Outlier Weighed Layerwise Sparsity (OWL): A Missing Secret Sauce for Pruning LLMs to High Sparsity

Lu Yin [1 2 3] **You Wu** [3] **Zhenyu Zhang** [4] **Cheng-Yu Hsieh** [5] **Yaqing Wang** [3] **Yiling Jia** [3] **Gen Li** [6] **Ajay Jaiswal** [4] **Mykola Pechenizkiy** [2] **Yi Liang** [3] **Michael Bendersky** [3] **Zhangyang Wang** [4] **Shiwei Liu** [7 2]

Abstract

Large Language Models (LLMs), renowned for their remarkable performance across diverse domains, present a challenge when it comes to practical deployment due to their colossal model size. In response to this challenge, efforts have been directed toward the application of traditional network pruning techniques to LLMs, uncovering a massive number of parameters that can be pruned in one-shot without hurting performance. Building upon insights gained from previous work, prevailing LLM pruning strategies have consistently adhered to the practice of uniformly pruning all layers at equivalent sparsity, resulting in robust performance. However, this observation stands in contrast to the prevailing trends observed in the field of vision models, where non-uniform layerwise sparsity typically yields stronger results. To understand the underlying reasons for this disparity, we conduct a comprehensive study and discover a strong correlation with the emergence of activation outliers in LLMs, which are output features exhibiting significantly greater magnitudes compared to their counterparts. Inspired by this finding, we introduce a novel LLM pruning methodology that incorporates a tailored set of **non-uniform layerwise sparsity ratios**, termed as **O**utlier **W**eighed **L**ayerwise sparsity (**OWL**). The sparsity ratio of OWL is proportional to the outlier ratio observed within each layer, facilitating a more effective alignment between layerwise weight sparsity and outlier ratios. Our empirical evaluation, conducted across the LLaMA-V1/V2, Vicuna, OPT, and Mistral, spanning various benchmarks, demonstrates the distinct advantages offered by OWL over previous methods. For instance, OWL exhibits a remarkable performance gain, surpassing the state-of-the-art Wanda and SparseGPT by **61.22** and **6.80** perplexity at a high sparsity level of 70%, respectively, while delivering **2.6**× end-to-end inference speed-up in the DeepSparse inference engine. Code is available at https://github.com/luuyin/OWL.git.

[1]University of Surrey [2]Eindhoven University of Technology [3]Google Research [4]University of Texas at Austin [5]University of Washington [6]Clemson University [7]University of Oxford. Correspondence to: Shiwei Liu <shiwei.liu@maths.ox.ac.uk>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

1. Introduction

The remarkable performance exhibited by Large Language Models (LLMs) across a diverse spectrum of applications has ignited an unparalleled race among tech giants and academic institutions to build LLMs at the billion-parameter scale (Brown et al., 2020; Touvron et al., 2023a;b; Brown et al., 2020). The compelling performance of Large Language Models (LLMs) demonstrated in various applications triggers an unprecedented competition of building billion-level LLMs among tech giants and academic institutions (Brown et al., 2020; Touvron et al., 2023a;b; Brown et al., 2020). While their exceptional capabilities are undeniable, the colossal size and computational demands of these models have also raised substantial concerns, particularly in terms of financial expenditure and environment (Luccioni et al., 2022; Patterson et al., 2021).

Network pruning (Mozer & Smolensky, 1989; Janowsky, 1989; LeCun et al., 1989; Han et al., 2015), as a long-established model compression method, is expected to serve as an effective solution for reducing the size of LLMs. However, network pruning usually favors a certain time of fine-tuning or re-training to reacquire the original optimal performance. Given the extensive text corpus and model size associated with LLMs, conventional fine-tuning becomes exceedingly challenging and less desirable. Fortunately, recent endeavors have explored the possibility of LLM pruning without the need for fine-tuning, showcasing that LLMs contain a substantial number of parameters that can be removed in a single step with minimal performance degradation (Frantar & Alistarh, 2023; Sun et al., 2023; Jaiswal

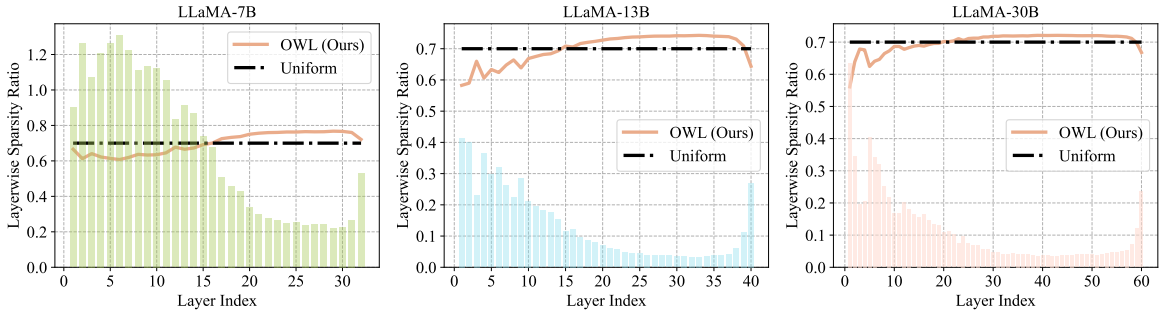

Figure 1: The demonstration of the OWL layerwise sparsity and Uniform layerwise sparsity at 70% sparsity. The bar chart in the background corresponds to the Layerwise Outlier Distribution (LOD), as elaborated in Section 3.2

et al., 2023b; Ma et al., 2023). SparseGPT (Frantar & Alistarh, 2023) addresses the challenge of LLM pruning from the perspective of layerwise reconstruction problem. In this context, the primary goal is to minimize the output discrepancy in terms of the reconstruction error between dense and sparse LLMs. It adopts an iterative strategy to handle the computational hurdle posed by the row-Hessian problem. Specifically, it employs the Optimal Brain Surgeon (OBS) algorithm (Hassibi et al., 1993) to selectively prune and update weights in a column-wise manner. Wanda (Sun et al., 2023), on the other hand, introduces a novel pruning metric that takes into account both the weight magnitudes and their corresponding input activations. Remarkably, it achieves performance on par with SparseGPT without relying on computationally expensive second-order information. The effectiveness of Wanda stems from the emergence of the outlier features residing within LLMs. These outliers, which tend to be significantly larger than typical features, are nonetheless crucial for optimizing LLM performance (Kovaleva et al., 2021; Puccetti et al., 2022; Timkey & van Schijndel, 2021; Dettmers et al., 2022). Both SparseGPT and Wanda exhibit appealing performance, showcasing their ability to reduce model parameters by up to 50% while incurring only a modest increase of perplexity (Sun et al., 2023).

It is worth noting that SparseGPT and Wanda unanimously follow previous work on BERT pruning (Sanh et al., 2020; Kurtic et al., 2022) and choose to prune LLMs with a uniform sparsity ratio per layer, *i.e.,* each layer will be pruned at the same sparsity. Such choice is reasonable for LLMs, as the pruning process typically involves sorting the importance scores of weights. Conducting such sorting globally across layers could become a computational bottleneck, especially for models at the billion-parameter scale. Nevertheless, before it has been taken root that uniform layerwise sparsity is the default choice for LLMs, we raise a timely inquiry: *are there any pivotal aspects that have been inadvertently omitted in the context of favorable layerwise sparsity ratios for LLM pruning?*

Three reasons behoove us to pose the above research question: *First*, it is widely acknowledged that within Transformer architectures, certain components hold greater significance than others, and thus, they merit distinct treatment during the pruning process (Wang & Tu, 2020; Bhojanapalli et al., 2021); *Second*, a consensus view has been reached in computer vision that non-uniform layerwise sparsity typically achieves stronger results than uniform sparsity (Liu et al., 2022a; Lee et al., 2020); *More importantly*, LLMs demonstrate astonishingly emergent behaviors (Wei et al., 2022; Schaeffer et al., 2023; Dettmers et al., 2022) as model size continuously scales up, a phenomenon distinct from smaller-scale language models. These emergent behaviors offer fresh insights into the domain of LLM pruning. For instance, the existence of outlier features within LLMs, with magnitudes up to 20 times larger than others, exerts a profound influence across all Transformer layers (Kovaleva et al., 2021; Dettmers et al., 2022).

