MI-PRUN : Pruning the Blocks in Large Language Models with Minimal Performance Impact

Anonymous ACL submission

Abstract

Large language models (LLMs) have become crucial across various domains, yet it comes at the expense of considerable computational and memory resources. Model pruning refines deep learning models by excising redundant elements. However, current pruning methods often fail to substantially achieve end-to-end acceleration. In this paper, we propose MI-PRUN, a novel approach that uses mutual information to identify low impact blocks for efficient model pruning. Furthermore, we incorporate the Data Processing Inequality (DPI) to elucidate the relationship between the importance of contiguous blocks and that of individual blocks. We utilize an iterative block selection algorithm to continuously update the combination of blocks that have the minimal impact on model performance, thereby obtaining a globally optimal solution. To enhance the efficiency of pruning, we develop the Fast-Block-Select algorithm to accelerate the pruning process. Comprehensive experiments on a wide range of models and datasets have demonstrated the rationality and effectiveness of our method.

1 Introduction

Large language models (LLMs) have made significant strides, showing notable language skills in both comprehension and creation (Brown et al., 2020; Touvron et al., 2023a; Chiang et al., 2023; Zhu et al., 2023; Li et al., 2023a; Zhang et al., 2023; Huang et al., 2023; Wang et al., 2023). However, as the scale of models expands, the challenges faced in practical deployment also increase. The large size and computational requirements of the models lead to high deployment costs and inference delays. The exponential growth of parameters in large language models has become a striking phenomenon. For instance, the size of the LLaMA series models has soared from several billion parameters to hundreds of billions, and it may even go higher. These enormous leaps in numbers deeply reveal that as the models grow in size, the complexity of their deployment and application in practical scenarios also increases.

In pursuit of lightweight deployment for large language models, the research community has developed an array of model compression strategies, such as model pruning (Ma et al., 2023; Ashkboos et al., 2024; Li et al., 2023c; Han et al., 2015a), quantization (Zhou et al., 2023; Cai et al., 2023; Zhou et al., 2024) and knowledge distillation (Yang et al., 2021; Zhang et al., 2024). These approaches aim to lighten the computational load of large language models, enhancing their deployability across platforms with constrained resources.

Model pruning is essential for enhancing the efficiency of deep learning models by eliminating redundant weights or neurons without compromising performance. Depending on the granularity of the pruning operation, pruning techniques can be categorized into structured and unstructured pruning. Structured pruning removes entire neurons, attention heads or layers, which is more hardware friendly and preserves the efficiency of matrix multiplication (Ashkboos et al., 2024; Yang and Zhang, 2022). Unstructured pruning removes individual weights based on criteria such as magnitude, achieving high compression rates but complicating hardware acceleration due to irregular sparsity patterns (Liao et al., 2023; Anonymous, 2024).

While the benefits of pruning are clear, applying it to large language models still remains a formidable challenge (Ma et al., 2023; Ashkboos et al., 2024; Li et al., 2023c; Han et al., 2015b; Fang et al., 2023). Current pruning techniques often fail to deliver significant acceleration. To address this issue, some studies have proposed pruning entire blocks rather than individual components (Men et al., 2024; Yang et al., 2024b; Chen et al., 2024; Song et al., 2024; Kim et al., 2024). For instance, methods that utilize cosine similarity to evaluate block importance and adopt greedy pruning strategies have emerged (Men et al., 2024). However, these greedy approaches tend to converge to locally optimal solutions, rather than identifying the globally optimal combination of blocks for pruning. Other techniques, such as LaCo (Yang et al., 2024b), attempt to reduce model size by merging subsequent layers into preceding ones. Unfortunately, this method often underperforms compared to direct layer removal in terms of effectiveness. LLM-Streamline achieves pruning by consolidating several blocks with the highest cosine similarity into a single block (Chen et al., 2024). However, distilling too many blocks can sometimes degrade the model's accuracy and necessitates fine-tuning training strategies, thereby introducing additional training overhead. Moreover, it remains a greedy selection method. Meanwhile, methods like SLEB (Song et al., 2024) and Shortened LLaMA (Kim et al., 2024) iteratively prune layers based on importance metrics. Nevertheless, the need to measure corresponding indicators using a calibration set after each layer removal often results in high computational complexity.

To design a method that can achieve the globally optimal combination of pruned blocks, we propose a Mutual Information Based Pruning (MI-PRUN) method designed for large language models. MI-PRUN uses mutual information to identify and remove non essential weight blocks by evaluating the transitions of hidden states. Furthermore, we incorporate the Data Processing Inequality (DPI) (Beaudry and Renner, 2011; Merhav, 2012; Braverman et al., 2016) to elucidate the relationship between the importance of contiguous blocks and that of individual blocks. We utilize an iterative block selection algorithm to continuously update the combination of blocks that have the minimal impact on model performance, thereby obtaining a globally optimal solution. To enhance efficiency, we develop the Fast-Block-Select algorithm, which utilizes heuristic methods to efficiently pinpoint optimal candidates for pruning and streamline the pruning process.

The main contributions of our paper can be summarized as follows:

- We leverage mutual information to quantify the transition of hidden states between different blocks, thereby identifying and pruning weight blocks that contribute less to the model's performance.
- We incorporate the Data Processing Inequality (DPI) (Beaudry and Renner, 2011; Merhav, 2012; Braverman et al., 2016) to elucidate the relationship between the importance of contiguous and individual blocks, and utilize an iterative block selection algorithm to continuously update the combination of blocks with minimal impact on model performance, thereby achieving a globally optimal solution.
- We develop the Fast-Block-Select algorithm to significantly enhance the efficiency of the pruning process.

2 Related Work

Mutual information is a powerful metric that quantifies the dependency between two variables by measuring the amount of information about one variable that can be inferred from the other (Liu and Motani, 2022; Nguyen et al., 2014; Veyrat-Charvillon and Standaert, 2009). Its ability to detect non-linear relationships sets it apart from conventional linear measurement techniques, making it an essential tool for uncovering intricate data patterns and understanding complex model dynamics (Vinh et al., 2012; Pascoal et al., 2017). This allows it to delve into the complex interplays within data, surpassing traditional methods in capturing nuanced relationships (Pluim et al., 2003; Steuer et al., 2002). The application of mutual information is remarkably broad, extending to the assessment of complex connections among multiple variables. In the field of feature selection, mutual information plays a particularly crucial role (Battiti, 1994; Liu et al., 2009; Vergara and Estévez, 2014).

