ratios affect the performance, and first find that the outlier value issue in LLMs pruning. To reveal the root cause, we conduct an in-depth analysis and find that the outlier value issue is attributed to uniform sparsity.
- **GCoT driven Self-pruning.** To address the concern of black-LLMs, we propose a GCoT to build a reasoning engine, shaping self-pruning by obtaining positive reward signals from target tasks.
- **Numerical Verification.** Extensive experiments show that our method consistently outperforms previous state-of-the-art competitors.

2. Rethinking the Design of LLMs Pruning

The main goal of unstructured pruning for LLMs is to learn a sparse and smaller subnetwork with minimal performance degradation through a training-free way. This raises a key problem: *how can we accurately measure the importance*

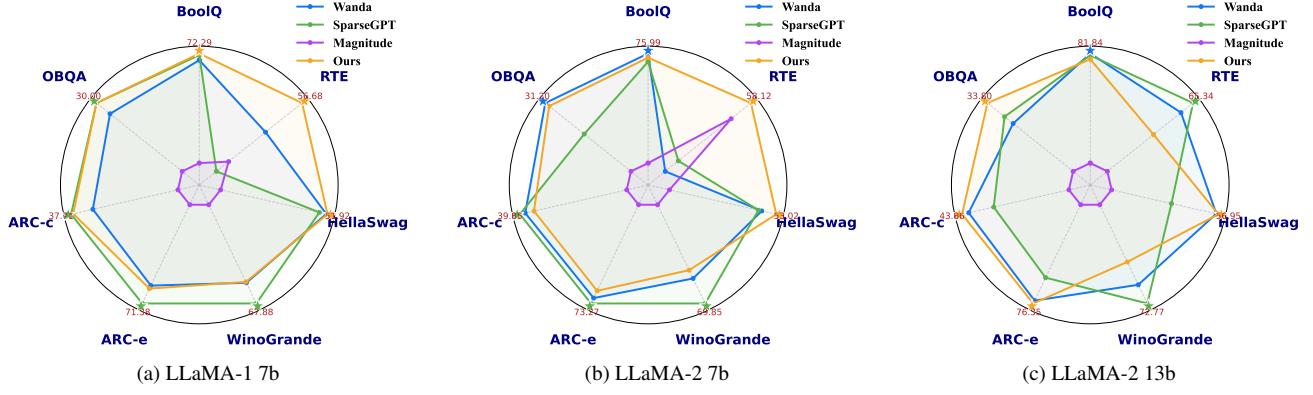


Figure 2. Our AutoPrune v.s. peer competitors on 7 zero-shot tasks. (a) LLaMA-1 7b. (b) LLaMA-2 7b. (c) LLaMA-2 13b.

Table 1. Comparison of different reasoning strategies on **LLaMA-2 7B**. Metrics are reported on MMLU and WikiText-V2 (%).

Method	Step-by-step CoT	GCoT	MMLU	Wiki
Naive	✗	✗	27.25	16.55
Naive (step-by-step)	✓	✗	29.15	7.17
AutoPrune (ours)	✓	✓	29.69	6.28

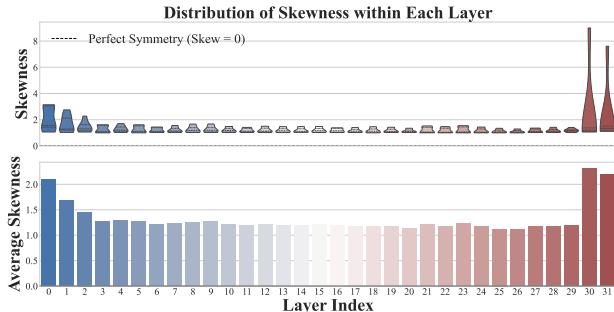


Figure 3. Validation on outlier value issue by skewness. Top: per-layer skewness. Bottom: mean skewness.

of each weight? Naturally, we can leverage some statistical or theoretical properties of model weights or activations as an effective metric to obtain the importance of each weight. For instance, Wanda utilizes the magnitude of weights and input activations, enabling LLMs to effectively prune insignificant weights. However, those methods heavily rely on human experts (as shown in Table 2), which results in huge labor costs. Additionally, we validate that existing methods suffer from the outlier value issue, which leads to severe performance degradation under high pruning ratios. For the sake of clarity, we provide a series of strict validations to support our statements and reveal the root cause.

2.1. Experimentation Outlier Value Issue

To verify and understand the outlier value issue, we conduct a detailed experiment on LLaMA-1-7B by skewness sensitivity analysis. To be specific, we utilize skewness (see Eq.

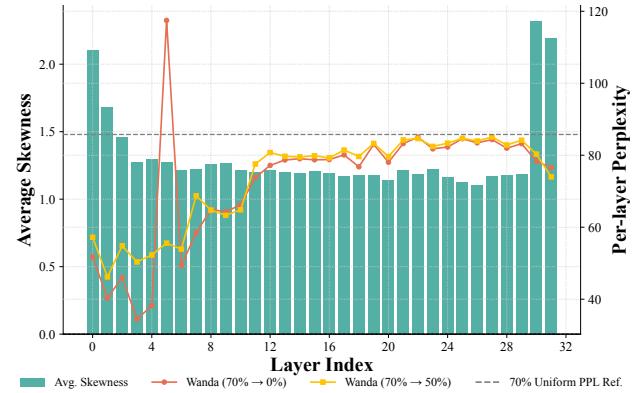


Figure 4. Validation of layer sensitivity to pruning ratios.

4 for its definition) as a layer importance metric, accurately measuring skewness for the per-layer weights. As shown in Fig. 3, we show the distribution of skewness and average skewness for LLaMA-1-7B. Fig. 3 can yield the conclusion: Layers whose weight distributions exhibit stronger positive skewness tend to be more sensitive to pruning. Removing a small number of large weights can disproportionately degrade performance. For instance, removing the weights of “layer 1” at 70% will result in PPL increasing (from 45 to 95). In this paper, we define a layer with stronger positive skewness as “Outlier Value Issue”.

2.2. Further Analysis

What factor of pruning causes the outlier value issue?

To answer the above questions, we perform a series of strict validations using Wanda on LLaMA-1-7B in WikiText-2. To be specific, Fig. 4 shows the layer sensitivity to different pruning ratios, where experiments are conducted on LLaMA-1-7B in WikiText-2 via varying pruning ratios of layer l_i to 0% and 50%, keeping others to 70%. As shown, the layer with the outlier value will result in high PPL (lower is better) at 70% high pruning ratios. By contrast, if we as-

Table 2. AutoPrune v.s. previous methods. “ d ” denotes hidden dimension.

Name	Formula	Human Expert	Calibration Data	Calibration Sample	Weight Update	Complexity
Magnitude	$ \mathbf{W}_{ij} $	✗	✗	0	✓	$O(1)$
SparseGPT	$[\ \mathbf{W}\ / \text{diag}[(\mathbf{X}\mathbf{X}^T + \lambda I)^{-1}]]_{ij}$	✓	✓	128	✓	$O(d^3)$
Wanda	$ \mathbf{W}_{ij} \cdot \ X_j\ _2$	✓	✓	128	✗	$O(d^2)$
AutoPrune (ours)	$\frac{ \mathbf{W}_{ij} }{\ \mathbf{W}_i\ _1} \sqrt{\ X_{:,j}\ _1 + \ X_{:,j}\ _2^2}$	✗	✓	128	✗	$O(d^2)$

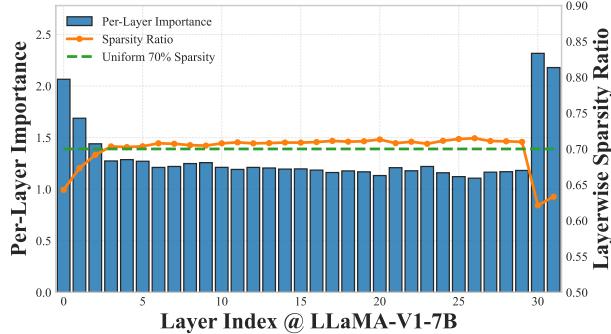


Figure 5. SDAS v.s. Uniform allocation at 70% sparsity.

