Beyond Size: How Gradients Shape Pruning Decisions in Large Language Models

Anonymous authors
Paper under double-blind review

Abstract

Large Language Models (LLMs) with billions of parameters are prime targets for network pruning, which removes some model weights without hurting performance. Prior approaches such as magnitude pruning, SparseGPT, and Wanda either concentrated solely on weights or integrated weights with activations for sparsity. However, they overlooked the informative gradients derived from pretrained LLMs. In this paper, we present a novel sparsity-centric pruning method for pretrained LLMs, termed **G**radient-**b**ased **L**anguage **M**odel **P**runer (`GBLM-Pruner`). `GBLM-Pruner` leverages the first-order term of the Taylor expansion, operating in a training-free manner by harnessing properly normalized gradients from a few calibration samples to determine the pruning metric, and substantially outperforms competitive counterparts like SparseGPT and Wanda in multiple benchmarks. Intriguingly, by incorporating gradients, unstructured pruning with our method tends to reveal some structural patterns, which mirrors the geometric interdependence inherent in the LLMs' parameter structure. Additionally, `GBLM-Pruner` functions without any subsequent retraining or weight updates to maintain its simplicity as other counterparts. Extensive evaluations on LLaMA-series, Qwen-Series and Deepseek models across zero-shot, STEM-Reasoning and Coding benchmarks show that `GBLM-Pruner` surpasses magnitude pruning and Wanda by significant margins and performs competitively against SparseGPT which involves intensive weight updates. We further extend our approach on Vision Transformer and Vision Language Models. Our code is available in the supplementary material.

1 Introduction

Large Language Models (LLMs) like OpenAI's GPT series (Radford et al., 2018; 2019; Brown et al., 2020; Achiam et al., 2023), BERT (Devlin et al., 2018), LLaMA (Touvron et al., 2023a;b) and others have made significant strides in recent years, leading to a paradigm shift in natural language processing (Achiam et al., 2023; Anil et al., 2023; Touvron et al., 2023b) and multimodal learning (Alayrac et al., 2022; Li et al., 2023). Many industries have integrated LLMs into their workflows, such as in chatbots (Achiam et al., 2023), code completion tools (e.g., GitHub Copilot) (Chen et al., 2021), and assistive technologies (Zdravkova et al., 2022). While enjoying impressive generalization capabilities, LLMs come with a set of practical challenges and disadvantages. The presence of abundant parameters, large memory consumption, and high computational cost during inference present several concerns in real-world applications. Previous literature proposed multiple solutions to address these disadvantages, such as distillation (Hinton et al., 2015), quantization (Jacob et al., 2018), pruning (Han et al., 2016), and hardware acceleration (Chen et al., 2020).

Among them, pruning refers to the removal of certain weights or entire neurons/layers based on some criteria, e.g., the smallest weights. A pruned model can maintain similar performance with fewer parameters, resulting in a reduction in storage and computational requirements. Inducing unstructural sparsity in pruning is a widely embraced method aimed at minimizing the memory requirements of neural networks with only a minimal sacrifice in accuracy. Pruning methods stand out as notably simple and efficient mechanisms for model compression, serving to eliminate weights contingent on their significance. Reduced models can be conveniently dispatched to edge devices, and also exhibit substantially lower energy consumption, a sizable

portion of energy is expended in transferring model parameters from a device's long-term storage to its memory (Dao et al., 2022).

However, given the constraints of training-free conditions, existing solutions for pruning LLMs primarily employ either weight magnitude (Han et al., 2015; 2016) or a combination of magnitude and activation (Frantar and Alistarh, 2023; Sun et al., 2023). While these methods are substantiated with empirical ablations and experiments, they are, to a degree, either too complex to use like SparseGPT by computing matrix inverses and updating weights, or heuristic and lack profound theoretical justification like Wanda, especially concerning the application to the recently developed, highly advanced large language models.

In this study, we tackle the aforementioned complexity and interpretability challenges in LLM pruning methods by presenting a simple yet effective approach named `GBLM-Pruner` (Gradient-Based Language Model Pruner) that can be theoretically explained using the adapted Optimal Brain Surgeon framework (OBS) (Hassibi et al., 1993). This method proficiently prunes LLMs to significant levels of sparsity, eliminating the need to update the residual weights. Specifically, we employ normalization of gradients across calibration samples to formulate an importance indicator matrix $\mathbf{A} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, where $\mathbf{A}[i,j] = \|\mathbf{G}[:,i,j]\|_p$ represents the aggregated gradient magnitude at each weight position. Here, $\mathbf{G}[k,i,j] = \frac{\partial E}{\partial \mathbf{W}[i,j]}\Big|_{\text{sample } k}$ is the gradient for calibration sample k, and $\|\cdot\|_p$ denotes the ℓ_p norm across samples. This gradient-based indicator quantifies weight importance and can either replace or supplement activation-based metrics. This matrix can serve as activations and can either replace or supplement them. This method maintains simplicity over SparseGPT (Frantar and Alistarh, 2023) while showcasing enhanced robustness and improved interpretation than Wanda (Sun et al., 2023) on large language models. Furthermore, it is notable that although we employ gradients in our approach, there is no necessity for retraining or any updates to parameters.

We conducted extensive evaluations of `GBLM-Pruner` on LLaMA-1 and 2 (Touvron et al., 2023a;b), among the most influential families of open-source LLMs. Results across various language benchmarks highlight that `GBLM-Pruner` is proficient in identifying effective sparse networks directly from pretrained LLMs. Notably, `GBLM-Pruner` substantially surpasses magnitude pruning and recent state-of-the-art methods. Our contributions in this work form a foundational basis for ensuing advancements in this domain. Furthermore, we advocate for continued exploration of sparsity within LLMs through the underexplored *gradients* angle. While prior gradient-based methods focus on structured pruning or require task-specific finetuning, we introduce the first training-free gradient-based solution for fine-grained unstructured LLM pruning that maintains zero-shot generalization capabilities. Last, we demonstrate the effectiveness of our approach on Vision Transformers (Dosovitskiy et al., 2020) (see Appendix A).

Overall, our contributions can be summarized as follows:

- We present `GBLM-Pruner`, the first training-free gradient-based pruning method for LLMs that maintains task-agnostic zero-shot generalization across models up to 70B parameters.
- We provide theoretical motivation by extending the OBS framework to retain first-order gradient information, validated through extensive ablations.
- We demonstrate consistent improvements over magnitude pruning and Wanda (up to 11.19% relative gain on coding tasks) while being 636∼754× faster than weight update based methods like SparseGPT, with comprehensive evaluation across 15+ model families and 20+ benchmarks.

2 Approach

2.1 Notation

We establish the following notation used throughout this paper. Matrices are denoted by bold uppercase letters (e.g., $\mathbf{W}$, $\mathbf{X}$), vectors by bold lowercase letters (e.g., $\mathbf{w}$, $\mathbf{g}$), and scalars by regular lowercase letters (e.g., w_m, α). For a matrix $\mathbf{A}$, we use $\mathbf{A}[i,j]$ to denote the element at row i and column j, and $\mathbf{A}[:,j]$ to denote the j-th column vector.

Symbol	Dimension	Description
$\mathbf{W}$	$\mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$	Weight matrix of a neural network layer
$\mathbf{X}$	$\mathbb{R}^{N \times L \times d_{\text{in}}}$	Input activation matrix across samples
$\mathbf{G}$	$\mathbb{R}^{N \times d_{\text{out}} \times d_{\text{in}}}$	Gradient matrix, $\mathbf{G}[k,i,j] = \frac{\partial E}{\partial \mathbf{W}[i,j]}\Big\vert_k$
$\mathbf{S}$	$\mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$	Importance score matrix (pruning metric)
$\mathbf{M}$	$\{0,1\}^{d_{\text{out}} \times d_{\text{in}}}$	Binary pruning mask
$\mathbf{H}$	$\mathbb{R}^{d \times d}$	Hessian matrix, $\mathbf{H} = \frac{\partial^2 E}{\partial \mathbf{w}^2}$
$\mathbf{w}$	$\mathbb{R}^{d}$	Vectorized network weights
$\mathbf{g}$	$\mathbb{R}^{d}$	Gradient vector, $\mathbf{g} = \frac{\partial E}{\partial \mathbf{w}}$
$\delta\mathbf{w}$	$\mathbb{R}^{d}$	Change in weight vector
$\mathbf{e}_m$	$\mathbb{R}^{d}$	Unit vector (1 at position m, 0 elsewhere)
N	$\mathbb{N}$	Number of calibration samples
L	$\mathbb{N}$	Sequence length
s	$(0,1)$	Sparsity ratio
α	$\mathbb{R}^{+}$	Scaling factor for gradient contribution
w_m	$\mathbb{R}$	Scalar weight element at position m
λ	$\mathbb{R}$	Lagrange multiplier
$E(\cdot)$	$\mathbb{R}$	Global loss function
$\lvert\cdot\rvert$	–	Absolute value (element-wise for matrices)
$\lVert\cdot\rVert_p$	–	ℓ_p norm (aggregation across samples)
$\lVert\cdot\rVert_2$	–	ℓ_2 (Euclidean) norm
$(\cdot)^\top$	–	Matrix or vector transpose

2.2 Preliminaries

Weights Magnitude. Magnitude pruning, which retains weights of significant absolute values, is the predominant approach for weight pruning. It usually induces unstructured sparsity and has been employed across various architectures spanning computer vision (Han et al., 2015; 2016) and language processing (Gale et al., 2019). Furthermore, it has recently become integral to the lottery ticket hypothesis (Frankle and Carbin, 2018).