Some studies determine whether to prune neurons or activations between layers by calculating the mutual information between neurons (Fan et al., 2021; Huang et al., 2024; Westphal et al., 2024; Ganesh et al., 2021). If the mutual information value exceeds a certain threshold, it indicates that their information overlaps, and some of them can be pruned without causing significant performance loss. However, these methods still have several drawbacks. Pruning based on measuring the correlation between neurons does not always lead to substantial acceleration, as the inference speed is far lower than that of block pruning. Moreover, simply measuring the correlation between two units (neurons or blocks) is often insufficient. It is necessary to consider the relationships among multiple units using the Data Processing Inequality. Additionally, calculating the mutual information for each pair of neurons individually is computationally intensive. Therefore, it is essential to design more efficient methods to reduce the computational load. Our method fully overcomes the aforementioned challenges.

3 Methodology

3.1 Mutual Information Measures Block Importance

During the inference phase of LLMs, the sequence outputs of the Transformer layers exhibit a high degree of similarity. This similarity primarily stems from a crucial design feature of the model: the use of residual connections. Specifically, the output of each layer is added to the output of the previous layer through these residual connections, thereby enabling the continuous transfer and accumulation of information across different layers of the model. Mathematically, the output of the $(i+1)$-th Transformer layer can be represented as follows:

$$h_{i+1} = Transformer_{i+1}(h_i) + h_i \qquad (1)$$

Here, h_i and h_{i+1} denote the outputs of the i-th and $(i+1)$-th layers, respectively, while $Transformer_{i+1}$ represents the transformation function of the $(i+1)$-th layer. This additive mechanism ensures that the information from previous layers is retained and built upon, contributing to the overall robustness and depth of the model's representations.

Some studies identify less important blocks using cosine similarity (Men et al., 2024). However, the effectiveness of cosine similarity often decreases when dealing with non-linear relationships or complex, skewed data distributions, and it may not delve deeply into the specific interactions between variables. In contrast, mutual information (Liu and Motani, 2022; Nguyen et al., 2014; Veyrat-Charvillon and Standaert, 2009) presents a more robust and in-depth analytical approach. Unlike cosine similarity, which primarily focuses on the directionality of hidden states, mutual information delves into the actual information flow and dependency relationships between hidden states. This holistic perspective allows for a richer understanding of the underlying interactions within the model, thereby providing a more effective basis for optimization and deeper insights into model performance. This is also demonstrated in Section 4 of our work.

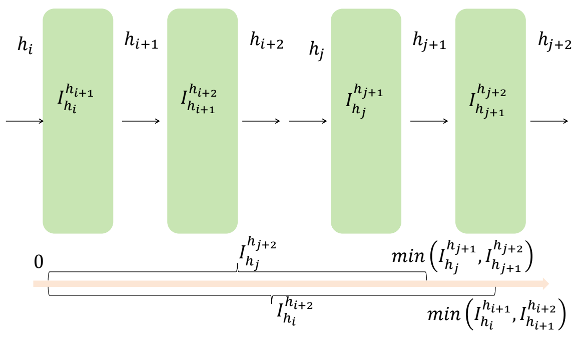

Figure 1: The relationship between the mutual information of the global continuous block and the local individual blocks.

When the mutual information (Liu and Motani, 2022; Nguyen et al., 2014; Veyrat-Charvillon and Standaert, 2009) between the input and output states is unusually high, this typically indicates that the block's output is largely a direct reflection of its input state. This suggests that the block holds lower importance within the model. In other words, the output state does not significantly add new or important information but rather largely replicates the information from the input state, thereby introducing redundancy. In this case, the block may play a minor role in the overall functionality of the model.

The transformation effect of the $block_{i+1}$ is maximized when h_i and h_{i+1} are independent, and the mutual information is zero. Conversely, the transformation function is minimal when h_{i+1} is completely determined by h_i, such that for any h_i there exists a corresponding h_{i+1} for which $P(h_{i+1} \mid h_i) = 1$, and the mutual information is maximal. Beyond these two extreme cases, as the transformation function exerts a greater effect, $H(h_{i+1} \mid h_i)$ increases, while the mutual information $I(h_i, h_{i+1})$ decreases.

3.2 Importance of Continuous Blocks

In the previous section, we analyze the importance of blocks from the perspective of individual, isolated blocks. A potential issue arises: some individual blocks may appear unimportant on their own, but when considered in the context of the surrounding blocks as a whole, they can significantly contribute to the model performance. To gain a clearer understanding of this issue, we introduce the Data Processing Inequality (DPI) (Beaudry and Renner, 2011; Merhav, 2012; Braverman et al., 2016). The DPI states that when input h_i passes through a first-level block to obtain information h_{i+1}, and then

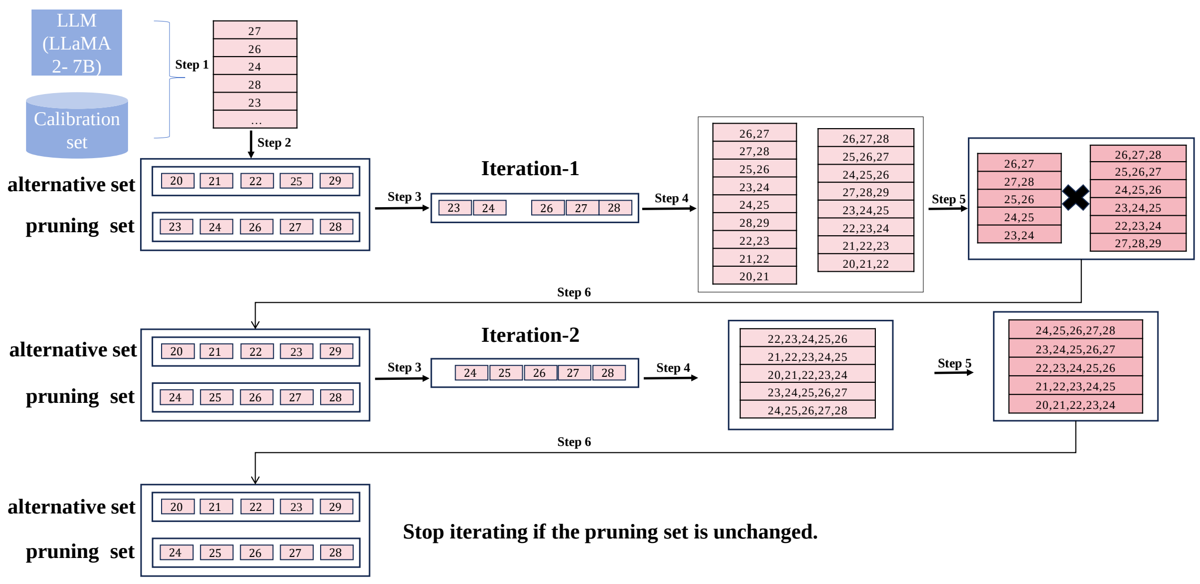

Figure 2: An overview of our method. The process of pruning 5 blocks in the LLaMA2-7B model. We provide a detailed description of the implementation for each step in Section 3.3.