Table 3. Empirical validation of our SDSA with LLaMA-1/2 on WikiText at 60% high sparsity.

Methods	LLaMA-1 7B	LLaMA-1 13B	LLaMA-2 7B	LLaMA-2 13B
SparseGPT	10.51	8.56	9.58	7.80
SparseGPT + SDSA	8.85	7.01	7.69	6.46
Wanda	10.66	8.56	9.71	7.75
Wanda + SDSA	9.88	7.65	8.89	6.98
AutoPrune	9.63	7.63	8.93	7.02
AutoPrune + SDSA	9.42	7.58	8.88	6.81

sign dynamic pruning ratios to those layers, the PPL will significantly decrease. In brief, outlier value is associated with uniform sparsity. To validate our statement, we conduct an additional experiment using Wanda on LLaMA-1-7B with our SDSA. As shown in Fig. 5, our SDSA provides a better dynamic sparsity. More validations are provided in App. A.

Conclusion. Consequently, the root cause of the outlier value issue can be attributed to uniform sparsity. This motivated us to design SDSA to mitigate the outlier value issue. As shown in Table 3, our SDSA successfully boosts the performance of all pruning methods (i.e., Wanda).

3. AutoPrune: Self-Pruning for LLMs

Can LLMs Self-Design Pruning Algorithms? We argue that LLMs are not only capable of this complex design task but are uniquely positioned to excel at it. LLMs possess vast, latent knowledge of network principles. They are trained on a colossal corpus of scientific literature, technical blogs, and open-source code, effectively internalizing decades of collective human knowledge on neural networks and optimization. The models’ parameters hold a representation of what makes architectures work, what causes performance drops, and which patterns are redundant.

However, even advanced proprietary models such as GPT-o3 [27] and DeepSeek-R1 [9], which demonstrate strong emergent reasoning abilities, still suffer from significant limitations in reasoning. Even with reasoning capabilities, when these models face specific, specialized downstream tasks that demand structured and goal-directed thinking, they still inevitably run into problems with synthesizing information, making logical inferences, or handling multi-step planning. To address this gap, we introduce a Graph-driven Chain-of-Thought (GCoT) reasoning module. GCoT is explicitly designed to augment the reasoning ability of LLMs, providing a structured mechanism to guide pruning decisions through interpretable multi-step reasoning, thereby enabling our method to leverage the model’s latent knowledge more effectively.

3.1. GCoT Driven Self-Pruning

Chain of Thought. To harness the LLM’s potential, we design a Chain of Thought (CoT) pipeline that guides the model from a high-level goal to an implementable pruning algorithm. As illustrated in Fig. 6, the process includes four stages: (1) **Analysis**. the LLM analyzes model statistics to identify signals (e.g., weights, activations) relevant to pruning sensitivity; (2) **Hypothesis**. it synthesizes a concise causal statement linking these signals to parameter importance; (3) **Conceptual Formula**. the hypothesis is formalized into a semi-structured computational relation; and (4) **Computable Concept**. the relation is decomposed into modular, machine-readable components defining inputs, outputs, and operations.

This CoT procedure yields a complete, traceable algorithm, yet its linear nature restricts exploration to a single reasoning path. Suboptimal outcomes may arise simply because one trajectory fails to express a promising idea. This limitation motivates our Graph-driven CoT (GCoT), which systematically explores multiple reasoning branches to search for stronger pruning algorithms.

Graph-Driven CoT. To explore a broader algorithmic space beyond a single reasoning trajectory, we generalize linear CoT into a directed acyclic reasoning graph. Let each node s represent a reasoning state (e.g., *Analysis*, *Hypothesis*), and each edge be an LLM expansion step. At selected reasoning stages, we perform **temperature-based branching**:

$$\text{Expand}(s) = \{\text{LLM}(s; T_i)\}_{i=1}^k, \quad (1)$$

where T_i is the sampling temperature and k is the XXX.

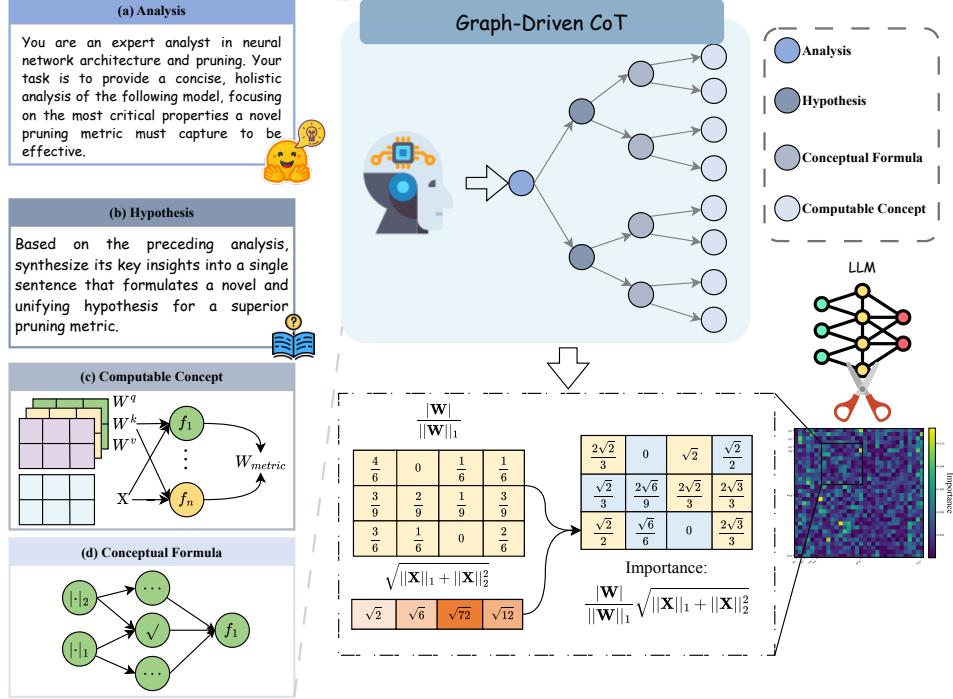


Figure 6. The intuitive framework of GCoT-driven self-pruning. This framework leverages an LLM to automatically generate pruning algorithms effectively, with the example below illustrating an optimal method discovered through this process.

his yields a graph \mathcal{G} containing multiple distinct reasoning paths, and each root-to-leaf path p corresponds to a complete pruning algorithm.

Each candidate algorithm induces an importance score function $I^{(p)}(W)$, which we evaluate by pruning the target model to sparsity α and computing perplexity on wikitext-2:

$$\mathcal{E} = \text{PPL}(\text{Prune}(W, I^{(p)}, \alpha)). \quad (2)$$

We then select the best algorithm by:

$$p^* = \arg \min_p \mathcal{E}(p). \quad (3)$$

This graph-driven process enables systematic exploration of diverse algorithmic structures while reusing shared reasoning components, improving both search efficiency and final pruning performance.

3.2. Skew-aware Dynamic Sparsity Allocation

Grounded in aforementioned observations of outlier value, we propose **Skew-Aware Dynamic Sparsity Allocation (SDSA)**, which combines (i) a simple yet effective skewness-indexed layer prior that adaptively reduces pruning in highly skewed outlier-bearing layers and (ii) a global sparsity-aware calibration term that tempers the skewness prior under low global sparsity, preventing both disproportionate pruning of some layers and excessive protection of others, and yielding a more stable adaptation trajectory.

Skewness Measurement. Skewness serves in our work as a quantitative indicator of the outlier value issue. Intuitively, layers whose weight magnitude distribution exhibits stronger positive skewness contain a small fraction of extremely large weights that disproportionately influence the model’s output. For layer ℓ , we compute the biased sample skewness of its weight distribution, intentionally omitting the small-sample correction since we analyze the LLMs weights rather than infer a population moment.