Weights and Activations. SparseGPT (Frantar and Alistarh, 2023) conceptualizes the problem of pruning large language models by addressing a local, layer-wise reconstruction problem. The approach for determining pruning metrics and the process for updating weights in SparseGPT draws inspiration from the Optimal Brain Surgeon (OBS) (Hassibi et al., 1993) approach. The pruning metric employed within SparseGPT is defined as follows:

$$\mathbf{S}[i,j] = \frac{|\mathbf{W}[i,j]|^2}{\text{diag}\left(\mathbf{H}^{-1}\right)[j,j]} \tag{1}$$

where $\mathbf{H} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)$ is the Hessian matrix, and $\mathbf{H}^{-1}$ is the inverse Hessian. $\mathbf{S}$ is the importance score for a given weight $\mathbf{W}$, and $[i,j]$ is the element at index i,j of the matrix.

Wanda (Sun et al., 2023) suggests assessing the significance of each individual weight by calculating the product of its magnitude and the norm of the corresponding input feature. More precisely, the score $\mathbf{S}$ for a given weight, $\mathbf{W}[i,j]$, is determined as follows:

$$\mathbf{S}[i,j] = |\mathbf{W}[i,j]| \cdot \|\mathbf{X}[:,j]\|_2 \tag{2}$$

where the element-wise product between the weight magnitude and the norm of input activations is performed within each row in $\mathbf{W}$.

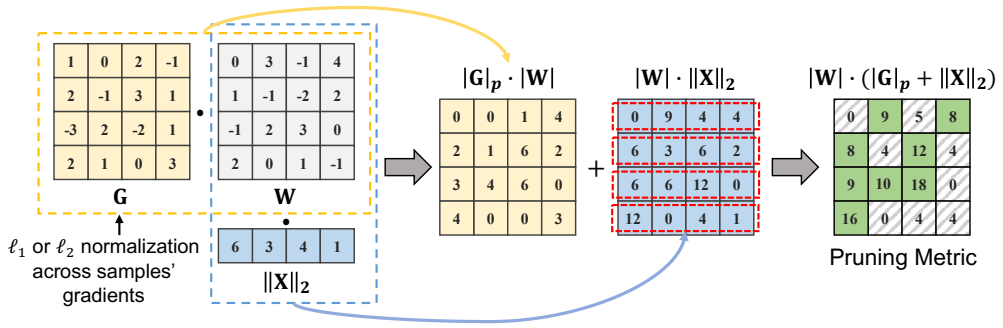

Figure 1: Illustration of our method `GBLM-Pruner`. Given a weight matrix, $\mathbf{W}$, a gradient matrix, $\mathbf{G}$, and an input feature activation, $\mathbf{X}$, weight importance is computed as an element-wise multiplication of weight magnitude and ℓ_1 or ℓ_2 norm of the gradients across multiple samples, denoted as $\|\mathbf{G}\|_p \cdot |\mathbf{W}|$, optionally, it is promotable to add the multiplication of weight and the ℓ_2 norm of input activations, denoted as $|\mathbf{W}| \cdot \|\mathbf{X}\|_2$.

Algorithm 1 The `GBLM-Pruner` algorithm

$\mathbf{W} \leftarrow$ weight matrix $\in (\boldsymbol{d}_{out}, \boldsymbol{d}_{in})$
$\mathbf{X} \leftarrow$ activation matrix $\in (\boldsymbol{N} \times \boldsymbol{L}, \boldsymbol{d}_{in})$
$\mathbf{G} \leftarrow$ gradient matrix $\in (\boldsymbol{N}, \boldsymbol{d}_{out}, \boldsymbol{d}_{in})$
$\alpha \leftarrow$ scaling factor for gradient term $\in \mathbb{R}^+$
$\boldsymbol{s} \leftarrow$ sparsity ratio $\in (0, 1)$
$\mathbf{S} \leftarrow$ pruning metric $\in (\boldsymbol{d}_{out}, \boldsymbol{d}_{in})$
$\mathbf{M} \leftarrow$ pruning mask $\in (\boldsymbol{d}_{out}, \boldsymbol{d}_{in})$

for i $\in (1, \boldsymbol{d}_{out})$ **do**
  for j $\in (1, \boldsymbol{d}_{in})$ **do**
    $\mathbf{S}[i,j] = \alpha \cdot |\mathbf{W}[i,j]| \cdot \|\mathbf{G}[:,i,j]\|_p + |\mathbf{W}[i,j]| \cdot \|\mathbf{X}[:,j]\|_2$
  end for
end for

for i $\in (1, \boldsymbol{d}_{out})$ **do**
  $\mathbf{M}[i,:] =$ mask of $\boldsymbol{s}\%$ weights with smallest $\mathbf{S}[i,:]$
end for

$\mathbf{W}[\mathbf{M}] = 0$

2.3 Gradients Matter

Gradients. According to Optimal Brain Damage (LeCun et al., 1989) and Optimal Brain Surgeon (Hassibi et al., 1993), gradients and higher order derivatives are naturally correlated to the importance of weights for LLM pruning, which is the theoretical basis of our approach. However, they ignore the gradients in their pruning framework under the assumption that gradients of the fully trained network are small and do not provide any additional information when the higher-order terms are considered. Our work shows that gradients are still crucial and provide non-trivial information.

Previous gradient-based structured pruning methods, such as feature map pruning (Molchanov et al., 2016), channel pruning (Yang et al., 2022), and head pruning (Michel et al., 2019) utilize the first-order Taylor approximation of loss $\mathcal{L}$ around activation $z = 0$ or weight $w = 0$ as importance score, the formulation is:

$$\mathbf{S} = \mathbb{E}_{\boldsymbol{x} \sim \mathbf{X}} \left| \frac{\partial \mathcal{L}(\boldsymbol{x})}{\partial \mathbf{A}} \mathbf{A} \right| \tag{3}$$

where $\mathbf{X}$ is the sampled data distribution and A is either activation matrix Z or weight matrix W. Most of these structured pruning methods are proposed for transfer learning to a particular task and require

significant finetuning on the specific task to maintain the model performance. In contrast, our work leverages gradient information to do unstructured and N:M semi-structured pruning without any subsequent weight update. Additionally, we illustrate the integration of activations into our pruning metric through the use of a scaling factor for best performance. Furthermore, our pruned model is task-agnostic and generalizable to any downstream task as showcased by the Zero-shot evaluation on several tasks included in the Eleuther AI lm-evaluation harness benchmark (Gao et al., 2021).

Pruning Metric. Consider a layer in LLMs characterized by the weight $\mathbf{W}$, possessing a shape of $(\boldsymbol{d}_{out}, \boldsymbol{d}_{in})$. In the context of Transformer models, this layer has the gradient $\mathbf{G}$, exhibiting the same shape of weight $\mathbf{W}$. We propose evaluating the importance of each individual weight by aggregating the corresponding gradients across different samples and then computing the product of its magnitude with the weights. More precisely, the importance score attributed to a specific weight, $\mathbf{W}[i,j]$, is determined as follows:

$$\mathbf{S}[i,j] = |\mathbf{W}[i,j]| \cdot \|\mathbf{G}[:,i,j]\|_p \tag{4}$$

While competitive results can be achieved with gradients solely, we can combine feature activations to get better performance, which forms our final pruning metric:

$$\mathbf{S}[i,j] = \alpha \cdot |\mathbf{W}[i,j]| \cdot \|\mathbf{G}[:,i,j]\|_p + |\mathbf{W}[i,j]| \cdot \|\mathbf{X}[:,j]\|_2 \tag{5}$$

where α is the scaling factor used to account for the small magnitude of gradients, which makes the contribution of gradient balanced to the large magnitude of activations. The α is chosen using a held-out validation set.

Comparison Group. The comparison group of pruning is pivotal in unstructured pruning, owing to the fact that varying granularities yield disparate pruning patterns. Previously, unstructured magnitude pruning approaches have leveraged both layer-wise and global pruning. In these methods, weights are contrasted either within the same layer or throughout the entirety of the model. Through a comprehensive study, we observe that the highest accuracy is achieved when weights are analyzed on a row-wise basis. This insight is consistent with the findings in Sun et al. (2023).

We illustrate in detail our proposed pruning approach `GBLM-Pruner` in Algorithm 1.

2.4 A Theoretical Motivation

In this section, we have revisited and refined the Optimal Brain Surgeon (OBS) framework (Hassibi et al., 1993) framework by incorporating considerations of the gradient, i.e., the first-order term in Taylor approximation. The closed-form solution of the increase in error for removing a weight from the model, given by this analysis serves as the fundamental basis for our novel gradient-based pruning metric. For the sake of simplicity, we will consider weights and gradients as one-dimensional vectors denoted by $\boldsymbol{w}$ and $\boldsymbol{g}$ respectively in our analysis.

We work with the squared error loss function:

$$E = \frac{1}{2N} \sum_{k=1}^{N} (\mathbf{t}^{[k]} - \mathbf{o}^{[k]})^\top (\mathbf{t}^{[k]} - \mathbf{o}^{[k]}) \tag{6}$$

where N is the number of calibration samples, $\mathbf{t}^{[k]}$ is the target output, and $\mathbf{o}^{[k]}$ is the network output for sample k. The optimization problem for network pruning using both the first and second-order terms can be depicted in Equation 7. Here, $\boldsymbol{E}$ is the error or loss function, $\boldsymbol{w}$ is the weight vector for the neural network, and $\delta\boldsymbol{w}$ is the change in the weight vector. Additionally, $\mathbf{e}_m$ is the unit vector in weight space corresponding to the pruned weight w_m, $\mathbf{H} = \frac{\partial^2 \boldsymbol{E}}{\partial \boldsymbol{w}^2}$ denotes the Hessian Matrix, and the superscript $\top$ signifies vector transpose.

$$\min_m \left\{ \min_{\delta\boldsymbol{w}} \left(\left(\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{w}} \right)^\top \cdot \delta\boldsymbol{w} + \frac{1}{2} \delta\boldsymbol{w}^\top \cdot \mathbf{H} \cdot \delta\boldsymbol{w} \right) \middle| \mathbf{e}_m^\top \cdot \delta\boldsymbol{w} + w_m = 0 \right\} \tag{7}$$

The constraint $\mathbf{e}_m^\top \cdot \delta\mathbf{w} + w_m = 0$ enforces that after weight adjustment, the pruned weight w_m is set to exactly zero. Specifically, if the new weights are $\mathbf{w}' = \mathbf{w} + \delta\mathbf{w}$, we require $w'_m = 0$, which gives $\delta w_m = -w_m$. Since $\mathbf{e}_m$ is a unit vector with 1 at position m and 0 elsewhere, we have $\mathbf{e}_m^\top \delta\mathbf{w} = \delta w_m$, thus $\mathbf{e}_m^\top \delta\mathbf{w} + w_m = 0$.