h_{i+1} is processed further by a second-level block to produce the output h_{i+2}, the mutual information (Liu and Motani, 2022; Nguyen et al., 2014; Veyrat-Charvillon and Standaert, 2009) between h_i and h_{i+2} ($I_{h_i}^{h_{i+2}}$) satisfies the following relationship:

$$I_{h_i}^{h_{i+2}} \leq \min(I_{h_i}^{h_{i+1}}, I_{h_{i+1}}^{h_{i+2}}) \tag{2}$$

As illustrated in Figure 1, consider the combination of two blocks as an example. Suppose the importance of any block on the left ($h_i \rightarrow h_{i+1} \rightarrow h_{i+2}$) is lower than that of any block on the right ($h_j \rightarrow h_{j+1} \rightarrow h_{j+2}$). Under this scenario, the mutual information follows the following characteristics:

$$\min(I_{h_i}^{h_{i+1}}, I_{h_{i+1}}^{h_{i+2}}) \geq \max(I_{h_j}^{h_{j+1}}, I_{h_{j+1}}^{h_{j+2}}) \tag{3}$$

Based on the DPI (Beaudry and Renner, 2011; Merhav, 2012; Braverman et al., 2016), the following formula can be derived:

$$\overline{I_{h_i}^{h_{i+2}}} \geq \overline{I_{h_j}^{h_{j+2}}} \tag{4}$$

Here, $\overline{I_{h_j}^{h_{j+2}}}$ ($\min(I_{h_j}^{h_{j+1}}, I_{h_{j+1}}^{h_{j+2}})$) denotes the upper bound of $I_{h_j}^{h_{j+2}}$, and the same applies to the $\overline{I_{h_i}^{h_{i+2}}}$ ($\min(I_{h_i}^{h_{i+1}}, I_{h_{i+1}}^{h_{i+2}})$). The mutual information $I_{h_i}^{h_{i+2}}$ is only guaranteed to have a higher upper bound than $I_{h_j}^{h_{j+2}}$, but it is still possible that $I_{h_i}^{h_{i+2}}$ is less than $I_{h_j}^{h_{j+2}}$. Thus, a continuous block that follows two unimportant blocks is more likely to be unimportant as well. Nevertheless, the possibility that it could be important still exists. The same reasoning applies to the case of multiple continuous blocks, extending from the scenario of two continuous blocks.

To enhance the accuracy of block selection, we assess the importance of continuous blocks as a whole rather than individually. This approach prevents the unintended removal of blocks that, while seemingly less important on their own, significantly contribute to the overall performance of the model. For example, consider the LLaMA3.1-8B model, which consists of 32 blocks. When pruning 5 blocks, we need to evaluate the mutual information of 150 blocks, including both individual blocks and continuous block. We will elaborate on our detailed iterative block selection algorithm in Section 3.3.

3.3 Fast-Block-Select (Iterative Updates)

In practical applications, calculating the importance (measured by mutual information) of all contiguous blocks can be highly time consuming and resource intensive. For example, in the LLaMA2-7B model, there are as many as 528 possible combinations of contiguous blocks, making the task extremely complex. Therefore, it is crucial to explore methods that can accelerate the block selection process. A straightforward idea might be to use dynamic programming to tackle this problem. However, since the importance of contiguous blocks does not have a strict quantitative relationship with the importance of their sub-blocks, this approach is not viable. To address this challenge, we have designed a heuristic block selection method called Fast-Block-Select to significantly enhance the speed of block selection. Assuming the model consists of a total of T blocks, we aim to prune N blocks. The process can be broken down into the following steps (Figure 2 provides a detailed illustration of our process for pruning 5 blocks in the LLaMA2-7B model):

Step 1 : We employ a calibration set to obtain the importance of each independent block within the model and subsequently rank them in ascending order of importance. In Figure 2, we employ a calibration set to perform inference on LLaMA2-7B and obtain a list of blocks sorted in ascending order of importance (27, 26, 24, 28, 23 ...).

Step 2 : We select the top N blocks according to their importance ranking to form the pruning set. Additionally, we identify the M least important blocks that are outside the pruning set to form the alternative set, where M satisfies the following conditions:

$$M = \min(N, T - N) \tag{5}$$

The number of elements in the alternative set (M), is typically set to match the number of elements in the pruning set (N), and should not exceed the total number of remaining blocks ($T - N$). In Figure 2, both N and M are set to 5. The pruning set is defined as $\{23, 24, 26, 27, 28\}$, and the alternative set is defined as $\{20, 21, 22, 25, 29\}$.

Step 3 : We categorize the blocks in the pruning set into groups to form contiguous block sets. In Figure 2, in Iteration-1, we group the pruning set into $\{23, 24\}$ and $\{26, 27, 28\}$. In Iteration-2, since all elements in the pruning set are contiguous, there is only one group: $\{24, 25, 26, 27, 28\}$.

Step 4 : For each contiguous block within the contiguous blocks set, we generate all possible contiguous block sets of matching lengths by utilizing blocks from both the pruning set and the alternative set. We assess the importance of each contiguous block by summing the importance values of its constituent individual blocks, and subsequently sort these contiguous blocks in ascending order of their estimated importance. This method provides a reasonable approximation of the importance of contiguous blocks, as a sequence composed of multiple important individual blocks is more likely to be significant. Importantly, this approach eliminates the need to directly compute the mutual information between the inputs and outputs of the contiguous blocks; instead, it simply involves aggregating the mutual information of the individual blocks. In Figure 2, during Iteration-1 (with contiguous block lengths of 2 and 3), we construct possible contiguous block sets using elements from the pruning set and the candidate set. We also estimate the importance of these contiguous blocks and sort them, resulting in the sets $\{\{26, 27\}, \{27, 28\}, \{25, 26\} \ldots\}$ for the set $\{23, 34\}$. For the set $\{26, 27, 28\}$, we construct the contiguous block sets $\{\{26, 27, 28\}, \{25, 26, 27\} \ldots\}$. In Iteration-2 (with contiguous block length of 5), we construct the contiguous block sets $\{\{22, 23, 24, 25, 26\} \ldots\}$.