Contributions. Given the pivotal role that outliers play in LLMs, coupled with the demonstrated effectiveness of Wanda (Sun et al., 2023), our initial investigation centers on a systematic examination of the impact of existing LLM pruning methodologies on outliers. We uncover a strong correlation between pruning efficacy and the retention ratio of outliers: contemporary state-of-the-art LLM pruning approaches, such as SparseGPT and Wanda, exhibit remarkable preservation of outliers, even though the former was not originally designed with this intent. Moreover, we conduct an in-depth analysis of the distribution of outliers across different layers and observe **a notably non-uniform pattern**. This non-uniform distribution emerges as a valuable indicator for the formulation of layerwise sparsity strategies tailored specifically for LLMs. Building upon this newfound insight, we introduce an LLM pruning paradigm characterized by a novel layerwise sparsity ratio, denoted as **O**utlier **W**eighed **L**ayerwise sparsity (**OWL**). OWL inherently assigns greater emphasis to layers housing a higher prevalence of outliers, thereby facilitating more nuanced coordination between sparsity in weight matrices and the presence of outliers within the layer.

We conduct extensive experiments to evaluate the performance OWL across a spectrum of LLMs, including LLaMA-V1 (Touvron et al., 2023a), and OPT (Zhang et al., 2022), from 7B to 65B. Our empirical results show that OWL consistently outperforms existing top-performing LLM pruning methods, particularly at high sparsity levels. For instance, we observe significant improvements achieved by OWL over Wanda with LLaMa-7B on WikiText (Merity et al., 2016a), with perplexity reductions of more than 60 and 3300 perplexity at sparsity levels of 70% and 80%, respectively. When evaluated in the DeepSparse (DeepSparse, 2021) inference engine, OWL delivers a 2.6× - 3.9× end-to-end speedup on CPUs with 70% - 90% sparsity.

Overall, our research provides a compelling counter-argument to previous studies by highlighting the previously overlooked yet crucial role of layerwise sparsity ratios in LLM pruning. This change in perspective has enabled us to push the boundaries of achievable one-shot LLM pruning ratios to 70%. *Note that while non-uniform layerwise sparsity has been extensively explored in network pruning, our paper represents the first effort to make it applicable in LLM pruning, challenging the conventional belief that uniform layerwise sparsity is the default and optimal choice for LLM pruning.*

2. Related Work

Pruning and LLM Pruning. Since the 1980s, network pruning has been a well-established technique for simplifying neural networks in various applications while maintaining accuracy (Mozer & Smolensky, 1989; Han et al., 2015; Mocanu et al., 2018; Wen et al., 2017; Lin et al., 2019). However, when it comes to pruning Large Language Models (LLMs), progress has been limited. Traditional pruning typically requires a round of re-training to restore performance, which can be challenging for LLMs. To address this challenge, researchers have developed pruning algorithms specifically tailored for LLM compression. For example, Ma et al. (2023) explored structured sparse LLMs using Taylor pruning to remove entire weight rows, followed by LoRA fine-tuning (Hu et al., 2021). Recent research has shifted toward unstructured pruning without the need for fine-tuning, showing substantial advancements. SparseGPT (Frantar & Alistarh, 2023) utilizes the Hessian inverse for pruning and with subsequent weight updates to reduce reconstruction error of dense and sparse weights, while Wanda (Sun et al., 2023) produces a criterion incorporating weight magnitude with their input activations, aiming to preserve outlier features. Zhang et al. (2023) extended dynamic sparsity (Mocanu et al., 2018; Evci et al., 2020; Liu et al., 2021b) to efficiently fine-tune sparse LLM without weight updating. Our work for the first time probes the crucial role of non-uniform layerwise sparsity for LLM pruning, making good progress in this field.

Layerwise Sparsity for Pruning. While it is common to use uniform layerwise sparsity (Zhu & Gupta, 2017; Gale et al., 2019) to prune language models (Sanh et al., 2020; Kurtic et al., 2022), there is a well-established line of work that explore non-uniform layerwise sparsity in terms of pruning vision models. Mocanu et al. (2016) propose a non-uniform and scale-free topology inspired from graph theory, showing better performance than the dense counterpart when applied to restricted Boltzmann machines. Follow-up works significantly improve its scalability based on *Erdős-Rényi* graph (Erdős & Rényi, 1959), extending to fully-connected layers (Mocanu et al., 2018) and convolutional layers (Evci et al., 2020; Liu et al., 2022a) as data-free and feedforward-free layerwise sparsity. Another group of work produces non-uniform sparsity by applying a global threshold on every layer (Frankle & Carbin, 2019; Lee et al., 2019; Wang et al., 2020; Lee et al., 2020; Liu et al., 2021a). However, global pruning becomes extremely expensive and inefficacious in the context of LLM pruning as shown in Table 2. We also provide a comparison among the most common layerwise sparsity for LLMs in Section 5, and all of them fail to perform on LLMs.

Outliers in LLMs. Recent studies have revealed certain emergent characteristics unique to language models at scale. Specifically, one intriguing trait of LLMs is the exhibition of *outlier features*, which are the features with significantly larger magnitudes than others (Kovaleva et al., 2021; Puccetti et al., 2022; Timkey & van Schijndel, 2021; Dettmers et al., 2022). Kovaleva et al. (2021) first discovered that a very small number of outlier dimensions appears in the pre-trained encoder layers of GPT-2. Puccetti et al. (2022) further demonstrated that outliers are causally related to high-frequency tokens in pre-training data. While constituting only a very small portion of the entire feature dimensions, these outliers play an imperative role in models' predictive performance. Given the importance of outliers, several recent works have developed techniques to effectively quantize LLMs with minimal performance drop (Dettmers et al., 2022; Xiao et al., 2023; Lin et al., 2023). However, in the context of LLM pruning, this unique characteristic has scarcely been explored except for Sun et al. (2023). Our work draws inspiration from the emergent outliers in LLMs, and provides a new technique that leverages the distribution of outliers to guide layerwise LLM pruning.

3. Outlier Weighed Layerwise Sparsity

In this section, we will introduce **O**utlier-**W**eighed **L**ayerwise sparsity (**OWL**) step by step, from rationale to empirical studies, and eventually to our algorithm.

3.1. Rationale

The success of pruning in pre-LLM model compression is closely intertwined with the feasibility of fine-tuning or

re-training. It has been observed that even the random removal of components can ultimately restore the original performance through adequate re-training (Liu et al., 2022a; Mittal et al., 2019). However, fine-tuning encounters significant challenges when applied to LLMs, rendering pruning less effective. Notably, Wanda (Sun et al., 2023) achieves remarkable performance by augmenting input activation with weight magnitude, underscoring the critical importance of preserving outlier features in LLM pruning. Considering the vital role that outliers play in the context of LLMs (Kovaleva et al., 2021; Dettmers et al., 2022) and the success of Wanda, we conjecture that the performance of different pruning methods has a strong correlation with their ability to preserve outlier features. To assess our conjecture, we undertake several preliminary investigations outlined below based on Layerwise Outlier Distribution.

