MI-PRUN : Leveraging Mutual Information for Efficient LLM Pruning

Anonymous ACL submission

Abstract

Large language models 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 present *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* to ensure the preservation of contiguous blocks essential for overall model performance, avoiding their accidental pruning. Concurrently, we develop the *Fast-Block-Select* algorithm to enhance the efficiency of the pruning process. Comprehensive experiments show that our proposed method surpasses the previous state-of-the-art (SOTA) model pruning methods.

1 Introduction

Recently, 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). 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. As shown in Figure 1, the growth in the size of large language models has become a striking phenomenon, a trend that not only captures attention but also sparks profound contemplation and challenges. The LLaMA models have experienced a remarkable surge in size, soaring from 7B to 65B, and potentially even greater heights. These enormous leaps in numbers are not only impressive but also deeply reveal that as the models grow in size, the complexity of their deployment and application in practical scenarios also increases.

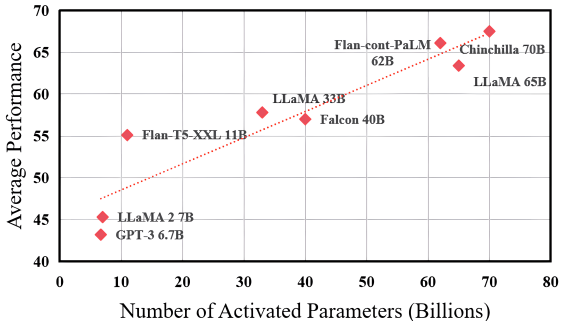

Figure 1: The trend of growth in the size of large language models.

Model pruning is essential for enhancing the efficiency of deep learning models by eliminating redundant weights or neurons without compromising performance. While the benefits of pruning are clear, applying it to large language models still presents several challenges (Ma et al., 2023; Ashkboos et al., 2024). There are four main key points: *1) Complexity in Weight Selection:* Many pruning methods rely on intricate gradient analysis (Li et al., 2023), which is both resource-intensive and time-consuming. *2) Challenges in Hardware Utilization:* The sparsity introduced by unstructured pruning complicates the efficient use of hardware (Han et al., 2015a). *3) Necessity for Model Fine-tuning:* Pruning can impact model accuracy, often requiring fine-tuning to regain pre-pruning levels of precision, which is particularly costly for large language models (Han et al., 2015b). *4) Overemphasis on Model Width Reduction:* Some approaches focus primarily on reducing the model's width by pruning certain connections within the network (Fang et al., 2023), potentially neglecting the optimization of parameters in depth and other dimensions (Men et al., 2024; Yang et al., 2024; Kim et al., 2024; Song et al., 2024). Addressing these challenges is crucial for advancing pruning techniques in large language models.

Recently, some studies begin to use *mutual in-*

formation for optimizing neural network pruning. Some methods retain neurons with high *mutual information* with the preserved neurons in the upper layer for each layer of the neural network, starting pruning from the top softmax layer and moving downward (Fan et al., 2021). Others decide on pruning by calculating the *mutual information* between neurons; if the MI value between two neurons exceeds a threshold, it indicates overlapping information, allowing for the pruning of one without significant loss (Huang et al., 2024). MIPP maintains the *mutual information* of activations between layers during pruning and uses the Transfer Entropy Redundancy Criterion to remove neurons that do not contribute to downstream layers (Westphal et al., 2024). However, these methods optimize network structure and enhance pruning efficiency by evaluating the relationships between neurons for width pruning, have not yet been extended to depth layer pruning to achieve effective acceleration.

To design a simple pruning algorithm that is easy to deploy and use on hardware for large language models, some studies propose identifying less important blocks in large language models using cosine similarity and performing greedy pruning based on block importance (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 (Kim et al., 2022). Moreover, greedy block selection strategies based on this approach tend to find the local optimal solutions, and cannot identify the global optimal solution for the combination of pruning blocks. LaCo (Yang et al., 2024) reduces the model size by merging subsequent layers into the previous ones. However, its effectiveness is often not as good as direct layer removal. SLEB (Song et al., 2024) and Shortened LLaMA (Kim et al., 2024) determine importance metrics to iteratively remove layers. However, the requirement to measure corresponding indicators with a calibration set after each layer removal often leads to high complexity.

In this paper, 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. It incorporates *Data Processing Inequality (DPI)* (Beaudry and Renner, 2011) to enhance the accuracy of identifying less important blocks by assessing the *MI* of inputs and outputs across continuous blocks. This helps in recognizing blocks that contribute significantly when part of a larger unit. To enhance efficiency, we develop the *Fast-Block-Select* algorithm. This innovative approach utilizes heuristic methods to efficiently pinpoint optimal candidates for pruning, thereby streamlining the pruning process.

The main contributions of our paper are summarized as follows:

- We leverages *mutual information* to quantify the transition of hidden states between different blocks, identifying and pruning weight blocks that contribute less to the model's performance.
- We incorporate the *Data Processing Inequality* to assess the overall importance of weight blocks, employing an iterative block updating strategy. This ensures that blocks that may seem unimportant individually but significantly contribute to the model as a whole are not mistakenly pruned.
- We develope the *Fast-Block-Select* algorithm to enhance the efficiency of the pruning process.
- Comprehensive experiments show that our proposed pruning method surpasses the previous state-of-the-art (SOTA) model pruning methods.

