OptiPrune: Effective Pruning Approach for Every Target Sparsity

Nguyen-Khang Le[1], Ryo Sato[2], Dai Nakashima[2], Takeshi Suzuki[2], Minh Le Nguyen[1]

[1]Japan Advanced Institute of Science and Technology, [2]RICOH

Correspondence: lnkhang@jaist.ac.jp, nguyenml@jaist.ac.jp

Abstract

Large language models (LLMs) have achieved notable success across various tasks but are hindered by their large size and high computational demands. Post-training pruning (PTP) offers a promising solution by reducing model size through parameter removal while preserving performance. However, current PTP methods perform optimally only within specific sparsity ranges. This paper presents two key findings: (1) Layerwise uniform sparsity is effective at low sparsity, while non-uniform sparsity excels at high levels; (2) Relative importance-based pruning works best at low sparsity, whereas Hessian-based weight reconstruction is superior at high sparsity. We design and conduct experiments to validate these findings. Based on these insights, we introduce OPTIPRUNE, a robust pruning method effective across all sparsity levels. OPTIPRUNE adapts non-uniform sparsity with adaptive deviation and employs a threshold to select the optimal pruning strategy. Empirical results across diverse datasets, architectures, and languages validate its performance and robustness. These findings provide valuable directions for future LLM pruning research. Our code and data are publicly available.

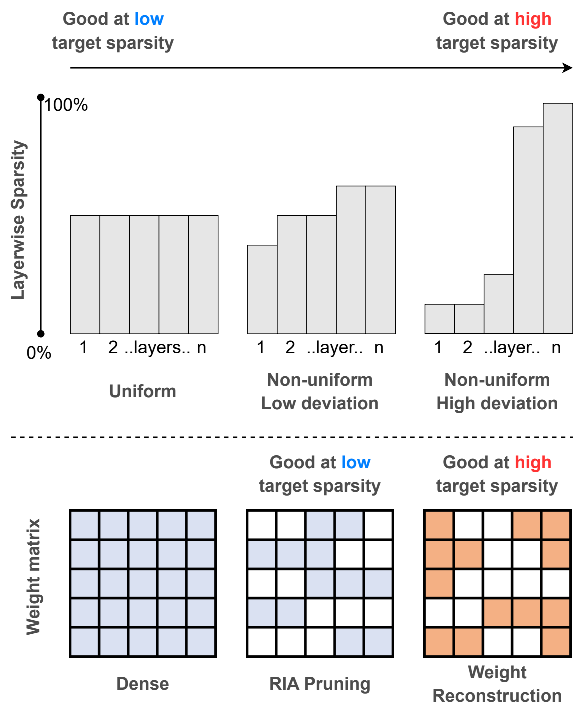


Figure 1: Illustration of findings. Top: Layerwise uniform sparsity outperforms at low sparsity; Non-uniform sparsity excels at high sparsity. Bottom: Relative importance-based pruning (RIA) is better at low sparsity; Hessian-based weight reconstruction is superior at high sparsity. Orange cells indicate changes in weights.

1 Introduction

Large language models (LLMs) have demonstrated exceptional performance across various tasks. However, their large size and high computational demands often constrain their deployment. To address this, network compression techniques like model pruning have been widely explored (LeCun et al., 1989; Hassibi et al., 1993; Mocanu et al., 2018; Sun et al., 2024; Frantar and Alistarh, 2023; Zhang et al., 2024; Yin et al., 2024). Model pruning reduces the model size by eliminating redundant elements, either as individual parameters (unstructured pruning), parameters with structural constraints (semi-structured pruning), or entire structures such as rows, columns, or layers (structured pruning). *Post-training pruning*, a one-shot approach that avoids retraining, is particularly effective for LLMs, mainly when focusing on unstructured and semi-structured methods. Given a target sparsity, these methods create sparsity in weight matrices by removing unnecessary parameters until the target sparsity is reached (e.g., removing 70% of parameters to achieve 70% target sparsity).

State-of-the-art (SOTA) methods typically use layerwise pruning, where each layer is pruned separately to minimize the difference between the out-

Proceedings of the 31st International Conference on Computational Linguistics, pages 3600–3612
January 19–24, 2025.

puts of the pruned and original layer. These methods often follow uniform sparsity, which prune every layer to the target sparsity. Pruning can be achieved by finding a binary sparsity mask or reconstructing weight via the Hessian matrix. The sparsity mask approaches can be based on magnitude (Han et al., 2015) or relative importance (Zhang et al., 2024). The weight reconstruction approaches (Li and Louri, 2021; Frantar and Alistarh, 2022, 2023) leverage the Hessian matrix and typically find the mask adaptively based on the reconstruction. Some approaches explore non-uniform sparsity, varying the sparsity per layer while maintaining overall target sparsity (Frankle and Carbin, 2019; Lee et al., 2019; Wang et al., 2020; Lee et al., 2020; Liu et al., 2021; Yin et al., 2024). However, our study reveals that these methods perform optimally within specific sparsity ranges and lose effectiveness beyond them. Furthermore, there is limited research on their ability to retain multilingual capabilities post-pruning. We examine these techniques across different target sparsity levels and present our key findings.

- $\mathcal{F}1$: Layerwise uniform sparsity performs better at low sparsity, while non-uniform sparsity excels at high sparsity. Additionally, as non-uniform sparsity deviates further from uniform sparsity, performance improves at high sparsity but declines at low sparsity.
- $\mathcal{F}2$: Pruning with a sparsity mask based on relative importance, without modifying the weights, performs better at low sparsity. At high sparsity, weight reconstruction using the Hessian matrix yields better results.

Figure 1 illustrates our findings. We experimented with SOTA pruning methods and popular LLM architectures to validate our findings. To enable this, we re-implemented the methods, allowing for the combination and investigation of their various aspects, and adapted them for modern architectures such as Llama3 (Dubey et al., 2024). Based on these results, we introduce OPTIPRUNE, a versatile pruning approach that optimizes performance across every target sparsity. Additionally, we find that calibration data, though limited in size, significantly affects the multilingual perplexity of pruned models. To address this, we develop a perplexity benchmark in six languages and evaluate OPTIPRUNE on multilingual calibration. The key contributions of this paper are as follows.

- We present findings $\mathcal{F}1$ and $\mathcal{F}2$ and validate them through experiments on SOTA techniques (Section 4).
- We propose OPTIPRUNE, a method for effective pruning of LLMs across varying target sparsity levels (Section 5).
- Empirical results show that OPTIPRUNE outperforms other SOTA pruning methods across most sparsity levels, benchmarks, and language calibrations (Section 7).