3.2. Empirical Study

Layerwise Outlier Distribution (LOD). Our preliminary studies are predominantly based on Layerwise Outlier Distribution (LOD), a concept used to measure the across-layer outlier distribution. Since we focus on weight pruning in this paper, we opt to prioritize the weight outliers instead of the activation outliers, which are identified as weights whose outlier scores are at least $\mathbf{M}$ times larger than the mean. The outlier score of weight is calculated as the accumulation of all input features connected to that weight, multiplied by its magnitude, which also serves as the pruning metric used by Wanda (Sun et al., 2023). By measuring the ratio of weight outliers in each layer, we can obtain the LOD of the whole model.

To formalize our approach, let us consider the input of a layer as $\mathbf{X}$ with dimensions $(N \times L, C_{\text{in}})$, where N and L represent the batch and sequence dimensions, respectively; and the weight matrix $\mathbf{W}$ has dimensions $(C_{\text{out}}, C_{\text{in}})$. The outlier score of weight $\mathbf{W}_{\text{ij}}$ is computed as $\mathbf{A}_{\text{ij}} = \|\mathbf{X}_{\text{j}}\|_2 \cdot |\mathbf{W}_{\text{ij}}|$, which is the aggregation of all input features connected to weight $\mathbf{W}_{\text{ij}}$, multiplied by its magnitude $|\mathbf{W}_{\text{ij}}|$. Here, $\|\mathbf{X}_{\text{j}}\|_2$ is the ℓ_2 norm of input feature connected to the weight. This computation is performed across all $N \times L$ tokens. Subsequently, after obtaining the outlier score for all weights, we proceed to calculate the "outlier ratio" of $\mathbf{A}$ by identifying elements whose magnitude is $\mathbf{M}$ times greater than the averaged value in each layer. We empirically find that $\mathbf{M} = 5$ or $\mathbf{M} = 7$ usually works well to sketch the distribution of weight outliers. This process enables us to derive a vector, denoted as LOD = $[D^1, D^2, ..., D^n]$, where D^l characterizes the outlier distribution of layer l. That is,

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

where $\bar{\mathbf{A}}^l$ is the mean of $\mathbf{A}^l$ and $\mathbb{I}(\cdot)$ is the indicator function, returning 1 if $\mathbf{A}^l_{\text{ij}}$ is larger than $\mathbf{M} \cdot \bar{\mathbf{A}}^l$, else 0. Based on LOD, we conduct three empirical studies outlined below to better understand the effect of current LLM pruning approaches on outliers.

Empirical Study I: Dense LLMs vs. LOD. To investigate whether sparsifying LLMs necessitates differential treatment of individual layers, we employ LOD to gauge the layerwise distribution of outliers within dense LLMs. If LOD in dense LLMs exhibits a relatively uniform pattern, it suggests that we do not need a non-uniform layerwise sparsity ratio to preserve outliers, and vice versa. We assess the LOD with LLaMA-7B, 13B, and 30B.

Table 1: Effects of various pruning methods on Layerwise Outlier Distribution (LOD) and Perplexity with LLaMA-13B on WikiText. LOD is calculated as the *summation* across all layers with $\mathbf{M} = 7$.

Sparsity	Method	LOD (%) ↑	ΔLOD (%) ↑	Perplexity ↓
	Dense	5.432	-	5.090
70%	Wanda	5.716	0.284	55.900
	SparseGPT	**6.645**	**1.213**	**19.235**
	Magnitude	5.322	-0.110	84539.445
60%	Wanda	5.433	0.001	8.761
	SparseGPT	**6.044**	**0.612**	**8.458**
	Magnitude	5.322	-0.110	229.451

Empirical Study II: Pruning Metric vs. LOD. We further delve into the impact of different pruning metrics on LOD. The primary objective of this study is to explore whether there exists a robust correlation between the performance of various pruning methods and their ability to preserve outliers. To achieve this, we *aggregate* the LOD values across layers for various LLM pruning methods, including magnitude, Wanda, and SparseGPT, and compare them with their dense counterparts. To mitigate the influence of pruning on the mean of outlier score $\mathbf{A}$, we use the pre-pruning mean value to measure the outlier score after pruning. Subsequently, the number of outlier weights after pruning is then divided by the total number of weights (including both zero and non-zero weights) to obtain the updated weight outlier ratio. Doing so helps avoid the impact of pruning on the mean outlier score, ensuring a precise evaluation of alterations in the outlier ratio. All sparse models are pruned with uniform layerwise sparsity. These experiments are conducted using LLaMA-13B at sparsity levels of 60% and 70% with $\mathbf{M} = 7$.

Empirical Study III: Pruning Granularity. It is well-established that non-uniform or global layerwise sparsity often leads to more accurate sparser networks at high sparsity than the uniform layerwise sparsity for pre-LLM pruning. However, endeavors unanimously point out that uniform sparsity is more favorable for LLM pruning. To provide more insights into these two seemingly contradictory arguments, we study the effect of various pruning granularities

Table 2: WikiText perplexity with LLaMA-7B of various pruning granularity.

Method	Layerwise Uniform	Output Balanced	Sparsity 10%	20%	30%	40%	50%	60%	70%
Wanda	✓	✓	5.697	5.817	5.999	6.388	7.260	10	86
Wanda	✓	✗	5.695	5.819	6.029	6.572	7.942	20	238
Wanda	✗	✗	14.117	3134	10293	10762	14848	17765	5147
Magnitude	✓	✓	5.803	6.018	6.622	8.041	13.349	152	25304
Magnitude	✓	✗	5.806	6.020	6.669	8.601	17.287	559	48419
Magnitude	✗	✗	5.821	6.111	7.012	9.825	48.627	38335	29283

on LLMs. Specifically, we study two sets of pruning granularities: (1) **Across different layers,** we compare the performance of uniform pruning and global pruning; (2) **Within the same layer,** we study the output-imbalanced sparsity used by SparseGPT against the output-balanced sparsity adopted by Wanda. Output-balanced sparsity eliminates the same amount of weights for all outputs. We conduct experiments with magnitude pruning and Wanda using LLaMA-7B. While this part of the study is irrelevant to LOD, we place it here due to the crucial role of pruning granularity.

Results: We present our findings from Study 1-3, in Figure 1, Table 1, and Table 2, respectively. These results provide positive support for our conjecture, and we summarize the key observations below:

① **LOD of dense LLMs exhibits a highly non-uniform distribution across layers.** In essence, the distribution of dense LLMs shown in Figure 1 loosely follows a "U" shape, with notable proportions at both ends, while the central region displays a monotonic descending trend. This finding validates our conjecture that individual layers need unique consideration during the pruning procedure. Employing uniform pruning across all layers would inevitably disrupt the outlier structure in layers characterized by a large outlier ratio, such as those layers at the beginning or end of models.

② **The performance of sparse pruning methods on LLMs is closely correlated with their ability to retain outlier features.** Leading pruning techniques like Wanda and SparseGPT all excel in outlier, resulting in an overall increase in LOD. In contrast, the naive baseline of magnitude pruning performs no better than random selection at 70% sparsity, as evidenced by a negative change of -0.110 in LOD, indicating the removal of important outliers. It is interesting to see that despite SparseGPT not being explicitly designed for outlier preservation, it achieves the highest LOD as well as performance, providing further insight into the underlying reason for its success. The potential explanation could be that the weight update involved within SparseGPT contributes to the increase in LOD.

