FINERCUT: Finer-grained Interpretable Layer Pruning for Large Language Models

Yang Zhang [*1] **Yawei Li** [*2,3] **Xinpeng Wang** [2,3] **Qianli Shen** [1]
Barbara Plank [2,3] **Bernd Bischl** [2,3] **Mina Rezaei** [2,3] **Kenji Kawaguchi** [1]
[1]National University of Singapore [2]LMU Munich
[3]Munich Center for Machine Learning (MCML) [*] Equal contribution

Abstract

Overparametrized transformer networks are the state-of-the-art architecture for Large Language Models (LLMs). However, such models contain billions of parameters making large compute a necessity, while raising environmental concerns. To address these issues, we propose FINERCUT, a new form of fine-grained layer pruning, which in contrast to prior work at the transformer block level, considers all self-attention and feed-forward network (FFN) layers within blocks as individual pruning candidates. FINERCUT prunes layers whose removal causes minimal alternation to the model's output—contributing to a new, lean, interpretable, and task-agnostic pruning method. Tested across 9 benchmarks, our approach retains 90% performance of Llama3-8B with 25% layers removed, and 95% performance of Llama3-70B with 30% layers removed, all without fine-tuning or post-pruning reconstruction. Strikingly, we observe intriguing results with FINERCUT: 42% (34 out of 80) of the self-attention layers in Llama3-70B can be removed while preserving 99% of its performance—without additional fine-tuning after removal. Moreover, FINERCUT provides a tool to inspect the types and locations of pruned layers, allowing to observe interesting pruning behaviors. For instance, we observe a preference for pruning self-attention layers, often at deeper consecutive decoder layers. We hope our insights inspire future efficient LLM architecture designs.

1 Introduction

Large language models (LLMs) have shown impressive performance improvement in recent years and have exhibited emergent abilities [31, 7, 41, 42, 1, 3, 35]. The success of LLMs lies in their scale with billions of parameters and their pretraining on millions of tokens [22]. Nevertheless, deploying an LLM usually needs multiple GPUs and the huge amount of parameters induces long latencies for an LLM to complete its computation. These limitations have raised significant sustainability concerns [32]. In response, there is an ongoing quest to enhance the efficiency of LLMs, such as model distillation[21, 17, 24, 39, 40], model quantization[46, 5, 47, 44, 8], and model pruning[26, 13, 18].

Pruning reduces the size of an LLM by removing components while aiming to keep performance. Pruning methods can be categorized into structured pruning and unstructured pruning. Unstructured pruning, which removes connections between neurons, can often preserve performance effectively but requires specialized hardware to accelerate the pruned model. In contrast, structured pruning targets the removal of entire neurons, attention heads, channels, and layers, offering intrinsic speedup without the need for specific hardware. However, some structured pruning methods impose specific constraints during pruning. For instance, several works [4, 14, 27, 43] enforce the same sparsity across all layers, implicitly assuming equal importance among the layers—a premise that is not necessarily true. Besides, recent approaches [28, 16] selectively remove transformer blocks, treating the attention layer and feed-forward network (FFN) within the same transformer block as a whole. This constraint

(a) Overview of our iterative layer pruning algorithm. We repeat step 1-3 until designated amount of layers are removed.

(b) Overview of pruned layers at 25% layer pruning rate.

(c) We observe three major pruning behaviors.
From left to right: Remove a transformer block; Merge two transformer blocks into one; One attention associated with many FFN layers.
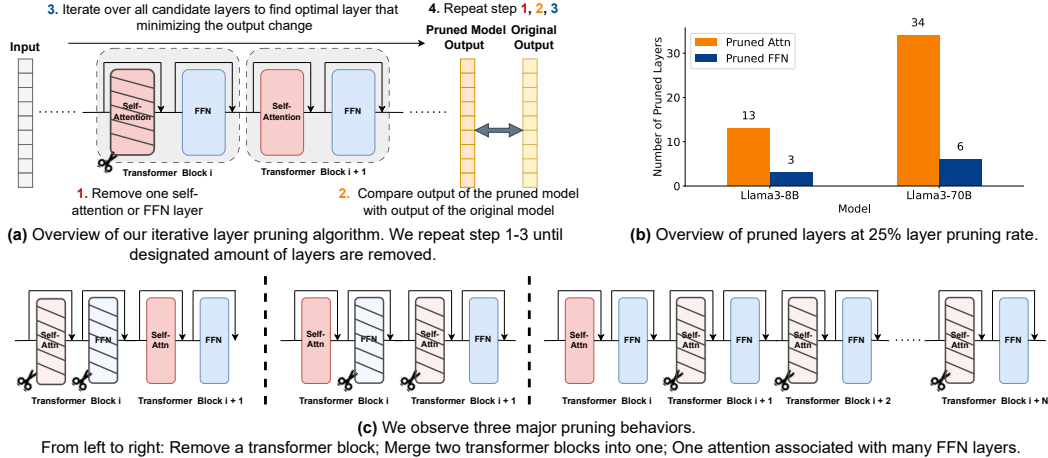
Figure 1: **(a)**: Overview of FINERCUT. FINERCUT iteratively examines candidate attention and FFN layers to find the next pruning target that minimizes output discrepancy compared to the original model. **(b)**: Overview of pruned layers in type. More attention layers are removed than FFN layers. **(c)**: Three major pruning behaviors we observed. Apart from pruning a transformer block or merging multiple transformer blocks to one transformer block through pruning, FINERCUT tends to remove attention layers in consecutive transformer blocks.

presumes similar and interdependent importance between attention and FFNs. Although these constraints simplify the pruning problem, they introduce rigidity when searching for candidates, potentially limiting the flexibility needed to achieve the best pruning outcomes.

In this work, we introduce FINERCUT, a flexible and effective layer pruning method that treats self-attention and FFN layers as separate pruning candidates. This approach offers a finer granularity compared to previous layer pruning methods. Another significant distinction lies in how we assess the impact of layers. Unlike previous layer pruning methods that measure the similarity between layer input and output, focusing on the local effects of the layers, our method evaluates the global impact by identifying layers *whose removal causes minimal alteration to the model's output*. To identify which layers to prune, we employ an iterative algorithm (Fig. 1 (a)). The algorithm not only evaluates individual layers but also considers the interdependencies among them, where the importance of remaining layers may shift following the removal of others. Evaluations on multiple tasks across various LLMs demonstrate that our pruning technique surpasses existing baselines and better preserves the capabilities of the original models.

Our contribution: (i) We propose FINERCUT, a novel layer pruning method that aims to reduce the computation of LLMs by treating self-attention and FFN layers as individual pruning candidates. (ii) We introduce a new formulation in model pruning aimed at minimizing the pruning effect on the model's output, measured by the shift in the predictive distribution. This formulation considers the characteristics of the entire model instead of only a target layer. Furthermore, it is task-agnostic. (iii) Compared with baseline approaches on various models and tasks, our model can effectively reduce the amount of computation of an LLM while better preserving its zero-shot ability on many tasks.

2 Related works

Unstructured pruning of LLMs: The most primitive unstructured pruning is through magnitude-based removal of weights by setting weights with small values to zero, which performs poorly on LLMs [14]. One better way is through Optimal Brain Surgeon (OBS) [18, 26] that systematically removes weights that have the least impact on the loss function. However, OBS is impractical for large models such as LLM due to the complexity of calculating the Hessian. Optimal Brain Compression [13] is a variant that decomposes the full-model pruning into per-layer unstructured pruning subproblems to reduce the computation of OBS and makes OBS applicable to LLMs. Furthermore, Singh and Alistarh [34] approximate Hessian matrix through block Fischer matrix. SparseGPT [14] formalizes the layer-wise unstructured pruning as a layer output reconstruction

problem and solves it iteratively. Sun et al. [36] proposed to prune connections based on both weights and activations that achieved good performance without needing post-pruning reconstruction.

Structured pruning of LLMs: Structured pruning methods do not rely on special hardware realization and can intrinsically speed up a model. LayerDrop [11], Block Pruning [25], and ShearedLlama [43] introduce structured pruning as a regularization during pre-training or fine-tuning. Another line of research is to perform structured pruning after a model is already trained. SliceGPT [4] removes rows and columns of a weight matrix, equivalent to reducing a layer's input/output dimension. LLM-Pruner [27] removes parts of LLMs such as neurons, attention heads, or channels based on an importance score consisting of gradients of connected weights. Recently, more works have focused on reducing layers of LLMs. LaCo [45] selects and merges multiple layers into one layer. ShortGPT [28] removes decoder layers of LLMs based on cosine similarity between the input and output of decoder layers. Gromov et al. [16] prunes a block of consecutive decoder layers according to angular distances between the block input and block output. Compared to other layer pruning methods, ours first proposes to look into decoder layers and treat self-attention layers and FFN layers separately as independent components to be pruned.