By solving the optimization problem, we obtain the optimal change in error, $\delta\boldsymbol{E}_m$, for removing weight w_m as shown in Equation 8. We provide detailed derivation in Appendix I.

$$\begin{aligned}\delta\boldsymbol{E}_m = & \frac{w_m^2}{2\left(\mathbf{H}^{-1}\right)_{mm}} - \frac{w_m\left(\boldsymbol{g}^\top \cdot \mathbf{H}^{-1} \cdot \mathbf{e}_m\right)}{\left(\mathbf{H}^{-1}\right)_{mm}} \\ & + \frac{\left(\mathbf{e}_m^\top \cdot \mathbf{H}^{-1} \cdot \boldsymbol{g}\right)^2}{2\left(\mathbf{H}^{-1}\right)_{mm}} - \frac{1}{2}\boldsymbol{g}^\top \cdot \mathbf{H}^{-1} \cdot \boldsymbol{g}\end{aligned} \tag{8}$$

For the error, $\delta\boldsymbol{E}_m$, since the gradients are already small, we can consider the quadratic or square term of the gradient to be insignificant. Thus, ignoring the third and fourth terms, we have:

$$\delta\boldsymbol{E}_m = \frac{w_m^2}{2\left(\mathbf{H}^{-1}\right)_{mm}} - \frac{w_m\left(\boldsymbol{g}^\top \cdot \mathbf{H}^{-1} \cdot \mathbf{e}_m\right)}{\left(\mathbf{H}^{-1}\right)_{mm}} \tag{9}$$

Hessian Approximation. To compute the Hessian matrix $\mathbf{H}$ of the global loss function, we follow the approximation from the original OBS framework (Hassibi et al., 1993). Under the $(t - o) \to 0$ approximation for well-trained networks, OBS derives (their Equation 11):

$$\mathbf{H} \approx \frac{1}{N}\sum_{k=1}^{N} \mathbf{J}^{[k]\top}\mathbf{J}^{[k]} \tag{10}$$

where $\mathbf{J}^{[k]} = \frac{\partial \mathbf{F}(\mathbf{w}, \mathbf{x}^{[k]})}{\partial \mathbf{w}}$ is the Jacobian of network outputs with respect to weights for sample k.

For layer-wise analysis of a linear transformation $\mathbf{a}^{[k]} = \mathbf{W}\mathbf{x}^{[k]}$, the Jacobian with respect to weight $W[i, j]$ involves the input activation $x_j^{[k]}$. Through the chain rule, this yields $\mathbf{H} \approx \mathbf{X}^\top\mathbf{X}$ where $\mathbf{X}$ contains the input activation features across samples. We provide the complete derivation in Appendix L.

We make two simplifying assumptions to enable tractable computation:

Assumption A1 (Diagonal across neurons): Following Optimal Brain Damage (LeCun et al., 1989), we assume interactions between different output neurons in the same layer are negligible, giving a block-diagonal Hessian structure.

Assumption A2 (Uniform curvature): We approximate the second-order derivatives $\frac{\partial^2 E}{\partial a_i^2}$ as approximately constant across neurons within a layer. While this is a strong assumption that may not hold exactly, it enables computational efficiency. The same approximations are implicitly made in prior accepted work such as Wanda (Sun et al., 2023), whose metric omits curvature terms entirely.

Under these assumptions, we restrict our focus to the diagonal elements:

$$\mathbf{H} \approx 2 \cdot \operatorname{diag}\left(\left\{\|\boldsymbol{x}_j\|_2^2, 1 \le j \le n\right\}\right) \tag{11}$$

where $\boldsymbol{x}_j$ is the activation vector corresponding to component j across samples, and n is the total number of components within the activation tensor for the layer.

The first term of Equation 9 then becomes:

$$\frac{w_m^2}{2\left(\mathbf{H}^{-1}\right)_{mm}} = w_m^2 \|\boldsymbol{x}_m\|_2^2 \tag{12}$$

Table 1: Zero-Shot evaluation on 50% unstructured sparsity pruned models for different benchmarks.

Models	Method	BoolQ	RTE	HellaSwag	WinoGrande	ARC-e	OBQA	Mean
LLaMA-1-7B	Dense	75.11	66.43	76.21	69.85	72.81	44.40	67.47
	Magnitude	54.65	54.15	60.90	59.43	54.38	35.20	53.12
	SparseGPT	72.87	53.07	69.77	**67.88**	**66.46**	40.60	61.77
	Wanda	71.25	54.87	70.12	66.06	65.11	39.60	61.17
	Ours	**73.43**	**59.93**	**70.29**	67.40	65.99	**41.40**	**63.07**
LLaMA-1-13B	Dense	77.98	70.40	79.07	72.77	74.75	44.80	69.96
	Magnitude	54.95	50.90	59.69	63.54	54.25	39.80	53.86
	SparseGPT	**76.67**	63.18	74.09	71.59	68.48	**43.60**	66.27
	Wanda	76.02	63.18	74.80	**71.90**	69.82	43.00	66.45
	Ours	76.61	**63.18**	**74.90**	71.67	**70.37**	43.20	**66.65**
LLaMA-1-30B	Dense	82.72	66.79	82.62	75.77	78.91	48.20	72.50
	Magnitude	64.25	49.82	67.29	66.61	70.71	41.20	59.98
	SparseGPT	**82.91**	55.96	79.31	**74.27**	77.53	46.00	69.33
	Wanda	81.71	65.34	79.91	73.56	**78.11**	**46.40**	70.84
	Ours	82.69	**67.15**	**80.23**	73.95	76.98	46.00	**71.17**
LLaMA-2-13B	Dense	80.55	65.34	79.38	72.22	77.44	45.20	70.02
	Magnitude	57.65	55.96	73.02	65.35	67.17	40.80	59.99
	SparseGPT	**81.25**	**62.82**	75.34	70.48	71.34	44.00	67.54
	Wanda	81.07	60.65	**76.08**	71.67	71.63	44.60	67.62
	Ours	80.89	60.65	76.03	**71.82**	**72.26**	**44.80**	**67.74**
Qwen3-14B	Dense	89.33	77.62	78.86	73.09	82.83	46.20	74.65
	SparseGPT	87.86	79.06	**75.24**	71.90	74.75	**45.60**	72.40
	Wanda	**88.04**	**80.87**	73.00	72.45	**79.80**	43.00	72.86
	Ours	87.95	80.51	74.86	**72.61**	79.25	43.80	**73.16**

Similarly, considering only diagonal elements and using the diagonal Hessian approximation, the second term in Equation 9 simplifies to:

$$-\frac{w_m\left(\boldsymbol{g}^\top \cdot \mathbf{H}^{-1} \cdot \mathbf{e}_m\right)}{\left(\mathbf{H}^{-1}\right)_{mm}} = -\frac{w_m g_m \left(\mathbf{H}^{-1}\right)_{mm}}{\left(\mathbf{H}^{-1}\right)_{mm}} = w_m(-g_m) \tag{13}$$

Thus, the final solution for the optimization problem in Equation 7 can be expressed as:

$$\delta \boldsymbol{E}_m = \left(w_m \left\|\boldsymbol{x}_m\right\|_2\right)^2 + w_m\left(-g_m\right) \tag{14}$$

Equation 14 provides theoretical motivation for combining weight magnitude, activation norms, and gradient information. However, through extensive empirical investigation in Section 3.7, we find that the following practical formulation yields superior performance:

$$\mathbf{S}[i,j] = |\mathbf{W}[i,j]| \cdot \|\mathbf{X}[:,j]\|_2 + \alpha \cdot |\mathbf{W}[i,j]| \cdot \|\mathbf{G}[:,i,j]\|_p \tag{15}$$

where $\mathbf{S}$ is the importance score matrix, α is an empirically determined scaling factor (we use $\alpha = 100$ based on validation experiments in Table 13), and $\|\mathbf{G}[:,i,j]\|_p$ represents the ℓ_p norm of gradients across samples.

The key modifications from the theoretical form are: (1) removing the square from the activation term, (2) introducing absolute values for stability, and (3) using ℓ_p norm aggregation of gradients rather than individual gradient values. These adjustments improve empirical performance while preserving the core theoretical insight: weight importance should balance gradient magnitude with parameter sensitivity to input activations.

3 Experiments

3.1 Implementation and Setup Details

We conduct all our experiments using PyTorch (Paszke et al., 2017) for `GBLM-Pruner`. Experiments are performed with five models from the LLaMA-1 series (7B, 13B, 30B) (Touvron et al., 2023a), the LLaMA-2-13B (Touvron et al., 2023b) and the Qwen3-14B (Yang et al., 2025a). The Huggingface transformer library

is used (Wolf et al., 2019) for handling models. The experiments are conducted on NVIDIA A100 GPUs with 40/80GB of memory. `GBLM-Pruner` requires calibration data for the computation of gradients and activations. Following previous works (Frantar et al., 2022; Frantar and Alistarh, 2023; Sun et al., 2023), we use 128 sequences with 2048 tokens randomly sampled from the first shard of the C4 (Raffel et al., 2019) training data as our calibration data. The gradients are computed with language modeling on the input sequence as the objective function. This represents the pretraining objective of the language models and remains agnostic to the downstream tasks for which the language models are used. For scaling factor α, we use 100 after ablation using a validation set.