③ **Pruning with coarser granularity results in diminished performance.** In general, we observe a consistent trend of improved perplexity as the pruning granularity becomes finer, transitioning from global layerwise sparsity to uniform layerwise sparsity at the macro level, and from output imbalanced sparsity to output balanced sparsity at the micro level. These findings align with the conclusions presented by Sun et al. (2023). This observation suggests the importance of a nuanced design of pruning ratios to mitigate aggressive sparsity differences among different components in LLMs. This motivation led us to constrain the sparsity ratio of each layer to fluctuate only around the target sparsity, with a hyperparameter λ introduced in Section 3.3.

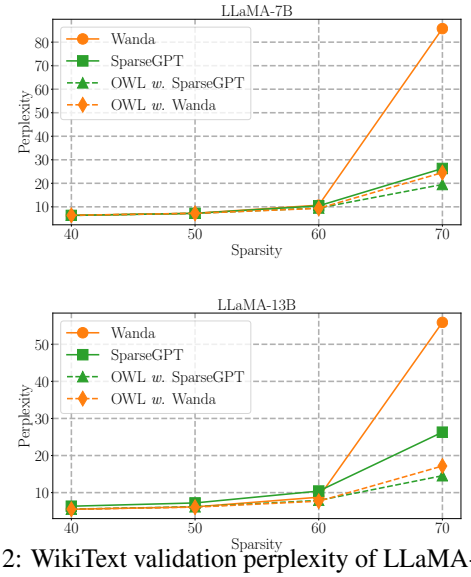

Figure 2: WikiText validation perplexity of LLaMA-7B and LLaMA-13B pruned by various approaches.

3.3. Outlier Weighed Layerwise Sparsity (OWL)

The above empirical studies underscore the critical significance of preserving outliers in the context of LLM pruning. Consequently, it becomes imperative to implement layerwise pruning strategies that take into account the non-uniform distribution of outliers across different layers. However, global pruning can be costly and lead to the collapse of outliers, resulting in significant performance degradation. On the other hand, uniform pruning does not adequately consider the highly non-uniform distribution of outlier features across various layers. This negligence inevitably disrupts the structure of outliers in layers characterized by a substantial outlier ratio, particularly at high sparsity levels. Therefore, there is a need of an ideal layerwise sparsity that aligns effectively with the layerwise outlier distribution while maintaining computational and memory efficiency.

To address this issue, we propose a novel layerwise sparsity ratio strategy, referred to as **O**utlier-**W**eighed **L**ayer-wise sparsity (**OWL**) explicitly tailored for LLMs, which can better coordinate with the outlier distribution by taking the layerwise outlier ratio into consideration. Given a l-layer large language model with a target model sparsity S, we aim to calculate the target layerwise sparsity $[S^1, S^2, ..., S^n]$. We first calculate LOD, $\mathbf{D} = [D^1, D^2, ..., D^n]$, based on the approach proposed in Section 3.2. Guided by the principle that layers with a higher proportion of outliers should

Table 3: WikiText validation perplexity of pruning methods for LLaMA-V1 family and OPT-6.7B at 70% sparsity. The best performance method is indicated in **bold**, and the gain in perplexity achieved by OWL is highlighted in blue.

Method	Layerwise Sparsity	Weight Update	LLaMA-V1 7B	LLaMA-V1 13B	LLaMA-V1 30B	LLaMA-V1 65B	OPT 6.7B
Dense	-	-	5.68	5.09	4.10	4.77	10.13
Magnitude	Uniform	✗	48419.12	84539.45	977.73	46.89	290985.03
Wanda	Uniform	✗	85.77	55.90	17.37	15.23	162.92
OWL *w.* Wanda	Non-Uni	✗	**24.55** (-61.22)	**17.17** (-38.73)	**10.75** (-6.62)	**8.61** (-6.62)	**40.22** (-120.70)
SparseGPT	Uniform	✓	26.30	19.24	12.56	10.45	**20.29**
OWL *w.* SparseGPT	Non-Uni	✓	**19.49** (-6.81)	**14.55** (-4.69)	**10.28** (-2.28)	**8.28** (-0.64)	22.48 (2.19)

have a lower sparsity, we set $S_i \propto 1 - D_i$. Additionally, we introduce a hyperparameter λ which constrains the layerwise sparsity to fall within a small range, specifically, $S_i \in [S - \lambda, S + \lambda]$, while maintaining an average sparsity of S across all layers. This helps prevent excessive differences in sparsity between layers, ensuring a robust performance. This constraint is inspired by the insights gained from "Empirical Study III" which highlight the detrimental impact of overly aggressive layerwise sparsity, akin to global pruning, on sparse LLMs. To obtain a favorable number for λ and M, we conduct a small hyperparameter sweep within the range of $\lambda \in$ [0.02, 0.05, 0.08, 0.1, 0.2] and for $M \in$ [3, 5, 7, 10]. Note that we assign a distinct pruning ratio for each Transformer block instead of each layer, as the former provides better performance (see Appendix C for more detail). The visualization of our layerwise sparsity ratio is demonstrated in Figure 1, where we can see that the layerwise sparsity level of OWL nuancedly aligns well with the model's `LOD`.

4. Experiments

4.1. Main Experiments

Models and Dataset. We assess OWL's performance across a range of LLMs, encompassing the LLaMA-V1 model family (Touvron et al., 2023b) with parameter counts ranging from 7 billion to 65 billion, as well as OPT-6.7B (Zhang et al., 2022). Our evaluation protocol aligns with established LLM pruning methodologies (Frantar & Alistarh, 2023; Sun et al., 2023), encompassing assessments of language modeling proficiency and zero-shot capabilities of sparse LLMs. Specifically, we measure the Perplexity metric on the WikiText (Merity et al., 2016b) validation dataset for language modeling performance, and employ the Accuracy metric for zero-shot evaluations on seven common sense benchmarks, including BoolQ (Clark et al., 2019), RTE (Wang et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2019), ARC Easy and Challenge (Clark et al., 2018), and OpenbookQA (Mihaylov et al., 2018).

Baselines. We choose the three current LLM-pruning baselines, including magnitude (Jaiswal et al., 2023b), SparseGPT (Frantar & Alistarh, 2023), Wanda (Sun et al., 2023). Magnitude pruning serves as a naive baseline for LLMs, with an expected sharp decline in performance at modest sparsity levels, typically ranging from 10% to 30%. SparseGPT and Wanda, on the other hand, are established baselines known for their ability to maintain reasonable performance even at relatively high sparsity levels, typically around 50% to 60%. Notably, in contrast to our approach, all baseline methods employ with uniform layerwise sparsity. We primarily focus on high sparsity levels, not falling below 50%, as regions with low sparsity pose challenges for existing sparse GPU kernels to outperform their dense counterparts (Gale et al., 2020). To ensure equitable comparisons, we have employed the identical set of calibration data as utilized by SparseGPT and Wanda for model pruning, *i.e.,* comprising 128 sequences with 2048 tokens for each, randomly sampled from the first shard of the C4 (Raffel et al., 2020) dataset. We incorporate OWL directly into Wanda and SparseGPT, resulting in two variants: "OWL *w.* Wanda" and "OWL *w.* SparseGPT". The only distinction between these variants lies in their layerwise sparsity ratios, with OWL providing a more tailored layerwise sparsity in this regard. Hyperparameters are shared in Appendix D.

Language Modelling. We first report the performance of various LLM pruning methods on language modelling with WikiText. The results is presented in Table 3 and Figure 2. We summarize the key observation below:

① OWL serves as a general layerwise sparsity method suitable for various scenarios. As illustrated in Table 3, OWL exhibits effectiveness across different pruning methods (such as Wanda and SparseGPT), architectural variants (including LLaMA-V1 and OPT), and diverse model sizes (ranging from 7B, 13B, 30B, to 65B parameters), resulting in substantial reductions in perplexity scores. Notably, even when applied to SparseGPT, a strong pruning method incorporating second-order information, OWL still achieves significant perplexity reductions, exemplified by a reduction of 6.81 for LLaMA-7B.