3 Method

3.1 Preliminaries of LLMs

Decoder-only LLMs generally comprise an embedding layer E, succeeded by L transformer decoder layers $H_1, H_2, \ldots, H_L$, and conclude with a prediction head layer C. Each decoder layer H_l incorporates an attention layer and an FFN layer. Consider an input prompt $\boldsymbol{x} \in |\mathcal{V}|^N$, where $\mathcal{V}$ represents the vocabulary and N is the length of the prompt sequence. The LLM, denoted by f, processes the input as follows: Initially, $\boldsymbol{x}$ is mapped into a hidden space as $\boldsymbol{h}_0 = E(\boldsymbol{x}) \in \mathbb{R}^{N \times d}$. Subsequently, the hidden state $\boldsymbol{h}_0$ is passed through the decoder layers: $\boldsymbol{h}'_l = \mathsf{Attn}(\boldsymbol{h}_{l-1}) + \boldsymbol{h}_{l-1}$, and $\boldsymbol{h}_l = \mathsf{FFN}(\boldsymbol{h}'_l) + \boldsymbol{h}'_l$ for $l = 1, 2, \ldots, L$. This formulation intentionally omits positional embeddings and normalization steps in Attn and the normalization in FFN to enhance clarity. Next, the head layer C predicts the logits $C(\boldsymbol{h}_L) = [\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \ldots, \boldsymbol{z}^{(N)}]$, where $\boldsymbol{z}^{(i)} \in \mathbb{R}^{|\mathcal{V}|}$ represents the predicted logits for the $(i+1)$-th output token. Although next-word prediction can be implemented in an auto-regressive manner, this mode of prediction is not employed during model pruning. Instead, we concentrate on the non-auto-regressive prediction approach.

3.2 Formulation of structured pruning for LLMs

When pruning the model, we selectively drop some layers from the $2L$ Attn or FFN layers. The pruning layer selection can be parametrized by an *indicator vector* $\boldsymbol{m} \in \{0, 1\}^{2L}$, where a value of 1 signifies that the parameters within the corresponding Attn or FFN layer should be dropped. Then, the layer pruning ratio r can mathematically described as: $r = (\boldsymbol{m}^\top \mathbf{1})/(2L)$, where $\mathbf{1} = [1, \ldots, 1]^\top$.

The primary objective of pruning, for a specific r, is to identify the layers whose removal minimally impacts the model performance. Although performance evaluations across different tasks utilize diverse metrics, these metrics are intrinsically dependent on the model's output. Therefore, to minimize performance degradation, the output of the pruned model $f(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{m}) = [\tilde{\boldsymbol{z}}^{(1)}, \ldots, \tilde{\boldsymbol{z}}^{(N)}]$ should closely approximate the original model output $f(\boldsymbol{x}; \boldsymbol{\theta}) = [\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(N)}]$. This leads to the following optimization formulation for an optimal pruning layer selection $\boldsymbol{m}^*$:

$$\boldsymbol{m}^* = \arg\min_{\boldsymbol{m}} \quad \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x})} \left[\frac{1}{N} \sum_{i=1}^{N} q\left(\boldsymbol{z}^{(i)}, \tilde{\boldsymbol{z}}^{(i)}\right) \right] \tag{1}$$
$$\text{s.t.} \quad \boldsymbol{m}^\top \mathbf{1} = 2L \cdot r,$$

where $q : \mathbb{R}^{|\mathcal{V}|} \times \mathbb{R}^{|\mathcal{V}|} \to \mathbb{R}_{\geq 0}$ is a metric used to measure changes in the model output. We can perform Monte-Carlo simulation on $\boldsymbol{x}$ to get an approximation of the expectation. Although the optimization problem in Eq. (1) appears straightforward, solving it is computationally challenging. A brute-force search for the global optimum would have *exponential* computational complexity, which is infeasible to compute for a large L. Furthermore, the optimization parameter is a binary vector, making the gradient-based optimization not applicable. We discuss an iterative algorithm in the subsequent section to address these challenges.

3.3 Iterative search algorithm as an efficient and approximate solver

In this work, we utilize an iterative search algorithm as an approximation method, which reduces the complexity to $O(L^2)$. This algorithm progressively ablates one layer at a time, selecting the layer whose removal least affects the model output. This procedure is iterated upon, layer by layer, until the desired layer pruning ratio r is achieved. The specifics of this method are detailed in Algorithm 1.

Algorithm 1 Iterative layer pruning

$\boldsymbol{m} \leftarrow [0, 0, \ldots, 0]$
while r is not reached: **do**
    $Q_{\min} \leftarrow \infty$, and $l_{\min} \leftarrow 0$ ▷ Initialize the change at output and the best pruning layer.
    for l in argwhere($\boldsymbol{m}_l \neq 1$) **do**   ▷ Loop over the remaining layers.
        $\boldsymbol{m}_l \leftarrow 1$ ▷ Try pruning layer l; update $\boldsymbol{m}$ temporarily.
        $Q \leftarrow$ EvalOutputChange($f, \boldsymbol{\theta}, \boldsymbol{m}$)   ▷ The change at output after the removal of layer l.
        if $Q \leq Q_{\min}$ **then**
            $Q_{\min} \leftarrow Q$, and $l_{\min} \leftarrow l$ ▷ Update the best pruning layer to layer l.
        end if
        $\boldsymbol{m}_l \leftarrow 0$ ▷ Reset $\boldsymbol{m}$ to previous state; prepare for ablating the next layer.
    end for
    $\boldsymbol{m}_{l_{\min}} \leftarrow 1$ ▷ Prune the optimal single layer at the current step.
end while

The function EvalOutputChange($f, \boldsymbol{\theta}, \boldsymbol{m}$) evaluates the output change between the original model and the pruned model with $q\left(\boldsymbol{z}^{(i)}, \tilde{\boldsymbol{z}}^{(i)}\right)$ in Eq. (1). We discuss the choice of metric functions for measuring the output change in detail in Section 3.4. Algorithm 1 provides an efficient approximation instead of solving the optimization problem exactly. Nevertheless, the algorithm can be time-consuming, as modern LLMs usually have many layers. For a single pruning step in Algorithm 1, we still need to perform many forward passes on all remaining layers to find the pruning target. To further reduce the runtime, we adapt a finding from prior work [16] and consider only a subset of the last 60% layers when the layer pruning ratio does not exceed 40%. For layer pruning ratios larger than 40%, we consider all layers as pruning candidates. Subsequently, we always consider only 60% of all layers at any pruning step, hence further reducing the runtime of our pruning method.

3.4 Choices of metric functions

To evaluate changes in the model output, we consider multiple distance metrics for $q(\cdot, \cdot)$ in Eq. (1).

Angular distance: Angular distance measures the distance between two vectors in hypersphere coordinates. This metric is used in several prior works [28, 16]. Formally, we leverage the cosine-similarity to measure the angular distance:

$$q(\boldsymbol{z}, \tilde{\boldsymbol{z}}) = \arccos\left(\frac{\boldsymbol{z}^T \tilde{\boldsymbol{z}}}{\|\boldsymbol{z}\|_2 \|\tilde{\boldsymbol{z}}\|_2}\right) \tag{2}$$

Euclidean distance: To improve the pruning performance, we can apply more strict distance metrics. Since we can consider the model output as vectors in high dimensional space, it is natural to measure the Euclidean distance between two outputs. Formally, we have:

$$q(\boldsymbol{z}, \tilde{\boldsymbol{z}}) = \sqrt{\sum_{j=1}^{|\mathcal{V}|} (\boldsymbol{z}_j - \tilde{\boldsymbol{z}}_j)^2} \tag{3}$$

Statistical distance: Formally, given that $\boldsymbol{z}$ and $\tilde{\boldsymbol{z}}$ represent logits, we apply the softmax function to derive the predicted probability distributions $\boldsymbol{s} = \mathsf{softmax}(\boldsymbol{z})$ and $\tilde{\boldsymbol{s}} = \mathsf{softmax}(\tilde{\boldsymbol{z}})$. We then use a statistical distance to quantify the discrepancy between these two distributions. There are many options for statistical distance measurement. In this work, we choose the Jensen-Shannon divergence, a variant of Kullback–Leibler divergence with symmetric property:

$$q(\boldsymbol{z}, \tilde{\boldsymbol{z}}) = \frac{1}{2}\left[\mathbb{D}_{KL}\left(\boldsymbol{s} \,\|\, \frac{1}{2}(\boldsymbol{s} + \tilde{\boldsymbol{s}})\right) + \mathbb{D}_{KL}\left(\tilde{\boldsymbol{s}} \,\|\, \frac{1}{2}(\boldsymbol{s} + \tilde{\boldsymbol{s}})\right)\right] \tag{4}$$

where $\mathbb{D}_{KL}$ represents the Kullback–Leibler divergence, defined as $\mathbb{D}_{KL}(\boldsymbol{u}, \boldsymbol{v}) = \sum_j \boldsymbol{u}_j \log(\boldsymbol{u}_j / \boldsymbol{v}_j)$ for discrete random variables, with vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ representing discrete probability distributions.