Step 5 : For each group, we compute the exact mutual information of the top K contiguous blocks, and subsequently rank these blocks based on their mutual information. Here, K is determined by the following conditions:

$$K = \min(\lfloor \log L \rfloor + k, l) \tag{6}$$

Here, L denotes the length of the corresponding contiguous block, while l represents the number of elements in the continuous block set. Additionally, k signifies the number of extra elements we consider (a hyperparameter). As L increases, the error in measuring the importance of a continuous block based solely on the sum of the importance of individual blocks also increases. Therefore, we incorporate $\log L$ into the calculation of K, ensuring that K increases with L. Additionally, the upper bound of K cannot exceed l. In Figure 2, in Iteration-1, L is 2 and 3, while K is 5 and 6, respectively. In Iteration-2, L is 5, and K is 5.

Step 6 : For each group of contiguous blocks, we select the combination with the highest sum of mutual information that does not conflict as the updated block combination for this iteration. In Figure 2, in Iteration-1, we select $\{24, 25\}$ and $\{26, 27, 28\}$, resulting in a new pruning set $\{24, 25, 26, 27, 28\}$. In Iteration-2, we select $\{24, 25, 26, 27, 28\}$. The pruning set changes between before and after Iteration-1, so the iterative algorithm continues. However, the pruning set remains unchanged between before and after Iteration-2, so the iteration stops.

The steps 3, 4, 5, and 6 are executed in each iteration. This process continues until the selected blocks remain unchanged from one iteration to the next. Our method takes into account the influence of blocks more comprehensively and thoroughly both before and after each iteration, which enables it to obtain either better or equivalent solutions (never worse). This provides a significant guarantee of convergence. Extensive testing on a wide range of models and data also show that our method does not exhibit oscillation phenomena, further demonstrating its effectiveness.

4 Experiments

4.1 Experimental Setup

Models and Benchmarks. To demonstrate the effectiveness of our method, we conduct extensive evaluations on four representative LLMs with diverse architectures and scales, including LLaMA3.1-8B (Grattafiori et al., 2024), LLaMA2-13B (Touvron et al., 2023b), Qwen2.5-7B (Yang et al., 2024a) and Qwen2.5-14B (Yang et al., 2024a). In some experiments, we also use models like LLaMA2-7B (Touvron et al., 2023b). We employ a wide range of benchmarks. These benchmarks include MMLU (Hendrycks et al., 2020), CMMLU (Li et al., 2023b), PIQA (Bisk et al., 2020b), Winogrande (ai2, 2019), HellaSwag (Zellers et al., 2019), BoolQ (Clark et al., 2019), MathQA (Amini et al., 2019), ARC-Easy and ARC-Challenge (Clark et al., 2018), RTE (Wang et al., 2018), WNLI (Wang et al., 2018), CB (Wang et al., 2019) and SST-2 (Wang et al., 2018). All benchmark results are reported using accuracy as the evaluation metric. More details can be found in Appendix A.1.

Baselines. We conduct extensive comparative evaluations against other pruning algorithms, including LLM-Pruner (Ma et al., 2023), FLAP (An et al., 2024), Shortened LLaMA (abbreviated as Shortened)(Kim et al., 2024), ShortGPT (Men et al., 2024) and SLEB (Song et al., 2024). Additionally, we also use SliceGPT to test inference speed (Ashkboos et al., 2024). Through these comprehensive comparisons, we thoroughly assess the strengths of our approach. More details can be found in Appendix A.2.

Implementation Details. We implement our approach using PyTorch (Paszke et al., 2019) and the HuggingFace Transformers library (Wolf, 2020), conducting experiments on NVIDIA A100 GPUs with 80GB memory. If the product of a model's total Transformer blocks and target sparsity is not an integer, we round up to determine the number of blocks to prune. We set the value of the hyperparameter k to 5. For performance comparisons, we maintain consistent experimental settings, including the calibration set and pruning rate.

4.2 Main Results

To validate the effectiveness of our proposed method, comparative experiments are conducted on the LLaMA3.1, LLaMA2 and Qwen2.5 series, employing standard benchmarks and baselines commonly utilized in the assessment of large language models. Dense denotes an unpruned model. The experimental results are shown in Table 1. These results suggest that the models pruned via our proposed method demonstrate enhanced overall performance when compared with the baseline methods, maintaining most of the large language model's capabilities. Although our method performs slightly worse on a small subset of datasets, this is still acceptable. The average accuracy across all tasks further highlights the effectiveness of our approach.

4.3 Ablation Study

Iterative Update Blocks. We investigate the impact of incorporating the DPI into the MI-PRUN method for iterative pruning block updates by meticulously comparing two pruning strategies. The experiment begins with an assessment of the mutual information of each block and employs a greedy strategy to prune the block with the highest mutual information value. Meanwhile, the MI-PRUN method calculates the mutual information between continuous blocks, ensuring that the collective impact of these blocks on the overall model performance is considered. An iterative optimization algorithm enhances the selection process, guaranteeing that the pruning results are both comprehensive and accurate. We prune 15.32% parameter on LLaMA2-7B. As shown in Figure 3, the results compare the model accuracy under the two strategies, vividly demonstrating the significant impact of considering iterative update blocks on enhancing model performance.