② The benefits of OWL increase significantly as model size decreases. There is a clear trend that the performance gain of OWL monotonically increases as LLaMA-V1 scales down from 65B to 7B. While the performance improvement

Table 4: Accuracies (%) for 7 zero-shot tasks with 70% sparsity using LLaMA-V1 family.

Params	Method	BoolQ	RTE	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Mean
7B	Dense	75.14	66.43	74.80	70.01	67.67	41.38	41.40	62.40
	Magnitude	38.29	52.71	24.68	51.46	26.98	22.35	25.80	34.61
	Wanda	55.11	57.40	31.83	51.38	34.22	19.80	26.00	39.39
	OWL *w.* Wanda	**62.48**	**58.48**	**44.79**	**58.72**	**45.03**	**26.19**	**29.60**	**46.47**
	SparseGPT	64.53	**53.79**	42.11	58.64	43.06	24.57	27.80	44.93
	OWL *w.* SparseGPT	**67.13**	53.43	**48.56**	**62.03**	**45.41**	**27.65**	**32.00**	**48.03**
13B	Dense	77.86	70.40	78.08	72.77	69.19	47.18	43.80	65.61
	Magnitude	52.94	50.54	27.67	50.91	28.24	23.38	24.80	36.93
	Wanda	61.71	**52.71**	34.31	52.33	37.16	20.90	29.60	41.25
	OWL *w.* Wanda	**62.69**	**52.71**	**51.03**	**63.14**	**49.54**	**28.67**	**34.40**	**48.88**
	SparseGPT	**66.94**	52.71	47.91	62.90	45.03	27.99	35.20	48.38
	OWL *w.* SparseGPT	64.95	**53.07**	**54.39**	**66.54**	**48.86**	**30.12**	**38.00**	**50.85**
30B	Dense	82.69	66.79	81.19	75.85	73.48	50.77	44.60	67.91
	Magnitude	39.14	46.21	24.31	52.33	24.66	22.87	29.00	34.07
	Wanda	66.12	**57.76**	58.84	67.32	59.26	33.11	**40.20**	54.66
	OWL *w.* Wanda	**66.42**	52.35	**62.94**	**69.30**	**61.83**	**35.84**	40.00	**55.53**
	SparseGPT	66.51	**63.90**	60.38	69.85	58.54	33.70	40.60	55.78
	OWL *w.* SparseGPT	**67.58**	58.48	**64.88**	**70.72**	**60.82**	**35.07**	**42.20**	**57.11**
65B	Dense	84.86	69.68	82.94	77.35	75.08	52.56	44.20	69.52
	Magnitude	52.17	54.87	49.87	56.67	49.71	30.63	38.80	47.53
	Wanda	76.30	56.68	61.26	70.48	63.47	35.67	39.40	57.61
	OWL *w.* Wanda	**80.12**	**58.84**	**66.16**	**73.56**	**65.45**	**39.93**	**42.20**	**60.89**
	SparseGPT	80.64	59.57	66.42	72.61	**60.52**	38.57	**40.80**	59.88
	OWL *w.* SparseGPT	**82.63**	**67.15**	**68.52**	**75.06**	60.10	**39.59**	39.00	**61.72**

Table 5: WikiText perplexity of "OWL *w.* SparseGPT" with LoRA fine-tuning.

Method	Model	Sparsity	Perplexity
Without FT	7B	0.7	19.49
With FT	7B	0.7	11.15
Without FT	13B	0.7	14.55
With FT	13B	0.7	9.0

Table 6: Comparison of time overhead used for computing the pruning metric across layers of LLaMA (in seconds).

Method	7B	13B	30B	65B
SparseGPT	266	436	869	1395
OWL *w.* SparseGPT	268	438	870	1397
Wanda	0.3	0.4	1.1	1.8
OWL *w.* Wanda	0.3	0.5	1.2	2.0

of OWL *.w* Wanda for LLaMA-65B is already encouraging i.e., 6.62, it achieves a remarkable gain of 61.22 for LLaMA-7B, resulting in a reasonable 24.55 perplexity.

Zero-Shot Tasks. While perplexity is a widely used metric for language modeling, it primarily serves as a statistical measure of how confidently a language model predicts a text sample and does not necessarily align with the quality of the generated text (Jaiswal et al., 2023a). To draw more robust conclusions, we conducted experiments to evaluate the zero-shot ability of various sparse LLMs on diverse zero-shot downstream tasks with prompting. These experiments were performed using the LLaMA-V1 family at 70% sparsity, and the results are presented in Table 4. It's noteworthy that OWL consistently improves accuracy across nearly all settings, with very few exceptions on RTE dataset. For example, OWL achieves an average perplexity gain of 4.72 and 2.19 over 7 tasks and 4 model sizes compared to Wanda and SparseGPT alone, respectively. This result highlights the promise of OWL still holds for more challenging tasks.

Fine-tuning Performance. We also explore the impact of fine-tuning on the performance recovery of OWL. In alignment with Wanda, we utilize LoRA (Hu et al., 2021) as our fine-tuning method and refrain from merging the adapter back to preserve the sparse pattern. We fine-tune models pruned by "OWL + SparseGPT" using only a minimal 30,000 tokens from the C4 training dataset. Remarkably, our results demonstrate that the perplexity drop caused by aggressive pruning can be significantly narrowed through a very short time of fine-tuning, reducing the perplexity from 19.49 to 11.15 of LLaMA-7B and from 14.55 to 9.0 for LLaMA-13B. We anticipate achieving a significantly lower perplexity by employing more advanced sparse fine-tuning approaches (Zimmer et al., 2023), or by extending the fine-tuning duration.

Pruning Efficiency. Since we utilize the pruning metric of Wanda to determine our layerwise sparsity, the computa-

Table 7: End-to-end decode latency speedup of LLaMA-V2-7B-chat-hf using OWL with the DeepSparse (DeepSparse, 2021) inference engine.

Sparsity	Dense	10%	20%	30%	40%	50%	60%	70%	80%	90%
Latency (ms)	213.83	216.86	221.62	218.01	167.54	121.25	101.41	81.89	64.57	54.24
Throughput (tokens/sec)	4.68	4.61	4.51	4.59	5.97	8.25	9.86	12.21	15.48	18.43
Speedup	1.0x	1.0x	1.0x	1.0x	1.3x	1.8x	2.1x	2.6x	3.3x	3.9x

Table 8: WikiText validation perplexity of LLaMA-7B with various layerwise sparsity using Wanda.

Sparsity/Perplexity	10%	20%	30%	40%	50%	60%	70%	80%
Global	14.11	3134	10293	10762	14848	17765	5147	39918.56
ER-plus	5.70	5.82	6.05	6.62	8.00	14.04	229.17	6013.91
ER	5.69	5.80	6.02	6.55	7.74	12.16	112.03	11151.18
Uniform	5.69	5.81	5.99	6.38	7.26	10.70	85.77	3499.88
OWL-inverse	5.72	5.83	6.04	6.51	8.03	26.05	822.23	9616.08
OWL (ours)	5.70	5.80	6.01	6.39	7.22	9.35	24.54	1002.87

tional complexity of OWL is comparable to that of Wanda. To demonstrate this, we measure the total pruning time, excluding the forward pass process, following the methodology outlined by Sun et al. (2023). These results were obtained using NVIDIA A100 GPUs. Our results in Table 6 indicate that OWL incurs negligible time overhead (maximum 2 seconds) relative to previous pruning approaches.