Table 1: Performance on Llama3-8B at 25% layer pruning ratio. "Average" shows the mean accuracy across 8 reasoning tasks. Acos, Norm, and JS are our approaches with different measures. Our pruning method outperforms baselines on all evaluated tasks. The * symbol denotes that we set sparsity to be 25% instead of layer pruning ratio. Evaluation: Wikitext is ppl, other tasks are accuracy.

Model	Speedup	Wikitext	BoolQ	ARC-C	ARC-E	WG	HS	PIQA	OBQA	MMLU	Average
Original	-	11.31	80.8	53.1	76.9	71.7	68.8	81.3	45.0	66.2	68.0
ShortGPT	1.31×	10^3	37.8	31.2	37.7	53.4	37.9	62.6	28.2	35.7	40.6
DeeperLayers	1.31×	10^3	37.8	31.2	37.7	53.4	37.9	62.6	28.2	35.7	40.6
SliceGPT*	1.29×	45.65	37.6	28.8	45.8	59.5	51.9	58.5	33.6	24.2	42.5
Acos	1.30×	23.57	72.7	35.8	62.2	65.9	60.1	**74.8**	37.8	41.2	56.3
Norm	1.30×	38.14	**78.1**	37.5	63.6	**67.6**	60.7	73.8	**39.0**	46.7	58.4
JS	1.30×	**23.06**	76.0	**38.2**	**65.6**	65.2	**60.5**	74.2	38.2	**62.8**	**60.1**

Table 2: Model statistics of a pruned Llama3-70B model using FINERCUT (JS).

Layer pruning ratio	MACs	Time (per sample)	# parameters	Memory
0%	734690 GMacs	9.50 s	70.5 B	134 GB
25%	587756 GMacs	7.42 s	61.1 B	116 GB
50%	386504 GMacs	5.09 s	39.6 B	76 GB
75%	202535 GMacs	2.39 s	22.0 B	43 GB

4 Experiments

4.1 Experiment setup

Models: We choose some member models of Llama family[37, 38] including Llama2-7B, Llama2-13B, Llama3-8B, and Llama3-70B. In addition, we include Mixtral-8x7B [23] in our experiments to assess the pruning performance on Mixture-of-Experts (MoE) models [12]. Due to the page limit, we present the results of Llama2 models in Appendix E.1 and Appendix E.2.

Benchmarks: We conduct our experiment on two types of tasks: generative tasks and zero/few-shot tasks. We evaluate the perplexity on WikiText2 [29]. Furthermore, we evaluate accuracy on commonsense reasoning datasets: BoolQ [9], PIQA [6], HellaSwag [48], WinoGrande [33], ARC-easy/challenge [10],OpenbookQA [30] and MMLU [19, 20]. More details are in Appendix C.

Baselines: We include other four structured pruning methods in our evaluation: LLM-Pruner [27], SliceGPT [4], ShortGPT [28], and the pruning method discussed in Gromov et al. [16], which we refer it as *DeeperLayers* in the following text.

4.2 Main result

Table 1 show the performance of Llama3-8B and Llama3-70B when 25% of the layers are pruned using different pruning methods. On Llama3-8B, FINERCUT outperforms all other baselines and achieves 60% mean accuracy on 8 different reasoning tasks, which is around 20% higher than the best prior work method. Furthermore, with 25% of the layers removed, our pruned model still retains 88% of the dense model's performance. On Llama3-70B, FINERCUT retains 98% of the original performance. We also evaluate the language modeling ability of pruned models with the WikiText2 dataset. According to the perplexity results, FINERCUT better preserves the text generation ability than other baselines. Based on the performance drop on Llama3-70B and Llama3-8B, it is worth mentioning that a larger model is also an easier pruning target. This suggests more redundancy on larger models despite their superior performance compared to smaller models.

To gauge the performance of pruning methods at different layer pruning ratios, we summarize our results in Fig. 2. Here, we only present our results with JS divergence. From Fig. 2 we see that our method performs better on zero-shot and few-shot reasoning tasks in the Llama3 model family. For Llama3-8B, our method still retains the model performance at 25% layer pruning ratio, while other baselines have already collapsed. Our model performs substantially better than other baselines and can have more than 10 times smaller perplexity on the WikiText2 generation task. In contrast, other methods like ShortGPT exhibit a stronger performance degradation on language generation
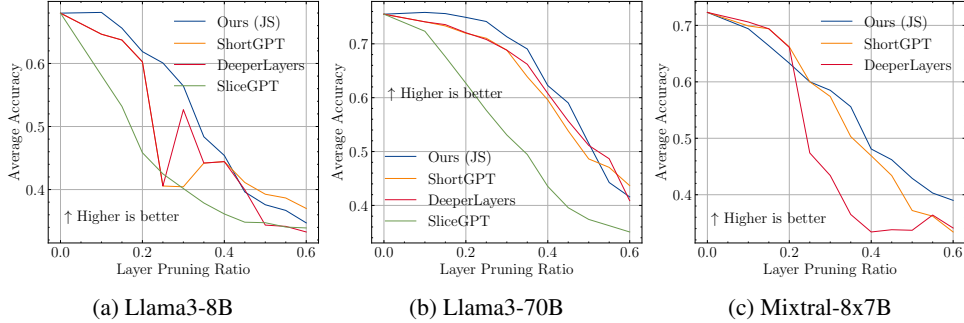
Figure 2: Average zero/few-shot performance at different layer pruning ratios. We omit SliceGPT in (c) because it does not support MoE.

Table 3: Time and memory consumption for running different pruning methods on Llama2-7B. We use the Llama2-7B model such that we can include LLM-Pruner in our comparison.

	SliceGPT	LLM-Pruner	ShortGPT	DeeperLayers	FINERCUT
Time consumption	4 min	1 min	1 min	1 min	27 min
Memory usage	25.2 GB	40.71 GB	27.15 GB	27.15 GB	27.19 GB

tasks than on zero/few-shot tasks. Additionally, we demonstrate text examples generated by the FINERCUT-pruned models in Appendix H.

In addition, Table 2 shows the computation and memory reduction of a FINERCUT-pruned model. For memory measurements, we consider only the memory required to load the model. We measure MACs and runtime under an 8k context length, as the 8k context length is applied for pretraining. At 25% layer pruning ratio, our methods can effectively reduce the amount of computation by around 25%, which is calculated in MACs. We observe that the runtime reduction is proportional to the MAC reduction as expected. At the 25% layer pruning ratio, the memory usage of the pruned model is close to the memory usage of the original model. This is because our pruning methods choose to prune self-attention layers at early pruning iterations (details of pruned layers are presented in Section F). Since GQA in Llama3-8B is memory efficient, removing them can effectively reduce the runtime but barely reduces the memory usage. At larger pruning ratios, our method removes more FFN layers, hence resulting in more significant memory reduction.

In Table 3, we compare the runtime and memory usage of our method with other baselines. Unlike the LLM-Pruner, our method does not require gradient information during pruning and relies solely on forward passes. This significantly reduces the GPU memory requirements, making our method's memory usage comparable to other layer pruning methods that also operate on forward passes alone. Conversely, methods that utilize gradient information consume more GPU memory because they must store activations during forward passes. This requirement constrains their scalability and limits their usage under low-resource conditions. Regarding runtime, our method is lengthier because it iterates over all remaining layers before selecting the next layer to prune. However, we believe this increase in runtime is justifiable since the pruning process is a one-time requirement. The performance of the pruned model, which is a critical metric for any pruning method, outweighs the longer runtime.

5 Conclusion and future works

In this work, we propose FINERCUT, an effective layer pruning method with a finer scope that prunes attention and FFN layers. We evaluate the efficacy of FINERCUT on three sets of distinct models, Llama2 family, Llama3 family, and Mixtral-8x7B. Compared to other baselines, FINERCUT leads to better models that in general preserve 90% of their performance with around 30% layers removed, without fine-tuning or post-pruning reconstruction. Furthermore, our pruning results (detailed in Appendix F) demonstrate that self-attention layers in LLMs are highly redundant in certain model architectures, which provides new insights into the inner mechanism of LLMs. These findings carry significant implications for the design of future LLMs.