Let $W_\ell = \{w_{\ell,i}\}_{i=1}^{n_\ell}$ denote all weights in layer ℓ . Because our pruning strategy depends on the prevalence of large-magnitude weights rather than the sign structure, we operate on the absolute values $A_\ell = \{|w_{\ell,i}|\}_{i=1}^{n_\ell}$. We then characterize the asymmetry of the magnitude distribution via the biased sample skewness, computed as:

$$\begin{aligned} \bar{w}_\ell &= \frac{1}{n_\ell} \sum_{i=1}^{n_\ell} |w_{\ell,i}|, \quad m_{2,\ell} = \frac{1}{n_\ell} \sum_{i=1}^{n_\ell} (|w_{\ell,i}| - \bar{w}_\ell)^2, \\ \gamma_\ell &= \frac{1}{n_\ell} \sum_{i=1}^{n_\ell} \left(\frac{|w_{\ell,i}| - \bar{w}_\ell}{\sqrt{m_{2,\ell}}} \right)^3. \end{aligned} \quad (4)$$

Raw skewness magnitudes can vary across layers. To obtain a scale-stable index that is comparable across layers within a model, we center and range-normalize the $\{\gamma_\ell\}$ to $[-1, 1]$. Let:

$$\bar{\gamma} = \frac{1}{L} \sum_{\ell=1}^L \gamma_\ell, \quad \Delta\gamma = \max_j |\gamma_j - \bar{\gamma}|, \quad (5)$$

and define

$$\tilde{\gamma}_\ell = \frac{\gamma_\ell - \bar{\gamma}}{\Delta\gamma + \varepsilon}, \quad \tilde{\gamma}_\ell \in [-1, 1], \quad (6)$$

where ε is a constant, typically set to 1e-8 to avoid division by zero. Positive $\tilde{\gamma}_\ell$ indicates a layer whose magnitude distribution is more positively skewed than the model-wide mean. Negative values indicate the opposite. We then convert the normalized skewness scores into relative protection weights through a softmax:

$$\omega_\ell = \frac{\exp(\beta\tilde{\gamma}_\ell)}{\sum_{j=1}^L \exp(\beta\tilde{\gamma}_j)}, \quad (7)$$

where β is a coefficient that is dynamically modulated by the global sparsity level and skewness, and ω_ℓ denotes the retention fraction for layer ℓ .

Global-Sparsity-Aware Schedule. Based on the aforementioned insights, instead of hand-tuning β directly, we devise an intuitive schedule to adapt β automatically. We aim to ensure the ratio between the largest and smallest protection weights ω_{\max} and ω_{\min} remains bounded:

$$\frac{\omega_{\max}}{\omega_{\min}} = \exp(\beta\Delta\tilde{\gamma}) \leq M, \quad \Delta\tilde{\gamma} = \max_\ell \tilde{\gamma}_\ell - \min_\ell \tilde{\gamma}. \quad (8)$$

Solving for β yields $\beta = \frac{\ln M}{\Delta\tilde{\gamma} + \varepsilon}$. To flatten allocations when global sparsity ratio $S_g \in [0, 1]$ is small, we modulate this contrast linearly: $\beta(S_g) = S_g \frac{\ln M}{\Delta\tilde{\gamma} + \varepsilon}$.

Thus, $\beta(0) = 0$ gives a uniform start, and $\beta(S_g)$ grows smoothly with the demanded pruning level, gradually releasing skewness sensitivity. In practice, we cap the allowable contrast with a modest value ($M=1.8$ for layerwise).

4. Experiments

We conduct the experimental validation of AutoPrune across multiple mainstream 7 LLMs, including LLaMA-1-(7B/13B/30B/65B) [31] and LLaMA-2-(7B/13B/70B) [32]. For tasks and datasets, we evaluate AutoPrune’s performance on the language modeling task in the WikiText dataset [21] and 7 zero-shot tasks from EleutherAI LM Harness [8]. Notably, we use the perplexity on the held-out WikiText and accuracy as evaluation metrics. We further include evaluations on the *From Passive to Active Reasoning* benchmark [41] under a zero-shot setting, using GPT-4o [14] as the response model and a 50%-pruned Llama-3-8B-Instruct [22] as the policy model to examine the generality of our reasoning framework beyond pruning scenarios. All experiments are conducted on an NVIDIA H100 GPU. The detailed experimental settings are presented in App. B.

4.1. Results on Language Modeling

Table 4 reports the perplexity of our method (AutoPrune) and several strong baselines on pruned LLaMA-1 and LLaMA-2 models evaluated on the WikiText dataset. In this case, we observe that our method consistently outperforms existing training-free pruning approaches across various sparsity settings and model scales. To be specific, under 50% sparsity, our method achieves lower perplexity than the state-of-the-art training-free method Wanda across nearly all model sizes. For example, on LLaMA-1 13B and 30B, AutoPrune reduces perplexity from Wanda’s 6.15 and 5.24 to 6.02 and 5.18, respectively, which represents absolute reductions of 0.13 and 0.06. On LLaMA-2 13B, AutoPrune achieves a notable 1.30 decrease in perplexity (from 5.56 to 4.26), suggesting that the automatic strategy is especially beneficial for large-scale models.

Under 60% sparsity, where the pruning task becomes more challenging, our method maintains its superiority. Compared to Wanda, AutoPrune reduces perplexity by 0.93 on LLaMA-1 13B (from 8.56 to 7.63), and by 0.75 on LLaMA-2 13B (from 7.75 to 7.02). These results indicate that our method is more robust to higher sparsity levels and better preserves model performance under aggressive compression. In the structured pruning settings (2:4 and 4:8 patterns), AutoPrune continues to outperform all baselines. In the 2:4 case on LLaMA-2 70B, AutoPrune reduces perplexity from Wanda’s 5.16 to 4.97, an absolute improvement of 0.19. Similarly, in the 4:8 setting on LLaMA-1 30B, our method achieves 5.79 perplexity, outperforming Wanda’s 5.97 and SparseGPT’s 6.17.

4.2. Results on Zero-shot Tasks

To scrutinize the generalizability of AutoPrune, we conduct the experiments on different zero-shot downstream tasks with prompting, including BoolQ [4], RTE [33], Hellaswag [42], Winogrande [28], ARC Easy and Challenge [5], and OpenBookQA [23].

LLaMA-1. As shown in Table 5, we can clearly observe that our method achieves state-of-the-art performance across multiple zero-shot tasks under 50% unstructured sparsity. Notably, on the LLaMA-1 7B model, our method outperforms the SOTA training-free baseline “Wanda” by 1.08% on RTE and achieves the best accuracy of 72.29% on BoolQ. Moreover, on the larger 30B and 65B models, our method delivers highly competitive results, reaching 35.20% on OBQA (vs. 34.60% by Wanda) and 72.56% on RTE (vs. 71.48% by Wanda).

LLaMA-2. As shown in Table 6, the same conclusion can be drawn from evaluations on LLaMA-2 models: our method consistently delivers competitive or superior accuracy across various zero-shot tasks under 50% unstructured sparsity. For instance, on the LLaMA-2 7B model, our method achieves the best result on RTE with 58.12%, sur-

Table 4. A performance comparison (lower is better) of pruned LLaMA-1 and LLaMA-2 models on the mainstream WikiText dataset. To make a fair comparison, we use “uniform” as the sample method used in previous methods (e.g., Wanda). Deep cyan, light cyan, and pink indicate the best, second-best, and good results, respectively.