2 Related Work

2.1 Pruning Strategies

Model pruning reduces model size by removing unnecessary parameters. Pruning strategies can be categorized into three approaches: *sparse training* (Lee et al., 2018; Mocanu et al., 2018; Evci et al., 2020; Sanh et al., 2020; Yuan et al., 2021; Hoang et al., 2022; Zhang et al., 2023), *pruning-aware training* (Han et al., 2015; Liu et al., 2021), and *post-training pruning* (Hassibi et al., 1993; Li and Louri, 2021; Frantar and Alistarh, 2023; Sun et al., 2024). While *sparse training* and *pruning-aware training* involve iterations of training and are costly for LLMs, *post-training pruning* avoids retraining, making it a more practical approach for LLMs (Zhang et al., 2024; Frantar and Alistarh, 2023). This paper focuses on *post-training pruning*.

2.2 Post-training pruning

Post-training pruning (PTP) has a long history where early work prune model using Hessian Matrix (LeCun et al., 1989; Hassibi et al., 1993). As LLMs advanced, more recent techniques like Iterative AdaPrune (Li and Louri, 2021), and AdaPrune (Frantar and Alistarh, 2022) have leveraged the Hessian matrix for weight pruning and reconstruction. However, due to the computational complexity of $O(N^4)$, newer methods reduce this to $O(N^3)$, improving pruning efficiency (Frantar and Alistarh, 2023; Sun et al., 2024).

Other approaches use pruning masks to remove less important parameters without weight reconstruction, often determining parameter importance by magnitude (Zhu and Gupta, 2017). The recent work RIA (Zhang et al., 2024), accounting for parameter relative connections and activations in calculating importance scores, obtains SOTA results.

2.3 Uniform & Non-uniform Sparsity

Although uniform layerwise sparsity (Zhu and Gupta, 2017; Gale et al., 2019) is a common pruning approach (Sanh et al., 2020; Kurtic et al., 2022), non-uniform layerwise sparsity, where different layers have different sparsity levels, has been actively studied. Early work focused on vision models (Mocanu et al., 2016; Erdos and Renyi, 1959; Mocanu et al., 2018), while more recent efforts apply a global threshold across all layers for non-uniform sparsity (Frankle and Carbin, 2019; Lee et al., 2019; Wang et al., 2020; Lee et al., 2020; Liu et al., 2021). Studies on LLMs have uncovered "outlier" features with magnitudes significantly larger than others (Kovaleva et al., 2021; Puccetti et al., 2022; Timkey and van Schijndel, 2021; Dettmers et al., 2022). OWL (Yin et al., 2024) exploits this outlier distribution to design layerwise non-uniform sparsity for improved pruning performance.

3 Preliminaries

3.1 Layer-wise Pruning

Post-training pruning is often performed through layer-wise pruning, which breaks down the pruning process into subproblems for each layer. The goal is to minimize the ℓ_2 error between the original dense layer and the pruned layer. Formally, for a given input $\mathbf{X}_l$ and weight matrix $\mathbf{W}_l \in \mathbb{R}^{r \times c}$ in the l-th layer, where r and c represent the number of output and input channels, respectively, the objective is to find a binary mask $\mathbf{M}_l \in \{0,1\}^{r \times c}$ and potentially reconstructed weights $\hat{\mathbf{W}}_l$ such that:

$$\text{argmin}_{\mathbf{M}_\ell, \hat{\mathbf{W}}_\ell} \|\mathbf{W}_\ell \mathbf{X}_\ell - (\mathbf{M}_\ell \odot \hat{\mathbf{W}}_\ell)\mathbf{X}_\ell\|_2^2 \quad (1)$$

3.2 State-of-the-art Solvers

We refer to layerwise pruning methods as *solvers*. The first category of solvers focuses on finding the sparsity mask $\mathbf{M}_l$ while keeping the weights unchanged ($\hat{\mathbf{W}}_l = \mathbf{W}_l$). This is typically done by calculating the importance of each weight parameter and pruning the least important ones until the target sparsity is reached. While parameter importance is often determined by magnitude (Zhu and Gupta, 2017), recent SOTA method RIA (Zhang et al., 2024) incorporate parameter connections and activations for more accurate importance estimates (Details in Appendix A.2). The second category of solvers focuses on reconstructing the weights $\hat{\mathbf{W}}$ using the Hessian matrix (LeCun et al., 1989; Hassibi et al., 1993; Li and Louri, 2021; Frantar and Alistarh, 2022, 2023; Sun et al., 2024). Here, the mask $\mathbf{M}_l$ is selected adaptively during the reconstruction. Recently, SparseGPT (Frantar and Alistarh, 2023) reduced pruning complexity and achieved SOTA results among these approaches.

3.3 Non-uniform Sparsity

Given a target sparsity S, layerwise uniform sparsity assigns the same sparsity level to each layer i, such that $S_i = S, \forall i$. Instead of applying a uniform sparsity across all layers, non-uniform sparsity can be used, where each layer has a different sparsity while maintaining the overall target sparsity. This removes the constraint $S_i = S$ but ensures that the average sparsity across layers satisfies $\sum_i S_i/N = S$ where N is the number of layers.

OWL (Yin et al., 2024) computes the non-uniform sparsity ratios by identifying outliers in each layer through the layerwise outlier distribution (LOD). Layers with a high number of outliers are assigned lower sparsity, while those with fewer outliers have higher sparsity. For layer l-th, OWL computes the outlier distribution D^l. The detailed calculation of D^l is described in Appendix A.1. Given a global target sparsity S and the LOD for each layer $\mathbf{D} = [D^1, D^2, \ldots, D^n]$, the sparsity for layer i-th is set as $S_i \propto 1 - D_i$. A hyperparameter λ controls the deviation of S_i from the target S, with the constraint $S_i \in [S - \lambda, S + \lambda]$. Figure 4a illustrates how non-uniform sparsity varies with different values of λ.

4 Empirical Study

4.1 Study on Uniform vs Non-uniform ($\mathcal{F}1$)

We examine the performance of layerwise uniform versus non-uniform sparsity across different target sparsity levels by comparing the perplexity of models pruned with uniform and non-uniform sparsity at each target level. To further assess how deviations from uniform sparsity affect performance, we analyze the impact of varying deviations on models pruned with non-uniform sparsity. We compare perplexity across different deviation levels at each target sparsity.