6

References

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

[2] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

[3] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

[4] Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. In *The Twelfth International Conference on Learning Representations*, 2023.

[5] Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jin Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. Binarybert: Pushing the limit of bert quantization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4334–4348, 2021.

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

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

[8] Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. Quip: 2-bit quantization of large language models with guarantees. *Advances in Neural Information Processing Systems*, 36, 2024.

[9] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL `https://aclanthology.org/N19-1300`.

[10] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

[11] Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*, 2019.

[12] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23 (120):1–39, 2022.

[13] Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35: 4475–4488, 2022.

[14] Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. *ICML*, 2023.

[15] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL `https://zenodo.org/records/10256836`.

[16] Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. The unreasonable ineffectiveness of the deeper layers, 2024.

[17] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2023.

[18] Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pages 293–299. IEEE, 1993.

[19] Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[20] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[21] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[22] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[23] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[24] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

[25] François Lagunas, Ella Charlaix, Victor Sanh, and Alexander Rush. Block pruning for faster transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10619–10629. Association for Computational Linguistics, November 2021. doi: 10.18653/v1/2021.emnlp-main.829.

[26] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

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

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

[29] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2016.

[30] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

[31] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[32] Matthias C Rillig, Marlene Ågerstrand, Mohan Bi, Kenneth A Gould, and Uli Sauerland. Risks and benefits of large language models for the environment. *Environmental Science & Technology*, 57(9):3464–3466, 2023.

[33] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019.

[34] Sidak Pal Singh and Dan Alistarh. Woodfisher: Efficient second-order approximation for neural network compression. *Advances in Neural Information Processing Systems*, 33:18098–18109, 2020.

[35] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

[36] Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*, 2023.

[37] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

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

[39] Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. Want to reduce labeling cost? gpt-3 can help. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4195–4205, 2021.

[40] Xinpeng Wang, Leonie Weissweiler, Hinrich Schütze, and Barbara Plank. How to distill your bert: An empirical study on the impact of weight initialisation and distillation objectives. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1843–1852, 2023.

[41] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2021.

[42] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022.

[43] Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared llama: Accelerating language model pre-training via structured pruning. In *The Twelfth International Conference on Learning Representations*, 2023.

[44] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR, 2023.

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

[46] Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183, 2022.

[47] Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. Q8bert: Quantized 8bit bert. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, pages 36–39. IEEE, 2019.

[48] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

A   Limitation

In this work, we demonstrate that we can remove some layers while effectively retaining the performance of a LLM. Although our layer pruning method has notable performance improvements compared to other baselines, several limitations warrant further investigation. Our study mainly focused on three distance metrics. There might exist distance metrics that can yield better pruning results. Additionally, despite the low computational complexity of our iterative pruning method, it only provides a local optimal solution to the optimization problem. Advanced combinatorial optimization methods may identify better selections of pruning layers.

B   Broader impact

The method we proposed in this paper, FINERCUT, is a layer pruning method that aims to prune large language models (LLMs). The development and implementation of methods to prune LLMs hold significant potential for both positive and negative impacts across various domains. Currently, LLMs are trained in computing clusters available only in certain countries and deployed in workstations equipped with costly GPUs. In a nutshell, they are not accessible to the public and people with limited computational resources. By reducing the size and computational requirements of LLMs, we can make LLMs, which have shown their capability and are publically considered powerful tools in many regards, more accessible to a wider range of users and applications. Potential use cases include LLMs on laptops, mobile devices, or IoT devices. This increased accessibility can foster innovation in many fields such as education, healthcare, and natural language processing by enabling smaller organizations and researchers with limited resources to leverage advanced AI technologies. In addition, increased accessibility also helps to avoid AI monopoly in the future.

Pruning LLMs can also contribute to environmental sustainability by decreasing the energy consumption and carbon footprint associated with training and deploying these models. As AI becomes increasingly integrated into everyday applications, the environmental impact of large-scale computations is a growing concern and cannot be neglected. Efficiently pruned models can help mitigate this issue by requiring fewer resources to operate effectively for both training and inference.

However, the broader adoption of pruned models must be approached with caution. Reducing model size can potentially lead to a loss of nuance and accuracy in language understanding and generation, which could have unintended consequences in critical applications such as medical diagnosis or legal analysis. Furthermore, the democratization of powerful language models, while generally positive, raises ethical considerations regarding the potential misuse of these technologies for generating misleading or harmful content. Hence, further research and collaboration across disciplines will be crucial in ensuring that the deployment of pruned LLMs is both responsible and beneficial to society.

C   Detailed experiment settings

Models: We consider a wide range of models for pruning. Specifically, we choose some member models of Llama family[37, 38] including Llama2-7B, Llama2-13B, Llama3-8B, and Llama3-70B. The rationale for selecting these models is to evaluate the pruning methods on models at different parameter levels and across different generations. In addition, we include Mixtral-8x7B [23] in our experiments to assess the pruning performance on Mixture-of-Experts (MoE) models [12]. Due to the page limit, we present the results of Llama2 models in Appendix E.1 and Appendix E.2.

Benchmarks: We conduct our experiment on two types of tasks: generative tasks and zero/few-shot tasks. For generative tasks, we evaluate the perplexity on WikiText2 [29]. For zero/few-shot tasks, we evaluate accuracy on commonsense reasoning datasets: BoolQ [9], PIQA [6], HellaSwag [48], WinoGrande [33], ARC-easy/challenge [10],OpenbookQA [30] and MMLU [19, 20]. We use the LM-Evaluate-Harness framework [15] to ensure a transparent and reproducible evaluation by following exactly the prompts and evaluation metrics in the framework.

Wikitext2   The WikiText language modeling dataset consists of over 100 million tokens that have been extracted from the set of verified Good and Featured articles on Wikipedia. The task is to measure the perplexity on the Wikitext dataset, via rolling loglikelihoods. A lower perplexity means better text generation performance.

BoolQ The BoolQ dataset contains 15942 examples of yes/no questions. The questions are generated naturally – they occur in an unprompted and unrestricted environment. Each example is a triplet (questions, passages, answers), with a title as an optional context.

ARC The ARC dataset includes 7,787 science exam questions selected from various sources, including questions provided by AI2 affiliates under license. The exam questions are text-only, English language questions that span several grade levels. Questions are structured with multiple-choice answers (usually four options). There are two sets of questions: the Challenge Set and the Easy Set. The Challenge Set has 2,590 "hard" questions (those that cannot be answered correctly by both retrieval and co-occurrence methods). The Easy Set has 5,197 questions.

WinoGrande WinoGrande consists of 44k fill-in-the-blank type problems with binary options. It requires common sense reasoning to choose the right option for a given sentence.

HellaSwag HellaSwag is a challenge dataset for evaluating commonsense Natural Language Inference that is especially hard for models, though its questions are trivial for humans (>95% accuracy). The questions are four-way multiple-choice problems.

PIQA PIQA (Physical Interaction: Question Answering) is a dataset for commonsense reasoning, designed to investigate the physical knowledge of the models. The underlying task is a binary choice question answering task.

OpenbookQA The OpenbookQA is based on open-book exams and is a question-answering dataset that assesses human understanding of subjects through question-answering. There are 5,957 multiple-choice questions addressing elementary science topics, which assess a candidate's understanding of 1,326 core science facts and the application of these facts in novel circumstances.

MMLU The MMLU consists of a series of 15,908 multiple-choice questions covering 57 academic disciplines including mathematics, philosophy, law and medicine as well as many other areas of academic study. The questions are in the form of four-way multiple-choice questions. We adopt the frequently-used 5-shot setting to provide four sample questions and answers from the dev subset of MMLU, then ask the model to answer the actual question with A,B,C, or D.

Baselines: We include other four structured pruning methods in our evaluation: LLM-Pruner [27], SliceGPT [4], ShortGPT [28], and the pruning method discussed in Gromov et al. [16], which we refer it as *DeeperLayers* in the following text. For ShortGPT and DeeperLayers, the implementation is not publicly available. Hence, we reproduce their methods according to their manuscripts. Our reproduced version is available in our codebase. For SliceGPT and LLM-Pruner, we use the official implementation. However, SliceGPT and LLM-Pruner are not designed to prune layers. Hence, we set the sparsity for these methods to be identical to our layer pruning methods. It is worth noting that LLM-Pruner currently does not support group-query attention (GQA) [2], so we only apply LLM-Pruner on Llama2 models.