Evaluation. We assess the performance of the pruned models using two distinct metrics: **(1)** Perplexity and **(2)** Zero-shot Evaluation on the Harness Benchmark (Gao et al., 2021). Perplexity is a well-established metric (Dettmers and Zettlemoyer, 2022; Yao et al., 2022; Frantar and Alistarh, 2022; Sun et al., 2023; Frantar and Alistarh, 2023) and provides stable and reliable results. The Zero-shot Harness evaluation, although known to be relatively noisy, offers a more readily interpretable assessment of model performance.

Table 2: Comparison group for `GBLM-Pruner`.

Comparison Group	Perplexity
`layer`	<u>7.45</u>
`input,1`	10.16
`input,128`	7.64
`output,1`	**6.86**
`output,128`	7.47

Sparsity and Comparison Group. Following recent methods (Frantar and Alistarh, 2023; Sanh et al., 2020), `GBLM-Pruner` prunes the linear layers of LLMs uniformly except for the embedding layer and the final classification head. In addition to unstructured pruning, we also position `GBLM-Pruner` in comparison to other baselines, exploring more rigorous yet hardware-accommodating 2:4 and 4:8 semi-structured sparsity patterns. We experiment with five different pruning configurations, as shown in Table 2. Our findings indicate that the (`output,1`) configuration yields the most favorable results, prompting its adoption as the standard for all our experiments.

3.2 Perplexity Evaluation

For all the comparing methods, we report the perplexity evaluated on WikiText (Merity et al., 2016) validation data for both unstructured and semi-structured N:M sparsity pruning in Table 3. For unstructured pruning, `GBLM-Pruner` with ℓ_1 norm outperforms both Wanda and reconstruction-based SparseGPT significantly across both LLaMA-1 and LLaMA-2 models. However, the N:M sparsity pruning is restrictive by definition, especially 2:4 sparsity, which imposes greater constraints and results in a noticeable increase in perplexity compared to unstructured pruning. As shown in Table 3, we can observe SparseGPT seems to perform better than both `GBLM-Pruner` and Wanda in the case of 2:4 sparsity pruning. Conversely, for 4:8 sparsity pruning, `GBLM-Pruner` outperforms other baselines for most of models, especially for the larger models.

3.3 Zero-Shot Tasks

In addition to our perplexity evaluations, we further assess the performance of our method across six Zero-shot common-sense tasks included in the Eleuther AI lm-evaluation-harness library (Gao et al., 2021): BoolQ (Clark et al., 2019), RTE (Wang et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2019), ARC-easy (Clark et al., 2018), and OBQA (Mihaylov et al., 2018). As noted by earlier work (Dettmers and Zettlemoyer, 2022; Frantar and Alistarh, 2023), zero-shot evaluation on these tasks is known to be noisy but aggregate performance across multiple tasks enhances interpretability.

Our comprehensive results for these tasks are presented in Table 1, where models are pruned to 50% unstructured sparsity. We evaluate our method across multiple model families including LLaMA-1 (Touvron et al., 2023a) (7B, 13B, 30B), LLaMA-2 (Touvron et al., 2023b) (13B, 70B), LLaMA-3.1 (Grattafiori et al., 2024) (8B), Qwen2.5 (Yang et al., 2025a) (7B), and Qwen3 (Yang et al., 2025a) (14B). Notably, while our proposed `GBLM-Pruner` outperforms both Wanda and SparseGPT in terms of perplexity, a consistent trend is

Table 3: WikiText perplexity of pruned LLaMA-1 and LLaMA-2 family of models.

		LLaMA-2			LLaMA-1		
Method	Sparsity	7B	13B	70B	7B	13B	30B
None	0	5.47	4.88	3.32	5.68	5.09	4.10
Magnitude	0.5	16.03	6.83	5.36	17.29	20.21	7.54
SparseGPT	0.5	7.00	6.03	4.25	7.22	6.19	5.32
Wanda	0.5	6.92	5.97	4.22	7.26	6.15	5.24
`GBLM-Pruner`$_{\ell 1}$	0.5	**6.86**	**5.88**	**4.17**	**7.15**	**6.11**	**5.18**
Magnitude	2:4	37.77	8.89	6.76	42.54	18.36	9.11
SparseGPT	2:4	**10.82**	**8.75**	5.68	**10.88**	**9.06**	7.12
Wanda	2:4	12.11	9.00	5.48	11.53	9.59	6.90
`GBLM-Pruner`$_{\ell 1}$	2:4	11.91	8.80	**5.47**	11.33	9.16	**6.87**
Magnitude	4:8	15.91	7.32	5.89	16.83	13.87	7.62
SparseGPT	4:8	**8.46**	7.01	4.91	**8.45**	7.44	6.18
Wanda	4:8	8.60	7.00	4.77	8.57	7.41	5.97
`GBLM-Pruner`$_{\ell 1}$	4:8	8.63	**6.90**	**4.72**	8.48	**7.26**	**5.89**

Table 4: Performance comparison of base and pruned language models with 50% unstructured sparsity on STEM and general purpose reasoning benchmarks.

Model	**Methods**	**MMLU** (5-s)	**GPQA-D** (5-s)	**GPQA** (5-s)	**MATH** (4-s)	**GSM8K** (5-s)	**Mean**
Llama-3.1-8B	Base	65.20	24.24	27.68	8.54	51.55	35.44
	SparseGPT	**53.21**	23.23	22.32	7.76	**16.60**	24.63
	Wanda	49.34	24.18	23.66	7.92	13.57	23.73
	Ours	50.16	**25.25**	**25.00**	**8.54**	14.86	**24.76**
Qwen3 8B	Base	74.95	19.70	28.57	25.70	87.34	47.25
	SparseGPT	**69.04**	21.72	**25.89**	16.74	73.09	41.30
	Wanda	67.36	**26.77**	24.33	17.02	73.62	41.82
	Ours	66.75	26.26	25.67	**17.18**	**74.15**	**42.00**
Qwen3-14B	Base	78.81	23.23	26.79	32.50	84.84	49.23
	SparseGPT	**74.29**	**31.82**	23.88	22.78	76.35	45.82
	Wanda	74.07	25.25	**29.69**	23.54	79.00	46.31
	Ours	74.01	29.80	29.46	**23.80**	**80.82**	**47.58**
Qwen2.5 7B	Base	74.34	24.24	25.22	24.88	82.18	46.17
	SparseGPT	**67.47**	**26.30**	23.66	15.82	58.15	38.28
	Wanda	66.76	20.20	23.88	16.50	60.27	37.52
	Ours	67.16	21.72	**25.22**	**16.70**	**62.24**	**38.61**

not observed across all the individual tasks, which aligns with existing literature (Frantar and Alistarh, 2023; Dettmers and Zettlemoyer, 2022). However, the mean accuracy across all six tasks surpasses the performance of both SparseGPT and Wanda for most of the models. This observation aligns with our findings from the perplexity evaluation, suggesting the robustness and effectiveness of our approach.

3.4 K-Shot Reasoning

We evaluate `GBLM-Pruner` on STEM/general benchmarks: MMLU (Hendrycks et al., 2020) (5-shot), GPQA (Rein et al., 2024) (5-shot) and GPQA-D, MATH (Hendrycks et al., 2021) (4-shot), and GSM8K (Cobbe et al., 2021) (5-shot), at 50% unstructured sparsity across LLaMA-3.1 (Grattafiori et al., 2024) (8B), Qwen3 (Yang et al., 2025a) (8B, 14B), and Qwen2.5 (Yang et al., 2025a) (7B). As shown in Table 4, our method yields the highest mean accuracy on all models; e.g., on Qwen3-14B it reaches 47.58% vs. Wanda 46.31% and SparseGPT 45.82%. Gains are most pronounced on MATH and GSM8K, indicating preserved multi-step STEM reasoning.

Table 5: Performance comparison of base and pruned language models on coding benchmarks. All pruned models use 50% unstructured sparsity.

Model	Method	MBPP (3-shot)	HumanEval (0-shot)	Mean
Qwen3-8B	Base	65.60	62.80	64.20
	SparseGPT	37.20	12.80	25.00
	Wanda	43.40	26.22	34.81
	Ours	**44.00**	**28.05**	**36.02**
Qwen3-14B	Base	71.60	55.49	63.54
	SparseGPT	25.61	**49.40**	37.50
	Wanda	**56.00**	44.51	50.26
	Ours	55.00	46.34	**50.67**
DeepSeek-R1-Distill-Qwen-7B	Base	47.60	10.98	29.29
	SparseGPT	25.40	0.00	12.70
	Wanda	27.40	1.20	14.30
	Ours	**30.00**	**1.80**	**15.90**
DeepSeek-R1-Distill-Qwen-14B	Base	60.00	31.71	45.85
	SparseGPT	41.80	2.44	22.12
	Wanda	43.60	3.05	23.32
	Ours	**44.80**	**4.27**	**24.53**

Table 6: Performance comparison of base and pruned vision-language models on multimodal benchmarks at 50% unstructured sparsity. Values are shown as percentages. Bold indicates best performance among pruning methods, underlined indicates second best.

Model	Method	MMMU	MTVQA	Mean
Qwen2.5-VL-3B	Base	48.60	24.34	36.47
	SparseGPT	30.33	19.66	25.00
	Wanda	**37.50**	19.96	28.73
	Ours	36.81	**20.97**	**28.89**
Qwen2.5-VL-7B	Base	49.55	29.59	39.57
	SparseGPT	39.33	**27.14**	33.24
	Wanda	42.66	25.95	34.31
	Ours	**43.77**	26.09	**34.93**

Table 7: Ablation on the pruning metrics.

Method	Sparsity	7B	13B
Magnitude	0.5	16.03	6.83
SparseGPT	0.5	7.00	6.03
Wanda	0.5	**6.92**	5.97
$\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{G}\rVert_1$ (Ours)	0.5	7.17	6.15
$\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{G}\rVert_2$ (Ours)	0.5	7.09	**5.96**

Table 8: Pruning metric on weight, gradient, activation.