Fast-Block-Select. We set out to compare two distinct block selection methods: the Brute Force approach and our heuristic based Fast-Block-Select algorithm. The Brute Force approach is exhaustive in nature, calculating the importance of all individual and contiguous blocks by evaluating every possible combination. While this method ensures no potential solution is overlooked, it is highly complex and resource-intensive, requiring an enormous number of solution attempts to identify the opti-

Models	Methods	Ratios	MMLU	CMMLU	ARC-E	ARC-C	PIQA	Winogrande	Benchmarks HellaSwag	BoolQ	MathQA	WNLI	SST-2	RTE	CB	Avg
LLaMA3.1-8B	Dense	0.00%	63.35	50.85	81.52	51.28	80.30	74.03	60.05	82.26	39.56	59.15	76.83	71.12	60.71	65.46
	LLM-Pruner	13.62%	52.01	41.12	67.36	42.06	74.54	69.32	51.50	73.78	32.35	50.78	69.37	49.40	55.28	56.07
	FLAP	13.21%	52.11	41.13	67.42	42.15	74.63	69.36	51.56	73.82	32.39	50.82	69.41	49.44	55.34	56.11
	Shortened	13.58%	33.54	34.28	72.31	42.15	72.63	69.61	47.96	45.23	34.27	56.34	52.52	67.15	69.64	53.66
	ShortGPT	13.58%	51.92	41.11	67.34	42.06	74.48	69.38	51.56	73.76	32.50	50.70	69.31	49.50	55.36	56.08
	SLEB	13.58%	28.35	25.51	71.04	36.18	75.46	62.98	49.70	57.89	27.30	46.48	55.85	57.04	35.71	48.42
	Ours	13.58%	60.74	46.65	73.06	43.83	76.97	73.47	53.71	69.76	32.61	60.48	82.96	70.72	73.16	**62.93**
LLaMA2-13B	Dense	0.00%	52.10	34.73	79.42	48.38	79.05	72.22	60.07	80.61	32.09	66.20	87.61	69.31	80.36	64.78
	LLM-Pruner	24.18%	50.11	33.57	61.16	37.67	71.38	70.71	47.37	62.53	24.41	43.24	65.37	59.19	51.38	52.16
	FLAP	24.16%	49.89	33.91	60.90	37.65	71.44	70.69	47.54	62.43	24.78	43.26	64.91	59.19	51.34	52.15
	Shortened	24.37%	26.71	25.50	26.26	22.61	51.14	48.22	25.76	38.93	18.89	43.66	46.56	52.71	37.50	35.73
	ShortGPT	24.37%	50.13	33.97	61.32	37.88	71.44	70.80	47.71	62.54	24.82	43.66	65.37	59.57	51.79	52.38
	SLEB	24.37%	23.76	25.41	67.85	33.87	75.41	63.77	48.76	62.42	25.46	45.07	50.92	58.84	41.07	47.89
	Ours	24.37%	51.89	33.73	64.33	38.44	72.93	69.97	48.55	58.36	26.74	54.05	59.79	63.79	50.64	**53.32**
Qwen2.5-7B	Dense	0.00%	71.92	81.69	80.56	48.21	78.73	72.93	59.92	84.46	43.22	71.83	91.86	81.23	87.50	73.39
	LLM-Pruner	15.21%	36.96	36.71	71.33	38.20	74.86	55.79	47.71	55.67	31.08	60.43	70.79	54.42	38.81	51.75
	FLAP	15.36%	37.17	36.39	71.47	37.83	76.21	55.65	47.47	57.23	30.11	61.61	70.68	54.59	37.32	51.82
	Shortened	15.30%	24.90	25.08	25.25	20.31	53.65	50.99	25.69	37.83	19.50	53.52	50.92	46.21	19.64	34.88
	ShortGPT	15.30%	36.89	31.07	71.46	37.80	76.12	55.80	48.67	63.21	30.59	42.25	80.62	54.87	41.07	51.57
	SLEB	15.30%	38.56	38.25	71.84	39.42	76.77	55.96	47.98	57.49	31.12	61.97	72.25	56.32	39.29	52.86
	Ours	15.30%	52.78	50.90	71.03	38.67	76.90	56.89	50.87	72.09	30.60	66.17	63.15	65.31	57.14	**58.09**
Qwen2.5-14B	Dense	0.00%	77.45	84.44	82.37	56.31	81.12	75.37	63.37	85.23	53.03	77.46	89.11	79.78	80.36	75.80
	LLM-Pruner	18.52%	43.09	42.01	73.33	40.54	73.58	58.47	47.79	61.89	31.53	48.79	54.26	56.88	49.27	52.42
	FLAP	18.89%	44.86	44.77	50.29	31.00	60.72	51.44	34.14	65.16	25.65	65.28	89.95	76.87	68.74	54.53
	Shortened	18.64%	24.63	25.31	25.04	20.14	52.88	50.43	25.69	37.92	18.93	52.11	48.62	51.99	37.50	36.25
	ShortGPT	18.64%	45.75	45.63	50.63	31.40	61.81	52.64	34.41	65.72	26.16	66.20	91.28	78.34	69.64	55.35
	SLEB	18.64%	43.77	42.90	74.07	41.04	74.37	58.96	48.43	62.57	32.23	49.30	54.93	57.40	50.00	53.07
	Ours	18.64%	71.26	76.07	67.92	44.11	71.08	69.94	47.72	64.05	34.68	61.47	81.95	73.71	69.52	**64.11**

Table 1: Comparison of pruning techniques in large language models through benchmark testing. We compare various pruning methods applied to four large language models using a diverse set of benchmark tests. For each combination of model and pruning technique, we document the pruning ratio, the accuracy scores for specific tasks, and the average accuracy across all tasks.

Method	Dense	SliceGPT	LLM-Pruner	FLAP	MI-PRUN
Time (ms)	330.33	297.40	307.15	294.98	258.59

Table 2: Inference time comparison of LLaMA3.1-8B model using different pruning methods.

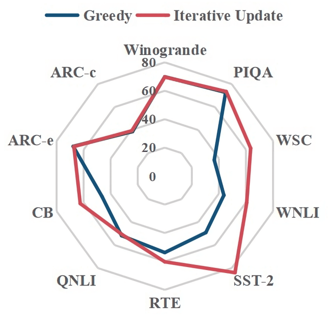

Figure 3: Performance comparison between Greedy Strategy and Iterative Update.

mal blocks for pruning. On the other hand, the Fast-Block-Select algorithm takes a more strategic approach. By employing a heuristic strategy, it efficiently identifies the most promising blocks for pruning without the need to evaluate every possible combination. This significantly reduces the computational burden and enhances the overall efficiency of the block selection process. Our primary focus is on comparing the computational efficiency of the two methods. We prune 15.32% parameter on LLaMA3.1-8B and 24.37% on LLaMA2-13B. As shown in Table 3, the Fast-Block-Select algorithm is significantly more efficient, highlighting its practicality and superiority over the Brute Force approach.

Models	Methods	Time (s)
LLaMA3.1-8B	Brute Force	183.88
	Fast-Block-Select	56.12
LLaMA2-13B	Brute Force	433.17
	Fast-Block-Select	67.35

Table 3: Comparison of pruning efficiency between Brute Force and Fast-Block-Select. The last line indicates that when pruning on LLaMA2-13B, the time required by Brute Force and Fast-Block-Select is 433.17s and 67.35s, respectively. The same applies to the above line.