Methods	Weight Update	Sparsity	Human Expert	LLaMA-1				LLaMA-2		
				7B	13B	30B	65B	7B	13B	70B
Dense	-	0%		5.68	5.09	4.77	3.56	5.12	4.57	3.12
Magnitude	X	50%	✓	17.29	20.21	7.54	5.90	14.89	6.37	4.98
SparseGPT	✓	50%	✓	7.22	6.21	5.31	4.57	6.51	5.63	3.98
Wanda	X	50%	✓	7.26	6.15	5.24	4.57	6.42	5.56	3.98
AutoPrune (ours)	X	50%	X	7.12↓(0.14)	6.02↓(0.13)	5.18↓(0.06)	4.71	6.28↓(0.04)	5.43↓(1.30)	4.00
Magnitude	X	60%	✓	6e2	2e2	27.67	9.34	4e3	11.23	8.21
SparseGPT	✓	60%	✓	10.51	8.56	6.66	5.82	9.58	7.80	4.98
Wanda	X	60%	✓	10.66	8.56	6.49	5.83	9.71	7.75	4.98
AutoPrune (ours)	X	60%	X	9.63↓(0.88)	7.63↓(0.93)	6.31↓(0.18)	5.79↓(0.03)	8.93↓(0.65)	7.02↓(0.75)	4.78↓(0.20)
Magnitude	X	2:4	✓	42.13	18.37	9.10	7.11	54.59	8.33	6.33
SparseGPT	✓	2:4	✓	11.00	9.11	7.16	6.28	10.17	8.32	5.40
Wanda	X	2:4	✓	11.53	9.58	6.90	6.25	11.02	8.27	5.16
AutoPrune (ours)	X	2:4	X	10.35↓(0.65)	8.25↓(0.86)	6.50↓(0.40)	6.04↓(0.21)	10.16↓(0.01)	7.48↓(0.79)	4.97↓(0.19)
Magnitude	X	4:8	✓	16.84	13.84	7.62	6.36	16.48	6.76	5.54
SparseGPT	✓	4:8	✓	8.61	7.40	6.17	5.38	8.12	6.60	4.59
Wanda	X	4:8	✓	8.57	7.40	5.97	5.30	7.97	6.55	4.47
AutoPrune (ours)	X	4:8	X	8.14↓(0.43)	6.89↓(0.51)	5.79↓(0.18)	5.29↓(0.01)	7.50↓(0.47)	6.16↓(0.39)	4.45↓(0.02)

Table 5. Accuracies (%) of LLaMA-1 for 7 zero-shot tasks with unstructured 50% sparsity. Because of the limited VRAM capacity of our available hardware, we restrict the 65B model to 8-bit quantization.

Params	Methods	BoolQ	RTE	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA
		Dense	75.05	66.43	56.92	69.93	75.34	34.40
LLaMA-1 7B	Magnitude [17]	54.59	54.51	45.49	59.19	58.84	33.53	22.40
	SparseGPT [7]	72.05	54.15	51.43	67.88	71.38	37.71	30.00
	Wanda [30]	71.22	55.60	51.85	66.06	69.11	36.86	28.80
	AutoPrune (ours)	72.29	56.68	51.92	65.98	69.48	37.62	30.00
LLaMA-1 13B	Dense	77.89	70.4	59.94	72.77	77.40	46.50	33.20
	Magnitude [17]	54.89	51.26	44.16	63.14	58.80	33.79	27.20
	SparseGPT [7]	76.97	61.01	54.95	71.67	72.47	41.98	31.20
	Wanda [30]	75.90	62.82	55.71	71.98	73.19	43.52	32.20
LLaMA-1 30B	Dense	82.69	66.79	63.35	75.69	80.30	52.82	36.00
	Magnitude [17]	64.34	50.18	50.59	66.54	72.39	43.77	29.00
	SparseGPT [7]	82.32	62.45	59.15	75.22	78.96	48.56	35.00
	Wanda [30]	81.90	65.34	60.93	73.48	79.29	49.66	34.60
LLaMA-1 65B	AutoPrune (ours)	82.11	63.54	60.64	74.43	80.06	49.83	35.20
	Dense	84.83	69.68	64.54	77.27	81.40	52.90	38.20
	Magnitude [17]	79.15	62.45	61.90	74.74	76.40	49.57	35.00
	SparseGPT [7]	84.60	70.76	63.90	77.43	79.35	50.85	37.20
	Wanda [30]	84.70	71.48	64.55	76.87	79.75	50.51	38.80
	AutoPrune (ours, 8bit)	84.79	72.56	61.76	75.77	79.89	50.97	38.80

passing SparseGPT (54.15%) and Wanda (53.43%). Similarly, on the 13B model, we reach 56.95% on HellaSwag, outperforming all baselines. On the largest 70B model, our method attains the highest score on BoolQ (83.60%) and matches or closely trails top baselines on other tasks.

4.3. Effectiveness on Outlier Value Issue

As shown in Table 3, we can find that the performance of pruning methods with SDSA surpasses that with Uniform at 60% sparsity. To further evaluate the effectiveness of our method, we compare SDSA with a broad range of baselines,

including Uniform, ER, ER-plus, Global, and OWL [36] on the WikiText dataset using LLaMA-1-7B under broad sparsity levels. As shown in Table 7, SDSA consistently achieves the lowest perplexity across all sparsity settings. Notably, we find that SDSA better mitigates the outlier value issue compared to its peer competitors (i.e., OWL). This is because SDSA is well-motivated and based on detailed skewness analysis for LLMs. Those results validate the effectiveness of our SDSA to outlier value issue, especially showing the superiority under high pruning ratios. Notably, we do not compare with ALS [19] and DSA [18]

Table 6. Accuracies (%) of LLaMA-2 for 7 zero-shot tasks with unstructured 50% sparsity.

Params	Method	BoolQ	RTE	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA
LLaMA-2 7B	Dense	77.74	62.82	57.17	68.90	76.39	43.52	31.40
	Magnitude [17]	63.00	57.04	49.13	63.30	64.10	34.64	26.80
	SparseGPT [7]	75.02	54.15	52.37	69.85	73.27	39.85	29.20
	Wanda [30]	75.99	53.43	52.49	68.19	72.77	39.59	31.20
LLaMA-2 13B	AutoPrune (ours)	75.50	58.12	53.02	67.64	74.75	39.16	31.00
	Dense	80.52	65.34	60.06	72.22	79.42	48.46	35.20
	Magnitude [17]	57.61	55.96	54.40	65.27	70.54	38.40	27.80
	SparseGPT [7]	81.44	65.34	55.83	72.77	74.83	42.24	32.60
LLaMA-2 70B	Wanda [30]	81.84	64.02	56.90	71.35	76.18	43.52	32.00
	AutoPrune (ours)	80.58	61.01	56.95	69.61	76.35	43.86	33.80
	Dense	83.40	67.87	66.10	78.06	82.55	54.44	37.20
	Magnitude [17]	70.55	60.65	61.50	73.48	75.70	49.23	35.40
LLaMA-2 70B	SparseGPT [7]	83.55	70.40	63.80	78.85	82.40	53.75	38.20
	Wanda [30]	82.50	73.65	64.10	78.14	80.80	52.65	37.40
	AutoPrune (ours, 8bit)	83.60	71.48	64.65	78.89	81.20	52.90	37.60

Table 7. A comparison of different sparsity using Wanda. As the official OWL code contains fatal bugs that impeded use, we reimplemented it and have released the code in our repository.

Sparsity/Perplexity	30%	40%	50%	60%	70%	80%
Global	10293	10762	14848	17765	5147	39918.56
ER-plus	6.05	6.62	8.00	14.04	229.17	6013.91
ER	6.02	6.55	7.74	12.16	112.03	11151.18
Uniform	5.99	6.38	7.26	10.70	85.77	3499.88
OWL	6.01	6.42	7.41	12.04	175.06	3244.62
SDSA(ours)	5.98	6.33	7.13	9.88	66.93	892.35

Table 8. Ablation study for calibration samples.

Models/samples	16	32	128	512	1024	2048
LLaMA-1 7b	7.12	7.10	7.12	7.11	7.11	7.11

because their reported results are significantly inconsistent with SparseGPT, Wanda, as well as OWL, and crucial code is unavailable. According to the CVPR reproducibility policy, unverifiable baselines can be excluded.

4.4. Ablation Study

The impact of calibration samples : We conduct an ablation study on LLaMA-1 7b in WikiText to study the impact of calibration samples using our method at 50% sparsity. As shown in Table 8, our method is non-sensitive to calibration samples, showing strong stability to calibration samples.