Experimental Settings. We use two pruning methods: SparseGPT (Frantar and Alistarh, 2023), which reconstructs weights using the Hessian matrix, and RIA (Zhang et al., 2024), which focuses on mask finding. Non-uniform sparsity is calculated using OWL (Yin et al., 2024), which

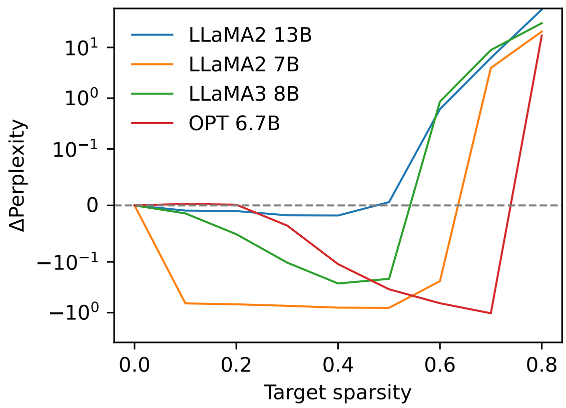


Figure 2: Perplexity difference (Δ Perplexity) between uniform and non-uniform sparsity, using SparseGPT. Lower perplexity indicates better performance. Negative Δ indicates uniform outperforms non-uniform. Positive Δ indicates non-uniform outperform uniform.

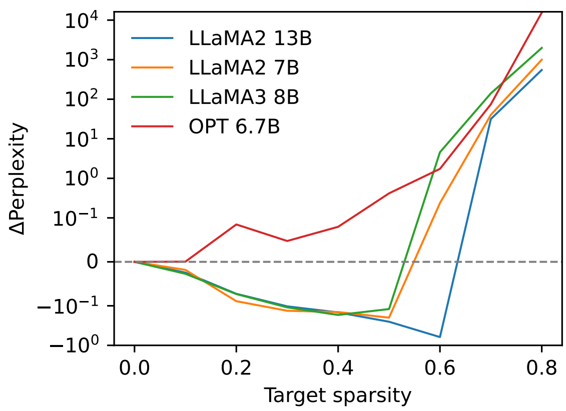
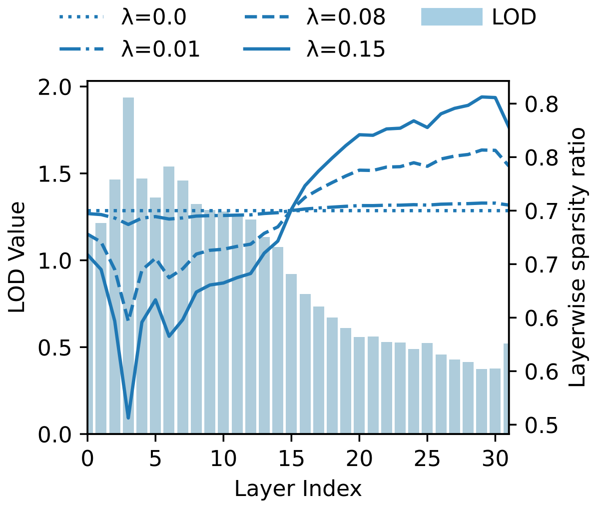
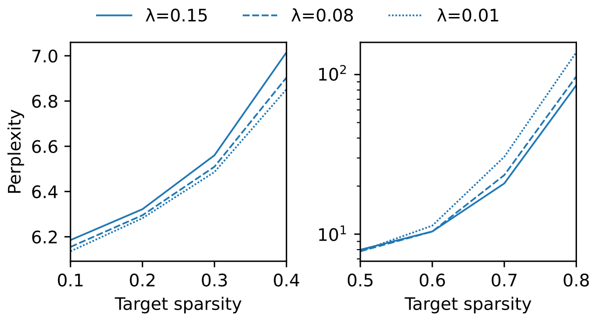


Figure 3: Perplexity difference (Δ Perplexity) RIA (mask finding via relative importance and activation) and SparseGPT (weight reconstruction via Hessian matrix). Lower perplexity indicates better performance. Negative Δ indicates RIA outperforms SparseGPT. Positive Δ indicates SparseGPT outperforms RIA.



(a) Illustration of layerwise non-uniform sparsity of different λ value at target sparsity 70%. $\lambda = 0.0$ indicates uniform sparsity. The bar chart shows each layer's Layerwise Outlier Distribution (LOD).



(b) Perplexity performance of different deviations controlled by λ, measured on Llama2(7B) pruned by SparseGPT. Lower perplexity indicates better performance. A lower λ is beneficial at low target sparsity (left), while a higher λ performs better at high target sparsity (right).

Figure 4: Effect of different values of λ

sets sparsity ratios based on outlier distribution. We evaluate target sparsity levels ranging from $[10\%, 20\%, ..., 80\%]$ on Llama2 (7B and 13B), Llama3 (8B), and OPT (6.7B), with perplexity measured on the Wikitext2 test set (Merity et al., 2016). To explore the effect of deviations from uniform sparsity, we experiment on Llama2 (7B) using SparseGPT with different values of λ (which controls deviation) set to $\lambda \in \{0.01, 0.08, 0.15\}$.

Results (Uniform vs Non-uniform). Figure 2 compares the perplexity of models pruned with uniform and non-uniform sparsity using SparseGPT. Non-uniform sparsity outperforms uniform sparsity at higher target sparsity levels. However, at lower sparsity levels, uniform sparsity shows better results. This pattern is consistent across all architectures. Results for models pruned with RIA follow a similar trend and are detailed in Appendix B.1.

Results (Deviations in Non-uniform). Figure 4a illustrates how non-uniform sparsity deviates from the target sparsity (70%) for different λ values. Figure 4b compares perplexity with varying λ values. The results show that at lower target sparsity levels (10% to 50%), smaller λ (closer to uniform sparsity) yields better performance. For higher target sparsity, larger λ (greater deviation from uniform) produces better results.

4.2 Study on State-of-the-art Solvers ($\mathcal{F}2$)

We evaluate two SOTA solvers, RIA and SparseGPT. RIA, which calculates parameter importance by considering connections and activations, leads to finding sparsity masks without modi-