2 Related Work

Mutual information (Liu and Motani, 2022; Nguyen et al., 2014; Veyrat-Charvillon and Standaert, 2009) is a metric that measures the dependency between two variables, quantifying the amount of information about one variable that can be obtained by observing the other. It is highly valued for its capacity to detect non-linear relationships between variables (Vinh et al., 2012; Pascoal et al., 2017), which is essential for uncovering intricate data patterns and understanding model dynamics. It surpasses conventional linear measurement techniques by delving into the complex interplays within data (Pluim et al., 2003; Steuer et al., 2002). Grounded in probability distributions, it not only tracks the trends and magnitudes of variables, but more importantly, uncovers their statistical dependencies(Walters-Williams and Li, 2009).

Furthermore, the application of *mutual information* is remarkably broad, enabling the assessment of complex connections among multiple variables. In the field of feature selection, the application of *mutual information* is particularly important (Battiti, 1994; Liu et al., 2009; Vergara and Estévez, 2014). By calculating the *mutual information* between features and the target variable, one can effectively identify the most informative features. This not only enhances the model's performance and interpretability but also simplifies the model and reduces computational costs.

3 Methodology

In this section, we discuss details of the proposed pruning method which leverages *mutual information* to identify blocks that contribute less to the model's performance. Furthermore, we integrate the *Data Processing Inequality* into the iterative refinement of the pruning blocks. Concurrently, we develop the *Fast-Block-Select* algorithm to enhance the efficiency of the pruning process.

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 feature in the model's design: the output of each layer is added to the output of the previous layer through residual connections, thereby enabling the continuous transfer of information across different layers of the model. Specifically, the output of the $(i+1)$-th Transformer layer can be represented as follows:

$$x_{i+1} = Transformer_{i+1}(x_i) + x_i \quad (1)$$

where x_i and x_{i+1} are the outputs of the i-th and $(i+1)$-th layers, $Transformer_{i+1}$ is the transformation of the $(i+1)$-th layer.

In the Transformer architecture, *mutual information* is a key metric for measuring the flow of information. The *mutual information* I between continuous random variables x and y can be calculated as:

$$I = \int P(x,y) \log \frac{P(x,y)}{P(x)\,P(y)} dxdy \quad (2)$$

When the *mutual information* between the input and output state is unusually high, it typically indicates that the block's output is largely a direct reflection of its input state, suggesting a lower importance. In other words, the output state does not significantly add new important information but largely replicates the information from the input state, thereby introducing redundancy in information. In this case, the Transformer block may play a minor role in the overall functionality of the model. This is because it does not significantly transform or refine the input data but simply passes on the original information. Suppose there is a block f with input x and output y, the specific proof is as follows:

① When x and y are independent, the transformation effect of f is maximized, and the *mutual information* I is zero.

$$P(x,y) = P(x)\,P(y) \quad (3)$$

$$I = \int P(x,y) \log \frac{P(x)\,P(y)}{P(x)\,P(y)} dxdy = 0 \quad (4)$$

② When y is completely determined by x, such that for any x there exists a y for which $P(y|x) = 1$, the transformation function is minimal, and the *mutual information* is maximal.

$$P(x,y) = P(x)\,P(y|x) \quad (5)$$

$$\int P(x,y)\,dx = P(y) \quad (6)$$

$$I = -\int P(y) \log P(y) dy \quad (7)$$

③ Aside from these two extreme cases, as the transformation function f exerts a greater effect, $H(y|x)$ becomes larger and $I(x,y)$ becomes smaller.

$$I(x,y) = H(y) - H(y|x) \quad (8)$$

This method based on *mutual information* offers a more comprehensive and in-depth analytical approach compared to traditional methods based on cosine similarity. It focuses not only on the directionality of vectors but also on the actual information flow and dependency relationships between vectors, thereby providing a richer perspective for model optimization and understanding. *MI-PRUN* leverages the advantages of *mutual information* to more accurately identify parts of the model that contribute less to its performance. It not only enhances the efficiency of the model but also aids in a deeper understanding of the model's internal workings.

3.2 Iterative Update Blocks

In the previous section, we analyze the importance of blocks from the perspective of individual independent blocks. A potential issue is that some individual blocks may seem unimportant, but when considered as a whole with surrounding blocks, they may significantly contribute to the model's performance. To gain a clearer understanding of this issue, we introduce the *Data Processing Inequality*. The *Data Processing Inequality* indicates that when data undergo multi-level processing, with each processing step, there is a potential loss of information. In other words, data processing can transform data into more useful forms but will never create new information. The *mutual information* between inputs and outputs in a two-stage cascade channel is bounded by the individual *mutual information* of each stage, never exceeding the levels at either stage. Specifically, when input information x is processed by the first level processor to obtain information y, and then information y is further processed by the second level processor to output the result z. For such a two-level processor system, if y is known, then x and z are mutually independent, that is, $I(x; z|y) = 0$. Thus, we can arrive at the following conclusion:

$$I(x; z) = I(x; y) - I(x; y|z) = I(y; z) - I(y; z|x) \tag{9}$$

$$I(x; z) \leq \min\{I(x; y), I(y; z)\} \tag{10}$$

From this perspective, we can extend the concept to the transformer model. As illustrated in Figure 2, taking the combination of two blocks as an example, suppose that the *mutual information* of the blocks meets the following conditions:

$$\min(I_{1_a}, I_{1_b}) \geq \max(I_{2_a}, I_{2_b}) \geq \min(I_{2_a}, I_{2_b}) \tag{11}$$

then the importance of the two blocks on the left side is lower than that of the two blocks on the right side. Based on the Data Processing Inequality, the following formula can be derived:

$$I(x_1, z_1) \leq \min(I_{1,a}, I_{1,b}) \tag{12}$$

$$I(x_2, z_2) \leq \min(I_{2,a}, I_{2,b}) \tag{13}$$

Therefore, the continuous block corresponding to two unimportant blocks is more likely to be unimportant. However, there is still a possibility that it could be important.

To improve the accuracy of our pruning process, we evaluate the combined effect of adjacent modules rather than assessing them individually. This prevents the accidental removal of modules that, while seemingly less critical on their own, significantly contribute to the model's overall performance. Take the Llama2-7B model, for instance, which consists of 32 modules. If we aim to prune 5, we must evaluate the importance of 150 modules. The *mutual information* among blocks for the Llama2-7B model can be found in Appendix C. We start by identifying modules for pruning based on individual importance, then merge and categorize continuous modules into groups, with adjacent modules in the same group and non-adjacent in different groups. We compare each group with other continuous blocks of the same length, replacing the current group if we find a less important one. After updating the modules in each group, we iterate this process to refine our pruning strategy. Once updated, we select the best solution for each group, ensuring no conflicts. If conflicts occur, we prioritize the least important modules, ranking continuous blocks in each group to choose a combination with the lowest overall importance and no conflicts.

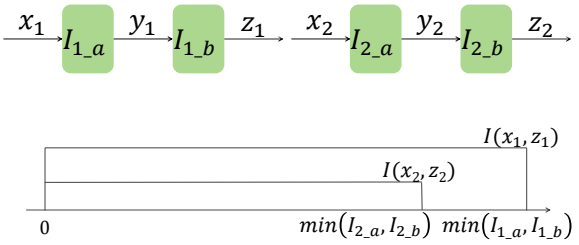

Figure 2: The relationship between the *MI* of the global continuous block and the local individual blocks.

3.3 Fast-Block-Select

In practical applications, calculating the importance of all contiguous blocks may require a significant amount of time and resources. For instance, in the Llama2-7B model, there are as many as 528 possible combinations of contiguous blocks, which is highly complex. Therefore, it is essential to explore methods that can expedite the block selection process. A simple idea might be to employ dynamic programming for the solution, but since the importance of contiguous blocks does not have a strict quantitative relationship with the importance of their sub-blocks, this approach is not feasible. To address this, we design a heuristic block selection method named *Fast-Block-Select* to enhance

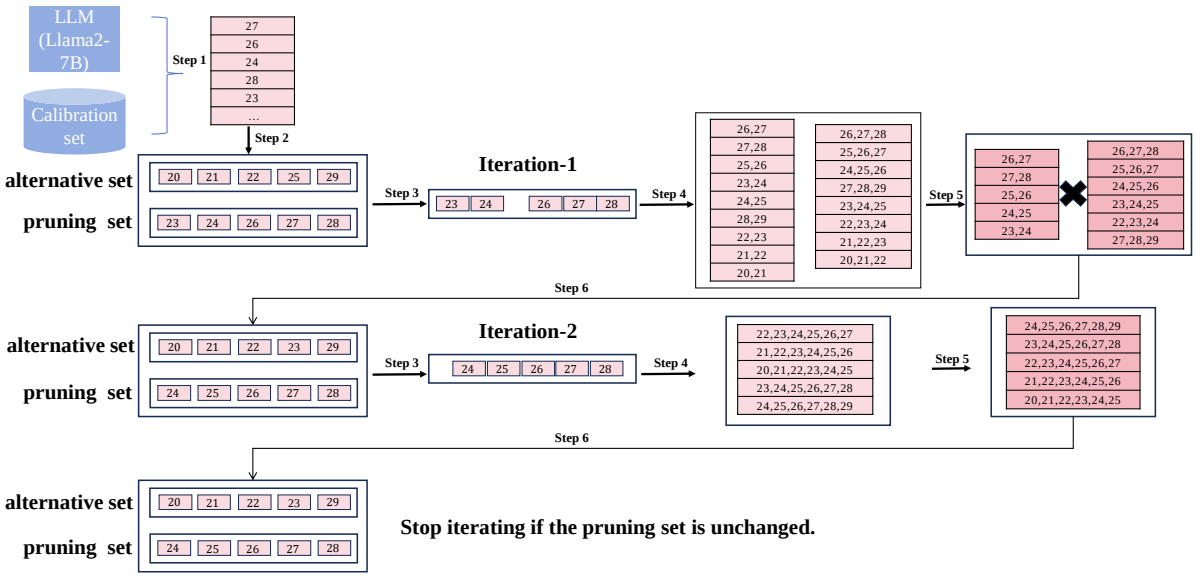

Figure 3: The process of pruning 5 blocks in the Llama2-7B model.

the speed of block selection. Figure 3 provides a detailed exposition of our process for pruning 5 blocks in the Llama2-7B model. When we need to prune N blocks, the process can be divided into the following steps:

① step1: We utilize a calibration set to obtain the importance of each independent block in the model and sort them in ascending order of importance.