Method	Sparsity	Perplexity
$\lvert\mathbf{W}\rvert \cdot \lvert\mathbf{G}_{acc}\rvert$	0.5	119.72
$\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{G}\rVert_1$	0.5	7.17
$\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{G}\rVert_2$	0.5	7.09
$(\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2)^2 + \alpha \cdot \lvert\mathbf{W}\rvert \cdot \lVert\mathbf{G}\rVert_1$	0.5	6.90
$(\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2)^2 + \alpha \cdot \lvert\mathbf{W}\rvert \cdot \lVert\mathbf{G}\rVert_2$	0.5	6.88
$(\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2)^2 - \alpha \cdot \lvert\mathbf{W}\rvert \cdot \lVert\mathbf{G}\rVert_1$	0.5	9743.65
$(\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2)^2 - \alpha \cdot \lvert\mathbf{W}\rvert \cdot \lVert\mathbf{G}\rVert_2$	0.5	9377.00
$\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2 + \alpha \cdot \lvert\mathbf{W}\rvert \cdot \lVert\mathbf{G}\rVert_1$	0.5	**6.86**
$\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2 + \alpha \cdot \lvert\mathbf{W}\rvert \cdot \lVert\mathbf{G}\rVert_2$	0.5	6.89

3.5 Coding Tasks

We assess code generation on MBPP (Austin et al., 2021) (3-shot) and HumanEval (Chen et al., 2021) (0-shot) with 50% sparsity, comparing against SparseGPT and Wanda on Qwen3 (Yang et al., 2025a) (8B, 14B) and DeepSeek-R1-Distill-Qwen (Guo et al., 2025) (7B, 14B). Table 5 shows `GBLM-Pruner` consistently achieves

the best mean accuracy; for Qwen3-14B it obtains 50.67% vs. Wanda 50.26% and SparseGPT 37.50%, with similar improvements on the DeepSeek-R1-Distill-Qwen models.

3.6 Evaluating Vision–Language Models (VLMs)

We extend to VLMs by pruning Qwen2.5-VL-3B and Qwen2.5-VL-7B to 50% sparsity and evaluating on MMMU (Yue et al., 2024) and MTVQA (Tang et al., 2024). As summarized in Table 6, `GBLM-Pruner` attains the top mean scores on both models (28.89% and 34.93%), indicating that its gradient-based optimization effectively transfers to multimodal architectures.

3.7 Ablation Study

Importance of Gradient. To emphasize the role of gradient, we perform an ablation experiment as shown in Table 7, wherein we only consider the Gradient-Weight term of the `GBLM-Pruner` pruning metric. Our experiments show a substantial enhancement over magnitude-based pruning when utilizing gradients solely with weights, evident in both LLaMA-2 7B and 13B models. Additionally, the performance of our metric closely aligns with that of Wanda and SparseGPT for LLaMA-2 13B model.

Pruning Metric. In Section 2.4, we revisited the OBS framework by incorporating the first order gradient which yields $\delta \boldsymbol{E}_m = (\boldsymbol{w}_m \|x_m\|_2)^2 + \boldsymbol{w}_m (-\boldsymbol{g}_m)$ as the pruning metric. To start with, we experiment with different ways of estimating the gradient magnitude from the calibration samples. We evaluated three methods: gradient accumulation, ℓ_1 norm and ℓ_2 norm applied to the gradient across calibration samples. For this experiment, we only utilize the pruning metric based on gradient alone with weight for better interpretability. From our experiment, we observe that gradient accumulation yields the least favorable results as depicted in Table 8. For a deeper understanding, we compared the pruning pattern of gradient accumulation with ℓ_1 and ℓ_2 norm which shows that gradient accumulation gives a noisy estimate of the gradient magnitude while ℓ_1 and ℓ_2 norm reveal more structured patterns. A comparison between gradient accumulation and ℓ_1 norm-based aggregation is shown in Figure 2. Based on this, we adopt ℓ_1 and ℓ_2 norm-based gradient estimation for subsequent analysis.

Subsequently, based on our theoretical pruning metric $\delta \boldsymbol{E}_m$, we experiment with two different ways of coupling the activations and gradients as shown in Table 8. We observe that in the case of $(|\mathbf{W}| \cdot \|\mathbf{X}\|_2)^2 - |\mathbf{W}| \cdot \|\mathbf{G}\|_p$ the pruning metric is completely disrupted. While for $(|\mathbf{W}| \cdot \|\mathbf{X}\|_2)^2 + |\mathbf{W}| \cdot \|\mathbf{G}\|_p$ gradients and activations complement each other and brings out the best performance. But, upon closer examination, we observe that the square of the first activation term significantly outweighs the contribution of the second term involving gradients. Consequently, we remove the square factor from the first term and add a scaling factor denoted as α to the second gradient term, resulting in the formulation of our final pruning metric as $|\mathbf{W}| \cdot \|\mathbf{X}\|_2 + \alpha \cdot |\mathbf{W}| \cdot \|\mathbf{G}\|_p$. This pruning metric with ℓ_1 norm-based gradient aggregation gives the best result for unstructured pruning across all models. We also conduct experiments to calibrate the scaling factor α as shown in Table 13. We vary the scaling factor and examine how the LLaMA-2-7B pruned model perplexity changes. For a scaling factor is equal to 100, we get the best perplexity.

Sparsity Variation. The objective of this ablation is to assess the robustness of our method across varying sparsity. For this, we compare the perplexity of the unstructured pruned model obtained by `GBLM-Pruner` to that of Wanda, SparseGPT, and magnitude pruning. We consider two distinct model sizes: a smaller LLaMA-2 13B model and a larger LLaMA-1 30B model, each is subjected to different degrees of sparsity. The results are shown in Figure 4. From the figure, it is evident that `GBLM-Pruner` exhibits a similar trend to SparseGPT and Wanda, showing a decline in performance as sparsity increases. However, `GBLM-Pruner` consistently outperforms all other baseline methods across various levels of sparsity for both models.

Dependence on Calibration Sample. `GBLM-Pruner` uses a set of calibration samples to calculate gradients and activations for the pruning metric. To understand the robustness of the pruned model to the calibration set, we conduct two ablations:

(1) Robustness to calibration set: We randomly sampled 5 different calibration sets with 128 samples each and prune LLaMA-2 7B model to 0.5 sparsity using `GBLM-Pruner`. The resultant pruned models have perplexities: 6.86, 6.87, 6.89, 6.86, and 6.87 respectively.

(2) Number of samples in the calibration set: here, we want to assess the influence of the calibration set size on the performance of `GBLM-Pruner`. For this, we prune the LLaMA-2 7B model using various calibration sets with the number of samples ranging from 1 to 512. The results are reported in Figure 3. From the figure, we can observe that in contrast to SparseGPT, our method exhibits a relatively lower sensitivity to variations in the number of calibration samples.

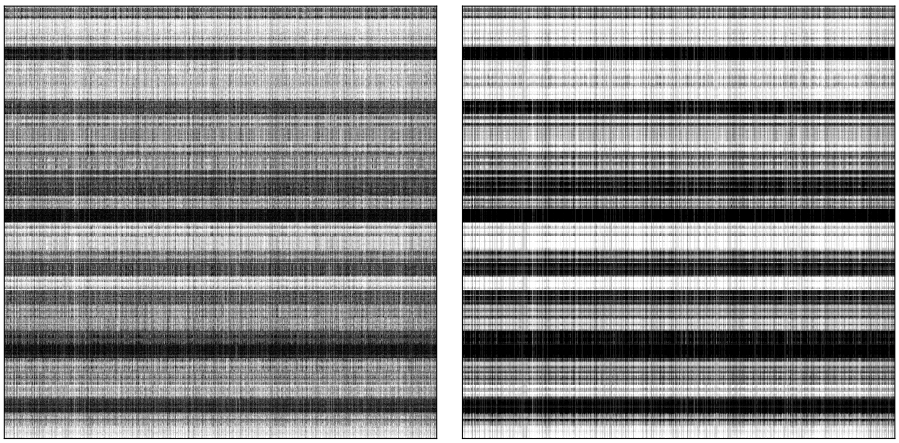

Figure 2: Illustration of learned pruning pattern.

3.8 Visualization of Pruned Pattern

Figure 2 visualizes 50% pruning masks for the first layer's key projection: left uses summation-aggregated gradients; right uses ℓ_1-norm–aggregated gradients. In each subfigure, x denotes input dimensions and y denotes output dimensions. The sum-aggregated mask appears noisier, whereas the ℓ_1-aggregated mask is sharper and more structured. After gradient aggregation, unstructured pruning reveals emergent structural patterns, reflecting geometric dependencies in LLM parameters aligned with gradient structure.

3.9 Inference Time and Memory Comparison

`GBLM-Pruner` has a one-time gradient pass and a layer-wise prune. Gradients require 71.3s/62GB (LLaMA1-7B) and 102.1s/96GB (LLaMA1-13B) on an A100 and are reusable across sparsity settings. The prune is $\mathcal{O}(d)$ like Wanda yet far faster than SparseGPT: 403.76ms vs. 256,732.95ms (636×) on 7B and 561.30ms vs. 423,550.22ms (754×) on 13B, with modest overhead vs. Wanda (257.04ms, 455.39ms). Speedups come from avoiding iterative weight updates/Hessian inverses. Also, 2:4 sparsity improves inference (overall 1.24×; per-layer 1.54–1.63× on A6000) and reduces storage.

Table 9: Computation requirements on an A100 GPU.

Model	Memory (GB)	Time (seconds)
LLaMA-7B	62	71.3
LLaMA-13B	96	102.1

Table 10: Pruning metric computation and weight removal time (milliseconds) on an A100 GPU.