4.4 Hyperparameter Impact Analysis

We evaluate the impact of the hyperparameter k in Step 6 of our method on performance. Here, k signifies the number of additional elements considered. As k increases, the consideration of blocks becomes more comprehensive, though this also introduces a higher computational load. We prune 24.37% parameter on LLaMA2-13B. As shown in Table 4, we experiment with several different configurations of k, and all achieve satisfactory results. This demonstrates the effectiveness and robustness of our approach.

k	MMLU	CMMLU	ARC-C	Avg
3	51.72	33.39	38.06	41.06
4	51.89	33.73	38.44	41.36
5	51.89	33.73	38.44	41.36

Table 4: Evaluation of the influence of the number of additional elements considered k on the LLaMA2-13B.

4.5 Comparison of Inference Time

We evaluate the execution time for a single prefill forward inference of the LLaMA3.1-8B model, using an input tensor of shape $[1, 1024]$ in the fp32 format. The remaining experimental settings are consistent with the details provided above. We apply a pruning strategy that reduces the model parameters by 13.58%. For comparison, we also include the Dense (unpruned) model as a baseline. Additionally, we test our approach against other non-block pruning methods, including SliceGPT (Ashkboos et al., 2024), LLM-Pruner (Ma et al., 2023), and FLAP (An et al., 2024). Table 2 presents a comparison of inference time between our method and the aforementioned approaches. Our method achieves the fastest runtime, significantly reducing the inference time. These results underscore the substantial acceleration achieved by our method.

4.6 Pruning Overhead Analysis

We compare the computational overhead of our proposed pruning method with that of SLEB when pruning 15.63% blocks on LLaMA3.1-8B. SLEB (Song et al., 2024) is also a strategy that considers the global optimal solution for block pruning, but it requires multiple calibration set tests in each iteration of pruning. The experimental settings are kept consistent with those described in the preceding section. We conduct inference on the PIQA (Bisk et al., 2020a) calibration set of the same scale using both methods. As shown in Table 5, our method has an average inference time of 56.12 seconds. In contrast, SLEB has an average inference time of 288.6 seconds. This efficiency improvement is attributed to our Fast-Block-Select strategy, which can more rapidly converge to an effective pruning configuration.

4.7 Additional Results

We test our method using various calibration sets and consistently achieve the same results, demonstrating the stability of our approach (Appendix B). We also evaluate the performance under different pruning ratios and find that the decrease in accuracy is relatively small as the pruning ratio increases (Appendix C). This indicates that by appropriately adjusting the pruning ratio, we can achieve a favorable balance between efficiency and performance. Moreover, we conduct extensive testing on the pruned model using a variety of specific generation prompts to thoroughly evaluate its performance. The results are highly encouraging, as we find that the quality of the generated content remains consistently high across different scenarios (Appendix D).

Method	Time (s)
SLEB	288.6
Ours	56.12

Table 5: Comparison of pruning time between our method and SLEB on LLaMA3.1-8B.

5 Discussion

In this study, we focus on optimizing LLMs by pruning intermediate blocks (layers) through the use of skip-connections. Specifically, we replace certain operations with identity matrices, thereby streamlining the network structure and significantly reducing computational overhead while maintaining performance. The versatility of our method is noteworthy, as it can be applied to a wide range of models, including those that already employ skip-connections, such as ResNet. Moreover, our approach supports fine-grained pruning, which not only enables flexible optimization but also enhances efficiency in models.

6 Conclusion

In this paper, we propose MI-PRUN, a structured pruning approach for large language models. MI-PRUN leverages mutual information and the Data Processing Inequality to iteratively refine the blocks that contribute less to the model's performance. Furthermore, it employs the Fast-Block-Select strategy to augments the efficiency. Our experimental results show that MI-PRUN effectively prunes the model, alleviating computational load without compromising its performance capabilities. Employing the strategy of eliminating entire blocks, MI-PRUN effectively enhances the inference speed of end-to-end LLM inference.

7 Limitations

Extensive experiments on a wide range of datasets and models demonstrate that our method outperforms other baselines in terms of performance. However, it is worth noting that our method may exhibit slightly inferior performance on a small number of datasets, although the results are still acceptable. As far as we know, this issue of varying performance across different datasets is also observed in current pruning methods, which may be attributed to differences in the distribution characteristics of the data. This is one of the key aspects that we will focus on in our future work.

References

2019. Winogrande: An adversarial winograd schema challenge at scale.

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*.

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

Anonymous. 2024. Unstructured pruning and low rank factorisation of self-supervised pre-trained speech models. *IEEE Transactions on Audio, Speech, and Language Processing*, pages 1046–1058.

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*.

Roberto Battiti. 1994. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on neural networks*, 5(4):537–550.

Normand J Beaudry and Renato Renner. 2011. An intuitive proof of the data processing inequality. *arXiv preprint arXiv:1107.0740*.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020a. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

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

Mark Braverman, Ankit Garg, Tengyu Ma, Huy L Nguyen, and David P Woodruff. 2016. Communication lower bounds for statistical estimation problems via a distributed data processing inequality. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 1011–1020.

T Brown, B Mann, N Ryder, M Subbiah, JD Kaplan, P Dhariwal, A Neelakantan, P Shyam, G Sastry, A Askell, et al. 2020. Language models are few-shot learners advances in neural information processing systems 33.

Yuchen Cai, Zhen Wang, Yujun Li, Sheng Wang, Zhiyuan Liu, and Maosong Sun. 2023. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2302.06557*.

Xiaodong Chen, Yuxuan Hu, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. 2024. Streamlining redundant layers to compress large language models. *arXiv preprint arXiv:2403.19135*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

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*.

Chun Fan, Jiwei Li, Xiang Ao, Fei Wu, Yuxian Meng, and Xiaofei Sun. 2021. Layer-wise model pruning based on mutual information. *arXiv preprint arXiv:2108.12594*.

Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. 2023. Depgraph: Towards any structural pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16091–16101.

Madan Ravi Ganesh, Jason J Corso, and Salimeh Yasaei Sekeh. 2021. Mint: Deep network compression via mutual information-based neuron trimming. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 8251–8258. IEEE.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Song Han, Huizi Mao, and William J Dally. 2015a. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

Song Han, Jeff Pool, John Tran, and William Dally. 2015b. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

E. Huang et al. 2023. Evaluating large language models in complex scenarios. *Journal of Computational Linguistics*.

Hanjuan Huang, Hao-Jia Song, and Hsing-Kuo Pao. 2024. Large language model pruning. *arXiv preprint arXiv:2406.00030*.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. 2024. Shortened llama: A simple depth pruning for large language models. *arXiv preprint arXiv:2402.02834*, 11.