4.5. Results on Active Reasoning

We further evaluate our reasoning framework on the latest “From Passive to Active Reasoning” benchmark [41] under a zero-shot setting. The response model is GPT-4o [14], while the policy model is a Llama-3-8B-Instruct [22] pruned to **50% sparsity**. As shown in Fig. 7, structured reasoning methods (APD, SparseGPT) consistently outperform the naive baseline, demonstrating that our approach generalizes beyond model compression tasks.

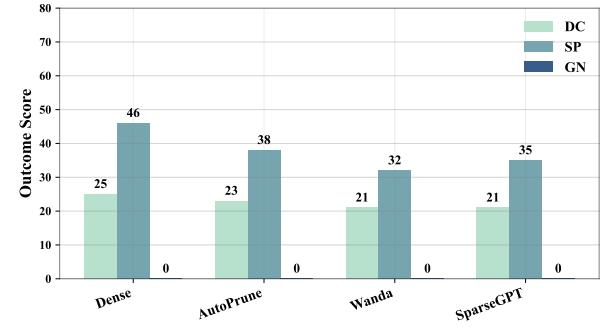


Figure 7. Results on the *From Passive to Active Reasoning* benchmark [41] under zero-shot settings. DC and GN are evaluated by accuracy, while SP is measured by F1 score.

4.6. Additional Evaluation

Due to the page limit of the main text, we provide more results in App. C-G. ① Generalization for more LLMs and zero-shot tasks are shown in App. C. ② More ablation studies are shown in App. D. ③ Detailed Prompt Engineering are shown in App. E. ④ Limitations are shown in App. F. ⑤ Related Work are shown in App. G.

5. Conclusion and Future Work

This paper identifies the outlier value issue that leads to severe performance drops in pruning large language models (LLMs) under high sparsity. Our method, AutoPrune, enables LLMs to automatically design optimal pruning algorithms without expert knowledge. By introducing Graph-driven Chain-of-Thought (GCoT) and Skew-aware Dynamic Sparsity Allocation (SDSA), AutoPrune effectively mitigates this issue and achieves superior pruning performance. We affirm that AutoPrune is practically valuable for efficient LLM pruning and hope it inspires further work on automated, adaptive sparsity design. In the future, we

plan to extend Autoprune to more application scenarios, i.e.,
multimodal large models.

Overview of the Appendix

The overview of this appendix are as follows:

- App. A: More Validations on the Outlier Value Issue.
- App. B: Detailed Experimental Settings.
- App. C: Additional Experimental Results.
- App. D: More Ablation Studies.
- App. E: Detailed Prompt Engineering.
- App. F: Limitations.
- App. G: Related Work.

A. More Validations on the Outlier Value Issue

To further validate our findings regarding the outlier value issue, which is attributed to uniform sparsity, we conducted a comprehensive sensitivity analysis. We used another three established pruning methods, SparseGPT, Magnitude, and our proposed AutoPrune, to evaluate the layer-wise sensitivity. Our results, which are detailed in Fig. 8, 9, and 10, clearly demonstrate a strong correlation between a layer’s sensitivity to pruning and its weight distribution’s skewness. Specifically, layers with higher skewness were found to be more susceptible to performance degradation when pruned, thus reinforcing our hypothesis that the outlier value issue is a critical factor to consider when designing effective LLM pruning strategies.

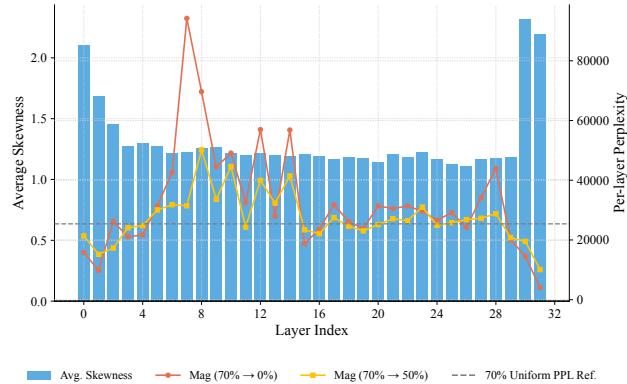


Figure 8. Validation of layer sensitivity to pruning ratios.

B. Detailed Experimental Settings

In this paper, our experimental settings are essentially identical to Wanda’s [30], with the sole exception of the weight update method. We conducted all experiments on LLaMA-1 and LLaMA-2 models. For the pruning process, we used a seqlen of 2048 and nsamples of 128. Notably, while some prior methods like Wanda also implement a sequential weight update strategy, they often find the iterative approach to yield better results. In contrast, our method directly employs the more efficient sequential weight update and still achieves excellent results.

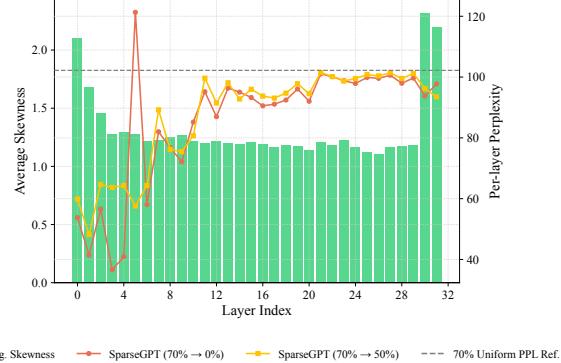


Figure 9. Validation of layer sensitivity to pruning ratios.

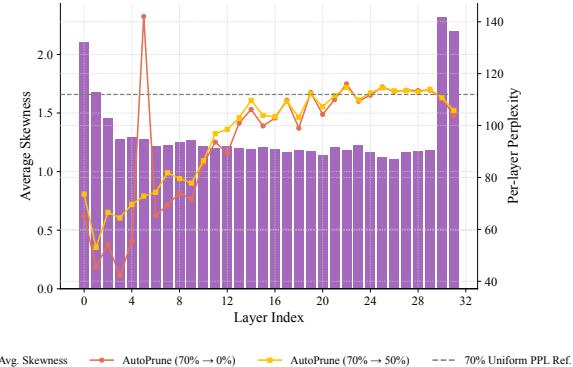


Figure 10. Validation of layer sensitivity to pruning ratios.

For the Skew-Aware Dynamic Soarsity Dynamic Soarsity scheduler, the hyperparameter M is set to 1.8 for layer-wise allocation and 1.5 for block-wise allocation. We found that, contrary to the findings in OWL [37], the layer-wise dynamic sparsity allocation in our experiments consistently yields superior results compared to the block-wise approach.

We follow the experimental protocol of From Passive to Active Reasoning [41] and evaluate three task types: DC, GN, and SP. For DC and SP, we use a zero-shot active-reasoning setup based on a policy-response architecture, where llama-3-8B-Instruct [22] serves as the policy model to generate candidate reasoning turns and GPT-4o [14] produces final responses. Both models use a temperature and top-p of 0.7, with a maximum of 25 turns.

C. Additional Experimental Results

C.1. Zero-Shot Tasks

In this section, we further demonstrate the effectiveness of our approach on task-wise evaluation of seven zero-shot tasks: BoolQ [4], RTE [33], HellaSwag [42], WinoGrande [28], ARC Easy and Challenge [5], and OpenBookQA [23].

We present the results of LLaMA V1 and LLaMA V2 family for 2:4 and 4:8 structured pruning in Table. 10, 12, 9, and 11.

C.2. Generalization for More LLMs

To further evaluate the generalizability of our method on large language models (LLMs), we conduct experiments on two representative architectures: **OPT-13B** and **Pythia-12B**. As shown in Table 13, AutoPrune consistently delivers strong performance across a wide range of sparsity levels, despite being applied to unseen model structures. Specifically, on Pythia-12B, AutoPrune matches or exceeds the performance of hand-crafted baselines such as Wanda and SparseGPT across all sparsity levels, achieving a perplexity of 8.59 at 10% sparsity and maintaining low degradation even at 50% sparsity (11.13 vs. 11.27 for Wanda). On OPT-13B, although SparseGPT performs slightly better at extreme sparsity levels (e.g., 11.19 at 50%), AutoPrune demonstrates stable performance and remains competitive across the board, showing robustness to model scale and architecture.