Pruning settings: We randomly selected 10 samples from the Wikitext2 dataset for running our iterative layer pruning algorithm. To compare the actual performance of the pruning methods, we **did not conduct post-pruning training** to heal or reconstruct the pruned LLMs. In the following text, we refer to the pruning results associated with angular distance as Acos, those associated with Euclidean distance as Norm, and those associated with statistical distance as JS. For Llama3-8B, the pruning algorithm was executed on a single NVIDIA A100-SMX-80GB GPU. For Llama3-70B and Mixtral-8x7B, the pruning algorithm was run on two NVIDIA A100-SMX-80GB GPUs.

D More evaluation result on Mixtral-8x7B

In complement to the Fig. 2c in the main text, we present a more detailed pruning result on Mixtral-8x7B with performance reports on each task. We exclude ShortGPT and LLM-Pruner for this model because their implementations do not support MoE layers. The results highlight the robustness and efficiency of FINERCUT across various benchmarks. Our experiments demonstrate that our proposed pruning method, FINERCUT, has many applications and can be used on MoE models effectively.

Table 4: Performance on Mixtral-8x7B at 25% layer pruning ratio.

Model	Wikitext	BoolQ	ARC-C	ARC-E	WG	HS	PIQA	OBQA	MMLU	Average
Original	8.50	84.5	59.7	83.2	75.6	72.8	83.9	46.6	72.2	72.3
ShortGPT	30.86	**83.2**	39.4	54.0	**69.5**	57.7	69.3	37.6	**69.6**	60.0
DeeperLayers	25.42	54.6	30.2	54.3	51.4	53.5	72.9	37.0	25.6	47.4
Acos	13.54	77.8	42.1	67.5	66.9	64.5	78.1	39.6	39.3	59.5
Norm	14.24	79.1	**43.9**	**68.5**	65.9	**65.9**	78.1	40.8	43.5	**60.7**
JS	**13.26**	77.3	42.9	68.2	66.3	64.8	**78.4**	**41.0**	40.7	60.0

Table 5: Performance on Mixtral-8x7B at 40% layer pruning ratio.

Model	Wikitext	BoolQ	ARC-C	ARC-E	WG	HS	PIQA	OBQA	MMLU	Average
Original	8.50	84.5	59.7	83.2	75.6	72.8	83.9	46.6	72.2	72.3
ShortGPT	47.64	**64.7**	31.9	**49.6**	52.8	50.2	66.5	33.2	26.3	46.9
DeeperLayers	5335.09	44.2	27.5	25.5	49.1	26.8	48.2	24.6	23.2	33.4
Acos	**43.31**	58.9	31.9	44.5	57.5	54.0	64.3	29.4	24.7	45.6
Norm	55.77	40.2	**35.2**	48.5	**60.3**	**57.3**	**66.7**	**33.8**	24.5	45.8
JS	43.93	62.5	34.4	47.4	58.9	56.0	66.3	32.6	**26.8**	**48.1**

E More evaluation results on the Llama2 model family

In this section, we present additional experimental results on Llama2 models. While the main text focuses on Llama3 models, we include Llama2 results to provide a comprehensive overview. We report results on Llama3 models in the main text because the conclusions drawn from Llama2 models are consistent with those from Llama3, and we believe that results on newer models are more relevant to our audience. However, Llama2 results are included for completeness and because they include results of LLM-Pruner, which does not apply to Llama3 models. We use Llama2-7B and Llama2-13B models as pruning targets. We do not include the Llama2-70B model, as this model requires more resources to run.

E.1 More evaluation results on Llama2-7B

In this section, we present the evaluation results for pruned Llama2-7B models. Our analysis includes the same performance metrics as shown in the main text. Table 6 shows the performance of a pruned model with 25% layer removal ratio. Additionally, Table 6 shows the performance of a pruned model with 40% layer removal ratio.

Table 6: Performance on Llama2-7B at 25% layer pruning ratio. Both SliceGPT and LLM-Pruner are applied with a desired 25% sparsity ratio according to their implementation.

Model	Wikitext	BoolQ	ARC-C	ARC-E	WG	HS	PIQA	OBQA	MMLU	Average
Original	10.14	76.2	46.0	74.0	68.3	66.5	79.7	44.2	47.1	62.8
ShortGPT	127.89	62.5	31.0	41.7	60.8	44.0	60.1	35.0	**44.4**	47.4
DeeperLayers	35.36	49.2	35.0	50.5	64.0	54.7	67.9	37.6	44.4	50.4
SliceGPT*	20.05	52.1	34.5	55.6	62.9	55.1	67.5	35.8	29.7	49.1
LLM-Pruner*	20.82	62.4	37.2	62.0	62.2	60.1	**75.7**	38.6	25.3	52.9
Acos	20.49	73.2	37.7	**62.8**	64.1	60.1	73.9	**40.2**	32.3	**55.5**
Norm	21.73	**74.5**	**38.4**	61.5	63.7	**60.3**	73.6	38.8	31.8	55.3
JS	**18.43**	69.3	38.4	62.1	**64.4**	59.5	75.4	39.8	28.5	54.7

To provide an overview how well pruning methods work in different cases. We include Fig. 3 to show the performance change at various layer pruning ratios. According to the figure, FINERCUT can in general better preserve the performance of pruned models. In addition, we include LLM-Pruner in all evaluations on Llama2-7B. LLM-Pruner performs better than other baseline methods but still less effective than FINERCUT. Importantly, FINERCUT can be applied to many models including models that do not exist yet, while LLM-Pruner needs to be adopted to those new structures.

E.2 More evaluation results on Llama2-13B

In this section, we present the evaluation results for pruned Llama2-13B models.

Table 7: Performance on Llama2-7B at 40% layer pruning ratio. Both SliceGPT and LLM-Pruner are applied with a desired 25% sparsity ratio according to their implementation.

| Model	Wikitext	BoolQ	ARC-C	ARC-E	WG	HS	PIQA	OBQA	MMLU	Average
Original	10.14	76.2	46.0	74.0	68.3	66.5	79.7	44.2	47.1	62.8
ShortGPT	1683.61	55.4	27.7	25.6	50.3	30.1	50.7	26.6	**39.6**	38.3
DeeperLayers	303.03	62.6	32.0	35.1	59.2	39.3	59.4	27.6	39.6	44.4
SliceGPT*	**46.63**	37.5	27.3	43.5	57.9	43.6	58.5	28.4	24.4	40.1
LLM-Pruner*	48.39	50.2	31.3	**50.7**	56.1	**53.5**	**70.7**	**36.6**	24.4	46.7
Acos	73.36	36.8	**34.0**	46.9	**62.0**	49.5	66.8	33.4	25.6	44.4
Norm	66.37	63.2	32.5	42.7	57.6	48.9	64.7	34.0	25.0	46.1
JS	61.49	**65.6**	33.9	49.3	60.3	48.1	64.5	34.6	25.3	**47.7**

(a)   (b)
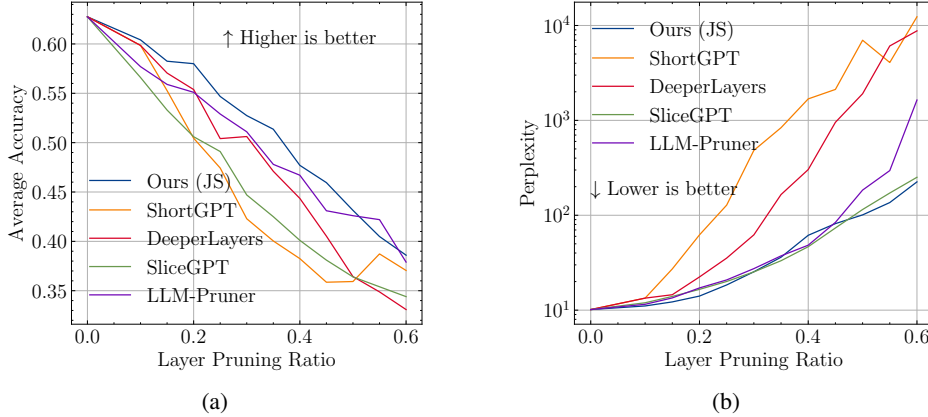
Figure 3: **(a)**: Average zero/few-shot performance at different layer pruning ratios on Llama2-7B. **(b)**: Perplexity (with a logarithmic scale) on WikiText2 at varying layer pruning ratios. FINERCUT outperforms other methods including LLM-Pruner (not evaluated on Llama3 models) on both text generation and QA tasks.

Fig. 4 shows the performance change at various layer pruning ratios.

