PIP: Perturbation-based Iterative Pruning for Large Language Models

Yi Cao[1] **Wei-Jie Xu**[2] **Yucheng Shen**[1]
Weijie Shi[3] **Chi-Min Chan**[3] **Jianfeng Qu**[1] **Jiajie Xu**[1*]
[1]School of Computer Science and Technology, Soochow University
[2]School of Artificial Intelligence, Nanjing University
[3]Department of Computer Science and Engineering, Hong Kong University of Science and Technology

Abstract

The rapid increase in the parameter counts of Large Language Models (LLMs), which often reach into the billions or even trillions, presents significant challenges for their practical deployment, particularly in resource-constrained environments. To address this issue, we propose PIP (Perturbation-based Iterative Pruning), a novel double-view structured pruning method to optimize LLMs, which combines information from two different views: the unperturbed view and the perturbed view. With the calculation of gradient differences, PIP iteratively prunes those that struggle to distinguish between these two views. Our experiments show that PIP reduces the parameter count by approximately 20% while retaining over 85% of the original model's accuracy across varied benchmarks. In some cases, the performance of the pruned model is within 5% of the unpruned version, demonstrating PIP's ability to preserve key aspects of model effectiveness. Moreover, PIP consistently outperforms existing state-of-the-art (SOTA) structured pruning methods, establishing it as a leading technique for optimizing LLMs in constrained environments.

1 Introduction

Large Language Models (LLMs) (Achiam et al., 2023; Dubey et al., 2024) based on the Transformer architecture (Vaswani, 2017) have demonstrated impressive capabilities across a wide range of tasks, but their capabilities come at the expense of massive parameter counts and high computational requirements (Kaplan et al., 2020). For illustration, the LLaMA3-405B model (Dubey et al., 2024), with about 405 billion parameters, demands at least 810 GB of memory with 11 A100 GPUs when using half-precision (FP16) format. Therefore, an issue presents itself that warrants further exploration: Can we produce a smaller, general-purpose, and competitive LLM by leveraging existing pretrained LLMs, while using much less compute than training one from scratch? (Xia et al., 2024)

To this end, researchers have been exploring strategies like pruning (Frantar and Alistarh, 2023; Ma et al., 2023; Ashkboos et al., 2024), quantization (Bai et al., 2021; Lin et al., 2024), knowledge distillation (Sun et al., 2020; Pan et al., 2021), and low-rank factorization (Saha et al., 2023; Yuan et al., 2023). Among them, pruning methods have gained considerable attention due to their potential to significantly reduce model size while preserving performance. Yet, traditional pruning strategies typically assess importance through isolated metrics such as weight magnitudes or input-output similarity (Men et al., 2024). While intuitive, these single-view approaches suffer from fundamental flaws: they overlook the capacity to preserve semantic robustness under adversarial or natural input variations, which is essential for ensuring reliable deployment of models in real-world applications.

To overcome the limitation, we propose a novel double-view approach that evaluates layer importance based on their awareness of text perturbations. For each input, we generate two complementary perspectives: an original sample and its perturbed counterpart—crafted via character-level edits that preserve syntax but maximally distort meaning (Section 3.1). By contrasting parameter gradients between these views through first-order Taylor approximation, we identify layers exhibiting weak semantic discrimination.

Beyond the initial double-view comparison, our approach employs an iterative gradient reassessment strategy to further refine the pruning process. After pruning the least sensitive layers identified in each cycle, we proceed to recompute gradient differences on the updated architecture. This dynamic process, which is akin to curriculum learning, progressively focuses on layers that are critical for semantic stability. By doing so, it ensures a thorough

---

[*] Corresponding author: xujj@suda.edu.cn

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 3685–3701
November 4-9, 2025 ©2025 Association for Computational Linguistics

and comprehensive importance evaluation through successive approximations.

Our contributions are summarized as follows:

- We introduce PIP, which is a structured pruning approach designed to iteratively remove low-importance layers identified by PertImport (detailed in Section 3.2) and recomputes gradients on the pruned architecture. PIP can be seamlessly integrated into popular LLM frameworks, such as Hugging Face, with minimal code modifications, offering a lightweight yet theoretically sound implementation.
- Through extensive experiments, we demonstrate PIP's consistent superiority over current state-of-the-art structured pruning methods. Ablation studies confirm that both perturbation (preserving semantic integrity) and the iteration process (dynamic importance reassessment) are essential for high-accuracy pruning. Additionally, comprehensive analyses further provide actionable insights for performance.

2 Related Work

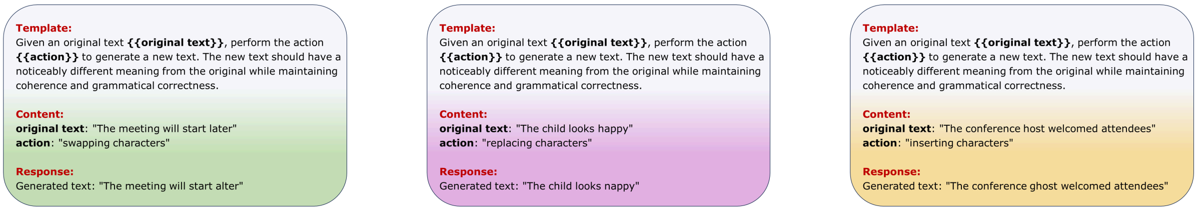
Pruning techniques for LLMs can generally be classified into two categories: unstructured pruning and structured pruning. Unstructured pruning sparsifies weight matrices by setting individual elements to zero, which often requires specialized hardware support. Notable works in this area include (Frantar and Alistarh, 2023; Sun et al., 2024).

Structured pruning, on the other hand, focuses on eliminating predefined units within the model, which makes it more compatible with hardware constraints. The concept of structured pruning for LLMs is introduced by (Wang et al., 2020), which proposes parameterizing each weight matrix via low-rank factorization and actually adaptively removing rank-1 components during training. This pioneering work has truly led to the development of several other methods, such as (Xia et al., 2022) and (Davy, 2024). These methods, however, are primarily designed for compression within specific domains or tasks, falling under the category of task-specific compression. While effective for their intended applications, they often limit the versatility of LLMs as general task solvers.

In contrast, (Ma et al., 2023) introduces a genetic pruning framework called LLM-Pruner, which aims to maintain task-agnostic capabilities while minimizing reliance on the original training dataset. Following the pipeline proposed by (Kwon et al., 2022), LLM-Pruner consists of three stages: Discovery, Estimation, and Recovery. It selectively removes non-critical coupled structures based on dependency analysis (Fang et al., 2023), preserving the core functionality of the model. However, its potential integration with LoRA (Hu et al., 2022) presents several challenges in achieving an optimal balance between efficiency and performance.

Inspired by the approach of LLM-Pruner, the research community has proposed several methods for structured pruning in general tasks. These methods can be broadly categorized into two main types: width pruning and depth pruning (Kim et al., 2024). Width pruning focuses on compressing the weight matrix by reducing its hidden dimension, while depth pruning targets the pruning of layers or blocks within the model. For example, ShearedLLaMA (Xia et al., 2024) implements structured pruning through a combination of targeted pruning and dynamic batch loading. Targeted pruning removes specific layers of the model in an end-to-end fashion to achieve a predefined compression ratio. Dynamic batch loading adjusts the composition of training data batches based on the varying losses from different domains. Although this method achieves competitive performance, it suffers from the same retraining challenges as LLM-Pruner (Ma et al., 2023).

To avoid retraining, which can be resource-intensive and time-consuming, (Men et al., 2024) proposes ShortGPT, a method based on layer importance. It introduces a novel importance metric called Block Influence, which quantifies the importance of each layer by calculating the similarity between the inputs and outputs of each layer. Layers with low importance scores are then removed. Similarly, (Kim et al., 2024) proposes Shortened LLaMA, a block-importance-based method that removes blocks based on a block-level importance metric. Another related work, SLEB (Song et al., 2024) evaluates the importance of Transformer blocks using the similarity between inputs and outputs, and removes the blocks with low importance scores. While these methods are straightforward to understand and implement, they fail to provide strong empirical results and lack rigorous theoretical support. Moreover, these single-view approaches are inherently limited as they neglect the necessity to maintain semantic robustness under adversarial or natural input variations, which is essential for reliable deployment.