These results highlight the strong generalizability of AutoPrune to diverse LLMs, confirming its effectiveness as a plug-and-play pruning algorithm that scales well across different transformer backbones without requiring weight updates or architecture-specific tuning.

C.3. Generalizability on MMLU

To further evaluate the reasoning capabilities of LLMs, we conduct experiments on the Massive Multitask Language Understanding (MMLU) benchmark [12] in a zero-shot setting, using LLaMA V2 and V3 models with both 50% unstructured sparsity and 2:4 structured sparsity. The results are presented in Table 14. As shown, our method achieves competitive or superior performance compared to existing pruning approaches across both model generations and sparsity types. On LLaMA V2 7B with 50% sparsity, our method attains 29.69 accuracy, slightly outperforming Wanda and significantly outperforming LLM-Pruner (+5.54). While SparseGPT leads on this setting with 34.62, our approach maintains higher stability across other configurations. In the more challenging 2:4 sparsity regime, our method surpasses Wanda by +1.61 and is comparable to SparseGPT (25.24 vs. 25.76), showing strong compatibility with structured pruning.

On LLaMA V3 8B, our method again delivers solid performance. With 50% sparsity, it achieves 37.35 accuracy, outperforming LLM-Pruner by a large margin (+7.45), though slightly behind SparseGPT and Wanda. Importantly, in the 2:4 sparsity case, our method matches Wanda (27.61 vs. 27.57) and closely trails SparseGPT (28.27), despite requiring no weight updates or task-specific tuning.

These results demonstrate that our pruning method is not

only effective in preserving reasoning ability in zero-shot settings, but also generalizes robustly across sparsity formats and model generations.

D. More Ablation Studies

D.1. Number of Calibration Samples

In line with the experimental setup of Wanda [30], we use a calibration set of 128 samples. To further investigate the stability and robustness of our proposed method, we conduct a detailed analysis of its effect on the LLaMA and LLaMA-2 model families using this setting. We show the results for pruning LLaMA-7B and LLaMA-2-7B with unstructured 50% sparsity in Table 15. We find that there is a slight improvement in performance of pruned LLMs when the size of the calibration set goes beyond 128, but we maintain 128 for a fair and efficient comparison.

D.2. Sensitivity Analysis on M

To evaluate the robustness of our method, we conduct a sensitivity analysis on the hyperparameter M . For this study, we apply 70% unstructured pruning on both the LLaMA-7B and LLaMA-2-7B models while varying the value of M . The results, summarized in Table. 16, demonstrates that our approach is not overly sensitive to the choice of M , as performance remains stable across the tested configurations.

D.3. Number of Branches

To investigate the sensitivity of our AutoPrune method to its branches, we analyze the impact of the number of branches explored during the Graph-Driven Chain-of-Thought search process. We conduct experiments on the WikiText-2 dataset, where we apply 50% unstructured pruning to the LLaMA-1 7B model while varying the number of candidate branches. This study allows us to understand the trade-off between the breadth of the search for a pruning algorithm and the final performance of the pruned model. The perplexity results for different numbers of branches are presented in Table 17.

E. Detailed Prompt Engineering

As detailed in the main text, our Graph-driven Chain-of-Thought framework utilizes four distinct prompts to guide the LLM through the progressive formalization of a new pruning algorithm. These prompts correspond to the four stages of the pipeline: Analysis, Hypothesis, Conceptual Formula, and Computational Concept. The specific prompts for each stage are as follows:

Params	Method	BoolQ	RTE	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA
7B	Dense	75.05	66.43	56.92	69.93	75.34	41.89	34.40
	Magnitude	51.19	50.54	46.73	60.69	58.96	30.89	23.20
	SparseGPT	73.06	58.12	47.88	65.98	66.75	32.42	25.40
	Wanda	70.97	58.24	46.81	65.83	65.53	33.97	28.00
13B	AutoPrune	68.59	58.84	48.37	64.33	64.90	33.19	27.00
	Dense	77.89	70.40	59.94	72.77	77.40	46.50	33.20
	Magnitude	61.07	51.26	48.91	65.11	63.26	35.67	28.40
	SparseGPT	76.61	57.76	51.24	70.17	71.17	37.20	27.80
30B	Wanda	74.89	57.89	51.26	70.56	70.29	37.97	29.80
	AutoPrune	73.30	54.51	52.50	69.37	71.25	38.90	29.20
	Dense	82.69	66.79	63.35	75.69	80.30	52.82	36.00
	Magnitude	63.55	50.18	49.45	65.75	73.36	42.83	29.60
65B	SparseGPT	78.69	61.73	56.15	74.35	76.94	46.08	31.60
	Wanda	77.38	58.80	58.79	74.28	77.34	46.46	34.00
	AutoPrune	78.69	63.18	57.30	72.85	75.72	46.33	32.40
	Dense	84.83	69.68	64.54	77.27	81.40	52.90	38.20
65B	Magnitude	74.95	68.23	60.85	74.27	76.45	47.61	32.80
	SparseGPT	84.35	68.95	61.00	77.19	78.75	48.46	35.40
	Wanda	84.29	70.92	59.54	76.64	79.00	48.83	35.60
	AutoPrune (8bit)	83.2	66.78	51.50	75.06	78.70	45.90	35.20

Table 9. Accuracies (%) of LLaMA-1 for 7 zero-shot tasks with 4:8 sparsity.

Params	Method	BoolQ	RTE	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA
7B	Dense	75.05	66.43	56.92	69.93	75.34	41.89	34.40
	Magnitude	53.09	55.60	42.30	59.91	53.28	27.13	21.80
	SparseGPT	70.46	60.65	42.99	64.88	61.49	30.12	23.60
	AutoPrune	69.11	51.26	43.67	61.41	60.52	30.29	24.20
13B	Dense	77.89	70.40	59.94	72.77	77.40	46.50	33.20
	Magnitude	60.95	49.10	45.81	62.75	58.75	31.06	27.60
	SparseGPT	72.14	55.23	48.11	68.98	66.71	34.98	26.40
	Wanda	70.21	53.43	46.74	68.82	65.82	33.87	27.20
30B	AutoPrune	71.59	53.43	48.68	67.96	66.29	33.36	27.60
	Dense	82.69	66.79	63.35	75.69	80.30	52.82	36.00
	Magnitude	65.11	52.35	51.72	66.22	70.88	38.23	27.60
	SparseGPT	75.60	62.13	53.10	72.61	75.13	41.98	31.80
65B	Wanda	74.68	63.80	54.41	72.93	74.41	42.06	32.20
	AutoPrune	75.44	63.17	54.43	71.74	73.57	41.64	31.40
	Dense	84.83	69.68	64.54	77.27	81.40	52.90	38.20
	Magnitude	77.9	64.98	58.65	72.85	75.15	45.05	34.40
65B	SparseGPT	83.15	65.34	57.20	76.72	78.20	45.18	32.20
	Wanda	83.58	66.79	56.36	75.82	78.23	45.56	33.60
	AutoPrune	83.80	58.49	50.15	74.82	75.70	44.28	32.60

Table 10. Accuracies (%) of LLaMA-1 for 7 zero-shot tasks with 2:4 sparsity.

Params	Method	BoolQ	RTE	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA
7B	Dense	77.74	62.82	57.17	68.90	76.39	43.52	31.40
	Magnitude	63.00	52.35	50.08	62.43	64.73	35.92	26.00
	SparseGPT	72.69	55.23	48.20	68.11	69.15	35.84	27.40
	Wanda	73.91	53.79	46.45	66.61	66.71	34.13	25.80
13B	AutoPrune	72.94	53.43	48.61	65.91	67.22	34.39	26.60
	Dense	80.52	65.34	60.06	72.22	79.42	48.46	35.20
	Magnitude	63.33	57.76	53.96	64.40	68.48	35.75	26.00
	SparseGPT	79.97	66.79	52.01	70.64	73.61	41.04	30.00
70B	Wanda	80.26	65.62	52.05	69.48	73.88	41.54	28.40
	AutoPrune	80.64	57.76	53.29	68.51	73.95	39.00	30.40
	Dense	83.40	67.87	66.10	78.06	82.55	54.44	37.20
	Magnitude	70.95	59.21	60.05	74.11	76.25	46.76	34.60
13B	SparseGPT	82.20	72.20	61.45	77.82	80.85	51.19	35.20
	Wanda	84.30	71.80	61.90	76.24	80.40	51.80	36.00
	AutoPrune	83.50	69.31	62.45	76.98	80.85	50.17	34.80

Table 11. Accuracies (%) of LLaMA-2 for 7 zero-shot tasks with 4:8 sparsity.