② step2: We select the topN blocks based on the importance ranking to form the pruning set, and identify the M least important blocks outside the pruning set to form the alternative set, where M satisfies the following conditions:

$$M = \min(N, Total\ blocks - N) \quad (14)$$

③ step3: We categorize the blocks in the pruning set into groups to obtain contiguous blocks sets.

④ step4: For any contiguous block within the contiguous blocks set, we construct all possible contiguous block sets of corresponding lengths using blocks from both the pruning set and the alternative set. We estimate the importance of each contiguous block by summing the importance of its individual blocks, and then sort these blocks in ascending order of importance.

⑤ step5: For the topK contiguous blocks in each group, we calculate their actual *mutual information* (rather than using estimated values) and then rank them accordingly, where K satisfies the following conditions:

$$K = \min(\lfloor \log L \rfloor + 5, S) \quad (15)$$

Here, L represents the length of the corresponding contiguous block and S corresponds to the number of elements in the continuous block set. In Figure 3, L is 2, 3 and 5 respectively, with corresponding K values of 5, 6 and 5.

⑥ step6: 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.

We then use the pruning set from this iteration to repeat steps 3, 4, 5 and 6. This process continues until the selected blocks remain unchanged from one iteration to the next.

4 Experiments

4.1 Experimental Setup

Models. To demonstrate our method's effectiveness, we conducted experiments on popular open-source models: Llama2-7B (Touvron et al., 2023b), Llama2-13B, Qwen-7B (Bai et al., 2023), and Qwen-14B. These Transformer-based decoder-only models include Llama2, trained on over two trillion tokens, and Qwen, pre-trained on a 3TB dataset featuring multilingual content, mainly Chinese and English.

Benchmarks. To assess the impact of pruning on large language models, we evaluate using key benchmarks: Winogrande (ai2, 2019), PIQA (Bisk et al., 2020), WSC (Kocijan et al., 2020), WNLI (ai2, 2019), SST-2 (Socher et al., 2013), RTE (Poliak, 2020), QNLI (Wang et al., 2018), CB (Talmor et al., 2019), ARC-e (Yadav et al., 2019), and ARC-

c (Yadav et al., 2019). These cover NLP tasks like QA, text entailment, sentiment analysis, and reasoning. We also include a perplexity analysis on C4 (Raffel et al., 2020).

Baselines. To evaluate the effectiveness of our method, we compare the following pruning methods for large language models:

- **LLM-Pruner:** The method adopts structural pruning that selectively removes non-critical coupled structures based on gradient information, maximally preserving the majority of the LLM's functionality (Ma et al., 2023).
- **SliceGPT:** SliceGPT is a new post-training sparsification scheme which replaces each weight matrix with a smaller matrix, reducing the embedding dimension of the network (Ashkboos et al., 2024).
- **ShortGPT:** It is a straightforward pruning approach: layer removal, in which we directly delete the redundant layers in LLMs based on their BI scores (Men et al., 2024).

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 40GB memory. If the product of a model's total transformer blocks and target sparsity isn't an integer, we round up to decide the number of blocks to prune. We use two calibration sets, WikiText-2 (Merity et al., 2016) and Alpaca (Taori et al., 2023), varying their size and sequence length. For performance comparisons, we maintain consistent experimental settings, including the calibration set and pruning rate. We run experiments with errors multiple times and average the results.

4.2 Results of Pruning Blocks on different Calibration Sets

To assess the influence of different calibration sets on pruning results, we conduct experiments on Qwen-7B and Qwen-14B, pruning 5 and 7 blocks respectively. We test pruning blocks with various data sizes and sequence lengths on the WikiText-2 and Alpaca datasets. Despite these differing experimental conditions, we achieve consistent pruning results. This indicates that our method maintains stability across different datasets. More details can be found in Appendix D.

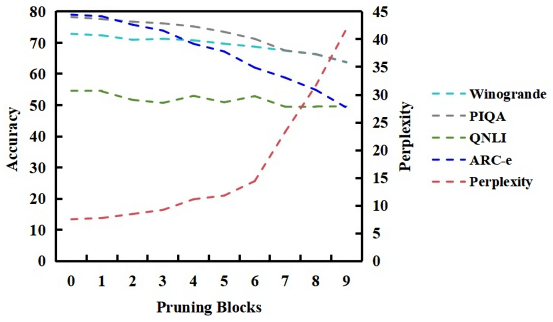

Figure 4: Performance of Llama2-7B with increasing pruning blocks.