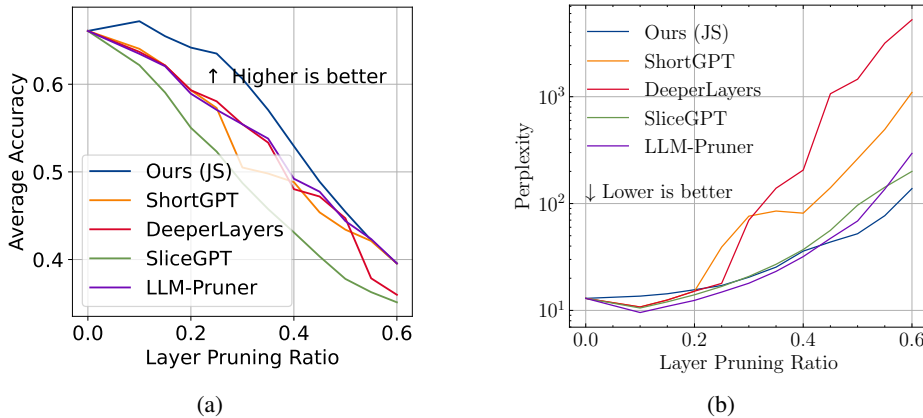(a)   (b)

Figure 4: **(a)**: Average zero/few-shot performance at different layer pruning ratios on Llama2-13B. **(b)**: Perplexity (with a logarithmic scale) on WikiText2 at varying layer pruning ratios. FINERCUT outperforms other methods including LLM-Pruner (not evaluated on Llama3 models) on both text generation and QA tasks.

E.3   Ablation study

In this section, we verify the design choices of FINERCUT and examine their significance through two experiments. First, we evaluate the option of pruning transformer blocks instead of attention

Table 8: Performance on Llama2-13B at 25% layer pruning ratio. Both SliceGPT and LLM-Pruner are applied with a desired 25% sparsity ratio according to their implementation.

Model	Wikitext	BoolQ	ARC-C	ARC-E	WG	HS	PIQA	OBQA	MMLU	Average
Original	12.96	79.8	49.0	77.2	71.6	69.6	81.3	45.2	55.0	66.1
ShortGPT	39.15	62.7	41.9	60.1	70.5	60.6	73.1	40.8	48.6	57.3
DeeperLayers	17.90	57.9	43.8	64.6	69.5	61.3	73.9	39.0	54.4	58.1
SliceGPT*	16.75	40.4	40.2	61.5	67.0	59.4	69.6	40.8	39.7	52.3
LLM-Pruner*	**14.84**	66.3	43.9	67.8	63.5	65.4	79.4	43.2	27.2	57.1
Acos	17.29	77.4	43.6	70.1	69.8	64.4	77.2	41.4	52.5	62.0
Norm	17.76	77.3	43.5	70.4	**71.5**	**66.4**	77.4	40.6	**54.6**	62.7
JS	15.61	**78.8**	**46.8**	**73.4**	70.5	66.1	**80.0**	**44.4**	53.4	**64.2**

Table 9: Performance on Llama2-13B at 40% layer pruning ratio. Both SliceGPT and LLM-Pruner are applied with a desired 25% sparsity ratio according to their implementation.

Model	Wikitext	BoolQ	ARC-C	ARC-E	WG	HS	PIQA	OBQA	MMLU	Average
Original	12.96	79.8	49.0	77.2	71.6	69.6	81.3	45.2	55.0	66.1
ShortGPT	81.29	62.5	32.3	44.8	62.8	47.8	62.4	35.4	42.6	48.8
DeeperLayers	205.75	43.5	33.5	45.8	65.7	51.8	64.5	33.4	46.1	48.0
SliceGPT*	37.06	37.5	29.2	44.1	61.6	49.6	59.9	35.6	27.3	43.1
LLM-Pruner*	**31.86**	43.4	35.4	**56.3**	57.8	**60.2**	**75.3**	**41.6**	23.8	49.2
Acos	38.65	57.5	**36.0**	53.0	65.4	54.9	69.8	33.6	**49.6**	52.5
Norm	38.65	57.5	36.0	53.0	65.4	54.9	69.8	33.6	49.6	52.5
JS	35.89	**65.0**	32.6	51.7	**66.7**	57.3	68.6	35.6	46.0	**52.9**

and FFN layers. In addition, we compare different metrics for distance measures. Fig. 5 (a) and (b) compare the pruning performance when pruning candidates are different. The performance is measured in two setups: (i) considering attention and FFN layers as separate pruning candidates; or (ii) only pruning transformer blocks. The results on zero/few-shot tasks demonstrate the advantage of pruning attention and FFN layers compared to transformer blocks, highlighting the importance of pruning at a finer level. Fig. 5 (c) and (d) show the pruning performance using three different distance measures: norm, arccos, and JS divergence. On zero/few-shot tasks, all three measures have similar performance. On perplexity results, JS divergence outperforms norm and arccos measures. Therefore, we recommend using JS divergence or other probabilistic measures as pruning metrics.

F   Analysis of pruned layers

Our layer pruning method can also work as an effective tool to study the mechanistic interpretability of LLMs. In this section, we present the results of pruned layers and study the layer importance of various LLM models based on the pruning results.

Fig. 6 shows the visualization of pruned layers in Llama3-8B and Llama3-70B models. At 25% layer pruning ratio, our pruning methods choose to mainly remove self-attention layers in the model. At this layer pruning ratio, pruned models exhibit minimal performance degradations, suggesting that these attention layers are unimportant. Moreover, the pruned self-attention layers are usually in consecutive transformer blocks. For instance, self-attention layers from index 40 to 70 are completely removed in Llama3-70B, and self-attention layers from index 18 to 28 are completely removed in Llama3-8B. Subsequently, we have a new structure at deeper layers, in which one self-attention layer is followed by multiple FFN layers. Surprisingly, on pruned Llama3-70B, there is even one attention layer followed by more than 20 FFN layers. Based on these observations, we hypothesize that at later parts of LLMs, multiple consecutive FFN layers work in cooperation to process the output from one attention layer. This observation suggests that the current transformer architecture may not be the optimal structure for LLMs. Instead, non-uniform transformer structures with more FFNs at the later stage can be potentially more parameter-efficient and may be a better design option for future LLMs.

We further inspect the pruning result on Mixtral-8x7B, an LLM equipped with Mixture-of-Experts (MoE) layers. Fig. 6 also shows pruned layers on Mixtral-8x7B. Intriguingly, the pruning selection is more balanced between attention and MoE layers than on Llama models. Though MoE layers are sparse structures, their representation capacity is larger as they contain multiple FFNs. Hence, we conjecture that the Mixtral-8x7B model is learned not to use multiple MoE layers to process
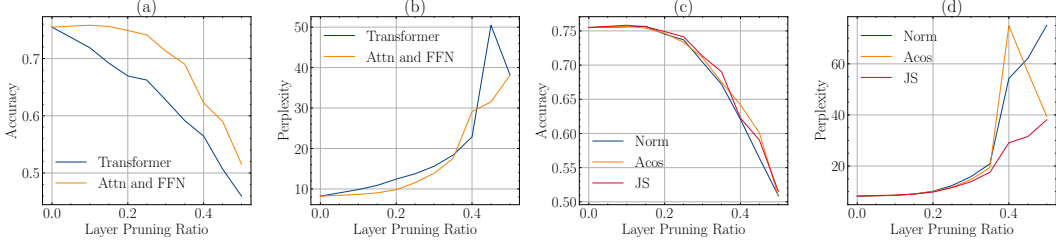
Figure 5: Ablation study on Llama3-70B. **(a)** and **(b)**: Pruning transformer blocks vs. pruning attention and FFN layers separately. **(c)** and **(d)**: Comparison of three distance metrics in FINERCUT.
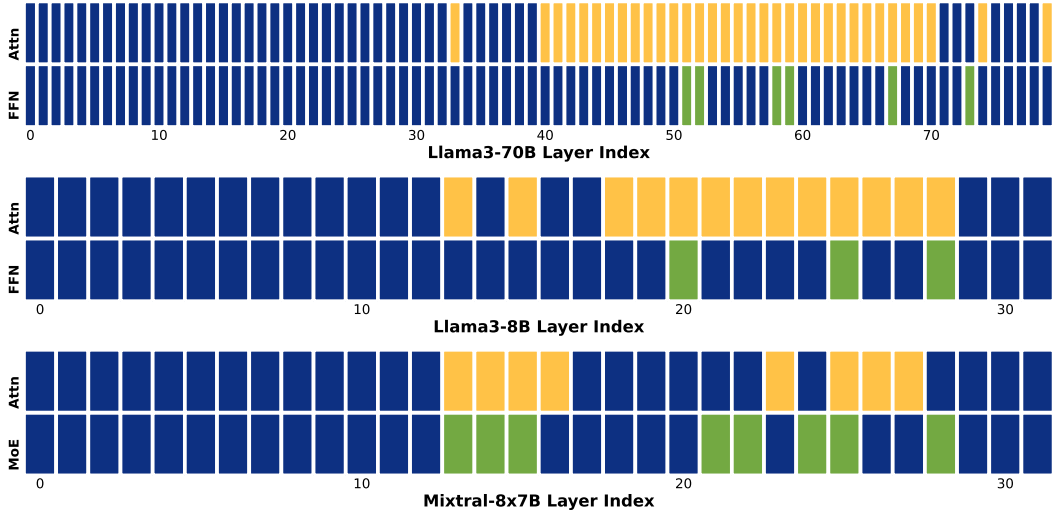

Figure 6: Visualization of pruned layers at 25% layer pruning ratio for Llama3-70B (top, with 1.9% performance drop), Llama3-8B (middle, with 11.6% performance drop), and Mixtral-8x7B (bottom, with 17.0% performance drop) using FINERCUT. ▮ indicates pruned self-attention layers, ▮ indicates pruned FFN layers, and ▮ indicates remaining layers. Notably, consecutive self-attention layers are removed, resulting in a heterogeneous structure where multiple FFNs process the output of one attention layer. More discussion in Section F.

for one attention layer. Our result suggests that MoE models may have less layer redundancy than conventional transformer models by design.