Params	Method	BoolQ	RTE	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA
7B	Dense	77.74	62.82	57.17	68.90	76.39	43.52	31.40
	Magnitude	56.23	51.35	42.27	60.93	59.18	27.31	21.80
	SparseGPT	70.52	58.84	43.26	66.69	64.10	29.97	23.20
	Wanda	67.65	53.07	40.92	62.43	61.78	31.20	24.20
13B	AutoPrune	70.25	54.15	43.20	63.06	64.56	30.63	25.60
	Dense	80.52	65.34	60.06	72.22	79.42	48.46	35.20
	Magnitude	65.69	54.15	50.13	62.04	62.46	31.74	23.00
	SparseGPT	76.79	59.38	46.58	68.67	70.62	36.60	25.40
70B	Wanda	76.80	61.22	47.82	66.90	69.24	36.82	26.40
	AutoPrune	77.77	56.68	47.76	66.93	69.66	35.07	26.20
	Dense	83.40	67.87	66.10	78.06	82.55	54.44	37.20
	Magnitude	73.20	57.04	58.40	74.27	76.15	45.22	35.40
13B	SparseGPT	79.50	70.76	59.00	76.64	78.95	48.55	33.80
	Wanda	82.20	69.85	59.34	76.23	79.30	47.26	34.80
	AutoPrune	83.05	68.59	59.10	75.92	79.17	47.61	33.00

Table 12. Accuracies (%) of LLaMA-2 for 7 zero-shot tasks with 2:4 sparsity.

Stage 1: Analysis

You are an expert analyst in neural network architecture and pruning. Your task is to provide a concise, holistic analysis of the following model, focusing on the most critical properties a novel pruning metric must capture to be effective.

TARGET MODEL FOR ANALYSIS:

{llm_description}

Based on the provided model's architecture and your expert knowledge, write a paragraph analyzing its core sensitivities and the most promising principles for effective pruning. Your analysis should identify the key trade-offs of simpler metrics and conclude by pointing towards the essential, multi-faceted characteristics that a superior metric needs

Model	Dense	Pruning Method	Weight Update	Sparsity				
				10%	20%	30%	40%	50%
OPT-13B	10.13	Magnitude	✗	14.45	9e3	1e4	1e4	1e4
		SparseGPT	✓	10.11	10.10	10.12	10.35	11.19
		Wanda	✗	10.09	10.07	10.09	10.63	11.42
		AutoPrune	✗	10.16	10.45	11.65	17.56	20.33
Pythia-12B	8.59	Magnitude	✗	127.76	2e5	7e5	2e5	3e5
		SparseGPT	✓	8.59	8.65	8.86	9.39	11.02
		Wanda	✗	8.59	8.60	8.85	9.31	11.27
		AutoPrune	✗	8.59	8.60	8.80	9.27	11.13

Table 13. Pruning Pythia-13B and OPT-13B with various sparsity levels.

Model & Param	Ours	Wanda	SparseGPT	LLM-Pruner
LLaMA V2 7B 50%	29.69	29.67	34.62	24.15
LLaMA V2 7B 2:4	25.24	23.63	25.76	–
LLaMA V3 8B 50%	37.35	40.59	48.33	29.90
LLaMA V3 8B 2:4	27.61	27.57	28.27	–

Table 14. Zero-shot accuracy comparison on the MMLU benchmark.

to measure. Provide only the analysis paragraph. Do not add any titles, introductory sentences, or concluding remarks.

Stage 2: Hypothesis

You are a principal scientist responsible for formulating novel research directions. Your task is to distill a detailed analysis into a single, powerful, and testable hypothesis.

PRECEDING ANALYSIS:

{analysis_text}

YOUR HYPOTHESIS TASK:

Based only on the preceding analysis, synthesize its key insights into a single sentence that formulates a novel and unifying hypothesis for a superior pruning metric.

The hypothesis should propose a clear relationship or dependency. For example: "A component's importance is a function of property X combined with property Y."

Provide only the single hypothesis sentence. Do not add any titles, introductory phrases, or explanations.

Stage 3: Conceptual Formula

You are a senior research engineer specializing in translating theoretical concepts into algorithmic structures. Your task is to convert a natural language hypothesis into a semi-formal conceptual formula.

PRECEDING HYPOTHESIS:

{hypothesis_text}

YOUR FORMALIZATION TASK:

Based on the provided hypothesis, create a conceptual formula that represents the core computational logic. This formula should be abstract, using mathematical notation and placeholders for functions and variables, to outline the structure without getting into implementation details.

For example, if the hypothesis is "Importance is the product of weight magnitude and activation norm," a valid conceptual formula would be: $Importance(W, X) = |W| * ||f(X)||$.

Provide only the conceptual formula. Do not add any titles, introductory sentences, or explanations.

Model	Method	1	16	32	64	128	256	512	1024	2048
LLaMA-7B	SparseGPT	10.22	7.61	7.36	7.29	7.26	7.20	7.19	7.23	7.20
	Wanda	7.46	7.27	7.28	7.28	7.26	7.30	7.26	7.25	7.26
	AutoPrune	7.15	7.12	7.11	7.12	7.11	7.11	7.11	7.11	7.11
LLaMA-2-7B	SparseGPT	8.63	6.67	6.62	6.61	6.53	6.52	6.50	6.49	6.49
	Wanda	6.53	6.45	6.46	6.45	6.45	6.45	6.45	6.45	6.45
	AutoPrune	6.33	6.30	6.29	6.29	6.28	6.29	6.29	6.28	6.28

Table 15. WikiText validation perplexity of pruned LLaMA-1 and LLaMA-2 under various number of calibration samples, with 50% sparsity.

Model	1.2	1.4	1.6	1.8	2.0	2.2	2.5	uniform
LLaMA-7B	99.30	90.45	81.06	79.94	79.01	82.26	87.24	111.98
LLaMA-V2-7B	99.09	97.26	95.42	95.19	95.46	93.60	100.03	127.92

Table 16. Sensitivity analysis of M , on the WikiText-2.

Model	1	3	5	10	uniform
LLaMA-7B	7.49	7.12	7.12	7.09	5.68
LLaMA-V2-7B	6.93	6.28	6.25	6.29	5.12

Table 17. Pruning LLaMA with various number of branches.

Stage 4: Computational Concept

You are a software architect designing a library of computational modules. Your task is to decompose a conceptual formula into a set of precise, machine-readable functional components.

CONCEPTUAL FORMULA:

{conceptual_formula}

YOUR DECOMPOSITION TASK:

Break down the conceptual formula into a series of modular, computable concepts. For each concept, define its precise functional signature, including a description, a list of inputs, and the expected output. This decomposition should create an unambiguous blueprint ready for direct implementation.

Provide only the list of computable concepts. Do not add any titles, introductory phrases, or concluding remarks.

F. Limitations

While AutoPrune significantly surpasses previous methods (i.e., SparseGPT, Wanda), some limitations remain. We organize them here for systematic discussion.

Black-box Optimization Although our method leverages an LLM to generate candidate pruning algorithms—rendering its internal token-level reasoning unobservable—we mitigate this black-box concern through a novel Graph-driven Chain-of-Thought (GCoT) framework.

GCoT structures the prompt optimization process into a progressive reasoning pipeline, guiding the LLM from vague goals to precise, executable algorithm blueprints. Specifically, the pipeline comprises four explicit stages: (1) Analysis, where the LLM grounds its reasoning in model-specific statistics; (2) Hypothesis, where it proposes a causal relationship between observed signals and pruning sensitivity; (3) Conceptual Formula, which formalizes the logic in a structured form; and (4) Computable Concept, where this logic is modularized into machine-readable functions.