Inference Speedup. We analyze the speedup achieved by OWL, as presented in Table 7. The reported speedups correspond to end-to-end decode latency using LLaMA-V2-7B-chat-hf in the DeepSparse inference engine (DeepSparse, 2021) on an Intel Xeon Platinum 8360Y CPU with 36 cores. It is evident that OWL delivers a significant inference speedup compared to the dense model, reaching 2.6× at 70% sparsity. Notably, the speedup gain becomes even more substantial with higher sparsity e.g., around 4× at 90% sparsity, showcasing additional motivation for future endeavors targeting extreme sparsity.

4.2. More Advanced LLMs

Table 9: WikiText Perplexity of Various LLMs.

Method	Models	Sparsity 60%	70%	80%
Wanda	LLaMA-V2-7B	11.63	57.10	3221.74
OWL *w.* Wanda	LLaMA-V2-7B	**10.30**	**29.07**	**523.30**
Wanda	Vicuna-7B	11.93	57.98	2347.49
OWL *w.* Wanda	Vicuna-7B	**10.58**	**34.44**	**719.11**
Wanda	Mistral-7B	11.27	60.67	306.26
OWL *w.* Wanda	Mistral-7B	**10.29**	**41.02**	**220.89**

To examine if the effectiveness of OWL is robust across various LLMs, we also apply OWL to more advanced LLMs including LLaMA-V2-7B-chat-hf (Touvron et al., 2023b), Vicuna-7B, and Mistral-7B (Jiang et al., 2023). Table 9 further illustrates the efficacy and robustness of OWL as a layerwise sparsity technique across a spectrum of Large Language Models (LLMs). Moreover, Mistral appears to preserve the perplexity better at high sparsity such as 80%.

4.3. More Practical Applications of OWL

Although GPUs provide limited support for unstructured sparsity, we highlight that OWL not only can be used to determine the sparsity ratio for unstructured pruning but also serves as a general approach to identify layer importance for LLMs, which exhibits substantial potential in many important scenarios. This section provides some preliminary results for low-rank approximation of LLMs. You can find more preliminary results across three more practical scenarios including N:M sparsity, structured pruning, and mixed-precision quantization in Appendix B.

Table 10: WikiText Perplexity of LLaMA-V1-7B Model.

Rank Reduction Ratio	0%	20%	30%	40%	50%	60%
Uniform - SVD	5.68	8.48	17.23	1909.34	13627.03	34354.9
OWL - SVD	5.68	**8.25**	**12.92**	**43.02**	**10707.41**	**20118.7**

SVD Low-rank Approximation. We expanded the OWL approach to incorporate low-rank compression as follows. In alignment with the OWL methodology, we evaluate the significance of each layer through the LoD. Subsequently, we utilize these LoD scores to assign SVD compression ratios to each layer. A higher LoD score signifies a layer of greater importance, warranting reduced compression to maintain its integrity. For any linear layer L equipped with a weight matrix $W \in \mathbb{R}^{d_1 \times d_2}$ and a designated preserved rank r, the compression ratio is defined by $\frac{r}{d_{\text{min}}}$, where d_{min} is the lesser of d_1 and d_2. In our evaluation, we present the perplexity metrics on the LLaMA-V1 7B model without any fine-tuning. The results clearly demonstrate that OWL-SVD surpasses uniform SVD low-rank compression across various rank reduction levels.

5. Analysis

Comparisons Among Various Layerwise Sparsity. We compare OWL layerwise sparsity with multiple commonly used layerwise sparsity, including **Global**: A global threshold is uniformly applied to all layers to satisfy the overall sparsity requirement, and the specific layerwise sparsity is automatically adjusted based on this threshold. **Uniform** (Zhu & Gupta, 2017): Every layer is pruned with the same target sparsity. **Erdős-Rényi (ER)** (Mocanu et al., 2018): The sparsity of the convolutional layer is scaled proportionally to $1 - \frac{n^{l-1}+n^l}{n^{l-1}\times n^l}$ where n^l refers to the number of neurons/channels in layer l. **ER-plus** (Liu et al., 2022a): ER-plus modifies ER by forcing the last layer as dense if it is not while keeping the overall parameter count the same. **OWL-inverse**: OWL-inverse metric is the inverse variant of OWL, whose outlier ratio is $1 - \texttt{LOD}$.

For this study, we apply Wanda to the LLaMA-7B model. The results are presented in Table 8. It is noteworthy that all approaches, except for the Global method, perform satisfactorily when the sparsity level is at or below 40%. This observation suggests that the region of low sparsity does not provide significant distinctions for performance comparison. However, as the sparsity level exceeds 50%, discrepancies between the various approaches become evident. Notably, the Uniform and OWL methods emerge as the top-performing approaches, with OWL consistently outperforming the former across all sparsity levels. On the other hand, the ER family of methods appears to be less suitable for LLM pruning. It's worth mentioning that the performance of OWL experiences a significant decline when we invert its outlier ratio, underscoring the effectiveness of LOD in identifying critical layers.

Vision Models. Moreover, we also evaluated OWL in vision models with vision models in Appendix A. The performance improvement of OWL on vision models is not as pronounced as observed in LLMs, likely as outliers are not particularly evident in vision models.

6. Conclusion

In this paper, we focus on a crucial aspect of LLM pruning that has been overlooked by previous works – layerwise sparsity ratios. Drawing inspiration from the emergence of outliers, characterized by features exhibiting significantly greater magnitudes compared to others, we introduced a novel layerwise sparsity ratio, Outlier Weighed Layerwise sparsity (OWL). OWL aligns the sparsity ratio with the outlier ratio of each layer to preserve outliers. Notably, our approach demonstrates substantial performance gains, surpassing the state-of-the-art Wanda and SparseGPT by 61.22 and 6.80 perplexity points at 70% sparsity, respectively. This work represents the first effort to make it applicable in LLM pruning, opening up new avenues for the development of specialized sparse algorithms that can further optimize the deployment of LLMs in practical applications.

Acknowledgements

S. Liu is supported by the Royal Society with the Newton International Fellowship. Part of this work used the Dutch national e-infrastructure with the support of the SURF Cooperative using grant no. NWO-2023.027/L1 and EINF-2943/L1. Z. Wang is supported by the NSF AI Institute for Foundations of Machine Learning (IFML) and a Google Research Scholar Award. We extend our gratitude to Eldar Kurtic for his invaluable effort in helping to measure the real speedup of the DeepSparse engine.

Impact Statement

This paper focuses on pruning large language models (LLMs). Through the implementation of effective layerwise sparsity, we can achieve a reduction of up to 70% in the parameters of SOTA LLMs while retaining their essential functionality. Consequently, this advancement facilitates the deployment of LLMs on resource-constrained devices, expedites inference processes, and contributes to the sustainability of AI technologies. The significant real speed-up demonstrated in our study underscores the considerable advantages of sparsity, extending beyond GPU to other commodity hardware like CPU and FPGA. This encourages researchers to broaden their focus beyond GPU-centric perspectives, fostering a deeper exploration of the benefits of sparsity across diverse hardware platforms. Moreover, our paper aligns with the imperative of developing energy-efficient and accessible artificial intelligence systems, catering to a wide array of applications and users.

References

Bhojanapalli, S., Chakrabarti, A., Veit, A., Lukasik, M., Jain, H., Liu, F., Chang, Y.-W., and Kumar, S. Leveraging redundancy in attention with reuse transformers. *arXiv preprint arXiv:2110.06821*, 2021.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems (NeurIPs)*, 33:1877–1901, 2020.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. BoolQ: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved

question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

DeepSparse. NeuralMagic DeepSparse Inference Engine, 2021. URL https://github.com/neuralmagic/deepsparse.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems (NeurIPs)*, 2022.