4.3 Comparison with Existing Work

To validate the effectiveness of our proposed method, comparative experiments are conducted on Llama2 and Qwen, employing standard benchmarks and baselines commonly utilized in the assessment of large language models. The experimental results are shown in table 1,3. The results suggest that the models pruned via our proposed method have demonstrated enhanced overall performance when compared with the baseline methods, maintaining most of the large language model's capabilities.

4.4 Sensitivity Analysis of Different Pruning Block Quantities

In pursuit of a holistic evaluation of our pruning method under diverse pruning ratios, we execute an extensive experimental regimen on the Llama2-7B model, as delineated in Figure 4. The assessment is conducted by scrutinizing the accuracy and perplexity indices on a broad dataset encompassing multiple tasks. The experimental results corroborate that our pruning methodology sustains commendable performance across a range of pruning ratios.

4.5 Statistics of the Compressed Model

Table 2 presents the performance metrics of the Llama2-7B model utilized in our experiments, encompassing reduced forward params size, reduced params size, ratio of reduced to forward-reduced params size and latency. The results indicate that our method not only reduces model params but also more effectively diminishes forward propagation params. These assessments are conducted with the model in inference mode, processing input data with a length of 1,024 sentence.

Models	Methods	Ratio	Winogrande	PIQA	WSC	WNLI	SST-2	RTE	QNLI	CB	ARC-e	ARC-c	Avg.
Llama2-7B	Dense	0.00%	72.85	78.24	54.81	59.15	85.89	66.06	54.6	60.71	79	47.78	65.91
	SliceGPT	15.34%	63.06	67.03	**63.46**	45.07	52.64	59.57	50.63	42.86	55.26	34.56	53.41
	LLM-Pruner	15.30%	64.72	**76.06**	36.54	56.34	65.83	52.71	**51.20**	46.43	64.86	36.69	55.14
	ShortGPT	15.32%	67.72	71.6	36.54	43.66	49.2	53.43	50.19	39.29	65.66	**39.42**	51.67
	MI-PRUN	15.32%	**69.69**	73.5	**63.46**	**60.56**	**83.94**	**60.29**	50.91	**62.5**	**67.21**	**39.42**	**63.15**
Llama2-13B	Dense	0.00%	75.61	79.71	53.85	66.2	87.61	69.31	58.56	80.36	81.82	53.16	70.62
	SliceGPT	25.28%	70.72	63.17	44.23	43.66	51.15	52.71	50.58	41.07	65.03	33.79	51.61
	LLM-Pruner	25.35%	**71.74**	**73.07**	36.54	43.66	66.86	52.71	49.90	42.86	**66.08**	32.42	53.58
	ShortGPT	24.37%	64.17	71.93	58.65	59.15	50.57	68.95	49.83	57.14	50.17	**40.53**	57.11
	MI-PRUN	24.37%	59.59	72.36	**59.62**	**66.2**	**70.87**	**69.31**	**50.61**	**78.57**	60.02	40.27	**62.74**
Qwen-7B	Dense	0.00%	71.98	79.22	73.08	64.79	94.15	83.75	74.24	76.79	79.04	48.04	74.51
	ShortGPT	13.11%	67.4	73.01	**63.46**	**63.38**	93.12	**80.87**	**50.54**	60.71	62.96	35.84	65.13
	MI-PRUN	13.11%	**69.3**	**73.78**	**63.46**	61.97	**94.04**	71.84	**50.54**	**62.5**	**66.25**	**39.93**	**65.36**
Qwen-14B	Dense	0.00%	75.3	80.01	69.23	73.24	95.07	80.51	74.81	87.5	81.14	52.47	76.93
	ShortGPT	15.58%	67.4	70.18	36.54	**71.83**	94.15	80.51	49.46	**87.5**	63.72	39.33	66.06
	MI-PRUN	15.58%	**69.46**	**72.25**	**81.73**	**71.83**	**95.41**	**80.87**	**69.89**	83.93	**67.38**	**43.69**	**73.64**

Table 1: Comparison of pruning methods on multiple natural language benchmarks.

Methods	Reduced Forward Params size (MB)	Reduced Params size (MB)	Ratio	Latency (ms)
SliceGPT	3281.53	1358.95	41.41%	287.35
LLM-Pruner	2061.34	714.37	34.66%	341.55
MI-PRUN	2023.83	1038.09	51.29%	221.14

Table 2: Statistics of the compressed models.

Models	Methods	Perplexity
Llama2-7B	Dense	7.53
	SliceGPT	12.49
	LLM-Pruner	13.70
	ShortGPT	15.93
	MI-PRUN	**11.80**
Llama2-13B	Dense	6.99
	SliceGPT	14.35
	LLM-Pruner	18.87
	ShortGPT	21.77
	MI-PRUN	**13.35**

Table 3: Comparison of pruning methods on perplexity.