Vid Kocijan, Thomas Lukasiewicz, Ernest Davis, Gary Marcus, and Leora Morgenstern. 2020. A review of winograd schema challenge datasets and approaches. *arXiv preprint arXiv:2004.13831*.

C. Li et al. 2023a. Fine-tuning techniques for efficient model adaptation. *AI Research Journal*.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2023b. Cmmlu: Measuring massive multitask language understanding in chinese. *arXiv preprint arXiv:2306.09212*.

Yong Li, Wei Du, Liquan Han, Zhenjian Zhang, and Tongtong Liu. 2023c. A communication-efficient, privacy-preserving federated learning algorithm based on two-stage gradient pruning and differentiated differential privacy. *Sensors*, 23(23):9305.

Sheng Liao et al. 2023. Can unstructured pruning reduce the depth in deep neural networks? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*.

Huawen Liu, Jigui Sun, Lei Liu, and Huijie Zhang. 2009. Feature selection with dynamic mutual information. *Pattern Recognition*, 42(7):1330–1339.

Shiyu Liu and Mehul Motani. 2022. Improving mutual information based feature selection by boosting unique relevance. *arXiv preprint arXiv:2212.06143*.

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

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2024. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*.

Neri Merhav. 2012. Data-processing inequalities based on a certain structured class of information measures with application to estimation theory. *IEEE transactions on information theory*, 58(8):5287–5301.

Xuan Vinh Nguyen, Jeffrey Chan, Simone Romano, and James Bailey. 2014. Effective global approaches for mutual information based feature selection. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 512–521.

Cláudia Pascoal, M Rosário Oliveira, António Pacheco, and Rui Valadas. 2017. Theoretical evaluation of feature selection methods based on mutual information. *Neurocomputing*, 226:168–181.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Josien PW Pluim, JB Antoine Maintz, and Max A Viergever. 2003. Mutual-information-based registration of medical images: a survey. *IEEE transactions on medical imaging*, 22(8):986–1004.

Adam Poliak. 2020. A survey on recognizing textual entailment as an nlp evaluation. *arXiv preprint arXiv:2010.03061*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. 2024. Sleb: Streamlining llms through redundancy verification and elimination of transformer blocks. *arXiv preprint arXiv:2402.09025*.

Ralf Steuer, Jürgen Kurths, Carsten O Daub, Janko Weise, and Joachim Selbig. 2002. The mutual information: detecting and evaluating dependencies between variables. *Bioinformatics*, 18(suppl_2):S231–S240.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. 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 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

H Touvron, T Lavril, G Izacard, X Martinet, MA Lachaux, T Lacroix, B Rozière, N Goyal, E Hambro, F Azhar, et al. 2023a. Open and efficient foundation language models. *Preprint at arXiv. https://doi.org/10.48550/arXiv*, 2302.

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

Jorge R Vergara and Pablo A Estévez. 2014. A review of feature selection methods based on mutual information. *Neural computing and applications*, 24:175–186.

Nicolas Veyrat-Charvillon and François-Xavier Standaert. 2009. Mutual information analysis: how, when and why? In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 429–443. Springer.

La The Vinh, Sungyoung Lee, Young-Tack Park, and Brian J d'Auriol. 2012. A novel feature selection method based on normalized mutual information. *Applied Intelligence*, 37:100–120.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

F. Wang et al. 2023. Practical applications of llms in specialized domains. *Specialized AI Applications*.

Charles Westphal, Stephen Hailes, and Mirco Musolesi. 2024. Mutual information preserving neural network pruning. *arXiv preprint arXiv:2411.00147*.

Thomas Wolf. 2020. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Vikas Yadav, Steven Bethard, and Mihai Surdeanu. 2019. Quick and (not so) dirty: Unsupervised selection of justification sentences for multi-hop question answering. *arXiv preprint arXiv:1911.07176*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024a. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Yifei Yang, Zouying Cao, and Hai Zhao. 2024b. Laco: Large language model pruning via layer collapse. *arXiv preprint arXiv:2402.11187*.

Zhen Yang, Zilun Zhang, Sheng Wang, Jie Li, Meishan Zhang, Zhiyuan Liu, and Maosong Sun. 2021. Knowledge distillation: A survey. *arXiv preprint arXiv:2106.05860*.

Zhengwu Yang and Han Zhang. 2022. Comparative analysis of structured pruning and unstructured pruning. In *Frontier Computing*, page 112. Springer.

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*.

D. Zhang et al. 2023. Parameter-efficient fine-tuning methods for llms. *Journal of Machine Learning Research*.

Qifan Zhang, Yunhui Guo, and Yu Xiang. 2024. Continual distillation learning: Knowledge distillation in prompt-based continual learning. *Preprint*, arXiv:2407.13911.

Yuxiao Zhou, Zhen Wang, Yujun Li, Sheng Wang, Zhiyuan Liu, and Maosong Sun. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. *arXiv preprint arXiv:2302.06557*.

Yuxiao Zhou, Zhen Wang, Yujun Li, Sheng Wang, Zhiyuan Liu, and Maosong Sun. 2024. Framequant: Flexible low-bit quantization for transformers. *arXiv preprint arXiv:2402.06557*.

B. Zhu et al. 2023. Expanding frontiers in large language models. *AI Frontier Research*.

A Experimental Setup

A.1 Benchmarks

In order to comprehensively assess the impact of pruning on the capabilities of large language models, we conduct an evaluation using widely prevalent benchmarks. Winogrande (ai2, 2019) is a comprehensive dataset designed to assess models' ability to reason with common sense. It consists of over 45,000 question-answer pairs that challenge models with complex, real-world scenarios. PIQA (Bisk et al., 2020a) is an innovative dataset that focuses on understanding physical interactions. It requires models to comprehend the relationships between objects to answer questions about their physical interactions. The WSC dataset (Kocijan et al., 2020) presents a series of text entailment tasks where models must discern whether one sentence logically implies another, with a focus on pronoun resolution and contextual understanding. WNLI (ai2, 2019) is a subset of the WinoGrande dataset, concentrating on natural language inference tasks. It tests models' capabilities in identifying whether one sentence entails another within a given context. SST-2 (Socher et al., 2013) is a sentiment analysis benchmark that includes movie reviews. Models are tasked with determining the sentiment expressed in the reviews, whether positive or negative. RTE (Poliak, 2020) challenges models to evaluate the logical relationships between sentences. It is a critical test for models' ability to understand textual entailment. QNLI (Wang et al., 2018) combines question answering with natural language inference, requiring models to assess the logical relationship between a given question and a set of candidate answers. CB (Talmor et al., 2019) is a question answering dataset that taps into general knowledge. It requires models to leverage common sense to provide accurate answers to a variety of questions. ARC-e (Yadav et al., 2019) is tailored for elementary-level science and math questions, designed to test models' understanding and reasoning abilities in these domains. ARC-c (Yadav et al., 2019) extends the challenge to college-level complexity, assessing models' proficiency in advanced scientific and mathematical reasoning. These datasets serve as critical benchmarks for evaluating the performance of language models across a spectrum of NLP tasks, including question answering, text entailment, sentiment analysis, and commonsense reasoning.