Template:
Given an original text **{{original text}}**, perform the action **{{action}}** to generate a new text. The new text should have a noticeably different meaning from the original while maintaining coherence and grammatical correctness.

Content:
original text: "The meeting will start later"
action: "swapping characters"

Response:
Generated text: "The meeting will start alter"

(a) Character Swap

Template:
Given an original text **{{original text}}**, perform the action **{{action}}** to generate a new text. The new text should have a noticeably different meaning from the original while maintaining coherence and grammatical correctness.

Content:
original text: "The child looks happy"
action: "replacing characters"

Response:
Generated text: "The child looks nappy"

(b) Character Replacement

Template:
Given an original text **{{original text}}**, perform the action **{{action}}** to generate a new text. The new text should have a noticeably different meaning from the original while maintaining coherence and grammatical correctness.

Content:
original text: "The conference host welcomed attendees"
action: "inserting characters"

Response:
Generated text: "The conference ghost welcomed attendees"

(c) Character Insertion

Figure 1: Generation of perturbed texts via auxiliary LLM.

In summary, while existing pruning methods offer trade-offs in terms of model efficiency and performance, they often either require retraining or lack solid guarantees in theory, limiting their applicability to real-world scenarios.

3 Methodology

3.1 Text Perturbation

Text perturbation is a data augmentation technique (Guerrero et al., 2023) that introduces variability into textual data by applying a suite of carefully designed transformations to the original text samples.

Inspired by adversarial training (Ganin et al., 2015), we design text perturbations that cause radical semantic shifts while preserving grammatical correctness. Using LLM-powered prompt templates (Figure 1), we propose methods generating perturbed text samples that challenge robustness:

3.1.1 Character Swap

- **Example:** Swapping the characters "l" and "a" in *"later"* gives us *"alter"*. Consequently, the sentence *"The meeting will start later"* becomes *"The meeting will start alter"*.

- **Impact:** In scheduling systems, this perturbation causes rescheduling forms to be generated instead of acknowledging delays, disrupting calendar management. In Q&A systems, models respond with operational directives like "How to adjust the meeting?" instead of factual answers, spreading incorrect procedural guidance and increasing inefficiencies.

3.1.2 Character Replacement

- **Example:** Replacing the character "h" with "n" in *"happy"* results in *"nappy"*. As a result, the sentence *"The child looks happy"* becomes *"The child looks nappy"*.

- **Impact:** In dialogue systems, this perturbation leads to inappropriate suggestions like "Check diaper supplies" instead of emotional support, causing nonsensical interactions in childcare applications. In healthcare chatbots, it can misinterpret "The patient feels nappy" as a clinical symptom, leading to incorrect medical advice and eroding trust in systems.

3.1.3 Character Insertion

- **Example:** Inserting the character "g" to *"host"* yields *"ghost"*. Then, the sentence *"The conference host welcomed attendees"* becomes *"The conference ghost welcomed attendees"*.

- **Impact:** In automated summarization, this perturbation can generate fictional narratives, misrepresenting factual events. In enterprise search, it may retrieve irrelevant documents, introducing noise into enterprise knowledge graphs and decision-making pipelines.

3.2 PertImport: A Perturbation-based Metric for Layer Importance Assessment

Building on the text perturbation framework defined in Section 3.1, we propose PertImport, a novel metric to measure the sensitivity to meaning-altering inputs. The rationale for this metric is grounded in the following analysis:

Consider a pre-trained large language model $\mathcal{M}$ with N layers. Each layer i has parameters $\mathbf{w}_i \in \mathbb{R}^n$. Excluding embedding and head layers, $\mathcal{M}$ can be seen as a mapping function f. For any input s, the function $f(s; \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N)$ generates an output that is consistent with s's semantics.

In Supervised Fine-Tuning (SFT) (Brown et al., 2020), when sample s is used as both input and label for model $\mathcal{M}$, the parameter update rule is:

$$\mathbf{w}_i' = \mathbf{w}_i - \alpha \cdot \nabla_{\mathbf{w}_i} \mathcal{L}(s; \mathbf{w}_{1:N}). \quad (1)$$

Introducing perturbation δs (Section 3.1) to s yields the perturbed sample $s + \delta s$ for SFT:

$$\mathbf{w}_i'' = \mathbf{w}_i - \alpha \cdot \nabla_{\mathbf{w}_i} \mathcal{L}(s + \delta s; \mathbf{w}_{1:N}). \quad (2)$$

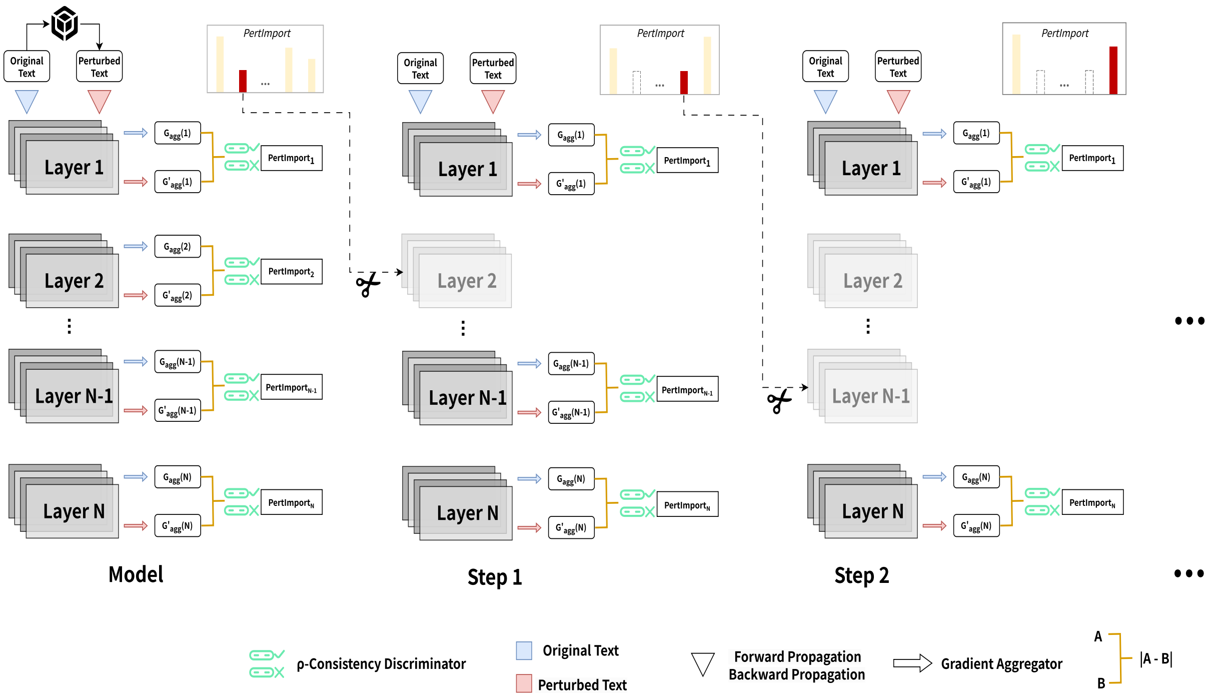


Figure 2: Overview of our PIP method, where an auxiliary LLM generates perturbed text (see Section 3.1). During pruning, the same original and perturbed texts are used to compute PertImport values, determining layer significance. The least significant layers (e.g., the 2nd, $(N-1)$-th, and N-th layers in the first three steps) are iteratively removed.