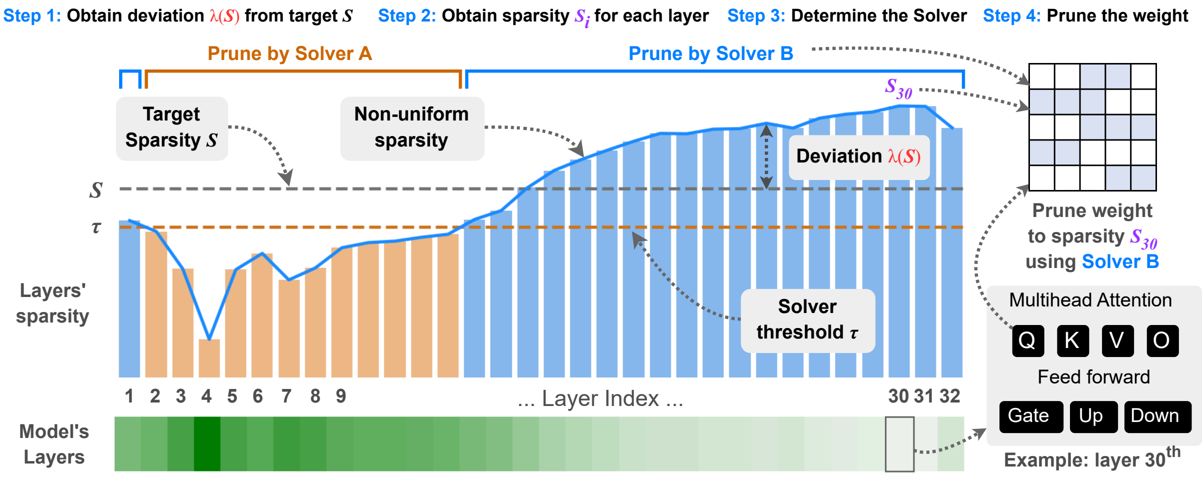


Figure 5: Illustration of OptiPrune applied to the Llama2 (7B) model, consisting of 32 layers. Each layer includes four weight matrices for multi-head attention (Q, K, V, O) and three for the feed-forward network (Gate, Up, Down). The importance of each layer, determined by the outlier distribution, is represented by varying shades of green.

fying weights. SparseGPT, which uses the Hessian matrix for weight reconstruction and adaptive mask selection, represents the SOTA for weight reconstruction methods. We compare the perplexity of models pruned by RIA and SparseGPT across different target sparsity levels.

Experimental Settings We assess RIA and SparseGPT across target sparsity levels ranging from $[10\%, 20\%, ..., 80\%]$ on several LLMs, including Llama2 (7B and 13B), Llama3 (8B), and OPT (6.7B). Perplexity is evaluated on the Wikitext2 test set (Merity et al., 2016).

Results. Figure 3 shows the perplexity performance of both solvers. The results indicate that RIA outperforms SparseGPT at lower target sparsity levels, while SparseGPT excels at higher target sparsity. This trend is consistent across all model architectures. Further analysis reveals that weight reconstruction degrades performance at lower sparsity but enhances it at higher sparsity levels, reinforcing our findings. Detailed results are provided in Appendix B.2.

5 OptiPrune

Building on our validated findings, we introduce a pruning method designed to effectively handle models across varying target sparsity levels. Figure 5 provides an overview of the method. Since lower deviation values enhance performance at low target sparsity and higher deviation values are better at high target sparsity (Finding $\mathcal{F}1$), we first compute the deviation level $\lambda(S)$ based on the target sparsity S (Step 1). Next, we determine the non-uniform sparsity for each layer, calculating the layer-wise sparsity S_i (Step 2). To optimize performance, we then compare S_i with a threshold τ (Step 3), as different solvers perform better at different sparsity levels (Finding $\mathcal{F}2$). Finally, for each layer, we prune the weight matrices using the chosen solver and calculated sparsity S_i (Step 4). The following subsections show the details of these key steps.

5.1 Determine the deviation levels

The parameter λ controls the deviation of non-uniform sparsity from uniform sparsity. While prior work (Yin et al., 2024) uses a fixed, predefined λ, we dynamically adjust λ based on the target sparsity level, following the strategy outlined in Finding $\mathcal{F}1$. Specifically, we use a low λ for low target sparsity and a high λ for high target sparsity. We define the range for λ as $[\lambda_{\text{min}}, \lambda_{\text{max}}]$ and compute λ as a function of the target sparsity S. Let S_{min} and S_{max} represent the minimum and maximum sparsity levels, respectively. One straightforward method is to linearly interpolate λ as follows.

$$\lambda(S) = \lambda_{\text{min}} + \frac{(S - S_{\text{min}}) \cdot (\lambda_{\text{max}} - \lambda_{\text{min}})}{S_{\text{max}} - S_{\text{min}}}$$

We also explore different approaches to calculate λ and detailed in Appendix C. For consistency, we use linear interpolation for all experiments. Through experimentation, we find that $\lambda_{\text{min}} = 0.01$ and $\lambda_{\text{max}} = 0.16$ provide a good range, with $S_{\text{min}} = 0\%$ and $S_{\text{max}} = 100\%$ as the sparsity bounds.

Method	Target sparsity 10%	20%	30%	40%	50%	60%	70%	80%	Avg
				Llama2-7B (Dense=53.83)					
OPTIPRUNE (Ours)	53.83	54.06	53.61	**53.39**	**51.50**	**48.05**	**42.83**	**36.79**	**49.26**
OWL	53.89	53.81	53.03	52.67	50.98	47.65	42.36	36.20	48.82
RIA	53.80	**54.17**	**53.79**	53.11	50.29	46.10	36.40	33.90	47.70
SparseGPT	53.88	53.49	53.35	52.36	50.12	47.25	39.89	33.48	47.98
Wanda	**53.90**	54.09	53.26	51.86	49.72	43.46	34.88	33.60	46.85
				Llama2-13B (Dense=56.60)					
OPTIPRUNE (Ours)	**56.35**	55.81	55.40	**55.79**	**54.35**	**51.69**	**46.25**	**38.35**	**51.75**
OWL	56.25	**56.14**	55.87	54.20	54.33	51.07	45.53	37.59	51.37
RIA	56.34	55.99	55.65	55.29	53.89	50.60	39.65	33.44	50.11
SparseGPT	56.35	55.98	**55.89**	55.12	54.08	50.51	42.74	35.30	50.75
Wanda	56.19	56.10	55.65	55.45	53.33	48.05	35.81	33.20	49.22
				Llama3-8B (Dense=58.62)					
OPTIPRUNE (Ours)	59.06	**58.82**	57.74	**56.72**	53.45	**50.74**	**42.45**	**36.41**	**51.92**
OWL	**59.15**	58.78	**58.06**	56.60	**54.05**	49.85	41.66	36.39	51.82
RIA	58.72	58.52	57.08	55.31	52.42	45.21	34.51	33.42	49.40
SparseGPT	59.10	58.70	57.75	56.59	53.42	48.64	40.20	34.91	51.16
Wanda	59.10	58.60	57.36	54.53	50.97	44.08	35.91	33.54	49.26
				OPT-6.7B (Dense=46.89)					
OPTIPRUNE (Ours)	**47.59**	46.98	47.11	**46.64**	45.52	44.18	**41.83**	**37.60**	**44.68**
OWL	47.00	**47.19**	**47.19**	46.59	45.56	44.18	41.54	37.15	44.55
RIA	46.94	46.88	46.72	46.41	45.50	43.31	36.10	35.63	43.44
SparseGPT	47.00	46.88	47.15	46.54	**45.92**	**44.61**	41.82	37.21	44.64
Wanda	46.97	46.93	46.89	46.24	43.35	38.06	35.11	33.15	42.09

Table 1: Zero-shot accuracy at each sparsity ratio, averaged across all benchmarks: Hellaswag, BoolQ, ARC (Challenge/Easy), MNLI, QNLI, RTE, OpenBookQA, Winogrande, and MathQA. The results of OPTIPRUNE are highlighted in blue. **Bold** numbers indicate the highest performance among the methods.