Lastly, we also observe the removal and merging of transformer blocks in all pruned models. From the pruning result, we can see cases where one entire transformer block is pruned, as observed for instance on Llama3-70B at layer 51, and on Mixtral-8x7B at layer 13. In addition, multiple transformer blocks can be merged together. As shown in Mixtral-8x7B, layer 22 and 23 are merged into one transformer block by removing the MoE layer and the subsequent attention layer.

G Complementary results for Figure 6

In this section, we provide two additional tables listing the pruned layers for readers interested in a more detailed breakdown of our pruning results. This table complements Fig. 4 by offering specific insights into which layers were pruned and the type of pruned layers (denoted with distinct colors). By including this supplementary information, we aim to enhance the transparency and reproducibility of our results, allowing for a deeper understanding of the pruning strategy employed in our study. Interested readers can use these results to reproduce pruned models for their purposes without running our code for pruning.

Layer pruning ratio	Pruned Layers	Performance Drop
25%	A13 A15 A18-19 T20 A21-24 T25 A26-27 T28	11.6%
40%	A13 F14 A15 F16 T17-20 A21 T22-23 A24 T25-26 A27 T28	33.2%

Table 10: Pruned layers in Llama-3-8B at various layer pruning ratios. A stands for a self-attention layer, F denotes an FFN layer. And T denotes a transformer block.

Layer pruning ratio	Pruned Layers	Performance Drop
25%	A33 A40-A50 T51-52 A53-A57 T58 T59 A60-A66 T67 A68-A70 F73 A74 A79	1.9%
40%	A33 F35 A36 A38-40 T41-44 A45 T46-55 A56 T57-T63 A64 T65-68 A69-70 F73 A74 A79	17.5%

Table 11: Pruned layers in Llama3-70B at various layer pruning ratios. A stands for a self-attention layer, F denotes an FFN layer. And T denotes a transformer block.

H   Text generation results on pruned models (w/o finetuning).

Below, we provide generated samples for readers to have a glance at the generation ability after pruning.

Table 12: Generated examples from the pruned LLama3-70B using FINERCUT. The underlined texts denote the input prompts.

Layer pruning ratio	Generated sentences
Original	<u>Neural Sequential Model, especially transformers, have been widely used in NLP tasks.</u> The sequential model is mainly based on the self-attention mechanism, which is used to capture the relationship between each token in the input sequence. The core of the self-attention mechanism is to calculate the similarity between the token in the input sequence, and then use the similarity to calculate the weight between each token. The self-attention mechanism can be divided into two types: local self-attention and global self-attention. Local self-attention only calculates the similarity between the token in the input sequence, while global self-attention calculates the similarity between the token in the input sequence and the token in the output sequence. The self-attention mechanism can be used to capture the relationship between each token in the input sequence, which is very important for NLP tasks.
	<u>Last night we all danced together in the rain,</u> the joy of the moment was so present that we didn't want it to end. The raindrops were the perfect accent to the rhythm of the music. The night was perfect. We danced in the rain. We danced for the rain. We danced for the joy of dancing in the rain. We danced for the joy of being alive, together, in the rain.
	<u>It fills me with such pride and joy</u> to see the work that our team has done to build a company that is committed to serving the people of our community. I am so proud of what we have accomplished and the legacy we have built. The first thing that I noticed about our company is that it is a family business. We are a family of four, and we are all very close.
0.25	<u>Neural Sequential Model, especially transformers, have been widely used</u> for many applications in natural language processing. However, it is still difficult for most of the neural models to incorporate structured external knowledge, which is essential for some tasks. This paper presents a new approach to integrating structured knowledge into sequential models. We introduce a new type of recurrent neural network, which is inspired from the transformer model. We first introduce a new type of neural network called a recurrent neural network (RN). We show that this architecture is able to capture the structure of the data in an end-to-end manner. The proposed method is eval uated on a large real-world data set of news articles. Experimental results show that our model outperforms the baseline in terms of accuracy and computation time.
	<u>Last night we all danced together in the rain,</u> and I thought of you. The kids were playing in the street, and I thought of you. As I watched the sunset, I thought of you. When the sun rose this morning, I thought o f you. I thought of you yesterday, I think of you today. I'll think of you again tomorrow. I have thought of you between the times I think I think of you. My thoughts are full of you. There is no room for another. You are my all, an d I am yours. It's time to come home. I'm ready. I've been ready, and waiting. The light is green, my love. The coast is clear.
	<u>It fills me with such pride and joy</u> to know that my son is being taught the way I want him to be taught. He is happy at school and is eager to go every day. He is learning about the Bible and how to be a good Chris tian. I have complete confidence that he is in great hands at St. John's. I am so thankful for the teachers and staff that help in the education and development of my child. Thank you for all you do!St. John's has been a wonderful place for my children to grow and learn. They have been taught Christian values in a caring, nurturing environment. The teachers are excellent and the curriculum is great.

Table 13: Generated examples from the pruned LLama3-70B using FINERCUT. The underlined texts denote the input prompts.

Layer pruning ratio	Generated sentences
0.4	Neural Sequential Model, especially transformers, are being applied in a diverse range of applications, from the traditional NLP tasks such as natural language generation, to the more general problems such as human relationship classification. In this workshop, we will review the recent advances and applications of transformers in a wide range of applications. Neural Sequential Model, especially transformers, have been widely applied in a diverse range of applications, from the traditional NLP tasks such as natural language generation, to the more general problems, such as human relations hip and interaction. In this workshop, we will review the recent advances and applications of transformers in a wide range of applications. The first part of this workshop will be devoted to the basics of transformers, including the history and evolution of the transformers. The second part of this workshop will be devoted to the applications of transformers, including the human relationship and human interaction. The last part of this workshop will be devoted to the future of transformers, including the future of the human interaction. The third part of this workshop will be devoted to the future of the human interaction.
	Last night we all danced together in the rain, the 3rd of 3 days of the 2018 Sante Festival. The final day of the festival was dedicated to the "Sante," a ceremonial blessing to the water, and a blessing for the pe ople. This is a time where the people of the tribe are welcomed to come and be blessed by the monks. As the monk chants and blesses, a ripple of people come to be blessed, and then they in-in-in-in... in an alternating rhythm - just as the monks chants, and the music. It is a ritual that takes hours, and is a spiritual experience for all.
	It fills me with such pride and joy when I come across a young man who has taken the initiative of his life. He is a man who is determined to be successful and to be a leader and not a victim. A man who has a desire to make the greatest use of the most important tool he possesses: his mind. I was watching TV when I was interviewed by a journalist. The story that attracted me was the story of a man who had been given the opportunity to enter the world of entrepreneurship and the results that he was experiencing.
0.55	Neural Sequential Model, especially transformers, have been proven to be a powerful tool for the language learning. The combination of the recent advances in the field and the current pandemic circumstances, is an excellent opportunity for me to build the first version of the 2023. by: 2023 2023: the first year. I have been looking for the first time.
	Last night we all danced together in the rain, while the sun was in the sky. And the rain was a joyful dance, the rain was the music. I think of the time of the sun and I of the wind, and the rain of the earth, is a gives all its power and strength. I am in the midst of a multitude and the sun and the rain, we shall be at the same time. The sun and the water, is the power and the life and the air, and the rain is the music.
	It fills me with such pride and joy that I am a product of the most powerful and effective Bible curriculum, the Catholic Church, of the Catholic Church. You are the product of the Catholic Church, and I am proud of the Catholic Church, and I am proud of the Catholic Church. The Catholic Church is a sacred and holy sacram, and it is a sacred sacram, which is a holy sacram.