Here, $\mathbf{w}_i \in \mathbb{R}^n$ denotes the original parameters of the i-th layer, $\mathbf{w}_i^{'} \in \mathbb{R}^n$ and $\mathbf{w}_i^{''} \in \mathbb{R}^n$ denote the updated parameters of the i-th layer. The notation $\mathbf{w}_{1:N}$ represents the collection of parameters from the first layer to the N-th layer. The learning rate is denoted by α , and the loss function $\mathcal{L}(\cdot)$ quantifies the difference between predictions and labels.

Given that f is differentiable and $\delta s \to \mathbf{0}$, we can use the first-order Taylor expansion to approximate the change in the function value. Specifically, we obtain the following approximation:

$$
\begin{aligned}
&f(s+\delta s;\mathbf{w}^{''}_{1:N}) - f(s;\mathbf{w}^{'}_{1:N}) \\
&\approx \nabla_s f \cdot \delta s + \sum_{i,j} (\nabla_{\mathbf{w}_i^{'}} f)_j \cdot (\mathbf{w}_i^{''} - \mathbf{w}_i)_j, \quad (3)
\end{aligned}
$$

where $(\nabla_{\mathbf{w}_i^{'}} f)_j \in \mathbb{R}$ represents the j-th element of the gradient vector $\nabla_{\mathbf{w}_i^{'}} f$ at the i-th layer, and $(\mathbf{w}_i^{''} - \mathbf{w}_i^{'})_j \in \mathbb{R}$ represents the j-th element of the parameter difference $\mathbf{w}_i^{''} - \mathbf{w}_i^{'}$ at the i-th layer.

Subsequently, we utilize Equation (3) to establish an upper bound for the estimation of the difference in output values with and without the perturbation δs. We introduce a constant sequence $\{C_i\}_{i=1}^N$ defined as $C_i = \underset{1\le j\le n}{max} \left|(\nabla_{\mathbf{w}_i^{'}} f)_j\right|$. Incorporating Equations (1), (2), and (3), we arrive at:

$$
\begin{aligned}
&\left| f(s+\delta s;\mathbf{w}^{''}_{1:N}) - f(s;\mathbf{w}^{'}_{1:N}) \right| \\
&\le |\nabla_s f \cdot \delta s| + \\
&C \sum_{i,j} |(\nabla_{\mathbf{w}_i}(\mathcal{L}(s+\delta s;\mathcal{W}) - \mathcal{L}(s;\mathcal{W})))_j| , \quad (4)
\end{aligned}
$$

where $C = \underset{1\le i\le N}{max} \{C_i\} \in \mathbb{R}$ is the maximum value, and $\mathcal{W}$ represents all the parameters of $\mathcal{M}$.

Theorem 1. *To enhance the robustness of the pruned model (defined as its capability to distinguish between s and $s+\delta s$), it's best to select parameters with smaller gradient differences between the perturbed and unperturbed views.*

Proof. Let $\mathbf{Y}$ be a random variable representing the output difference with and without the perturbation δs. Consider removing the parameter at the i_0-th layer and the j_0-th position, i.e., $(\mathbf{w}_{i_0})_{j_0}$. Suppose there exists another parameter $(\mathbf{w}_{i_0^{'}})_{j_0^{'}}$ that has a smaller gradient difference and a higher average probability of detecting the difference between s and $s+\delta s$. Assuming, without loss of generality, that $\mathbf{Y}$ follows a uniform distribution (as an analytical tool based on the principle of maximum entropy), we can derive the expectation of $\mathbf{Y}_{\overline{i_0,j_0}}$

after pruning the parameter $(\mathbf{w}_{i_0})_{j_0}$:

$$\sum_{(i,j)\neq(i_0,j_0)} |(\nabla_{\mathbf{w}_i}(\mathcal{L}(s+\delta s;\mathcal{W})-\mathcal{L}(s;\mathcal{W})))_j| = \frac{2}{C}\mathbb{E}[\mathbf{Y}_{\overline{i_0,j_0}}] - \frac{1}{C}\left|\nabla_s f\cdot\delta s\right|. \tag{5}$$

Similarly, when $(\mathbf{w}_{i_0'})_{j_0'}$ is pruned, we can derive the equation for the expectation of $\mathbf{Y}_{\overline{i_0',j_0'}}$:

$$\sum_{(i,j)\neq(i_0',j_0')} |(\nabla_{\mathbf{w}_i}(\mathcal{L}(s+\delta s;\mathcal{W})-\mathcal{L}(s;\mathcal{W})))_j| = \frac{2}{C}\mathbb{E}[\mathbf{Y}_{\overline{i_0',j_0'}}] - \frac{1}{C}\left|\nabla_s f\cdot\delta s\right|. \tag{6}$$

By Equations (5) and (6), we have $\mathbb{E}[\mathbf{Y}_{\overline{i_0,j_0}}] > \mathbb{E}[\mathbf{Y}_{\overline{i_0',j_0'}}]$. This contradicts the hypothesis that removing $(\mathbf{w}_{i_0'})_{j_0'}$ increases the likelihood of detecting the perturbation. Consequently, the assumption is invalid, implying that the theorem holds. □

Based on Theorem 1, we propose a robustness-aware importance metric, PertImport. For the i-th layer, PertImport quantifies its discriminative sensitivity through the following value:

$$\frac{1}{|\mathcal{D}|}\sum_{s\in\mathcal{D}}|g(\nabla_{\mathbf{w}_i}\mathcal{L}(s+\delta s;\mathcal{W})_{1:n}) - g(\nabla_{\mathbf{w}_i}\mathcal{L}(s;\mathcal{W})_{1:n})|, \tag{7}$$

where $\mathcal{D}$ is a small calibration dataset, and n is the count of parameters in the i-th layer of $\mathcal{M}$. The function $g:\mathbb{R}^n\rightarrow\mathbb{R}$ aggregates gradient information for a specific layer using norms like L_1, L_2, or L_∞, as shown in (Han et al., 2015). See Appendix A for the definitions of these norms.

3.3 PIP: Perturbation-based Iterative Pruning

After assessing layer importance via perturbation, we avoid making premature pruning decisions. Instead, PIP uses a more systematic approach by employing an iterative greedy strategy to progressively prune layers with minimal performance impact.

To enhance the robustness and accuracy of the importance evaluation, we introduce a consistency discriminator that filters out layers with unstable gradient differences. Specifically, it computes the standard deviation across multiple perturbations and excludes layers with high variability.

In summary, by integrating information from both the unperturbed and perturbed views, PIP effectively reduces the stochasticity inherent in pruning and facilitates a more stable and efficient optimization. The details are presented in Algorithm 1, and the overall workflow is illustrated in Figure 2.

Algorithm 1 Detailed Implementation of PIP

```
Input:
  pre-trained LLM $\mathcal{M}$,
  calibration dataset $\mathcal{D}$,
  text perturbation method $\delta(\cdot)$,
  gradient aggregation function $g(\cdot)$,
  # of layers to be pruned $L \in \mathbb{N}^*$,
  consistency threshold $\rho \in \mathbb{R}^+$
Output:
  indices of the pruned layers $\mathcal{P}$,
  pruned model $\mathcal{M}^*$
$\mathcal{D}' \leftarrow \delta(\mathcal{D})$;
$\mathcal{M}^* \leftarrow \mathcal{M}$;
$N_{\mathcal{M}} \leftarrow |\mathcal{M}.layers|$;
$\mathcal{P} \leftarrow \emptyset$;
for $\ell = 1$ to $L$ do
  $\{\boldsymbol{G}_{agg}(i)\}_{i=1}^{N_{\mathcal{M}}-\ell+1} \leftarrow (g \circ SFT)(\mathcal{M}^*, \mathcal{D})$;
  $\{\boldsymbol{G'}_{agg}(i)\}_{i=1}^{N_{\mathcal{M}}-\ell+1} \leftarrow (g \circ SFT)(\mathcal{M}^*, \mathcal{D}')$;
  for $i = 1$ to $N_{\mathcal{M}} - \ell + 1$ do
    if $std(\boldsymbol{G}_{agg}(i) - \boldsymbol{G'}_{agg}(i)) < \rho$ then
      $\boldsymbol{PI}(i) \leftarrow |\boldsymbol{G}_{agg}(i) - \boldsymbol{G'}_{agg}(i)|$;
    else
      $\boldsymbol{PI}(i) \leftarrow +\infty$;
    end if
  end for
  $p_\ell \leftarrow argmin_{1 \leq i \leq N_{\mathcal{M}}-\ell+1} \boldsymbol{PI}(i)$;
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{p_\ell\}$ ;
  $\mathcal{M}^* \leftarrow \mathcal{M}^* \setminus \{layer_{p_\ell}\}$ ;
end for
return $\mathcal{P}$, $\mathcal{M}^*$
```

4 Experiments

4.1 Experimental Setup

4.1.1 Model Selection

To compare with existing methods, we conduct experiments on LLaMA2 (Touvron et al., 2023) and LLaMA3 (Dubey et al., 2024) models of varying sizes. The architectural similarity to other LLMs allows our method, PIP, to generalize effectively to other models. Experiments on additional model architectures are provided in Appendix D.1.