Erdős, P. and Rényi, A. On random graphs i. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.

Evci, U., Gale, T., Menick, J., Castro, P. S., and Elsen, E. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning (ICML)*, pp. 2943–2952, 2020.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Frantar, E. and Alistarh, D. Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning (ICML)*, 2023.

Gale, T., Elsen, E., and Hooker, S. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

Gale, T., Zaharia, M., Young, C., and Elsen, E. Sparse gpu kernels for deep learning. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–14. IEEE, 2020.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1135–1143, 2015.

Hassibi, B., Stork, D. G., and Wolff, G. J. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pp. 293–299. IEEE, 1993.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Jaiswal, A., Gan, Z., Du, X., Zhang, B., Wang, Z., and Yang, Y. Compressing llms: The truth is rarely pure and never simple. *arXiv preprint arXiv:2310.01382*, 2023a.

Jaiswal, A., Liu, S., Chen, T., and Wang, Z. The emergence of essential sparsity in large pre-trained models: The weights that matter. *arXiv preprint arXiv:2306.03805*, 2023b.

Janowsky, S. A. Pruning versus clipping in neural networks. *Physical Review A*, 39(12):6600, 1989.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Kovaleva, O., Kulshreshtha, S., Rogers, A., and Rumshisky, A. Bert busters: Outlier dimensions that disrupt transformers. *arXiv preprint arXiv:2105.06990*, 2021.

Kurtic, E., Campos, D., Nguyen, T., Frantar, E., Kurtz, M., Fineran, B., Goin, M., and Alistarh, D. The optimal bert surgeon: Scalable and accurate second-order pruning for large language models. *arXiv preprint arXiv:2203.07259*, 2022.

LeCun, Y., Denker, J., and Solla, S. Optimal brain damage. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 598–605, 1989.

Lee, J., Park, S., Mo, S., Ahn, S., and Shin, J. Layer-adaptive sparsity for the magnitude-based pruning. *arXiv preprint arXiv:2010.07611*, 2020.

Lee, N., Ajanthan, T., and Torr, P. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations (ICLR)*, 2019.

Lin, J., Tang, J., Tang, H., Yang, S., Dang, X., and Han, S. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

Lin, S., Ji, R., Yan, C., Zhang, B., Cao, L., Ye, Q., Huang, F., and Doermann, D. Towards optimal structured cnn pruning via generative adversarial learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2790–2799, 2019.

Liu, S. and Wang, Z. Ten lessons we have learned in the new" sparseland": A short handbook for sparse neural network researchers. *arXiv preprint arXiv:2302.02596*, 2023.

Liu, S., Chen, T., Chen, X., Atashgahi, Z., Yin, L., Kou, H., Shen, L., Pechenizkiy, M., Wang, Z., and Mocanu, D. C. Sparse training via boosting pruning plasticity with

neuroregeneration. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021a.

Liu, S., Yin, L., Mocanu, D. C., and Pechenizkiy, M. Do we actually need dense over-parameterization? in-time over-parameterization in sparse training. In *International Conference on Machine Learning*, pp. 6989–7000. PMLR, 2021b.

Liu, S., Chen, T., Chen, X., Shen, L., Mocanu, D. C., Wang, Z., and Pechenizkiy, M. The unreasonable effectiveness of random pruning: Return of the most naive baseline for sparse training. *arXiv preprint arXiv:2202.02643*, 2022a.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11976–11986, 2022b.

Luccioni, A. S., Viguier, S., and Ligozat, A.-L. Estimating the carbon footprint of bloom, a 176b parameter language model. *arXiv preprint arXiv:2211.02001*, 2022.

Ma, X., Fang, G., and Wang, X. Llm-pruner: On the structural pruning of large language models. *arXiv preprint arXiv:2305.11627*, 2023.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016a.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016b.

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Mittal, D., Bhardwaj, S., Khapra, M. M., and Ravindran, B. Studying the plasticity in deep convolutional neural networks using random pruning. *Machine Vision and Applications*, 30(2):203–216, 2019.

Mocanu, D. C., Mocanu, E., Nguyen, P. H., Gibescu, M., and Liotta, A. A topological insight into restricted boltzmann machines. *Machine Learning*, 104(2):243–270, Sep 2016. ISSN 1573-0565. doi: 10.1007/s10994-016-5570-z. URL https://doi.org/10.1007/s10994-016-5570-z.

Mocanu, D. C., Mocanu, E., Stone, P., Nguyen, P. H., Gibescu, M., and Liotta, A. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 9: 1–12, 2018.

Mozer, M. C. and Smolensky, P. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 107–115, 1989.

Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L.-M., Rothchild, D., So, D., Texier, M., and Dean, J. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

Puccetti, G., Rogers, A., Drozd, A., and Dell'Orletta, F. Outliers dimensions that disrupt transformers are driven by frequency. *arXiv preprint arXiv:2205.11380*, 2022.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019.

Sanh, V., Wolf, T., and Rush, A. M. Movement pruning: Adaptive sparsity by fine-tuning. *arXiv preprint arXiv:2005.07683*, 2020.

Schaeffer, R., Miranda, B., and Koyejo, S. Are emergent abilities of large language models a mirage? *arXiv preprint arXiv:2304.15004*, 2023.

Sun, M., Liu, Z., Bair, A., and Kolter, J. Z. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

Sun, W., Zhou, A., Stuijk, S., Wijnhoven, R., Nelson, A. O., Corporaal, H., et al. Dominosearch: Find layer-wise fine-grained n: M sparse schemes from dense neural networks. *Advances in neural information processing systems*, 34: 20721–20732, 2021.

Tang, C., Ouyang, K., Wang, Z., Zhu, Y., Ji, W., Wang, Y., and Zhu, W. Mixed-precision neural network quantization via learned layer-wise importance. In *European Conference on Computer Vision*, pp. 259–275. Springer, 2022.

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

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pp. 10347–10357. PMLR, 2021.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Wang, C., Zhang, G., and Grosse, R. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations (ICLR)*, 2020.

Wang, W. and Tu, Z. Rethinking the value of transformer components. *arXiv preprint arXiv:2011.03803*, 2020.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022.

Wen, W., He, Y., Rajbhandari, S., Zhang, M., Wang, W., Liu, F., Hu, B., Chen, Y., and Li, H. Learning intrinsic sparse structures within long short-term memory. *arXiv preprint arXiv:1709.05027*, 2017.

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., and Han, S. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning (ICML)*, pp. 38087–38099. PMLR, 2023.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Zhang, Y., Zhao, L., Lin, M., Sun, Y., Yao, Y., Han, X., Tanner, J., Liu, S., and Ji, R. Dynamic sparse no training: Training-free fine-tuning for sparse llms. *arXiv preprint arXiv:2310.08915*, 2023.

Zhu, M. and Gupta, S. To prune, or not to prune: exploring the efficacy of pruning for model compression. In *International Conference on Learning Representations Workshop (ICLRW)*, 2017.

Zimmer, M., Andoni, M., Spiegel, C., and Pokutta, S. Perp: Rethinking the prune-retrain paradigm in the era of llms. *arXiv preprint arXiv:2312.15230*, 2023.

A. Vision Model Pruning

In this appendix, we study if the promise of OWL also holds for vision models. We apply OWL to two commonly used modern vision models, i.e., ConvNeXt-Base (Liu et al., 2022b) and DeiT-Base (Touvron et al., 2021) and evaluate them on the ImageNet-1K dataset (Deng et al., 2009). We adopt Wanda as the pruning approach and compare OWL with uniform layerwise sparsity. All models are pruned in one shot without fine-tuning.