Method	LLaMA-7B	LLaMA-13B
SparseGPT	256,732.95	423,550.22
Wanda	257.04	455.39
GBLM-Pruner	403.76	561.30

4 Related Work

Large Language Models (LLMs) based on transformer architecture (Vaswani et al., 2017) have ushered in a transformative era in the realm of natural language processing, achieving outstanding success. Their

Table 11: Inference speedup for 2:4 sparsity on an A6000 GPU across transformer layers of LLaMA-1.

LLaMA-1 layer	Dense	2:4	Speedup
q/k/v/o_proj	3.49	2.14	1.63×
up/gate_proj	9.82	6.10	1.61×
down_proj	9.92	6.45	1.54×

consistent and remarkable performance spans a wide array of tasks (Brown et al., 2020; Chung et al., 2022; Touvron et al., 2023a;b; Rozière et al., 2023; Achiam et al., 2023; Anil et al., 2023; Grattafiori et al., 2024; Yang et al., 2025a; Guo et al., 2025; Team et al., 2026; Zeng et al., 2026; Liu et al., 2026). For a long time, pruning has been identified as a powerful technique for reducing the size or complexity of a model by removing unnecessary or redundant components (LeCun et al., 1989; Hassibi et al., 1993). Pruning can be divided into structured and unstructured pruning. Structured pruning targets at removing a set of weights from a network at once such as channels or layers to reduce the model size and complexity while maintaining the network structure intact. In the realm of pruning LLMs, several studies (Frantar and Alistarh, 2022; 2023; Sun et al., 2023; Lee et al., 2025; Yang et al., 2026; Yin et al., 2026; Zou et al., 2026; Yu et al., 2026) have been undertaken in this area. Gradient-based importance estimation has proven effective in neural network pruning, as surveyed comprehensively by Hoefler et al. (2021). Early approaches utilized activation gradients during training (Ye et al., 2020) and position-based gradient scaling (Kim et al., 2020; Lv et al., 2024). Recent gradient-based methods for LLMs focus on structured pruning, employing first-order gradients with Hessian approximations (Ma et al., 2023), forward-pass gradient estimation (Cheng et al., 2024), second-order curvature information (van der Ouderaa et al., 2023; Kurtic et al., 2022; Singh and Alistarh, 2020), regional gradients (Yang et al., 2025b), gradient-guided layer pruning (Huang et al., 2026), and LoRA-aware approaches (Yu et al., 2025; Zhang et al., 2024). While prior gradient-based methods focus on structured pruning or require task-specific finetuning or weight update, we introduce the first weight update-free gradient-based solution for unstructured and semi-structured LLM pruning that maintains zero-shot generalization capabilities. Our work provides a unique angle from *gradient* along this direction, combining first-order gradients with activation magnitudes in a computationally efficient manner.

5 Conclusion

We presented a gradient-based pruning approach `GBLM-Pruner` for LLMs. Our approach performs in a training-free manner and applies gradient-based statistical magnitude to discern and selectively prune the model's parameters, enabling substantial reductions in model size while preserving the model's predictive accuracy. The proposed approach has surpassed all previous LLM pruning methods in terms of perplexity, zero-shot performance and interpretability, marking a pivotal advancement in the field. We also provided theoretical analyses on how gradients help identify the importance of weights in LLMs.

Ethics Statement

The primary goal of this research is to improve the efficiency of LLMs by proposing a new network pruning approach. Any potential ethical issues of pretrained LLMs, such as harmful biases, privacy and spreading disinformation, are likely to still exist after pruning.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al. (2022). Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736.

Anil, R., Dai, A. M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., et al. (2023). Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., et al. (2021). Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

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

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Chen, Y., Xie, Y., Song, L., Chen, F., and Tang, T. (2020). A survey of accelerator architectures for deep neural networks. *Engineering*, 6(3):264–274.

Cheng, H., Zhang, M., and Shi, J. Q. (2024). Mini-llm: Memory-efficient structured pruning for large language models. *arXiv preprint arXiv:2407.11681*.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Valter, D., Narang, S., Mishra, G., Yu, A. W., Zhao, V., Huang, Y., Dai, A. M., Yu, H., Petrov, S., hsin Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., and Wei, J. (2022). Scaling instruction-finetuned language models. *ArXiv*, abs/2210.11416.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. (2019). Boolq: Exploring the surprising difficulty of natural yes/no questions. *ArXiv*, abs/1905.10044.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. (2018). Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. (2021). Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. (2022). Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Dettmers, T. and Zettlemoyer, L. (2022). The case for 4-bit precision: k-bit inference scaling laws. In *International Conference on Machine Learning*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Frankle, J. and Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

Frantar, E. and Alistarh, D. (2022). Optimal brain compression: A framework for accurate post-training quantization and pruning. *ArXiv*, abs/2208.11580.

Frantar, E. and Alistarh, D. (2023). Sparsegpt: Massive language models can be accurately pruned in one-shot. *ArXiv*, abs/2301.00774.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. (2022). Gptq: Accurate post-training quantization for generative pre-trained transformers. *ArXiv*, abs/2210.17323.

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

Gao, L., Tow, J., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., McDonell, K., Muennighoff, N., Phang, J., Reynolds, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. (2021). A framework for few-shot language model evaluation.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. (2024). The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. (2025). Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Han, S., Mao, H., and Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*.

Han, S., Pool, J., Tran, J., and Dally, W. (2015). Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.

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

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. (2020). Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. (2021). Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., and Peste, A. (2021). Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124.

Huang, W., Cheng, A., and Wang, Y. (2026). Gradpruner: Gradient-guided layer pruning enabling efficient fine-tuning and inference for llms. *arXiv preprint arXiv:2601.19503*.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713.

Kim, J., Yoo, K., and Kwak, N. (2020). Position-based scaled gradient for model quantization and pruning. *Advances in neural information processing systems*, 33:20415–20426.

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

LeCun, Y., Denker, J., and Solla, S. (1989). Optimal brain damage. *Advances in neural information processing systems*, 2.

Lee, K., Jang, H., Lee, D., Alistarh, D., and Lee, N. (2025). The unseen frontier: Pushing the limits of llm sparsity with surrogate-free admm. *arXiv preprint arXiv:2510.01650*.

Li, J., Li, D., Savarese, S., and Hoi, S. (2023). Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*.

Liu, T., Wang, A. X., Panescu, A., Chen, L. X., Long, W., Wei, X., Jing, Y., Zeng, Z., Chen, J., Jiang, S., et al. (2026). Benchmarking ai agents for addressing scientific challenges across scales. *arXiv preprint arXiv:2606.12736*.

Lv, Q., Sun, J., Zhou, S., Zhang, X., Li, L., Gao, Y., Qiao, S., Song, J., and Bu, J. (2024). Fgp: Feature-gradient-prune for efficient convolutional layer pruning. *arXiv preprint arXiv:2411.12781*.

Ma, X., Fang, G., and Wang, X. (2023). Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. (2016). Pointer sentinel mixture models. *ArXiv*, abs/1609.07843.

Michel, P., Levy, O., and Neubig, G. (2019). Are sixteen heads really better than one? *Advances in neural information processing systems*, 32.

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. (2018). Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Conference on Empirical Methods in Natural Language Processing*.

Molchanov, P., Tyree, S., Karras, T., Aila, T., and Kautz, J. (2016). Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. In *https://openreview.net/pdf?id=BJJsrmfCZ*.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training. In *https://openai.com/research/language-unsupervised*. OpenAI.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Raffel, C., Shazeer, N. M., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. (2024). Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Rozière, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X., Adi, Y., Liu, J., Remez, T., Rapin, J., Kozhevnikov, A., Evtimov, I., Bitton, J., Bhatt, M. P., Ferrer, C. C., Grattafiori, A., Xiong, W., D'efossez, A., Copet, J., Azhar, F., Touvron, H., Martin, L., Usunier, N., Scialom, T., and Synnaeve, G. (2023). Code llama: Open foundation models for code. *ArXiv*, abs/2308.12950.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. (2019). Winogrande. *Communications of the ACM*, 64:99 – 106.

Sanh, V., Wolf, T., and Rush, A. (2020). Movement pruning: Adaptive sparsity by fine-tuning. *Advances in Neural Information Processing Systems*, 33:20378–20389.

Singh, S. P. and Alistarh, D. (2020). Woodfisher: Efficient second-order approximation for neural network compression. *Advances in Neural Information Processing Systems*, 33:18098–18109.

Sun, M., Liu, Z., Bair, A., and Kolter, J. Z. (2023). A simple and effective pruning approach for large language models. *ArXiv*, abs/2306.11695.

Tang, J., Liu, Q., Ye, Y., Lu, J., Wei, S., Lin, C., Li, W., Mahmood, M. F. F. B., Feng, H., Zhao, Z., et al. (2024). Mtvqa: Benchmarking multilingual text-centric visual question answering. *arXiv preprint arXiv:2405.11985*.

Team, K., Bai, T., Bai, Y., Bao, Y., Cai, S., Cao, Y., Charles, Y., Che, H., Chen, C., Chen, G., et al. (2026). Kimi k2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*.

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

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

van der Ouderaa, T. F., Nagel, M., Van Baalen, M., Asano, Y. M., and Blankevoort, T. (2023). The llm surgeon. *arXiv preprint arXiv:2312.17244*.

Vaswani, A., Shazeer, N. M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *NIPS*.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. In *BlackboxNLP@EMNLP*.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and Brew, J. (2019). Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. (2025a). Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yang, Y., Zhen, K., Ganesh, B., Galstyan, A., Huybrechts, G., Müller, M., Kübler, J. M., Swaminathan, R. V., Mouchtaris, A., Bodapati, S. B., et al. (2025b). Wanda++: Pruning large language models via regional gradients. *arXiv preprint arXiv:2503.04992*.

Yang, Z., Cui, Y., Yao, X., and Wang, S. (2022). Gradient-based intra-attention pruning on pre-trained language models. *arXiv preprint arXiv:2212.07634*.