4.1.2 Evaluation and Datasets

We evaluate accuracy using the following datasets: BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-Easy (Clark et al., 2018), ARC-Challenge (Clark et al., 2018), and OpenBookQA (Mihaylov et al., 2018), all of which have been widely utilized in previous structured pruning studies. To ensure a fair comparison across the aforementioned datasets, we use the LM Evaluation Harness framework (Gao et al., 2024) with its default settings for evaluation, without incorporating any shots as demonstrations. In addition, to assess the capability of predicting the next token, we evaluate perplexity (PPL) on the PTB dataset (Marcus et al., 1993), which is an established metric for evaluating the predictive capabilities of LLMs.

4.1.3 Baseline Methods

To show the effectiveness of our PIP method, we compare it with several state-of-the-art structured pruning methods specifically designed for LLMs:

- LLM-Pruner (Ma et al., 2023): A method which uses Taylor-based metrics to prune less important heads in MHA and neurons in FFN.
- SliceGPT (Ashkboos et al., 2024): A method which applies orthogonal transformations. By doing this, it can prune both rows and columns of the weight matrices, which in turn helps to reduce the hidden size within the LLM.
- ShortGPT (Men et al., 2024): A method which identifies redundant layers that have a small similarity between the inputs and outputs, pruning those to reduce the depth.

To compare with baselines, we follow the same experimental settings suggested in their studies.

4.1.4 Experimental Details

Following (Ashkboos et al., 2024), we randomly select a few samples (fewer than 10) from the WikiText2 dataset (Merity et al., 2016) for calibration, ensuring reproducibility with a fixed random seed. We aggregate gradients using the L_2-norm. Experiments are conducted using the Transformers library (Wolf, 2020) on a server with 8 NVIDIA A100 GPUs (80GB VRAM each, totaling 640GB).

4.1.5 Statistics of Pruned Models

Table 1 summarises the key characteristics of the pruned models used in our primary experiments, including parameter count, memory footprint, and Time-Per-Output-Token (TPOT). Evaluations use a randomly sampled sequence from WikiText2 with a fixed output length of 128 tokens. For hardware configuration: LLaMA2-8B and LLaMA3-13B are tested on a single NVIDIA A100 GPU, while LLaMA2-70B and LLaMA3-70B employ tensor parallelism across 4 NVIDIA A100 GPUs. All experiments are executed in half-precision mode.

Model	**Ratio**	**#Params**	**Memory**	**TPOT**
LLaMA3-8B	*Dense*	8.0B	15.0GiB	46.7ms
	19.0%	6.5B	12.1GiB	41.5ms
LLaMA3-70B	*Dense*	70.6B	131.4GiB	266.2ms
	19.4%	56.9B	99.5GiB	223.5ms
LLaMA2-13B	*Dense*	13.0B	24.4GiB	73.4ms
	19.5%	10.5B	19.6GiB	58.9ms
LLaMA2-70B	*Dense*	69.0B	128.5GiB	269.7ms
	19.9%	55.3B	96.6GiB	217.8ms

Table 1: Statistics of base and pruned models. *"Dense"* denotes the base model. "Ratio" is the pruning ratio, calculated as (#Pruned Params)/(#Base Params).

4.2 Zero-shot Performance

We compare PIP with baselines on zero-shot performance (see Table 2). On average, PIP retains over 85% of the base model's accuracy across benchmarks, with an approximate pruning ratio of 20%. In some cases, its performance is within 5% of the base model's, showing its ability to preserve the crucial aspects. PIP also consistently outperforms baselines, making it a superior LLM pruning technique. For more details, see Appendix C.

4.3 Ablation Analysis

To investigate the critical components of PIP, we conduct ablation studies on LLaMA2-13B, focusing on perturbation and greedy-search iteration. As shown in Table 3, we evaluate three pruning ratios (9.8%, 19.5%, 29.2%) under three configurations: (1) Full PIP implementation, (2) Perturbation-only, and (3) Iteration-only. The performance is measured using the PPL metric on the PTB dataset.

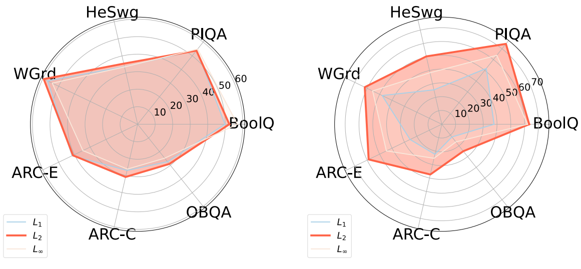
Our results show that perturbation and iteration work synergistically. At a 29.2% pruning ratio, the combined approach achieves optimal performance, while disabling iteration or perturbation degrades performance, with perturbation's absence causing more severe deterioration. In addition, similar trends are observed at lower pruning ratios.

Model	Method	PPL↓	BoolQ↑	PIQA↑	HeSwg↑	WGrd↑	ARC-E↑	ARC-C↑	OBQA↑	Avg.↑
LLaMA3-8B	*Dense*	10.6	81.4	79.7	60.2	72.5	80.1	50.5	34.8	65.6
	LLM-Pruner	56.5	63.5	69.5	42.6	62.4	52.3	29.5	**27.0**	49.5
	SliceGPT	72.3	40.8	65.4	39.8	63.2	**59.5**	29.4	23.8	46.0
	ShortGPT	67.9	65.2	68.9	**45.6**	**69.4**	57.2	**36.5**	25.4	52.6
	PIP (Ours)	**56.3**	**70.9**	**69.6**	44.7	**69.4**	57.9	35.1	26.8	**53.5**
LLaMA3-70B	*Dense*	8.2	85.2	82.4	66.4	80.3	86.9	60.3	38.2	71.4
	LLM-Pruner	-	-	-	-	-	-	-	-	-
	SliceGPT	78.2	59.4	69.6	44.4	72.1	69.7	41.1	**30.2**	55.2
	ShortGPT	13.9	**80.9**	76.0	**57.1**	60.5	**77.4**	**47.9**	19.0	59.8
	PIP (Ours)	**12.5**	79.0	**78.6**	56.1	**73.3**	75.7	45.3	29.6	**62.5**
LLaMA2-13B	*Dense*	28.9	80.6	79.1	60.0	72.4	79.4	48.5	35.2	65.0
	LLM-Pruner	150.2	57.7	60.3	31.9	53.9	37.4	22.9	15.8	40.1
	SliceGPT	64.3	38.2	65.0	39.5	**65.5**	**61.3**	33.4	**28.0**	47.3
	ShortGPT	44.6	49.8	55.5	39.3	57.1	49.3	29.9	25.4	43.8
	PIP (Ours)	**41.8**	**63.3**	**74.5**	**50.5**	62.0	58.8	**37.4**	25.0	**53.1**
LLaMA2-70B	*Dense*	14.4	76.6	81.1	64.0	77.0	77.8	51.2	34.8	66.1
	LLM-Pruner	-	-	-	-	-	-	-	-	-
	SliceGPT	33.9	70.1	76.3	52.7	**76.6**	**76.4**	**47.0**	**32.6**	61.7
	ShortGPT	18.5	73.5	73.9	56.0	72.5	66.7	39.2	26.8	58.4
	PIP (Ours)	**17.2**	**80.7**	**77.0**	**57.8**	73.4	70.7	43.6	29.4	**61.8**

Table 2: Zero-shot performance of LLMs with approximately 20% pruning ratio. "*Dense*" is the original unpruned model. "↑" means higher is better and "↓" means lower is better. Bold values denote the best performance among pruned models, underlined values the second-best. Abbreviations: HellaSwag (HeSwg), WinoGrande (WGrd), ARC-Easy (ARC-E), ARC-Challenge (ARC-C), OpenBookQA (OBQA). "Avg." represents the average score across the seven benchmarks. "-" indicates incompatibility between pruning methods and specific LLMs.