Crucially, all decisions that influence the search trajectory pass through this measurable, task-specific reasoning process, rather than being driven by opaque logits. By decomposing the algorithm search space into interpretable, modular components, GCoT exposes the full reasoning path behind each candidate algorithm. While the LLM itself remains a partially opaque system, the GCoT framework ensures that the optimization process is transparent, traceable, and grounded in explicit cognitive steps. This significantly enhances both the interpretability and trustworthiness of the learned pruning strategies.

Security and Stability of Executing LLM-Generated Code As LLM-generated proxies may contain unintended behaviors, including unsafe operations or excessive resource usage, executing them directly poses inherent risks. To mitigate these concerns, all code is executed within a controlled sandbox environment that enforces strict isolation, resource limits, and safety policies, ensuring stable and secure execution without compromising the host system.

G. Related Work

Layerwise Sparsity for Pruning. Early approaches used uniform pruning [13, 16], where all layers are pruned at the same sparsity. However, [6] highlighted that different layers have varying importance, making this strategy suboptimal.

To address this issue, some methods [24–26, 39] automatically determine layerwise sparsity by selecting critical parameters across the entire network. Other studies [11, 38] treat pruning as a search problem to identify layerwise sparsity. In contrast to these techniques for CNN models in vision tasks, our method focuses specifically on LLMs.

LLM Pruning. In contrast to traditional techniques, LLM pruning emphasizes data and time efficiency, meaning that pruned models do not require extensive retraining. LLM-Pruner [20] offers a structured pruning method based on model dependency relations and uses LoRA to recover the pruned model’s performance. SparseGPT [7] introduces an effective Hessian matrix estimation technique in large-scale models. In addition, Wanda [30] adopts a direct strategy based on the product of weights and activation values to eliminate weights. These methods apply a uniform pruning rate across all layers. Recently, several approaches have been proposed to achieve non-uniform layerwise sparsity. BESA [35] employs a differentiable approach to search for optimal layerwise sparsity. DSA [18] utilizes a distribution function to allocate sparsity to layers, with the function being searched through evolutionary algorithms. ALS [19] develops an adaptive sparsity allocation strategy based on evaluating the relevance matrix using linear optimization. OWL [37] linearly maps the non-outlier ratio of each layer to its sparsity. These methods rely on search processes or simple linear functions to derive sparsity or allocation strategies. However, for the high-dimensional search space of layerwise sparsity, they cannot guarantee achieving optimal solutions. To address this, we conduct dense empirical research to summarize LLM-specific pruning principles and propose a sparsity allocation strategy that satisfies these principles.

References

- [1] ChatGPT. <https://openai.com/blog/chatgpt>, 2022. 1
- [2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1
- [3] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023. 1
- [4] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019. 6, 10
- [5] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. 6, 10
- [6] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018. 15
- [7] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR, 2023. 1, 7, 8, 16
- [8] Leo Gao, Jonathan Tow, Stella Biderman, Shawn Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jasmine Hsu, Kyle McDonell, Niklas Muennighoff, et al. A framework for few-shot language model evaluation. *Version v0.1. Sept*, 10:8–9, 2021. 6
- [9] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. 1, 4
- [10] Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, pages 293–299, 1993. 1
- [11] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European conference on computer vision (ECCV)*, pages 784–800, 2018. 16
- [12] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *Proceedings of the International Conference on Learning Representations*, 2021. *arXiv:2009.03300*. 2, 11
- [13] H Hu. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016. 15
- [14] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024. 1, 6, 8, 10
- [15] Yao Lai, Sungyoung Lee, Guojin Chen, Souradip Poddar, Mengkang Hu, David Z Pan, and Ping Luo. Analogcoder: Analog circuit design via training-free code generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 379–387, 2025. 1
- [16] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989. 15
- [17] Jaeho Lee, Sejun Park, Sangwoo Mo, Sungsoo Ahn, and Jinwoo Shin. Layer-adaptive sparsity for the magnitude-based pruning. In *International Conference on Learning Representations*, 2021. 1, 7, 8
- [18] Lujun Li, Peijie Dong, Zhenheng Tang, Xiang Liu, Qiang Wang, Wenhan Luo, Wei Xue, Qifeng Liu, Xiaowen Chu, and Yike Guo. Discovering sparsity allocation for layer-wise pruning of large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 7, 16

- [19] Wei Li, Lujun Li, Mark G Lee, and Shengjie Sun. Adaptive layer sparsity for large language models via activation correlation assessment. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 7, 16
- [20] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023. 16
- [21] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016. 2, 6
- [22] Meta AI. Llama 3 8b instruct. https://github.com/meta-llama/Llama3/blob/main/MODEL_CARD.md, 2024. Instruction-tuned model with 8 billion parameters. 6, 8, 10
- [23] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018. 6, 10
- [24] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016. 16
- [25] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Froio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11264–11272, 2019.
- [26] Manuel Nonnenmacher, Thomas Pfeil, Ingo Steinwart, and David Reeb. Sosp: Efficiently capturing global correlations by second-order structured pruning. *arXiv preprint arXiv:2110.11395*, 2021. 16
- [27] OpenAI. Gpt-o3 system card. *OpenAI System Card*, 2025. Technical report, April 16, 2025. 4
- [28] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021. 6, 10
- [29] Per O Seglen. The skewness of science. *Journal of the American society for information science*, 43(9):628–638, 1992. 2
- [30] Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*, 2024. 1, 7, 8, 10, 11, 16
- [31] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 1, 6
- [32] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 6
- [33] Alex Wang. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018. 6, 10
- [34] Yanyue Xie, Zhi Zhang, Ding Zhou, Cong Xie, Ziang Song, Xin Liu, Yanzhi Wang, Xue Lin, and An Xu. Moe-pruner: Pruning mixture-of-experts large language model using the hints from its router. *arXiv preprint arXiv:2410.12013*, 2024. 1
- [35] Peng Xu, Wenqi Shao, Mengzhao Chen, Shitao Tang, Kaipeng Zhang, Peng Gao, Fengwei An, Yu Qiao, and Ping Luo. Besa: Pruning large language models with block-wise parameter-efficient sparsity allocation. *arXiv preprint arXiv:2402.16880*, 2024. 16
- [36] Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Jaiswal, Mykola Pechenizkiy, Yi Liang, et al. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity. In *International Conference on Machine Learning*, 2024. 7
- [37] Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Jaiswal, Mykola Pechenizkiy, Yi Liang, et al. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity. In *International Conference on Machine Learning*, 2024. 10, 16
- [38] Sixing Yu, Arya Mazaheri, and Ali Jannesari. Auto graph encoder-decoder for neural network pruning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6362–6372, 2021. 16
- [39] Yuxin Zhang, Mingbao Lin, Chia-Wen Lin, Jie Chen, Yongjian Wu, Yonghong Tian, and Rongrong Ji. Carrying out cnn channel pruning in a white box. *IEEE Transactions on Neural Networks and Learning Systems*, 34(10):7946–7955, 2022. 16
- [40] Yue Zheng, Yuhao Chen, Bin Qian, Xiufang Shi, Yuanchao Shu, and Jiming Chen. A review on edge large language models: Design, execution, and applications. *ACM Computing Surveys*, 57(8):1–35, 2025. 1
- [41] Zhanke Zhou, Xiao Feng, Zhaocheng Zhu, Jiangchao Yao, Sanmi Koyejo, and Bo Han. From passive to active reasoning: Can large language models ask the right questions under incomplete information? In *Proceedings of the 42nd International Conference on Machine Learning (ICML 2025)*, page –. ML Research Press, 2025. Poster. 6, 8, 10
- [42] Max Zimmer, Megi Andoni, Christoph Spiegel, and Sebastian Pokutta. Perp: Rethinking the prune-retrain paradigm in the era of llms. *arXiv preprint arXiv:2312.15230*, 2023. 6, 10