Table 14: Generated examples from the pruned LLama3-8B using FINERCUT. The underlined texts denote the input prompts.

Layer pruning ratio	Generated sentences
Original	<u>Neural Sequential Model, especially transformers,</u> are the state-of-the-art models for various NLP tasks. However, these models are often too large for practical deployment, due to the huge number of parameters and memory consumption. For example, BERT-Base has 110M parameters and takes 500M memory. The large model size and memory consumption limit their applications to mobile devices, which are often memory constrained and resource limited. In this paper, we propose a novel and efficient method, named model compression with knowledge distillation (MCKD), to compress the large models with knowledge distillation. Specifically, we first train a small teacher model, and the n train a student model with the knowledge from the teacher model. We find that a student model with 1/100 parameters and 1/1000 memory can achieve comparable performance with the teacher model. We also show that the student model c an be further compressed by removing unnecessary parameters. We evaluate our MCKD method on various NLP tasks, including text classification, sequence labeling, and machine translation, and demonstrate that the proposed MCKD method can significantly improve the performance of the student model.
	<u>Last night we all danced together in the rain,</u> under a full moon, on the beach. It was one of those nights that you always hope for, but never really think will happen. The stars were out, the moon was huge and ful l, the ocean was calm, and it was warm enough to stay outside and not freeze to death. We made s'mores, we danced, we laughed, we talked, we listened to music, and we had a great time. It was the kind of night that you can't help bu t smile at as you lay in bed, drifting off to sleep. The kind of night that makes you feel like anything is possible. The kind of night that makes you feel alive and like you are living in the moment.
	It fills me with such pride and joy to be able to share these photos with you. I am a mother to four boys. I am an artist and designer. I am a wife. I am a friend. I am a daughter. I am a sister. I am a writer. I am a baker. I am a cook.
0.25	<u>Neural Sequential Model, especially transformers,</u> is a powerful tool that has been very effective in generating text and music, and other artificial intelligence tasks. In the original Transformer, the Transformer block was a fully connected layer of the neural network, which is a dense layer and is the default setting in the neural sequential model. It is a very powerful model, but the problem is that the original Transformer is too time-consuming, which is a problem that the original Transformer is a bit of a problem. So, the Transformer model is now available as a module on the Open Neural Networks Library (C code) and a C++ version of the original Transformer. The TensorFlow implementation of the Neural Sequential Model is a good choice for tho se who are interested in the Neural Sequential Model.
	<u>Last night we all danced together in the rain,</u> and the rain poured on the dance floor as the crowd of 1,200, 3D Audio, and the 1971 film The Paper Man, a documentary, were all together with the music of the 1971 film The Godfather. The music is also a classic, and the music is a classic, the same as the 1971 film The Godfather, a documentary.
	<u>It fills me with such pride and joy</u> to hear the words "thank you for the book, thank you for the book" for the first time. A copy of my book is now in the hands of a little girl, a little girl who was born in 2012, and a little girl who is not the only one who is not the only one to the book. The book is the 2012 release of the book. The book is the first book in the "Pandora" trilogy trilogy, which is the second in a trilogy

Table 15: Generated examples from the pruned LLama3-8B using FINERCUT. The underlined texts denote the input prompts.

Layer pruning ratio	Generated sentences
0.40	<u>Neural Sequential Model, especially transformers,</u> has the the ability to capture the data of the new, and the Leberton, for a new, and the the ability to capture the data of the the ability to the the the the the and the the the and the the the the the and the the.
	<u>Last night we all danced together in the rain,</u> and in the clouds of the rain, it was the rain and the rain and the rain. This is the story of the rain, and the rain and the rain. The rain was a wonderful gift.
	<u>It fills me with such pride and joy</u> that I am now considered as a new and it's a new brand. The name is the name of a new Brand. I have a good name for the name. I will be a new brand is a new brand. I have a good name for the name. I will be a new I want to be a new brand. I want to be a new brand. I want to be a new brand. I want to be a new brand.
0.55	<u>Neural Sequential Model, especially transformers,</u> is a source of the German- hering, the German-7. The 2nd. The 10th of 20 the 22. 2. The 4. 2. 3 12. 2. 3 12.
	<u>Last night we all danced together in the rain,</u> that was the first of the two. The first of the two. The next one is one of the three, in the is one of the three, and the another one is one of the the four.
	<u>It fills me with such pride and joy,</u> I don't do the country, Toby Franey. And I don't dun credit I have the 4 of the 12 pack or whatever have you to I have the the.

Table A: FINERCUT vs. SLEB on Llama2-7B at the pruning ratio of 25% w/o fine-tuning.

Model	Wikitext ↓	BoolQ	ARC-C	ARC-E	WG	HS	PIQA	OBQA	MMLU	Average ↑
Ours (JS)	**18.43**	69.3	38.4	62.1	64.4	59.5	75.4	39.8	28.5	**54.7**
SLEB	22.9	62.8	32.7	58.7	57.8	55.1	73.8	35.2	24.7	50.1

Table B: FINERCUT vs. SLEB on Llama3-8B at the pruning ratio of 25% w/o fine-tuning.

Model	Wikitext ↓	BoolQ	ARC-C	ARC-E	WG	HS	PIQA	OBQA	MMLU	Average ↑
Ours (JS)	**23.06**	76.0	38.2	65.6	65.2	60.5	74.2	38.2	62.8	**60.1**
SLEB	31.75	54.5	32.1	57.3	57.1	59.1	72.9	33.4	26.3	49.1

Table C: FINERCUT vs. SLEB on Llama3-70B at the pruning ratio of 25% w/o fine-tuning.

Model	Wikitext ↓	BoolQ	ARC-C	ARC-E	WG	HS	PIQA	OBQA	MMLU	Average ↑
Ours (JS)	**11.55**	84.3	62.4	83.7	79.3	73.6	84.0	46.8	79.0	**74.1**
SLEB	16.57	80.9	49.8	75.2	75.3	66.1	78.5	42.6	70.3	67.3

Table D: FINERCUT using 10/50/100 samples for pruning Llama3-8B at 25% pruning ratio.

Sample size	Wikitext ↓	BoolQ	ARC-C	ARC-E	WG	HS	PIQA	OBQA	MMLU	Average ↑
10	23.06	76.0	38.2	65.6	65.2	60.5	74.2	38.2	62.8	60.1
50	18.81	76.4	46.8	73.5	69.2	64.8	78.1	41.4	64.1	64.3
100	**18.73**	80.2	52.7	76.3	72.8	66.9	79.5	42.8	66.3	**67.2**

Table E: FINERCUT using different datasets for pruning Llama3-8B at 25% pruning ratio.

Pruning dataset	Wikitext ↓	BoolQ	ARC-C	ARC-E	WG	HS	PIQA	OBQA	MMLU	Average ↑
WikiText	23.06	76.0	38.2	65.6	65.2	60.5	74.2	38.2	62.8	60.1
C4	**18.85**	80.4	50.7	76.5	73.7	67.9	79.8	43.0	66.4	**67.3**

Table F: Fine-tuning FINERCUT-pruned Llama3-8B on Alpaca dataset.

Ratio + FT	Wikitext ↓	BoolQ	ARC-C	ARC-E	WG	HS	PIQA	OBQA	MMLU	Average ↑
Original	11.31	80.8	53.1	76.9	71.7	68.8	81.3	45.0	66.2	68.0
25% w/o FT	23.06	76.0	38.2	65.6	65.2	60.5	74.2	38.2	62.8	60.1
25% w/ FT	**18.81**	80.6	48.0	74.0	73.0	64.6	78.6	40.2	65.8	**65.6**
40% w/o FT	123.94	67.4	29.3	38.9	58.0	46.1	61.9	32.0	29.6	45.4
40% w/ FT	**27.56**	72.1	38.4	66.9	65.8	62.9	75.1	38.4	61.8	**60.2**
50% w/o FT	249.51	38.7	25.6	32.1	53.7	39.2	58.0	27.6	25.8	37.6
50% w/ FT	**34.95**	60.6	33.4	60.0	63.1	59.1	73.4	34.4	46.1	**53.8**

Table G: Iterative pruning vs. one-shot pruning on Llama3-8B at 25% pruning ratio.

Algorithm	Wikitext ↓	BoolQ	ARC-C	ARC-E	WG	HS	PIQA	OBQA	MMLU	Average ↑
Iterative	**23.06**	76.0	38.2	65.6	65.2	60.5	74.2	38.2	62.8	**60.1**
One-shot	39.58	61.1	28.3	47.5	60.3	52.3	71.6	33.4	24.6	47.4