Ratio	Perturbation	Iteration	PPL↓
9.8%	✔	✔	30.7
	✔	✘	31.3
	✘	✔	43.4
19.5%	✔	✔	41.8
	✔	✘	42.0
	✘	✔	81.9
29.2%	✔	✔	53.3
	✔	✘	67.4
	✘	✔	99.8

Table 3: Ablation studies of LLaMA2-13B. "Ratio": Pruning Ratio; "✔": Enabled; "✘": Disabled.



(a) LLaMA3 (b) LLaMA2

Figure 3: Zero-shot performance of the pruned LLM using PIP, across various gradient aggregation strategies.

4.4 More Analysis

4.4.1 Effect of Gradient Aggregation

In this section, we explore different gradient aggregation strategies for model pruning, as shown in Figure 3. Overall, the L_2-norm outperforms the L_1 and L_∞ norms by assigning more weight to larger gradients and mitigating extreme values. Therefore, we recommend the L_2-norm as the default for general use. For more details, see Appendix D.2.

4.4.2 Effect of Calibration Data Volume

This section examines how calibration data volume affects the zero-shot performance of LLMs pruned using PIP. We use character swap and replacement

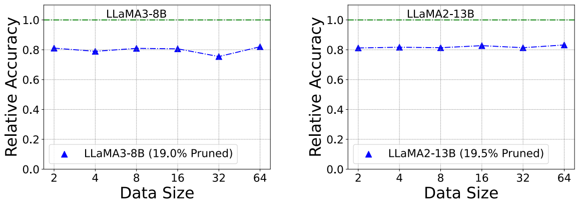


Figure 4: Experiments of performance at various data sizes. “Data Size” is the number of samples in the calibration dataset. “Relative Accuracy” is the ratio of the average accuracy of the pruned LLM on various benchmarks to the average accuracy of the Dense model.

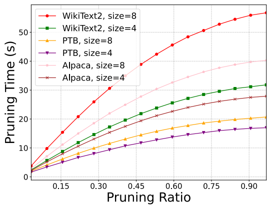


Figure 5: Impact of pruning ratio on pruning time.

techniques with the L_2-norm for gradient aggregation. Figure 4 shows that PIP achieves high efficiency with minimal calibration samples, demonstrating “few-shot” learning capabilities. This suggests that PIP can deliver strong performance without requiring large datasets, addressing a common limitation in practical applications. For more details, readers are referred to Appendix D.3.

4.4.3 Pruning Time Analysis

Let the average time for forward propagation in a layer be t_f, and backward propagation be t_b. Using the PIP method, the time required to prune m layers of model $\mathcal{M}$ is denoted as $PT_{\mathcal{M}}(m)$. The second finite difference of pruning time is calculated as:

$$\triangle^2 PT_{\mathcal{M}}(m) = -(t_f + t_b), \tag{8}$$

where $\triangle^2$ denotes the second finite difference. This result implies constant second differences in $PT_{\mathcal{M}}(m)$, a hallmark of quadratic sequences. For a detailed proof, please see Appendix B. This is consistent with Figure 5, which shows quadratic scaling of PIP pruning time for the LLaMA3-8B model across datasets and sample sizes.

4.4.4 Effect of Pruning Ratio

As shown in Table 4, increasing the pruning ratio from 29.2% to 39.0% reduces memory use and latency, while causing only modest increases in perplexity and minor decreases in accuracy. Beyond 39.0%, further pruning yields diminishing resource savings but leads to exponential increases in perplexity and substantial drops in accuracy. These results demonstrate that a pruning ratio of approximately 39.0% achieves the optimal balance between computational efficiency and model quality.

Ratio	Memory	TPOT	PPL↓	Acc↑
29.2%	17.2GiB	46.8ms	53.3	48.9
39.0%	14.9GiB	41.2ms	95.8	44.7
48.7%	12.5GiB	35.3ms	379.0	38.5
58.5%	10.1GiB	29.1ms	1030.2	34.2
68.2%	6.5GiB	20.1ms	2549.7	33.1
78.0%	4.2GiB	14.5ms	22026.5	34.1

Table 4: Performance of PIP-pruned LLaMA2-13B. “Ratio” is the pruning ratio, and “Acc” is the average accuracy of the pruned model on various benchmarks.

4.4.5 Orthogonal to Quantization

Ratio	Memory	TPOT	PPL↓	Acc↑
9.8%	11.4GiB	181.8ms	30.4	57.8
19.5%	10.2GiB	160.9ms	42.6	53.0
29.2%	9.0GiB	120.8ms	68.7	48.6
39.0%	7.7GiB	102.1ms	114.7	44.0

Table 5: Performance of quantized and PIP-pruned LLaMA2-13B. “Ratio” refers to the pruning ratio, and “Acc” represents the average accuracy of the quantized and pruned model on various benchmarks.

PIP is orthogonal to other compression approaches. As shown in Table 5, combining 8-bit quantization with PIP pruning achieves substantial memory reduction and speed-ups while maintaining acceptable perplexity and accuracy. This confirms that quantization and PIP pruning are fully orthogonal, enabling effective stacked compression.

5 Conclusion and Future Work

In this paper, we propose PIP, a novel perturbation-based iterative structured pruning method that unifies unperturbed and perturbed model views. It demonstrates theoretical rigor and SOTA performance across multiple benchmarks.

Future work could explore adaptive perturbation mechanisms, like dynamic scaling or task-specific perturbations, to enhance the precision of pruning. Additionally, we aim to collaborate with industry partners to deploy PIP in real-world applications, such as edge computing systems, to validate its practicality and address potential issues.

Limitations

A key limitation of PIP is its current incompatibility with multimodal models. Tailored for text, PIP's perturbation and gradient analysis aren't easily adaptable to diverse data types like images or audio. This poses a challenge due to the unique processing needs of different data forms. Consequently, to broaden PIP's applicability, it's crucial to develop methods that handle multimodal data complexities. Overcoming this limitation will enhance PIP's utility in various AI applications.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (Grant No. 62272334, 6257073827).

References

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

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. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jin Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. 2021. 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, Online. Association for Computational Linguistics.

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

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

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

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

Steven Davy. 2024. Tailored-llama: Optimizing few-shot learning in pruned llama models with task-specific prompts. In *European Conference on Artificial Intelligence*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10323–10337. PMLR.

Yaroslav Ganin, E. Ustinova, Hana Ajakan, Pascal Germain, H. Larochelle, François Laviolette, Mario Marchand, and Victor S. Lempitsky. 2015. Domain-adversarial training of neural networks. In *Journal of machine learning research*.

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

Jesus Guerrero, Gongbo Liang, and Izzat Alsmadi. 2023. Adversarial text perturbation generation and analysis. *2023 3rd Intelligent Cybersecurity Conference (ICSC)*, pages 67–73.

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

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

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.