5.2 Obtain the layerwise non-uniform sparsity

After calculating λ to control non-uniform deviation, we determine the layerwise non-uniform sparsity S_i based on outliers, following the OWL approach (Yin et al., 2024) (Section 3.3). First, we compute the outlier distribution for each layer as $\mathbf{D} = [D^1, D^2, ..., D^n]$. We then set $S_i \propto 1 - D_i$ and scale it within the range $[S-\lambda, S+\lambda]$. The principle is to assign lower sparsity to layers with more outliers and higher sparsity to those with fewer outliers while maintaining the overall target sparsity S.

5.3 Adaptive Solver

We select RIA and SparseGPT as the two SOTA solvers to consider. Our finding ($\mathcal{F}2$) shows that these solvers excel in different target sparsity ranges, with RIA suitable for low and SparseGPT suitable for high target sparsity levels. To determine which solver to use for pruning each layer, we introduce a hyperparameter τ. For the i-th layer, we compare the non-uniform sparsity S_i with τ. If $S_i < \tau$, we apply RIA to find the sparsity mask based on relative importance and activation. If $S_i \geq \tau$, we utilize SparseGPT to reconstruct the weights and adaptively generate the mask.

5.4 Model pruning

The final step is to prune each layer of the model based on the obtained non-uniform sparsity and the solver. For layer i-th, we use the solver (obtained in Step 3) to prune the weight matrices to the sparsity S_i (obtained in Step 2). We perform pruning on the main weight matrices of the layer. For common LLM architectures, this involves the multi-head attention, which contains 4 weight matrices Q, K, V, O, and feed-forward layers, which contain multiple

Method	ARC-C	ARC-E	BoolQ	HSwag	MathQA	MNLI	OBQA	QNLI	RTE	WGrande	AVG
						Llama-2 (7B) (Touvron et al., 2023)					
Dense	43.43	76.30	77.71	57.16	28.17	42.24	31.40	49.90	62.82	69.14	53.83
OPTIPRUNE	35.85	**65.28**	**73.56**	49.03	26.03	**40.37**	**27.50**	50.22	**58.57**	**66.17**	**49.26**
OWL	**36.09**	64.69	73.46	**49.05**	**26.31**	38.74	27.50	50.45	56.05	65.89	48.82
RIA	34.97	62.71	68.58	47.02	25.47	39.96	25.98	50.28	57.85	64.12	47.69
SparseGPT	35.41	63.60	70.15	48.25	26.25	37.73	27.05	50.51	55.42	65.40	47.98
Wanda	34.79	60.64	66.44	45.86	25.98	37.57	26.05	**50.88**	56.72	63.52	46.85
						Llama-2 (13B) (Touvron et al., 2023)					
Dense	48.46	79.38	80.61	60.07	32.06	43.19	35.00	49.53	65.34	72.38	56.60
OPTIPRUNE	**40.00**	**69.76**	**78.01**	**52.42**	**28.60**	40.42	**29.98**	**49.57**	**59.39**	**69.36**	**51.75**
OWL	39.88	69.49	77.25	52.08	28.46	40.09	29.88	49.51	58.21	68.88	51.37
RIA	38.46	66.94	73.12	50.43	28.32	39.92	28.62	49.52	58.98	66.75	50.11
SparseGPT	39.25	67.96	75.65	51.14	28.25	**40.69**	29.12	49.54	57.85	68.01	50.75
Wanda	38.01	63.89	71.15	49.32	27.73	40.12	28.32	49.28	58.08	66.34	49.22
						Llama-3 (8B) (Dubey et al., 2024)					
Dense	50.26	80.09	80.98	60.11	40.47	47.82	34.60	49.94	68.59	73.40	58.62
OPTIPRUNE	**39.19**	**67.03**	**77.40**	**50.85**	32.62	**42.80**	**27.55**	51.15	**62.59**	**68.06**	**51.92**
OWL	38.76	66.74	76.74	50.52	**32.82**	42.43	27.88	51.81	62.73	67.77	51.82
RIA	37.20	62.97	69.30	47.76	31.70	41.35	26.00	50.23	61.42	66.06	49.40
SparseGPT	38.77	66.00	74.87	49.77	32.73	42.18	27.20	**51.48**	61.60	67.04	51.16
Wanda	37.08	61.93	71.67	47.60	31.45	40.99	26.82	49.96	59.39	65.73	49.26
						OPT (6.7B) (Zhang et al., 2022)					
Dense	30.46	65.57	66.06	50.51	24.62	32.81	27.60	50.92	55.23	65.19	46.89
OPTIPRUNE	27.56	59.54	**65.28**	44.97	**24.34**	**34.56**	**24.38**	50.31	**54.18**	61.70	**44.68**
OWL	27.46	59.45	65.25	44.80	23.99	34.23	24.30	50.20	53.97	61.82	44.55
RIA	26.77	55.81	65.01	42.89	23.18	32.86	22.98	50.16	54.15	60.56	43.44
SparseGPT	**27.59**	**60.05**	65.25	**45.25**	23.92	33.81	24.30	**50.32**	53.70	**62.21**	44.64
Wanda	26.43	52.28	60.70	40.79	22.84	32.62	22.50	50.22	53.20	59.31	42.09

Table 2: Zero-shot accuracy of all benchmarks, averaged over all target sparsity (from 10% to 80% sparsity). OPTIPRUNE 's results are highlighted in blue . Dense models' results are highlighted in gray . **Bold** numbers indicate highest performance among methods.

matrices (e.g., 3 for Llama2 and 2 for OPT).

6 Experiments

6.1 Tasks and Datasets

We evaluate our model's performance on both language modeling and zero-shot classification tasks. For language modeling, we measure perplexity (PPL), with lower values indicating better performance. This is assessed using the Wikitext2. For zero-shot classification, we evaluate the models on multiple benchmarks: Hellaswag, BoolQ, ARC, MNLI, QNLI, RTE, OpenBookQA, Winogrande, and MathQA (Details in Appendix D).

6.2 Baselines

We evaluate our approach against strong publicly available LLMs, including Llama2 (7B, 13B) (Touvron et al., 2023) Llama3 (8B) (Dubey et al., 2024), and OPT (6.7B) (Zhang et al., 2022). We compare with recent SOTA methods for LLM pruning, such as SparseGPT (Frantar and Alistarh, 2023) Wanda (Sun et al., 2024), RIA (Zhang et al., 2024), and OWL (Yin et al., 2024). Since OWL only determines the non-uniform sparsity and requires a backbone method for pruning, we use SparseGPT as the backbone for OWL, given its reported highest performance in OWL's paper (Yin et al., 2024).