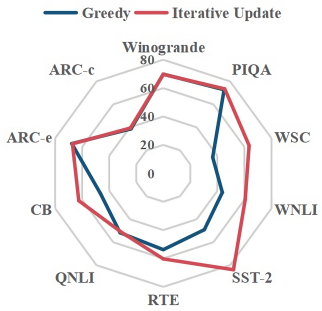

Figure 5: Performance comparison between Greedy Strategy and Iterative Update.

4.6 Ablation Study

Iterative Update Blocks. We delve into the impact of incorporating *Data Processing Inequality* within the *MI-PRUN* method, meticulously comparing two pruning strategies to assess its signifilnce. Initiating with an evaluation of *mutual information* for individual blocks, the experiment employ a greedy strategy to prune based on the *MI* values. Advancing from this, the *MI-PRUN* method is applied, integrating *DPI* to compute the *MI* across contiguous blocks, shedding light on their collective impact on the model's overall performance. An iterative refinement algorithm is harnessed to enhance the selection process, ensuring that the pruning outcomes were both exhaustive and accurate. The results, as depicted in Figure 5, contrast the model's accuracy under the two strategies, thereby vividly illustrating the notable influence of considering contiguous blocks during pruning on the model's enhanced performance.

Fast-Block-Select. We aim to evaluate two distinct block selection methods: one is the brute-force approach that calculates the importance of all individual and contiguous blocks, while the other is our heuristic-based *Fast-Block-Select* algorithm. The primary focus of the experiment is to compare the computational demand, specifically the number of required solution attempts, between these two methods. The brute-force approach must evaluate all possible combinations of contiguous blocks, a process that is exceedingly complex and resource-intensive. In stark contrast,

the *Fast-Block-Select* algorithm employs a heuristic strategy for block selection, substantially enhancing the selection efficiency. As depicted in Table 4, the *Fast-Block-Select* algorithm not only identifies the blocks slated for pruning but also significantly reduces the number of computations, thereby validating its practicality and superiority in real-world applications.

Models	Methods	MI Calculations
Llama2-7B	Brute Force	150
	Fast-Block-Select	46
Llama2-13B	Brute Force	355
	Fast-Block-Select	46

Table 4: Pruning efficiency of Brute Force and Fast-Block-Select.

4.7 Case Study

We present some examples of sentences generated by the model before and after pruning with our method. It is evident that the sentences produced by the compressed model are comparable in quality to those generated by the original model. These sentences exhibit fluency, relevance, and informativeness in relation to the given topics, demonstrating that our pruning method successfully maintains the model's ability to generate high-quality text while reducing its complexity. This further confirms the effectiveness of the method and its minimal impact on model performance in practical applications. More detailed results can be found in Appendix B.

4.8 Pruning Overhead Analysis

In our pruning approach, we calculate the *mutual information* between intermediate layer hidden states to evaluate inter-layer information flow and redundancy. To minimize GPU memory use and prevent data overload, we swiftly move each hidden state from GPU to CPU post-retrieval. This enables the GPU to focus on other tasks without interruption. Utilizing the CPU's multi-core capabilities, we efficiently compute MI, aided by optimized algorithms from libraries like scipy (Virtanen et al., 2020), torchmetrics (Detlefsen et al., 2022), and scikit-learn (Pedregosa et al., 2011), which are designed to exploit CPU parallelism for rapid *MI* calculations, thus supporting our pruning method effectively.

5 Discussion

In this study, we concentrate on pruning intermediate layers of large language models using skip-connections, replacing operations with identity matrices to streamline network structure and reduce computation without affecting performance. Our method's broad applicability extends to various models, including those with skip-connections like ResNet. It also allows for fine-grained pruning, enabling flexible optimization and enhanced efficiency in Transformer 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. This enhancement in speed is versatile, applicable in a multitude of implementation contexts, positioning *MI-PRUN* as a viable solution for practical LLM service scenarios.

7 Limitations

Although our paper focuses on measuring parameter redundancy from the perspective of depth, there are also well-established algorithms for neuron pruning in the width dimension. Determining which optimization method (width or depth) is more useful for a specific model, as well as integrating these two methods for easy hardware deployment, remains highly challenging. The complexity of model architectures and the diversity of task requirements make it difficult to universally determine the optimal optimization strategy. The implementation of joint optimization strategies also faces limitations. For example, the sequential approach of first pruning width and then depth may not always be optimal, and dynamically adjusting the pruning ratios increases the complexity and computational cost of the training process. Additionally, hardware deployment poses challenges, as different hardware platforms have varying preferences for

depth and width optimization, and re-adapting optimized models to hardware can involve additional development and debugging costs. Existing automated tools and frameworks, while providing some optimization support, still have limitations in terms of their universality and optimization accuracy, and may not fully adapt to specific LLM architectures and task requirements. Therefore, future research needs to further explore how to better integrate depth and width optimization methods and develop more efficient automated tools and hardware adaptation strategies to achieve efficient deployment of large language models.

References

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

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*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

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. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

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.

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.

Nicki Skafte Detlefsen, Jiri Borovec, Justus Schock, Ananya Harsh Jha, Teddy Koker, Luca Di Liello, Daniel Stancl, Changsheng Quan, Maxim Grechkin, and William Falcon. 2022. Torchmetrics-measuring reproducibility in pytorch. *Journal of Open Source Software*, 7(70):4101.