Yang, Z., Deng, C., Chen, Q., Zhou, Z., and Cheng, J. (2026). Lsa: Layer-wise sparsity allocation for large language model pruning based on minimal linear reconstruction error. In *The Fourteenth International Conference on Learning Representations*.

Yao, Z., Aminabadi, R. Y., Zhang, M., Wu, X., Li, C., and He, Y. (2022). Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *ArXiv*, abs/2206.01861.

Ye, X., Dai, P., Luo, J., Guo, X., Qi, Y., Yang, J., and Chen, Y. (2020). Accelerating cnn training by pruning activation gradients. In *European Conference on Computer Vision*, pages 322–338. Springer.

Yin, R., Li, Y., Lee, D., and Panda, P. (2026). Duogpt: Training-free dual sparsity through activation-aware pruning in llms. *Advances in Neural Information Processing Systems*, 38:146883–146912.

Yu, P., Wang, J., Sui, X., Ling, N., Wang, W., and Jiang, W. (2026). Efficient post-training pruning of large language models with statistical correction. *arXiv preprint arXiv:2602.07375*.

Yu, X., Xie, C., Zhao, Z., Fan, T., Xue, L., and Zhang, Z. (2025). Prunedlora: Robust gradient-based structured pruning for low-rank adaptation in fine-tuning. *arXiv preprint arXiv:2510.00192*.

Yue, X., Ni, Y., Zhang, K., Zheng, T., Liu, R., Zhang, G., Stevens, S., Jiang, D., Ren, W., Sun, Y., et al. (2024). Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9556–9567.

Zdravkova, K., Krasniqi, V., Dalipi, F., and Ferati, M. (2022). Cutting-edge communication and learning assistive technologies for disabled children: An artificial intelligence perspective. *Frontiers in Artificial Intelligence*, page 240.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. (2019). Hellaswag: Can a machine really finish your sentence? In *Annual Meeting of the Association for Computational Linguistics*.

Zeng, A., Lv, X., Hou, Z., Du, Z., Zheng, Q., Chen, B., Yin, D., Ge, C., Huang, C., Xie, C., et al. (2026). Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*.

Zhang, M., Chen, H., Shen, C., Yang, Z., Ou, L., Yu, X., and Zhuang, B. (2024). Loraprune: Structured pruning meets low-rank parameter-efficient fine-tuning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3013–3026.

Zou, L., Yin, S., Pei, Z., Ho, T.-Y., Farnia, F., and Yu, B. (2026). Permllm: Learnable channel permutation for n: M sparse large language models. *Advances in Neural Information Processing Systems*, 38:20183–20205.

Appendix

Contents

A Vision Transformers **20**

B Difference to Previous Gradient-based Methods **20**

C Baselines **20**

D Evaluation Metric **20**

E Comparison Group **21**

F LLaMA-Chat Models **21**

G OBS Weight Update **23**

H Correlations of Weights, Activations and Gradients. **23**

I Optimal Brain Surgeon Considering gradient **23**

J Different Pruning Metric **25**

K Higher Sparsity Evaluation **25**

L Detailed Hessian Derivation **25**

A Vision Transformers

To assess the generalizability of our method across models with different input modalities, we conduct experiments on the ViT-B model. We compare the performance of the pruned model obtained using `GBLM-Pruner` with those obtained through magnitude pruning and the Wanda method. We use 4,096 random samples from ImageNet-1k training set as our calibration data, and subsequently, we evaluate the pruned models on the standard ImageNet-1k classification task. The results of these evaluations are presented in Table 12. We can see that our method outperforms both Wanda and magnitude pruning, particularly when dealing with higher levels of sparsity.

Table 12: ViT-B model pruning.

Sparsity	Wanda	Magnitude	Ours ℓ_1	Ours ℓ_2
0	75.40	75.40	75.40	75.40
0.5	64.54	59.48	64.64	**64.86**
0.6	43.65	29.98	44.15	**44.23**
0.7	7.92	1.88	**8.89**	8.02
0.8	0.20	0.18	**0.32**	0.24

B Difference to Previous Gradient-based Methods

Although the use of gradients has been studied in the context of pruning, earlier methods (Molchanov et al., 2016; Sanh et al., 2020) used gradients in the context of transfer learning to obtain a pruned model that preserves the accuracy of the downstream task. This work is the first attempt to study the use of gradients for one-shot pruning of language models with billions of parameters while maintaining the zero-shot generalization capabilities of the language models to diverse downstream tasks. Additionally, our proposed method does not require weight update, which makes our proposed method computationally efficient and applicable for large language models with billions of parameters like LLaMA-1-30B and LLaMA-2-70B.

C Baselines

We compare our proposed method against three pruning baselines:

- **Magnitude pruning**: Magnitude pruning (Han et al., 2015) is a simple and scalable pruning method where the importance of LLM weights is decided based on the absolute value of their magnitude. Following Gale et al. (2019) and Sanh et al. (2020), we conduct a layer-wise comparison of model weights, subsequently removing those with smaller magnitudes.
- **SparseGPT**: SparseGPT (Frantar and Alistarh, 2023) is based on the second-order Optimal Brain Surgeon framework (Hassibi et al., 1993). It optimizes the accurate Optimal Brain Surgeon framework and introduces the first accurate one-shot pruning method that works efficiently at the scale of billions of parameters.
- **Wanda**: Wanda (Sun et al., 2023) proposed a simple pruning metric and showed the importance of activations in addition to weight magnitude while selecting weights for pruning. Unlike previous algorithms, it does not require any weight update of the remaining weights.

D Evaluation Metric

Perplexity and Zero-shot Evaluation on Harness are two well-established metrics for evaluating compressed models:

- **Perplexity**: Following previous work on model compression both in case of quantization (Dettmers and Zettlemoyer, 2022; Yao et al., 2022) and pruning (Frantar and Alistarh, 2022; Sun et al., 2023;

Frantar and Alistarh, 2023) we used perplexity as an evaluation metric to compare the pruned models. Perplexity is a stable, robust and challenging metric that is suited for evaluating the accuracy of compression methods. We used the WikiText (Merity et al., 2016) validation set for computing perplexity.

- **Zero-Shot Evaluation on Harness Benchmarks**: To complement perplexity, we provided the evaluation of the pruned model on the publicly available Eleuther AI LM Harness benchmark (Gao et al., 2021) for additional interpretability. We conducted evaluations on five standard common-sense reasoning tasks, including RTE (Wang et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2019), ARC-easy (Clark et al., 2018), OBQA (Mihaylov et al., 2018) and the BoolQ (Clark et al., 2019) reading comprehension task. Our evaluation primarily centers on assessing the pruned models' accuracy in comparison to the dense baseline, rather than emphasizing absolute numerical values.

Table 13: Ablation of scaling factor.

Scaling Factor, (α)	Perplexity
0.001	6.920
0.01	6.919
0.1	6.921
1	6.912
10	6.890
100	**6.858**
1000	6.902
10000	6.926
100000	6.952

E Comparison Group

Comparison group plays a pivotal role even in unstructured pruning. For `GBLM-Pruner`, we have experimented with 5 different comparison groups:

- `Layer-wise`: With layer-wise pruning, weights within same layer are compared for pruning.
- `(input, 1)`: For (input,1), weights connected within an input channel are grouped together for comparison.
- `(output, 1)`: Similarly in this approach, weights connected within an output channel are grouped together for comparison.
- `(input, 128)`: This comparison group involves forming blocks of 128 input channels, and weights within each block are compared for pruning.
- `(output, 128)`: Similar to (input,128), here blocks of 128 channels are formed along the output dimension for pruning.

F LLaMA-Chat Models

The LLaMA-2 series of models also includes fine-tuned chat versions. We sought to assess the generalization of our method to these chat models, specifically focusing on LLaMA-2-chat-7B and LLaMA-2-chat-13B as representative models. Similar to the pretrained LLaMA-2 series, our calibration data consisted of 128 samples, each comprising 2048 tokens from the C4 dataset. For evaluation purposes, we employed the Wiki-Text validation set.

Our approach to pruning was consistent with that applied to the pretrained LLaMA-2 models. We uniformly pruned every linear layer, except for the initial embedding layer and the final classification layer. We compare

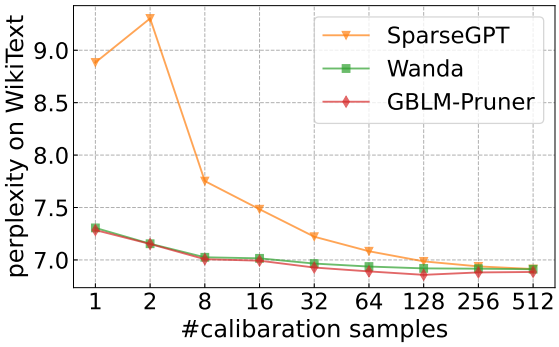

Figure 3: Robustness to calibration samples.

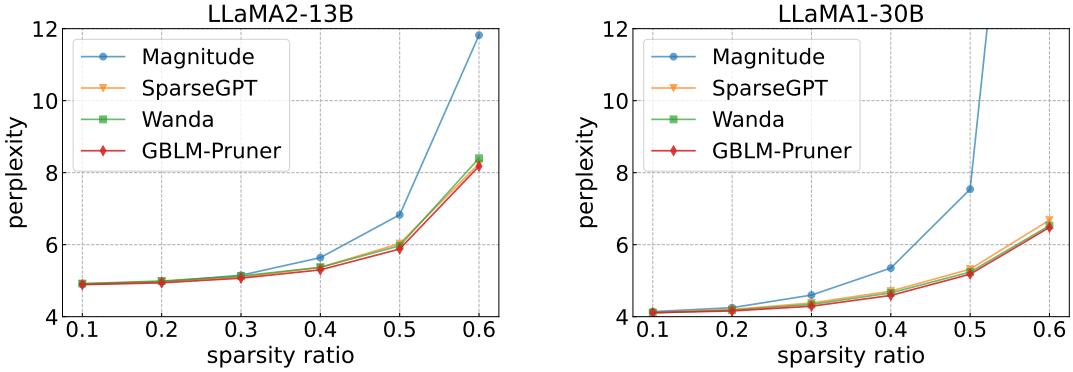

Figure 4: Sparsity variation results for a large and a small model where we compare the performance of our method against other baseline methods.

every weight of the linear layer on per output basis where pruning metric is compared within the output neuron.