7 Results

7.1 Zero-shot performance at every sparsity

Table 1 presents the zero-shot accuracy for various target sparsity levels. OPTIPRUNE consistently outperforms other baselines across nearly all sparsity levels. Averaged over all sparsity ratios, it achieves the highest accuracy compared to competing approaches. This trend remains consistent across different model architectures and sizes. Although OPTIPRUNE does not have the highest performance at a few sparsity levels, it is still competitive compared to the highest results. This result highlights the robustness and performance of OPTIPRUNE at every target sparsity.

7.2 Zero-shot performance by benchmarks

Table 2 shows the zero-shot accuracy for all benchmarks, averaged over all target sparsity levels. Our method consistently outperforms baseline methods across almost all benchmarks and achieves the highest average results. This improvement is evident across different model architectures and sizes. The strong performance across a diverse set of benchmarks highlights the robustness of OPTIPRUNE

7.3 Perplexity

Table 3 shows the average perplexity across all target sparsity levels. OPTIPRUNE outperforms all baselines for Llama models. For the OPT model, while OPTIPRUNE does not surpass OWL, it still demonstrates competitive performance compared to other SOTA methods.

7.4 Other Results

Semi-structured pruning with $N : M$ constraint, which requires that at least N out of every M consecutive elements be zero, allows for model pruning while preserving hardware efficiency (Details in Appendix F.1). Table 4 shows the perplexity performance of models pruned in semi-structured 2:4. The results show that OPTIPRUNE outperforms other methods in most architectures.

We also assess the ability of OPTIPRUNE to calibrate for specific languages. Calibration details are provided in Appendix E.1 with results in Appendix E.3. The findings show that the original OPTIPRUNE already outperforms other SOTA methods across languages. Furthermore, OPTIPRUNE with language calibration significantly enhances performance, surpassing the baselines by a substantial margin.

Additionally, in Appendix F.2, we discuss the inference acceleration of sparse models.

Method	Llama2-7B	Llama2-13B	Llama3-8B	OPT-6.7B
OPTIPRUNE	**18.98**	**13.68**	**39.42**	26.55
RIA	152.88	65.15	310.23	1967.77
SparseGPT	23.18	22.61	44.56	26.35
OWL	20.61	14.81	39.67	**24.49**
Wanda	743.84	251.90	447.14	592.75

Table 3: Perplexity performance, averaged across all sparsity.

Method	Llama2-7B	Llama2-13B	Llama3-8B	OPT-6.7B
OPTIPRUNE	**12.31**	9.86	**18.49**	**16.09**
RIA	12.75	**9.49**	25.81	17.63
SparseGPT	12.41	9.87	18.49	16.11
OWL	12.41	9.86	18.49	16.11
Wanda	13.72	10.17	27.71	17.80

Table 4: Semi-structured 2:4 pruning performance, measured by average perplexity across all sparsity.

8 Conclusions

This study addresses the limitations of existing pruning methods by exploring the effects of varying target sparsity levels. We present and validate two key findings: (1) Layerwise uniform sparsity is effective at low sparsity levels, while non-uniform sparsity excels at high sparsity levels; (2) Relative importance-based mask pruning performs better at low sparsity, whereas Hessian-based weight reconstruction is superior at high sparsity. Based on these insights, we introduce OPTIPRUNE , an effective pruning method that remains robust across all target sparsity levels. OPTIPRUNE uses non-uniform with adaptive deviation and employs a threshold to select the appropriate pruning solver. Empirical results across various datasets, model architectures, and languages confirm OPTIPRUNE 's robustness and performance. Our method and findings offer valuable insights for future research in LLM pruning.

9 Limitations

Specificity of Pruning Methods. When examining the effect of solvers across different target sparsity levels, we focused on two methods: RIA and SparseGPT. While SparseGPT aligns with other Hessian-based weight reconstruction methods, RIA is more tailored to its specific approach and may not represent the full spectrum of mask-finding techniques. Consequently, while the observation that each method excels at particular sparsity levels is valuable, it may be somewhat method-specific. This insight remains important for guiding future research.

Scale of Investigated Models. Due to computational constraints, we tested pruning methods on models ranging from 6 to 13 billion parameters. Larger LLMs were not explored, and future work should extend this investigation to larger models to confirm the generalizability of our results.

References

Aida Amini, Saadia Gabriel, Earl Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. 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 2357–2367.

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. 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.

Peter Clark, Liam 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*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems (NeurIPs)*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, and et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Paul Erdos and Alfred Renyi. 1959. On random graphs i. *Publicationes Mathematicae (Debrecen)*, 6:290–297.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. 2020. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pages 2943–2952. PMLR.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

Elias Frantar and Dan Alistarh. 2022. Spdy: Accurate pruning with speedup guarantees. In *International Conference on Machine Learning*, pages 6726–6743. PMLR.

Elias Frantar and Dan Alistarh. 2023. SparseGPT: massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Trevor Gale, Erich Elsen, and Sara Hooker. 2019. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*.

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

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

Duc NM Hoang, Shiwei Liu, Radu Marculescu, and Zhangyang Wang. 2022. Revisiting pruning at initialization through the lens of ramanujan graph. In *The Eleventh International Conference on Learning Representations*.

Olga Kovaleva, Suvarna Kulshreshtha, Anna Rogers, and Anna Rumshisky. 2021. Bert busters: Outlier dimensions that disrupt transformers. *arXiv preprint arXiv:2105.06990*.

Eldar Kurtic, Daniel Campos, Tri Nguyen, Elias Frantar, Mark Kurtz, Brendan Fineran, Matt Goin, and Dan Alistarh. 2022. The optimal bert surgeon: Scalable and accurate second-order pruning for large language models. *arXiv preprint arXiv:2203.07259*.

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

Jaehong Lee, Sejung Park, Seul-Kee Mo, Sungsoo Ahn, and Jinwoo Shin. 2020. Layer-adaptive sparsity for the magnitude-based pruning. *arXiv preprint arXiv:2010.07611*.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. 2019. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. 2018. Snip: Single-shot network pruning based on connection sensitivity.

Jiajun Li and Ahmed Louri. 2021. Adaprune: An accelerator-aware pruning technique for sustainable cnn accelerators. volume 7, pages 47–60.