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.

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.

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.

Byungmo Kim, Jaewon Oh, and Cheonhong Min. 2022. Investigation on applicability and limitation of cosine similarity-based structural condition monitoring for gageocho offshore structure. *Sensors*, 22(2):663.

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*.

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

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*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

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.

Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python journal of machine learning research. *Journal of machine learning research*, 12:2825–2830.

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*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

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.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Stanford alpaca: An instruction-following llama model.

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.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. 2020. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272.

Janett Walters-Williams and Yan Li. 2009. Estimation of mutual information: A survey. In *Rough Sets and*

Knowledge Technology: 4th International Conference, RSKT 2009, Gold Coast, Australia, July 14-16, 2009. Proceedings 4, pages 389–396. Springer.

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*.

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*.

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

A Evaluation 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., 2020) 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.

B Pruning Effects on Generated Sentences

We meticulously carry out a series of comparative experiments to thoroughly assess the impact of our proposed pruning technique on the sentence generation quality of large language models before and after the pruning process. The design of these experiments enables us to precisely quantify and compare the model's performance at different stages. The results, comprehensively presented in table B, reveal an encouraging phenomenon: despite undergoing pruning, the model's performance remains unaffected. In fact, the model continues to demonstrate exceptional capabilities in handling complex linguistic tasks, consistently producing responses that are not only of superior quality but also fluent and natural. These findings not only validate the scientific rationality of our approach but also provide strong empirical evidence for the potential to maintain or even enhance the performance of language models while reducing computational resource consumption in future applications.

C Mutual Information Value of Llama2-7B

Table 6 and figure 6 showcase our meticulous examination and detailed presentation of the mutual information values pertaining to the 15 least influential blocks within the Llama2-7B model, considering both isolated and contiguous blocks with lengths spanning from 2 to 5. This insightful anal-

Original Model	Pruned Model
Assess the impact of the Industrial Revolution on society. The impact of the Industrial Revolution on the environment was significant. The shift to industrial production and transportation led to increased pollution and damage to the environment ...	*Assess the impact of the Industrial Revolution on society.* The Industrial Revolution, which began in Britain in the late 18th century and spread to other parts of the world over the next century, had a profound impact on society. It transformed the ...
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 interface: Create a visual design for the app, including the layout, color scheme ...	*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 for beginners, as it is easier to learn and use. 3. Set up your development ...
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 data, learn from it, and make decisions or predictions ...	*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, including healthcare, finance, transportation, and ...
Describe the process of fermentation in bread making. Fermentation is a process that occurs when microorganisms such as yeast or bacteria convert sugars into alcohol or acids. In bread making, fermentation is used to convert the starches in flour into sugars, which are then converted into carbon dioxide gas by yeast. This process causes the dough to rise ...	*Describe the process of fermentation in bread making.* Fermentation is an important stage in bread making, which involves the conversion of sugars in the bread ingredients into alcohol and carbon dioxide gas. The fermentation stage is crucial for creating the characteristic flavor and texture of bread. Here are the key stages of fermentation in bread making ...
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 ...
Describe the process of DNA replication. DNA replication is the process by which a cell makes an exact copy of its DNA before cell division. This process is essential for the transmission of genetic information from one generation of cells to the next. Here is a step-by-step explanation of the process of DNA replication: 1. Initiation: The process of DNA replication begins with the unwinding of the double helix ...	*Describe the process of DNA replication.* DNA replication is the biological process that occurs during which the genetic material or DNA copy is produced. DNA replication is a complex and complex biological complex that involves many proteins and enzymes. DNA replication is divided into several stages, including:1. Initiation: The first step in DNA replication is the recognition of the DNA molecule by ...

Table 5: Some examples of generations obtained with the original model and the pruned model.

ysis affords us a transparent perspective on the model's internal information dynamics and, crucially, identifies components that could potentially be targeted for elimination during model pruning. The quantification of mutual information across these blocks allows us to discern the varying degrees of contribution each model segment makes to the overall performance, thereby laying down a solid scientific foundation for the strategic refinement of the model's architecture and the consequent improvement in operational efficiency.

D Results of Pruning Blocks on different Calibration Sets

To evaluate the impact of various calibration datasets on the pruning outcomes, we perform experiments on the Qwen-7B and Qwen-14B models, targeting the reduction of 5 and 7 blocks, respectively. We assess the pruning of blocks across a spectrum of data sizes and sequence lengths using the WikiText-2 and Alpaca datasets, as specified in tables 8 and 7. Despite the variation in experimental setups, we consistently achieved uniform pruning outcomes. This consistency suggests that our approach remains stable across diverse datasets.