Table 11: Top-1 accuracy of sparse vision models on ImageNet-1K.

Method	Model	Sparsity			
		50%	60%	70%	80%
Wanda	ConvNeXt-Base	82.72	**80.55**	68.18	**6.44**
OWL *w.* Wanda	ConvNeXt-Base	**82.76**	80.53	**68.28**	6.32
Wanda	DeiT-Base	78.23	71.14	49.20	6.86
OWL *w.* Wanda	DeiT-Base	**78.40**	**71.76**	**54.24**	**7.98**

Our findings in Table 11 reveal that OWL enhances the accuracy of sparse DeiT models in contrast to Wanda. However, for ConvNeXt models, it seems that OWL does not necessarily bring benefits to ConvNeXt (neither increases nor degrades accuracy). Overall, it seems that the performance improvement of OWL on vision models is not as pronounced as observed LLMs. Our hypothesis here is that the phenomenon of outliers is not particularly evident in vision models. According to Puccetti et al. (2022), outliers in LLMs are causally related to high-frequency tokens in pre-training data. These high-frequency tokens are more prevalent in textual datasets but are relatively scarce and challenging to identify within vision datasets. Hence, the phenomenon of outliers, crucial in OWL's effectiveness, may not be as evidently present or impactful within the domain of vision models, contributing to the differing performance improvements between LLMs and vision models.

B. More Practical Applications of OWL

OWL serves as a general approach to identify layer importance for LLMs, which exhibits substantial potential in many hardware-friendly scenarios. To examine this, we explore OWL in three hardware-friendly regimes, including N:M sparsity, structured pruning, and mixed-precision quantization. The preliminary results are shown below.

B.1. N:M Sparsity

Following DominoSearch (Sun et al., 2021), we choose a mixed N:8 sparsity configuration. Instead of employing a uniform N value across all layers, we allow individual layers to have distinct N values while maintaining the overall parameter count constant. We use OWL to determine the optimal value of N for individual layers. The results are presented in Table 12. It is evident that OWL consistently enhances performance compared to uniform N:M sparsity. Remarkably, in high sparsity scenarios like 3:8 and 2:8 sparsity, OWL demonstrates significant improvements with 2× and 8× perplexity reductions over the uniform baseline, respectively.

Table 12: Perplexity of mixed N:M sparsity (N refers to non-zero weights) with LLaMA-7B on WikiText.

Method	N:M Sparsity Structure	Perplexity
Wanda	4:8	8.57
OWL *w.* Wanda	Mixed 4:8	**8.55**
Wanda	3:8	42.56
OWL *w.* Wanda	Mixed 3:8	**21.49**
Wanda	2:8	2962.00
OWL *w.* Wanda	Mixed 2:8	**331.37**

B.2. Structured Pruning

Instead of pruning individual weights, structured pruning involves the selective removal of an entire group of weights, which are more amenable to hardware speedup, including weight blocks, neurons, filters/channels, and attention heads (Liu & Wang, 2023). We adhere to the recent methodology introduced in LLM Pruner (Ma et al., 2023), wherein entire neurons and attention heads are removed. This action facilitates the direct acceleration of pruned LLMs on GPUs or TPUs. We replace the uniform layerwise sparsity used by LLM pruner with the non-uniform layerwise sparsity discovered by OWL. Table 13 again demonstrates that OWL achieves preferable performance compared to the uniform layerwise sparsity in the context of structured pruning.

Table 13: Perplexity of Structure Pruning with LLaMA-7B on WikiText and PTB.

Dataset	Pruning Method	Layerwise Sparsity	20%	40%	60%	80%
WikiText	LLM Pruner	Uniform	19.09	30.39	90.02	1228.17
WikiText	LLM Pruner	OWL	**18.57**	**28.65**	**76.99**	**321.64**
PTB	LLM Pruner	Uniform	29.51	66.90	192.06	1691.87
PTB	LLM Pruner	OWL	**28.82**	**53.22**	**150.16**	**502.07**

B.3. Mixed-Precision Quantization

Leveraging our non-uniform layerwise sparsity, we can enhance mixed-precision quantization by assigning higher precision to layers exhibiting more outliers. Following the approach outlined in (Tang et al., 2022), we utilize OWL to assign different bit precision to different layers, thereby facilitating a mixed-precision quantization strategy. Our baseline here involves selecting layers randomly and based on the L_1 norm of weights. It is evident that OWL also serves as a valuable indicator for selecting important layers in mixed-precision quantization, leading to improved quantization performance as shown in Table 14.

Table 14: Perplexity of mixed-precision quantization with LLaMA-7B on WikiText.

Method	Precision	Perplexity
Same Bit-width	2 Bit	104151.84
Same Bit-width	3 Bit	25.82
Same Bit-width	4 Bit	6.29
Select with random	Mixed 3/4 Bit	12.04
Select with L_1 norm	Mixed 3/4 Bit	14.61
Select with OWL	Mixed 3/4 Bit	**9.09**
Select with random	Mixed 2/3/4 Bit	11455.54
Select with L_1 norm	Mixed 2/3/4 Bit	13959.422
Select with OWL	Mixed 2/3/4 Bit	**190.28**
Select with random	Mixed 2/4 Bit	14817.12
Select with L_1 norm	Mixed 2/4 Bit	33670.214
Select with OWL	Mixed 2/4 Bit	**7505.60**

C. Per-Block Vs. Per-Layer

As we mentioned before, we assign a distinct pruning ratio for each Transformer block instead of each layer. To provide more insights about this option, we provide the performance comparison between these two options and report their layerwise sparsity respectively. We report the sparsity ratio of 7 FC layers including `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `down_proj`, and `up_proj` of layers 1, 2, 15, 30, and 31. We found that applying OWL in a per-layer manner leads to sub-optimal performance (Perplexity: 86.285 vs 24.55). We can see that applying OWL in a per-layer manner will lead to nearly uniform sparsity of certain layers across Transformer blocks, such as `v_proj`, `gate_proj`, and `up_proj`, which might be undesirable for LLM pruning.

Table 15: Layerwise sparsity of LLaMA-7B pruned with per-layer OWL (Sparsity: 70%, Perplexity:86.285).

Layer	q_proj	k_proj	v_proj	o_proj	gate_proj	down_proj	up_proj
1	0.613	0.613	0.613	0.613	0.613	0.613	0.613
2	0.641	0.641	0.641	0.641	0.641	0.641	0.641
5	0.608	0.608	0.608	0.608	0.608	0.608	0.608
30	0.760	0.760	0.760	0.760	0.760	0.760	0.760
31	0.721	0.721	0.721	0.721	0.721	0.721	0.721

Table 16: Layerwise sparsity of LLaMA-7B pruned with per-block OWL (Sparsity: 70%, Perplexity:24.55).

Layer	q_proj	k_proj	v_proj	o_proj	gate_proj	down_proj	up_proj
1	0.639	0.638	0.691	0.598	0.710	0.696	0.713
2	0.680	0.677	0.707	0.679	0.711	0.703	0.713
15	0.698	0.693	0.710	0.706	0.709	0.703	0.712
30	0.705	0.704	0.713	0.663	0.710	0.657	0.712
31	0.702	0.701	0.712	0.670	0.710	0.621	0.711

D. Hyperparameters

In this section, we share the hyperparameters used to reproduce the results in our experiments in Table 17.

Table 17: Hyperparameters used to obtain the results in this paper.

Model	M	λ
LLaMA-7B	5	8%
LLaMA-13B	7	8%
LLaMA-30B	5	8%
LLaMA-65B	5	20%
OPT-6.7B	10	8%