Shiwei Liu, Tianlong Chen, Xiaohan Chen, Zahra Atashgahi, Lu Yin, Huanyu Kou, Li Shen, Mykola Pechenizkiy, Zhangyang Wang, and Decebal Constantin Mocanu. 2021. Sparse training via boosting pruning plasticity with neuroregeneration. *Advances in Neural Information Processing Systems*, 34:9908–9922.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv 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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. 2021. Accelerating sparse deep neural networks. *Preprint*, arXiv:2104.08378.

Decebal Constantin Mocanu, Elena Mocanu, Phuong H Nguyen, Madalina Gibescu, and Antonio Liotta. 2016. A topological insight into restricted boltzmann machines. *Machine Learning*, 104(2):243–270.

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. 2018. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):2383.

NeuralMagic. 2021. Deepsparse. `https://github.com/neuralmagic/deepsparse`.

Giacomo Puccetti, Anna Rogers, Alexander Drozd, and Felice Dell'Orletta. 2022. Outliers dimensions that disrupt transformers are driven by frequency. *arXiv preprint arXiv:2205.11380*.

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

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740.

Victor Sanh, Thomas Wolf, and Alexander Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems*, volume 33, pages 20378–20389.

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*.

Andrew Timkey and Marten van Schijndel. 2021. All bark and no bite: Rogue dimensions in transformer language models obscure representational quality. *arXiv preprint arXiv:2109.04404*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. 2020. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Mykola Pechenizkiy, Yi Liang, Zhangyang Wang, and Shiwei Liu. 2024. Outlier Weighed Layerwise Sparsity (OWL): A Missing Secret Sauce for Pruning LLMs to High Sparsity.

Geng Yuan, Xiaolong Ma, Wei Niu, Zhengang Li, Zhenglun Kong, Ning Liu, Yifan Gong, Zheng Zhan, Chaoyang He, and et al. Jin, Qing. 2021. Mest: Accurate and fast memory-economic sparse training framework on the edge. In *Advances in Neural Information Processing Systems*, volume 34, pages 20838–20850.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: Open Pre-trained Transformer Language Models.

Y Zhang, J Zhao, W Wu, A Muscoloni, and CV Cannistraci. 2023. Epitopological sparse ultra-deep learning: A brain-network topological theory carves communities in sparse and percolated hyperbolic anns.

Yingtao Zhang, Haoli Bai, Haokun Lin, Jialin Zhao, Lu Hou, and Carlo Vittorio Cannistraci. 2024. Plug-and-Play: An Efficient Post-training Pruning Method for Large Language Models. In *The Twelfth International Conference on Learning Representations*.

Michael Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. *Preprint*, arXiv:1710.01878.

A Detailed Calculation of Pruning Methods

A.1 Calculation of Outlier Distribution

This section outlines the calculation of OWL (Yin et al., 2024) for determining the layerwise outlier distribution. The outlier score for a given $\mathbf{W}_{ij}$ is computed as $\mathbf{A}_{ij} = ||\mathbf{X}_j||_2 * ||\mathbf{W}_{ij}||$, where $||\mathbf{X}_j||_2$ is the ℓ_2 norm of input feature connected to the weight. The outlier distribution of layer l is calculated as follows.

$$D^l = \frac{\sum_{i=1}^{C_{\text{out}}} \sum_{j=1}^{C_{\text{in}}} \mathbb{I}\left(\mathbf{A}_{ij}^l > \mathbf{M} \cdot \overline{\mathbf{A}}^l\right)}{C_{\text{in}} C_{\text{out}}} \quad (2)$$

where $(C_{\text{out}}, C_{\text{in}})$ are the dimensions of $\mathbf{W}$, $\overline{\mathbf{A}}^l$ is the mean of $\mathbf{A}^l$ and $\mathbb{I}()$ is an indicator function returning 1 if the condition is true, and 0 otherwise.

A.2 Calulation of Relative Importance and Activation

The importance of parameter $\mathbf{W}ij$ is calculated as follows.

$$\text{RIA}_{ij} = \left(\frac{|\mathbf{W}_{ij}|}{\sum |\mathbf{W}_{*j}|} + \frac{|\mathbf{W}_{ij}|}{\sum |\mathbf{W}_{i*}|} \right) (\|\mathbf{X}_i\|_2)^{\alpha} \quad (3)$$

where $\sum |\mathbf{W}_{*j}|$ is the sum of parameters in input channel j, $\sum |\mathbf{W}_{i*}|$ is the sum in output channel i, and α is a hyperparameter controlling activation strength.

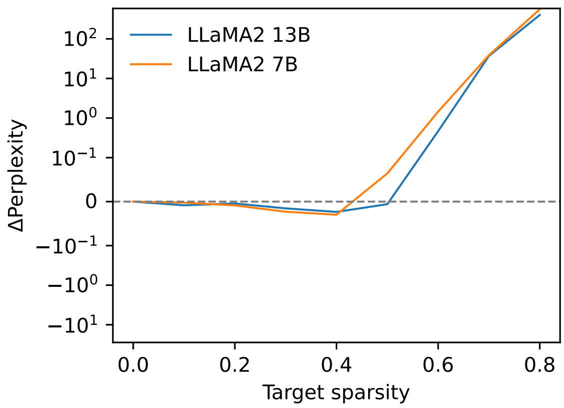


Figure 6: Perplexity difference (Δ Perplexity) between uniform and non-uniform sparsity, using RIA. Lower perplexity indicates better performance. Negative Δ indicates uniform outperforms non-uniform. Positive Δ indicates non-uniform outperform uniform.

B Additional Experiments

B.1 Experiments on Uniform vs Non-uniform

Figure 6 illustrates the perplexity difference between uniform and non-uniform sparsity in models pruned by RIA. Non-uniform sparsity outperforms uniform sparsity at higher sparsity ratios. However, at lower sparsity ratios, the benefits of non-uniform sparsity diminish, with uniform sparsity proving more effective. This pattern is consistent across all architectures.

B.2 Experiments on Solvers

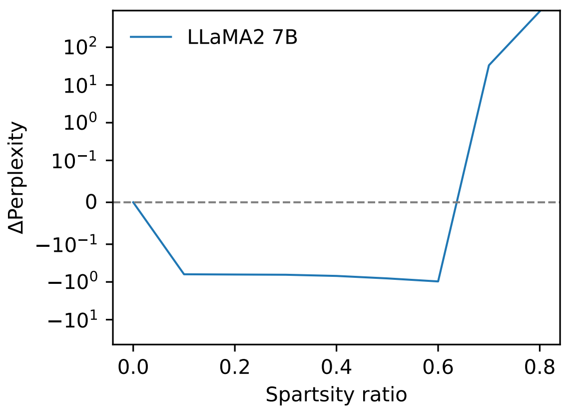


Figure 7: Perplexity difference (Δ Perplexity) between RIA with and without weight reconstruction. Lower perplexity indicates better performance. Negative Δ indicates weight reconstruction hurt the performance. Positive Δ indicates weight reconstruction improves the performance.