Woosuk Kwon, Sehoon Kim, Michael W Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. 2022. A fast post-training pruning framework for transformers. *Advances in Neural Information Processing Systems*, 35:24101–24116.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, pages 87–100.

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.

Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

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. *Preprint*, arXiv:1609.07843.

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

Haojie Pan, Chengyu Wang, Minghui Qiu, Yichang Zhang, Yaliang Li, and Jun Huang. 2021. Meta-KD: A meta knowledge distillation framework for language model compression across domains. 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 3026–3036, Online. Association for Computational Linguistics.

Rajarshi Saha, Varun Srivastava, and Mert Pilanci. 2023. Matrix compression via randomized low rank and low precision factorization. *Advances in Neural Information Processing Systems*, 36.

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

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. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 46136–46155. PMLR.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2024. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Siqi Sun, Zhe Gan, Yuwei Fang, Yu Cheng, Shuohang Wang, and Jingjing Liu. 2020. Contrastive distillation on intermediate representations for language model compression. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 498–508, Online. Association for Computational Linguistics.

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

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2020. Structured pruning of large language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6151–6162. Association for Computational Linguistics.

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

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2024. Sheared llama: Accelerating language model pre-training via structured pruning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 1513–1528. Association for Computational Linguistics.

Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guangyu Sun. 2023. Asvd: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*.

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

A Definition of Norm

In this section, we provide the formal definitions of the norms used in the main text, specifically the L_1, L_2, and L_∞ norms. These norms are commonly used in various mathematical and computational contexts to measure the magnitude of vectors.

A.1 L_1-norm

For a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, the L_1-norm (also known as the Manhattan norm or Taxicab norm) is defined as:

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|. \tag{9}$$

This norm represents the sum of the absolute values of the vector components.

A.2 L_2-norm

The L_2-norm (also known as the Euclidean norm) is perhaps the most commonly used norm. For a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, it is defined as:

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}. \tag{10}$$

This norm represents the Euclidean length of the vector, which is the geometric distance from the origin to the point represented by the vector in n-dimensional space.

A.3 L_∞-norm

The L_∞-norm (also known as the maximum norm or infinity norm) is defined as:

$$\|\mathbf{x}\|_\infty = \max_i |x_i|. \tag{11}$$

This norm represents the maximum absolute value among the vector components.

Model	Ratio	Pruned Layers
LLaMA3-8B	19.0%	22, 18, 23, 28, 19, 21, 27
LLaMA3-70B	19.4%	06, 11, 46, 50, 75, 34, 49, 04, 20, 25, 48, 57, 60, 56, 55, 58
LLaMA2-13B	19.5%	36, 31, 28, 13, 35, 25, 38, 23
LLaMA2-70B	19.9%	59, 29, 67, 26, 60, 61, 50, 43, 58, 57, 31, 17, 56, 74, 62, 49

Table 6: Pruned models by PIP in our main experiments.

B Proof of Constant Second Differences

We prove that if the second finite difference of pruning time satisfies $\triangle^2 PT_{\mathcal{M}}(m) = -(t_f + t_b)$ for all m, then $PT_{\mathcal{M}}(m)$ must be a quadratic sequence.

Proof. Let $\triangle^2 PT_{\mathcal{M}}(m) \equiv -(t_f + t_b)$ be constant. By definition, the second difference is the difference of consecutive first differences:

$$\triangle PT_{\mathcal{M}}(m+1) - \triangle PT_{\mathcal{M}}(m) = -(t_f + t_b). \tag{12}$$

This implies that the first differences $\triangle PT_{\mathcal{M}}(m)$ form an arithmetic sequence with common difference $-(t_f + t_b)$. Explicitly, for the initial first difference $\triangle PT_{\mathcal{M}}(0)$, we have:

$$\triangle PT_{\mathcal{M}}(m) = \triangle PT_{\mathcal{M}}(0) - (t_f + t_b)m. \tag{13}$$

The pruning time $PT_{\mathcal{M}}(m)$ is then the cumulative sum of these first differences:

$$\begin{aligned} & PT_{\mathcal{M}}(m) \\ & = PT_{\mathcal{M}}(0) + \sum_{k=0}^{m-1} \triangle PT_{\mathcal{M}}(k) \\ & = PT_{\mathcal{M}}(0) + \sum_{k=0}^{m-1} [\triangle PT_{\mathcal{M}}(0) - (t_f + t_b)k] \\ & = PT_{\mathcal{M}}(0) + m\triangle PT_{\mathcal{M}}(0) \\ & \quad - (t_f + t_b)\frac{m(m-1)}{2}. \end{aligned} \tag{14}$$

Letting $a \equiv -\frac{t_f + t_b}{2}$, $b \equiv \triangle PT_{\mathcal{M}}(0) + \frac{t_f + t_b}{2}$, and $c \equiv PT_{\mathcal{M}}(0)$, this simplifies to:

$$PT_{\mathcal{M}}(m) = am^2 + bm + c, \tag{15}$$

which is explicitly a quadratic function of m. □

C The Main Experiment

Our main experiments validate PIP across four model scales (8B-70B parameters), with pruned models in Table 6. All configurations use fewer than 10 calibration samples from WikiText2 and employ the L_2-norm gradient aggregation strategy.

D More Analysis

D.1 Generalization to Other Series' LLMs

We aim to demonstrate the generalization capability of our PIP pruning method beyond the LLaMA models. We have conducted extensive experiments on the OPT series of models with varying scales (ranging from 1.3B to 13B parameters) to validate the effectiveness and robustness of our approach. The results are summarized in Table 7.

D.2 Effect of Gradient Aggregation

We systematically evaluate gradient aggregation norms (L_1, L_2, L_∞) across model architectures under controlled pruning settings. The LLaMA3-8B experiments employ 8 calibration samples with character swap perturbation and 7-layer pruning, while LLaMA2-13B utilizes 4 samples with character replacement perturbation and 8-layer pruning. Table 8 presents the zero-shot benchmark results under various gradient aggregation strategies.

D.3 Effect of Calibration Data Volume

For LLaMA3-8B, experiments use WikiText2 as the calibration dataset. Layer importance scores are computed through L_2-norm aggregation of gradients, computing layer importance via L_2-norm aggregation of gradients under character swap perturbation. The LLaMA2-13B configuration maintains the L_2-norm aggregation and dataset while employing character-replacement perturbation. The complete lists of pruned layers (0-indexed) under each condition are provided in Table 9.

D.4 Effect of Pruning Ratio

We systematically evaluate the performance degradation of LLaMA3 and LLaMA2 models under increasing pruning ratios (10%–30%), as detailed in Table 10. As pruning ratios rise (10%→30%), both LLaMA3 and LLaMA2 show sharp performance drops. Commonsense tasks (ARC-Challenge, OBQA) decline the most, while language tasks (BoolQ, WinoGrande) are more robust.

D.5 Effect of Text Perturbation Method

The experiments are conducted on LLaMA3-8B under three text perturbation methods (*Swap*, *Replace*, *Insert*) with fixed configurations: 4 calibration samples, L_2-norm gradient aggregation. Table 11 compares the zero-shot performance across various benchmarks under different text perturbation methods. All perturbation methods degrade LLaMA3's zero-shot performance, with *Replace* showing the least decline.

D.6 Effect of Calibration Dataset

The experiments evaluate the LLaMA3-8B model under three calibration datasets (WikiText2, PTB, Alpaca) with fixed configurations: 8 calibration samples, swap-based text perturbation, L_2-norm gradient aggregation. Table 12 shows that PIQA and WinoGrande are stable across datasets, while ARC-Challenge declines sharply on PTB.

D.7 Short Generations from Pruned Models

The examples in Table 13 clearly demonstrate that the pruned LLMs, despite undergoing the PIP pruning method, retain robust language expression capabilities. For instance, the pruned LLaMA3 model is capable of generating a coherent and insightful statement about the impact of AI on the business world, highlighting its potential to change the future of work. Similarly, the pruned LLaMA2 model provides a comprehensive introduction to NLP, emphasizing its role in processing human languages and extracting valuable insights from unstructured text data. These examples validate our pruning methodology's effectiveness in preserving core linguistic competencies, particularly in: technical semantic preservation, syntactic coherence across multi-clause constructions, and domain-appropriate register consistency.