A.2 Baselines

We now introduce the pruning methods for comparison. Shortened LLaMA (Kim et al., 2024) selectively removes less important blocks based on block level importance scores, thereby accelerating model inference without significantly impacting performance. ShortGPT (Men et al., 2024) defines a BI metric to measure the importance of each layer within the model and directly removes those layers. SLEB (Song et al., 2024) employs a logit based approach to identify unnecessary transformer layers and updates the importance scores after each layer removal. Through these comprehensive comparisons, we thoroughly assess the strengths of our approach. SliceGPT (Ashkboos et al., 2024) is a post-training sparsification scheme which replaces each weight matrix with a smaller matrix. LLM-Pruner (Ma et al., 2023) adopts structural pruning that selectively removes non-critical coupled structures based on gradient information. FLAP (An et al., 2024) is a structured pruning framework that reduces storage by leveraging fluctuation based metrics and adaptive model compression.

B Results on Different Calibration Sets

In order to evaluate the impact of different calibration sets on pruning outcomes, we conduct experiments on Qwen-7B and Qwen-14B, pruning five and seven blocks, respectively. We test the pruned blocks under varying sizes using the WikiText-2 and Alpaca datasets. As illustrated in Table 6, despite the differences in experimental conditions, we achieve consistently stable pruning results. This demonstrates that our approach maintains robustness across diverse datasets.

C Sensitivity Analysis of Different Pruning Ratios

Figure 4 provides a comprehensive illustration of the influence of varying pruning ratios on the performance of LLaMA2-7B. As the pruning ratio escalates, the accuracy of the majority of tasks exhibits a relatively stable profile, albeit with a general tendency to decline. This stability underscores the remarkable robustness of our proposed method. The observed increase in perplexity at certain pruning levels is considered tolerable, given its inherently exponential correlation with loss. This phenomenon serves as compelling validation of the efficacy of our redundancy-aware iterative block pruning strategy. By selectively excising non-essential

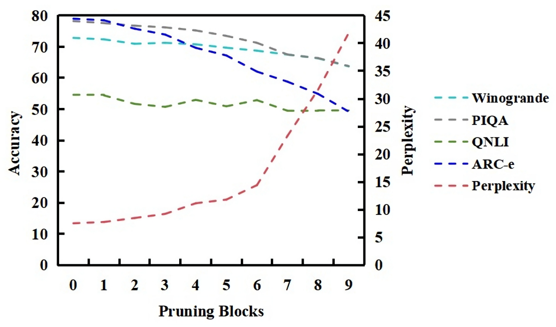

Figure 4: Performance of LLaMA2-7B with different pruning ratios (blocks).

blocks while meticulously preserving the model's core functional capabilities, our approach demonstrates its ability to maintain model performance even under significant pruning conditions.

D Case Study

We present a comprehensive comparison of sentences generated by the model both before and after pruning using our proposed method. The results clearly demonstrate that the sentences produced by the compressed model are of comparable quality to those generated by the original model. Table 7 presents the quality of the generated content after pruning 15.32% of the parameters on LLaMA2-7B. These sentences maintain fluency, relevance, and informativeness with respect to the given topics, thereby confirming that our pruning method effectively retains the model's ability to generate high-quality text while significantly reducing its complexity.

Models	Calibration set	Size	Results of pruning blocks
Qwen-7B	WikiText-2	256 512 1024	[24, 25, 26, 27, 28]
	Alpaca	256 512 1024	[24, 25, 26, 27, 28]
Qwen-14B	WikiText-2	256 512 1024	[29, 30, 31, 32, 33, 34, 35]
	Alpaca	256 512 1024	[29, 30, 31, 32, 33, 34, 35]

Table 6: Results of pruning blocks with varying sizes on diverse calibration sets.

Original Model	Pruned Model
10 steps to build an ios app: 1. Determine the purpose of your app: What problem does it solve or what goal does it achieve? Who is the audience and what features will they need? 2. Research the market: Look at similar apps in the Apple App Store and see what features they offer, what features are missing, and what can be improved. 3. Design the user ...	10 steps to build an ios app: 1. Determine the purpose of your app: Before starting to build an iOS app, you need to determine its purpose and what problem it will solve for the users. 2. Choose the right programming language: Swift and Objective-C are the two most popular programming languages used for iOS app development. Swift is recommended ...
Explain what artificial intelligence is. Artificial intelligence (AI) refers to the development of computer systems able to perform tasks that typically require human intelligence, such as visual perception, speech recognition, decision-making, and language translation. AI systems use machine learning, deep learning, and natural language processing to analyze ...	Explain what artificial intelligence is. Artificial intelligence (AI) is a branch of computer science dedicated to creating intelligent machines that can perform tasks that typically require human intelligence, such as understanding language, recognizing images, reasoning, and decision-making. AI is a rapidly growing field that is transforming many industries...
List the equipment needed for rock climbing. Rock climbing is a popular adventure sport that requires specific equipment to ensure safety and success. Here are some of the essential pieces of equipment needed for rock climbing:1. Climbing Shoes: Climbing shoes are specifically designed to provide a snug fit and good grip on the rock. They are usually ...	List the equipment needed for rock climbing. Rock climbing is a popular adventure sport that involves climbing a rock wall using a variety of techniques and equipment. The equipment needed for rock climbing includes: 1. Climbing shoes: These are specialized shoes designed specifically for rock climbing. They are typically tight-fitting and provide good grip ...

Table 7: Some examples of the generations obtained with the original model and the pruned model. The content marked with a line is the user's question.