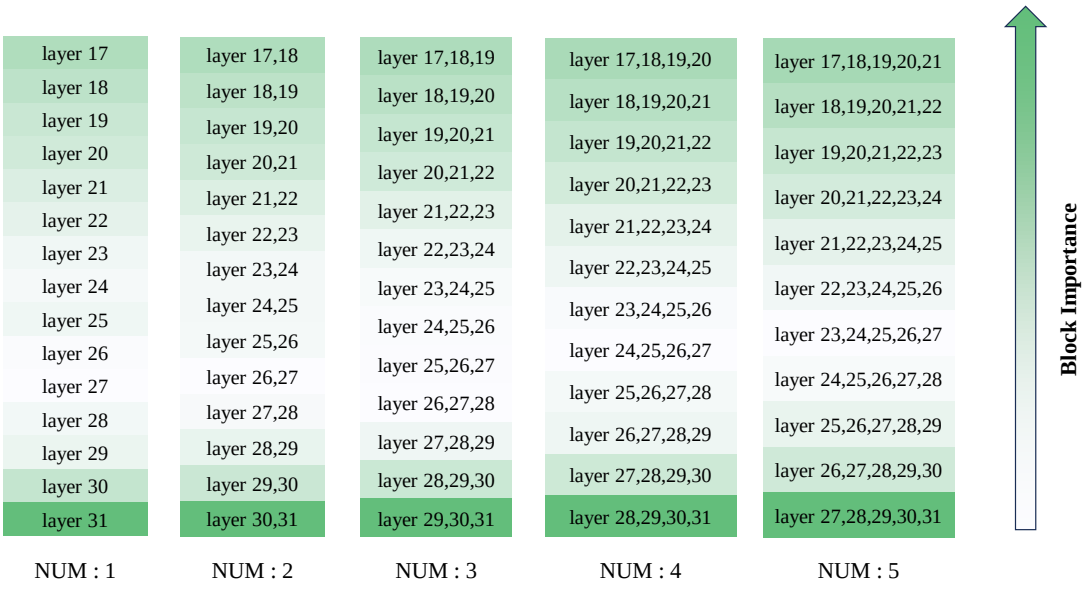

Figure 6: The results derived from the application of *mutual information* to assess the importance of individual and contiguous blocks within the Llama2-7B model.

NUM : 1		NUM : 2		NUM : 3		NUM : 4		NUM : 5	
layer 17	0.87498162	layer 17,18	0.63229072	layer 17,18,19	0.50139592	layer 17,18,19,20	0.40852264	layer 17,18,19,20,21	0.40852264
layer 18	0.95587192	layer 18,19	0.69475307	layer 18,19,20	0.54578185	layer 18,19,20,21	0.44966459	layer 18,19,20,21,22	0.44966459
layer 19	1.02724996	layer 19,20	0.7396465	layer 19,20,21	0.58558662	layer 19,20,21,22	0.48767546	layer 19,20,21,22,23	0.48767546
layer 20	1.06539699	layer 20,21	0.77436899	layer 20,21,22	0.62218076	layer 20,21,22,23	0.528129	layer 20,21,22,23,24	0.528129
layer 21	1.13227833	layer 21,22	0.84088689	layer 21,22,23	0.6884778	layer 21,22,23,24	0.58341424	layer 21,22,23,24,25	0.58341424
layer 22	1.1889242	layer 22,23	0.90008803	layer 22,23,24	0.73684974	layer 22,23,24,25	0.61361306	layer 22,23,24,25,26	0.61361306
layer 23	1.25624859	layer 23,24	0.94790796	layer 23,24,25	0.7637615	layer 23,24,25,26	0.64457198	layer 23,24,25,26,27	0.64457198
layer 24	1.28686095	layer 24,25	0.94960353	layer 24,25,26	0.77423227	layer 24,25,26,27	0.65406463	layer 24,25,26,27,28	0.65406463
layer 25	1.2529082	layer 25,26	0.95379755	layer 25,26,27	0.77904016	layer 25,26,27,28	0.63289751	layer 25,26,27,28,29	0.63289751
layer 26	1.31557265	layer 26,27	0.9882906	layer 26,27,28	0.78095716	layer 26,27,28,29	0.61478706	layer 26,27,28,29,30	0.61478706
layer 27	1.32509811	layer 27,28	0.96518432	layer 27,28,29	0.73501047	layer 27,28,29,30	0.51409194	layer 27,28,29,30,31	0.51409194
layer 28	1.27117968	layer 28,29	0.89805039	layer 28,29,30	0.60309149	layer 28,29,30,31	0.19466056	-	-
layer 29	1.22813231	layer 29,30	0.75716348	layer 29,30,31	0.22885524	-	-	-	-
layer 30	0.99682464	layer 30,31	0.27271845	-	-	-	-	-	-
layer 31	0.41455725	-	-	-	-	-	-	-	-

Table 6: The mutual information of individual and contiguous blocks within the Llama2-7B model.

Models	Calibration set	Sequence Length	Results of pruning blocks
Qwen-7B	WikiText-2	512 1024 2048	[24, 25, 26, 27, 28]
	Alpaca	512 1024 2048	[24, 25, 26, 27, 28]
Qwen-14B	WikiText-2	512 1024 2048	[29, 30, 31, 32, 33, 34, 35]
	Alpaca	512 1024 2048	[29, 30, 31, 32, 33, 34, 35]

Table 7: Results of pruning blocks with different sequence lengths on various calibration sets.

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 8: Results of pruning blocks with different sizes on various calibration sets.