D.8 Long Generations from Pruned Models

Tables 14 and 15 demonstrate that models pruned via the PIP method retain text generation quality comparable to their dense counterparts. Pruned models preserve logical coherence and domain-specific knowledge (e.g., technical terminology and contextual reasoning), with minimal degradation in fluency and factual accuracy, validating PIP's capability to identify and retain critical layers.

Model	Ratio	Memory	TPOT	PPL↓
OPT-1.3B	*Dense*	2.4508GiB	29.5556ms	16.9137
	12.5%	2.1694GiB	27.5401ms	20.4018
	20.8%	1.9818GiB	26.0304ms	38.7159
	29.2%	1.7942GiB	24.5175ms	98.8645
OPT-2.7B	*Dense*	4.9395GiB	41.3229ms	15.1614
	12.5%	4.3533GiB	40.8443ms	19.4676
	21.9%	3.9137GiB	30.1659ms	24.9969
	31.2%	3.4740GiB	23.1771ms	44.5617
OPT-6.7B	*Dense*	12.4024GiB	45.1802ms	13.1724
	12.5%	10.9020GiB	35.8956ms	15.6426
	21.9%	9.7767GiB	33.8391ms	19.7741
	31.2%	8.6514GiB	31.9243ms	35.2512
OPT-13B	*Dense*	23.9415GiB	71.5918ms	12.3743
	10.0%	21.5972GiB	57.4137ms	14.0220
	20.0%	19.2530GiB	50.1614ms	17.7254
	30.0%	16.9087GiB	46.7974ms	23.8522

Table 7: Pruning Results on OPT Models.

Model	Norm	Pruned Layers	BoolQ↑	PIQA↑	HeSwg↑	WGrd↑	ARC-E↑	ARC-C↑	OBQA↑	Avg.↑
LLaMA3	*Dense*	–	81.4	79.7	60.2	72.5	80.1	50.5	34.8	65.6
	L_1	23, 18, 29, 22, 11, 17, 10	49.8	54.1	33.7	57.1	41.5	27.0	27.4	41.5
	L_2	23, 22, 18, 19, 28, 20, 31	52.3	53.8	35.5	59.4	41.0	30.9	29.0	43.1
	L_∞	20, 13, 18, 23, 12, 28, 22	56.9	54.9	33.6	54.1	34.7	26.3	24.8	40.7
LLaMA2	*Dense*	–	80.6	79.1	60.0	72.4	79.4	48.5	35.2	65.0
	L_1	00, 34, 35, 09, 12, 27, 10, 33	37.9	51.7	25.6	48.0	25.5	22.4	11.8	31.8
	L_2	36, 31, 28, 13, 35, 25, 38, 23	63.3	74.5	50.5	62.0	58.8	37.4	25.0	53.1
	L_∞	30, 27, 24, 03, 28, 29, 17, 13	62.1	63.9	38.3	55.6	44.6	25.3	16.4	43.8

Table 8: Zero-shot performance under different gradient aggregation strategies.

Model	**Cnt.**	**Pruned Layers**	**BoolQ↑**	**PIQA↑**	**HeSwg↑**	**WGrd↑**	**ARC-E↑**	**ARC-C↑**	**OBQA↑**	**Avg.↑**
LLaMA3	*Dense*	–	81.4	79.7	60.2	72.5	80.1	50.5	34.8	65.6
	2	23, 22, 18, 30, 25, 27, 26	74.0	68.7	42.8	68.9	57.7	34.9	25.4	53.2
	4	31, 22, 24, 25, 30, 28, 07	62.3	70.1	41.8	61.2	61.8	37.1	28.4	51.8
	8	23, 18, 22, 19, 28, 21, 20	76.0	69.1	44.5	67.1	55.7	34.4	24.8	53.1
	16	23, 31, 24, 21, 25, 22, 18	70.5	68.4	45.5	63.6	56.7	38.3	27.4	52.9
	32	31, 30, 22, 10, 05, 17, 21	65.9	68.0	40.1	59.7	56.3	31.3	25.4	49.5
	64	31, 25, 23, 22, 28, 24, 19	66.2	70.2	47.8	65.4	58.5	39.5	28.8	53.8
LLaMA2	*Dense*	–	80.6	79.1	60.0	72.4	79.4	48.5	35.2	65.0
	2	09, 25, 34, 21, 14, 19, 31, 06	62.5	73.9	49.0	60.7	65.2	32.7	25.4	52.8
	4	36, 31, 28, 13, 35, 25, 38, 23	63.3	74.5	50.5	62.0	58.8	37.4	25.0	53.1
	8	34, 36, 31, 21, 26, 22, 07, 05	64.8	73.0	50.3	64.1	60.4	33.8	24.2	52.9
	16	29, 08, 27, 30, 25, 35, 23, 17	62.4	74.0	51.6	64.3	60.9	37.3	26.0	53.8
	32	31, 33, 17, 23, 32, 19, 16, 14	69.4	73.8	49.9	62.1	57.2	31.9	25.8	52.9
	64	36, 33, 17, 30, 24, 27, 13, 31	62.8	74.4	52.4	63.5	61.7	36.9	27.0	54.1

Table 9: Zero-shot performance across different calibration sample counts (Cnt.).

Model	**Ratio**	**Pruned Layers**	**BoolQ↑**	**PIQA↑**	**HeSwg↑**	**WGrd↑**	**ARC-E↑**	**ARC-C↑**	**OBQA↑**	**Avg.↑**
LLaMA3	*Dense*	–	81.4	79.7	60.2	72.5	80.1	50.5	34.8	65.6
	10.9%	22, 18, 23, 28	78.8	75.2	53.2	72.8	71.0	43.3	29.4	60.5
	19.0%	22, 18, 23, 28 19, 21, 27	70.9	69.6	44.7	69.4	57.9	35.1	26.8	53.5
	29.9%	22, 18, 23, 28 19, 21, 27, 10 25, 06, 31	43.5	63.0	36.4	56.2	43.3	30.0	25.0	42.5
LLaMA2	*Dense*	–	80.6	79.1	60.0	72.4	79.4	48.5	35.2	65.0
	9.8%	36, 31, 28, 13	63.0	76.1	56.0	66.1	67.4	41.6	30.4	57.2
	19.5%	36, 31, 28, 13 35, 25, 38, 23	63.3	74.5	50.5	62.0	58.8	37.4	25.0	53.1
	29.2%	36, 31, 28, 13 35, 25, 38, 23 17, 26, 29, 30	62.4	71.3	45.9	58.4	46.9	33.9	23.8	48.9

Table 10: Zero-shot performance comparison across different pruning ratios. “Ratio” refers to the pruning ratio.

Model	TPM	Pruned Layers	BoolQ↑	PIQA↑	HeSwg↑	WGrd↑	ARC-E↑	ARC-C↑	OBQA↑	Avg.↑
LLaMA3	*Dense*	–	81.4	79.7	60.2	72.5	80.1	50.5	34.8	65.6
	Swap	23, 22, 31, 25, 16, 26, 30	63.9	69.5	44.7	63.6	58.7	36.0	31.2	52.5
	Replace	22, 18, 23, 28, 19, 21, 27	70.9	69.7	44.8	69.6	58.0	35.1	27.4	53.6
	Insert	23, 18, 31, 28, 22, 27, 03	71.3	70.2	47.6	64.4	57.8	36.3	25.0	53.2

Table 11: Zero-shot performance under different text perturbation methods. "TPM": Text Perturbation Method.