We experiment on RIA, both with and without

	Sparsity ratio								
λ Calculation	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	Avg
Linear translate	**53.83**	54.06	53.61	**53.39**	51.50	**48.05**	**42.83**	36.79	**49.26**
Sigmoid Function	53.78	**54.20**	**53.70**	53.15	**50.56**	47.92	42.82	**36.85**	49.12

Table 5: Zero-shot Accuracy at every sparsity ratio, Llama2(7B).

weight reconstruction, across different target sparsities. Using RIA's mask, we applied SparseGPT's weight reconstruction technique and compared the perplexity results. Figure 7 shows the perplexity difference between RIA with and without weight reconstruction. The results indicate that while weight reconstruction negatively impacts performance at low sparsity, it improves performance at high sparsity levels.

C Approaches to calculate λ

This section investigates the two approaches to calculating λ. The first one is linear translation.

$$\lambda(S) = \lambda_{\min} + \frac{(S - S_{\min}) \cdot (\lambda_{\max} - \lambda_{\min})}{S_{\max} - S_{\min}}$$

Where S represents the target sparsity for the current layer or model. $\lambda_{\min}$ and $\lambda_{\max}$ are the boundaries for the deviation parameter. Another approach uses a sigmoid function to calculate λ.

$$\lambda(S) = \lambda_{\min} + (\lambda_{\max} - \lambda_{\min}) \cdot \frac{1}{1 + e^{-k\left(\frac{S - S_{\min}}{S_{\max} - S_{\min}} - 0.5\right)}}$$

Unlike linear scaling, the sigmoid function changes λ slowly near the boundaries of S and more rapidly around mid-range sparsity (50%).

Table 5 compares the performance of two approaches of calculating λ based on target sparsity. The results show that the two approaches perform equally well. However, linear translation is more stable and outperforms the translation by sigmoid function.

D Details on Dataset

This section lists the benchmarks used in the evaluation. Perplexity is evaluated on Wikitext2 (Merity et al., 2016). The benchmarks used for zero-shot evaluation includes Hellaswag (Zellers et al., 2019), BoolQ (Clark et al., 2019), ARC (Challenge/Easy) (Clark et al., 2018), MNLI (Williams et al., 2018), QNLI (Wang et al., 2018), RTE (Dagan et al., 2005), OpenBookQA (Mihaylov et al., 2018), Winogrande (Sakaguchi et al., 2020), and MathQA (Amini et al., 2019)

E Calibration on specific language

E.1 OPTIPRUNE with language calibration

Most existing pruning methods (Frantar and Alistarh, 2023; Sun et al., 2024; Zhang et al., 2024) use a small amount of calibration data—typically 128 examples from the C4 dataset (Raffel et al., 2020). Despite the small sample size, our study finds that this data can significantly impact multilingual performance. Our preliminary results show that by adjusting the calibration data to focus on a specific language, we can notably improve perplexity for that language, especially at high target sparsity levels. To leverage this insight, we develop a version of OPTIPRUNE calibrated for specific languages, using 128 examples from the Multilingual-C4 dataset (Xue et al., 2021) for calibration.

E.2 Evaluation data

To assess multilingual perplexity performance, we curated datasets comparable to the Wikitext2 test set (Merity et al., 2016) in multiple languages. We ensured that each dataset was similar in size to Wikitext2, specifically maintaining around 5 million tokens per dataset, as tokenized by the Llama tokenizer.

E.3 Results

Table 6 presents perplexity results on Wikitext for various languages, including German, Spanish, French, Vietnamese, Chinese, and Japanese, averaged across all target sparsity levels. The results indicate that the original OPTIPRUNE already outperforms other SOTA methods. Furthermore, OPTIPRUNE with language calibration significantly enhances performance, surpassing the baselines by a substantial margin.

Method	de	es	fr	vi	zh	ja
OPTIPRUNE -LC	**8.49**	**7.41**	**8.05**	**3.22**	**5.77**	**9.21**
OPTIPRUNE	24.04	16.23	14.87	7.01	11.23	9.25
RIA	60.18	32.76	32.82	10.87	473.72	579.88
SparseGPT	66.67	33.41	28.78	10.33	19.80	18.82
OWL	28.13	17.45	16.00	7.69	12.17	10.19
Wanda	223.80	108.37	126.10	49.84	407.26	1751.95

Table 6: Average perplexity across all target sparsity levels, measured on Wikitext for various languages: German (de), Spanish (es), French (fr), Vietnamese (vi), Chinese (zh), and Japanese (ja). Original OPTIPRUNE and the version with specific language calibration (OPTIPRUNE-LC) are compared with baselines.

F Inference of Pruned Models

F.1 N:M Pruning

NVIDIA recently introduced N:M sparsity (Mishra et al., 2021) as a technique to compress neural networks while maintaining hardware efficiency. This approach requires that at least N out of every M consecutive elements be zero, facilitating faster matrix-multiply-accumulate operations. For example, a 2:4 sparsity ratio yields 50% sparsity, which can effectively double inference speed on NVIDIA's Ampere GPUs.

F.2 Inference Acceleration

Strategy	Q/K/V/Out	Up/Gate	Down	Overall
Unstructured 50%	0.98x	0.98x	0.97x	0.98x
2:4 (cuTLASS)	1.21x	1.23x	1.23x	1.22x
2:4 (cuSPARSELT)	1.64x	1.65x	1.62x	1.63x

Table 7: Inference time on Llama2 (13B)

Unstructured sparsity	40%	50%	60%
Speedup	1.57x	1.82x	2.16x

Table 8: Speedup over dense version of OPT(2.7B) in DeepSparse

Pruning of models can improve inference efficiency. The improvements in inference have been actively studied in previous studies. Although the post-training pruning methods are different, they all follow the same strategy (unstructured, semi-structured pruning). The only difference is which parameters the method chooses to prune. Theoretically, the inference efficiency should be approximately the same for these pruning methods at the same target sparsity and model architecture. This section includes the inference acceleration in pruning that has been previously studied. Theoretically, the acceleration observed in these studies should apply to OptiPrune.

Table 7 shows the inference time on Llama2(13B), reported by Zhang et al. (2024), experimented on NVIDIA Tesla A100 with cuTLASS and cuSPARSELt library for Sparse Matrix-Matrix Multiplication (Mishra et al., 2021). The results show that 50% semi-structure pruned model can achieve up to 1.6 times speed up.

Table 8 shows the CPU inference time speedup on pruned OPT(2.7B), reported by Frantar and Alistarh (2023). The experiments use DeepSparse engine (NeuralMagic, 2021).