Model	CD	Pruned Layers	BoolQ↑	PIQA↑	HeSwg↑	WGrd↑	ARC-E↑	ARC-C↑	OBQA↑	Avg.↑
LLaMA3	*Dense*	–	81.4	79.7	60.2	72.5	80.1	50.5	34.8	65.6
	WikiText2	31, 28, 26, 29, 23, 25, 30	80.6	69.3	42.8	62.3	61.2	39.2	30.6	55.1
	PTB	23, 18, 28, 22, 19, 21, 13	72.1	70.6	44.6	66.6	54.8	32.9	24.0	52.2
	Alpaca	31, 30, 29, 28, 23, 25, 27	80.5	70.7	42.7	63.1	60.5	38.6	30.4	55.2

Table 12: Zero-shot performance comparison across different calibration datasets. "CD": Calibration Dataset.

Model	✂	Example
LLaMA3	✘	AI will be widely used in all areas of business, says Bjarne Corydon. The minister of business and growth, Bjarne Corydon is excited about what AI is doing to the Danish business world.
	✔	AI will change the world of work: this is what the Gartner report reveals. In the coming years, AI will have a large impact on the entire business, as well as the daily life of employees.
LLaMA2	✘	NLP is a way of applying computational processing to natural human languages that we use to communicate with each other. This course will give you an introduction to NLP, and how it can be leveraged to derive useful insights from unstructured text data.
	✔	NLP is the science that focuses on linguistic data. It is an AI methodology that combines computer science and artificial intelligence. This science focuses on linguistic input, output, understanding, processing, or interaction. It is used to process human languages.

Table 13: Examples from the original model and the pruned model. "✂" indicates whether it is pruned. The underline highlights the prompt provided to the LLM.

Model	✂	Example
LLaMA3	✘	Large Language Models will not replace us with AI – but they will help us automate our day-to-day work. AI has always been portrayed as the main reason behind replacing human labour with machine labour in many fields. For example, driverless cars and automated machines that replace employees are all examples of how AI will “replace” humans. However, AI has never replaced us, but has helped us automate many of our day-to-day operations and we will continue to see that trend of using AI in automation. One of the main reasons why AI has helped us improve our day-to-day work is that it is able to process, analyse and understand massive amounts of data. Large Language Models (LLMs) are a good example of this. LLMs can be a very useful tool to help us become more productive and improve our workflows. LLMs can also help us in many different ways, depending on what we are looking for, and they can be used in different ways. LLMs are able to process, analyse and understand massive amounts of text in many different ways. For example, they can be used to find information in a large amount of text, to do sentiment analysis, or to create new text. LLMs can also be used to do translation or to do a search in a large number of different languages. There are many tools available that allow us to work with LLMs, such as GPT-3 from OpenAI, which is one of the most powerful tools in the LLM space. LLMs can also be used to do sentiment analysis of text and we can use it in many different ways to help us become more productive and to improve our workflow. We can use LLMs to understand what is the sentiment of a large amount of text, to detect negative or positive sentiments in text, or to help us make decisions in our daily work. There are many ways to use LLMs to automate our day-to-day work. For example, we can use LLMs to help us become more productive and to improve our workflow by doing sentiment analysis. We can use LLMs to analyse large amounts of data and to analyse the sentiment of a large amount of text.
	✔	Large Language Models will be the next killer apps. The way these LLMs are developed is not only more efficient but also very innovative in terms of how the LLMs are built, the use of compute to build the LLM, and how they train and generate LLMs. What’s more, we are seeing a change in how LLMs are used for tasks, with an increasing number of LLMs being used to train other AI models. This has opened up a huge market for LLMs, creating a great opportunity for companies like Cytora and Credence to tap into. In this article, we explore the benefits of this new wave of AI development, and how companies can capitalize on the opportunities presented by these new AI innovations. Large language models (LLMs) have been a focus of AI innovation for some time, and there are many ways they can be developed. However, the most effective and efficient way to develop LLMs is to use a lab environment that is specific to LLM development. There are two main approaches to developing LLMs: traditional computing methods, which require large amounts of computing power and data, and cloud-based LLMs, which can be developed much more quickly and inexpensively. There are several reasons why it is better to develop LLMs in an AI innovation lab than in a traditional development environment. First, the cloud-based LLM development process allows for much greater scale. With only a few machines in use, LLM developers can achieve huge breakthroughs, such as improving text generation accuracy. Second, cloud-based LLM development also allows for a much more nimble process. Instead of waiting for computers to complete tasks one at a time, developers can use a variety of machines to speed up their work. This speeds up the entire process, making it easier to get feedback from stakeholders and improving the overall experience of working with LLMs. How LLMs Can Be Used to Train Other AI Models. Large language models (LLMs) have been a driving force in the development of Artificial Intelligence (AI) over the years.

Table 14: Examples generated by the original model and the pruned model. “✂” indicates whether it is pruned. The underline highlights the prompt provided to the LLM.

Model	✂	Example
LLaMA2	✘	Large Language Models will be ubiquitous in business and government within 5-7 years, predicts PwC by James Sanders in Artificial Intelligence on January 30, 2023, 11:41 AM PST. Large Language Models such as OpenAIś GPT-3 are becoming increasingly prevalent for a wide variety of business applications, the consulting firm said. Image: Bjorn Rune Lie, Getty Images. The adoption rate of large language models—AI models trained on massive amounts of natural language data—will increase rapidly as businesses look to improve customer engagement and operational efficiencies. According to a report by PwC, 65% of senior business executives indicated that large language models are a top investment area, and 53% said that this is the largest investment area for AI technology in the coming year. "Making AI more accessible, through advances such as large language models, is essential to the democratization of the technology, which could bring a range of business benefits to organizations," said John Garner, global AI leader at PwC, in a statement accompanying the report. SEE: The ethical dilemmas of AI (ZDNET/TechRepublic special feature) | Download the free PDF version (TechRepublic) PwCś 2023 Global Artificial Intelligence Survey found that the use of natural language processing technologies is rapidly growing in the workplace. The technology is being used by 36% of respondents to "identify risks or anomalies in client engagement," and by 35% of respondents to "increase the effectiveness and efficiency of operations." AI is also used to drive productivity: 25% of respondents indicated that AI is used to "enable the creation of new product and service offerings." The growing ubiquity of large language models in the workplace is also a factor driving widespread awareness: 55% of business executives indicated that large language models are "extremely important" to business success and operations, and 79% of employees said that they know of AI, a slight increase from last yearś survey.
	✔	Large Language Models will be used for everything from translation to financial services to healthcare. There are endless benefits to utilizing LLMs like ChatGPT, like saving money and time on repetitive tasks that are time-consuming or impossible to automate, and getting better answers than we could on our own. As AI gets more accessible to average users, a more accessible education in AI is more important than ever. The ChatGPT revolution has arrived. If you are a regular user of Google search or Twitter, you've probably already noticed. ChatGPT was released to the public in November 2022 by an organization called OpenAI. The platform uses artificial intelligence to create intelligent chatbot responses to user prompts. As a result, it has the potential to revolutionize the way we interact with technology. With ChatGPT, you can write essays, do your taxes, and ask questions like "Who wrote Romeo and Juliet?" or "Where is the nearest Walmart?" in chat format. It's accessible, fast, and most importantly, free. It's clear that LLMs are a powerful tool with enormous applications and capabilities. But what, exactly, is an LLM, and why is this technology so different from other language models that have been developed? What Are Large Language Models? An LLM is a type of language modeling that produces language using machine learning algorithms based on large amounts of training data. It's a relatively new development in the field of artificial intelligence, and it has become increasingly popular in recent years due to the advances in natural language processing and understanding that have been made. One of the main reasons for this is that large language models are capable of producing language that is more sophisticated and accurate than ever before. There are a few key reasons why large language models are different from other language models: They are based on very large amounts of data: This is the key characteristic that sets LLMs apart from other language models. Because of the amount of data used, these models can be trained to perform more complex tasks and generate more human-like text.

Table 15: Examples generated by the original model and the pruned model. "✂" indicates whether it is pruned. The underline highlights the prompt provided to the LLM.