The results are presented in Table 14. Examining the table, we can discern that our method consistently delivers superior performance, particularly evident in unstructured pruning. When it comes to N:M sparsity pruning, although SparseGPT achieves the lowest perplexity, our pruning metric significantly outperforms Wanda by a substantial margin.

Table 14: WikiText validation perplexity of different pruning methods for LLaMA 2 chat models.

Method	Sparsity	LLaMA-2-7B-chat	LLaMA-13B-chat
None	0	7.08	6.11
Magnitude	0.5	22.82	8.49
SparseGPT	0.5	8.66	7.26
Wanda	0.5	8.78	7.50
`GBLM-Pruner`$_{\ell 2}$	0.5	8.52	7.27
`GBLM-Pruner`$_{\ell 1}$	0.5	**8.40**	**7.10**
Magnitude	2:4	45.95	11.14
SparseGPT	2:4	**12.19**	**9.37**
Wanda	2:4	14.45	10.25
`GBLM-Pruner`$_{\ell 2}$	2:4	13.74	9.85
`GBLM-Pruner`$_{\ell 1}$	2:4	13.92	9.66
Magnitude	4:8	22.57	9.80
SparseGPT	4:8	**10.02**	**8.01**
Wanda	4:8	10.86	8.56
`GBLM-Pruner`$_{\ell 2}$	4:8	10.45	8.26
`GBLM-Pruner`$_{\ell 1}$	4:8	10.46	8.10

G OBS Weight Update

In this study, our objective was to assess whether the OBS (Optimal Brain Surgeon) weight update method enhances the performance of our pruned model. We implemented the OBS weight update using the efficient approach proposed by SparseGPT (Frantar and Alistarh, 2023).

The results, presented in Table 15, indicate that the OBS weight update does not lead to an improvement in the performance of our pruned model

Table 15: OBS weight update.

		Weight Update	
Method	Datasplit	no	yes
Magnitude	Calib	18.14	12.93
	Valid	17.29	12.55
Wanda	Calib	7.52	7.61
	Valid	7.26	7.36
Ours	Calib	7.54	7.64
	Valid	7.26	7.39

H Correlations of Weights, Activations and Gradients.

This section discusses an intuitive explanation of why gradient is essential. Weights are parameters in LLMs that are learned during the training process to minimize the loss function. They are fundamental in determining the strength of the connection between two neurons and subsequently the output of the network. Gradients of the loss with respect to weights, computed using an optimization algorithm like SGD (Ruder, 2016), are central to the learning process as they guide the updates made to the weights during training. On the otherhand, activations are the outputs of the neurons, typically computed as a weighted sum of inputs passed through an activation function. The activations are intrinsically impacted by the weights thus weight augmented with activation serves as a redundant indicator of weight importance. However, gradient being the guiding signal for the learning process serves as a valuable indicator by signalling the sensitivity of the loss to weight change and thus the importance of the weight in the pruning process.

I Optimal Brain Surgeon Considering gradient

As part of the theoretical justification for our proposed gradient-based metric, we revisited and redefined the OBS framework by incorporating considerations of the gradient information. The complete derivation of this process is meticulously presented within this section.

The Taylor Series expansion of the error with respect to weight is:

$$\delta \boldsymbol{E} = \left(\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{w}}\right)^{\top} \cdot \delta \boldsymbol{w} + \frac{1}{2} \delta \boldsymbol{w}^{\top} \cdot \mathbf{H} \cdot \delta \boldsymbol{w} + \mathcal{O}(||\delta \boldsymbol{w}||^3) \tag{16}$$

where $\boldsymbol{E}$ is the error or loss function and $\boldsymbol{w}$ is the weight vector for the neural network. The symbol $\mathbf{H} = \frac{\partial^2 \boldsymbol{E}}{\partial \boldsymbol{w}^2}$ denotes the Hessian Matrix, and the superscript $\top$ signifies vector transpose. Based on this we formulate the optimization problem for network pruning using both the first and second-order terms as depicted in Equation 17. Here, $\boldsymbol{w}_m$ is the pruned weight, $\delta \boldsymbol{w}$ is the change in weight magnitude for $\boldsymbol{w}_m$ and $\mathbf{e}_m$ is the unit vector in weight space corresponding to weight $\boldsymbol{w}_m$.

$$\begin{aligned} \min_{m} \Bigg\{ \min_{\delta \boldsymbol{w}} \Bigg(& \left(\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{w}}\right)^{\top} \cdot \delta \boldsymbol{w} \\ & + \frac{1}{2} \delta \boldsymbol{w}^{\top} \cdot \mathbf{H} \cdot \delta \boldsymbol{w} \Bigg) \Big| \mathbf{e}_m^{\top} \cdot \delta \boldsymbol{w} + w_m = 0 \Bigg\} \end{aligned} \tag{17}$$

Table 16: Pruning metric.

Method	Sparsity	Perplexity	Method	Sparsity	Perplexity
$\lvert\mathbf{W}\rvert \cdot \lvert\mathbf{G}_{acc}\rvert$	0.5	119.72	$(\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2)^2 + \alpha \cdot \mathbf{W} \cdot \mathbf{G}_{acc}$	0.5	7.04
$\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{G}\rVert_1$	0.5	7.17	$(\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2)^2 + \alpha \cdot \mathbf{W} \cdot \lVert\mathbf{G}\rVert_1$	0.5	180490.19
$\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{G}\rVert_2$	0.5	7.09	$(\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2)^2 + \alpha \cdot \mathbf{W} \cdot \lVert\mathbf{G}\rVert_2$	0.5	91781.49
$\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2 \cdot \lvert\mathbf{G}_{acc}\rvert$	0.5	69.59	$(\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2)^2 - \alpha \cdot \mathbf{W} \cdot \mathbf{G}_{acc}$	0.5	7.14
$\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2 \cdot \lVert\mathbf{G}\rVert_1$	0.5	7.31	$(\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2)^2 - \alpha \cdot \mathbf{W} \cdot \lVert\mathbf{G}\rVert_1$	0.5	246846.28
$\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2 \cdot \lVert\mathbf{G}\rVert_2$	0.5	7.31	$(\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2)^2 - \alpha \cdot \mathbf{W} \cdot \lVert\mathbf{G}\rVert_2$	0.5	283620.75
$\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2 + \alpha \cdot \lvert\mathbf{W}\rvert \cdot \lvert\mathbf{G}_{acc}\rvert$	0.5	6.92	$(\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2)^2 + \alpha \cdot \lvert\mathbf{W}\rvert \cdot \lvert\mathbf{G}_{acc}\rvert$	0.5	6.91
$\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2 + \alpha \cdot \lvert\mathbf{W}\rvert \cdot \lVert\mathbf{G}\rVert_1$	0.5	**6.86**	$(\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2)^2 + \alpha \cdot \lvert\mathbf{W}\rvert \cdot \lVert\mathbf{G}\rVert_1$	0.5	6.90
$\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2 + \alpha \cdot \lvert\mathbf{W}\rvert \cdot \lVert\mathbf{G}\rVert_2$	0.5	6.89	$(\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2)^2 + \alpha \cdot \lvert\mathbf{W}\rvert \cdot \lVert\mathbf{G}\rVert_2$	0.5	6.88
$\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2 - \alpha \cdot \lvert\mathbf{W}\rvert \cdot \lvert\mathbf{G}_{acc}\rvert$	0.5	6.92	$(\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2)^2 - \alpha \cdot \lvert\mathbf{W}\rvert \cdot \lvert\mathbf{G}_{acc}\rvert$	0.5	6.94
$\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2 - \alpha \cdot \lvert\mathbf{W}\rvert \cdot \lVert\mathbf{G}\rVert_1$	0.5	1180.67	$(\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2)^2 - \alpha \cdot \lvert\mathbf{W}\rvert \cdot \lVert\mathbf{G}\rVert_1$	0.5	9743.65
$\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2 - \alpha \cdot \lvert\mathbf{W}\rvert \cdot \lVert\mathbf{G}\rVert_2$	0.5	7.10	$(\lvert\mathbf{W}\rvert \cdot \lVert\mathbf{X}\rVert_2)^2 - \alpha \cdot \lvert\mathbf{W}\rvert \cdot \lVert\mathbf{G}\rVert_2$	0.5	9377.00

The Lagrangian formulation of the optimization problem is:

$$\mathcal{L} = \boldsymbol{g}^\top \cdot \delta\boldsymbol{w} + \frac{1}{2}\delta\boldsymbol{w}^\top \cdot \mathbf{H} \cdot \delta\boldsymbol{w} + \lambda\left(\mathbf{e}_m^\top \cdot \delta\boldsymbol{w} + w_m\right) \tag{18}$$

Now, differentiating Equation 18 w.r.t λ

$$\mathbf{e}_m^\top \cdot \delta\boldsymbol{w} + w_m = 0 \tag{19}$$

Differentiating w.r.t $\delta\boldsymbol{w}$

$$\begin{aligned} \boldsymbol{g} + \mathbf{H} \cdot \delta\boldsymbol{w} + \lambda\mathbf{e}_m &= 0 \\ \Rightarrow \delta\boldsymbol{w} &= -\mathbf{H}^{-1} \cdot (\lambda\mathbf{e}_m + \boldsymbol{g}) \end{aligned} \tag{20}$$

From 19 and 20, we have

$$\begin{aligned} &\mathbf{e}_m^\top\left(-\mathbf{H}^{-1} \cdot (\lambda\mathbf{e}_m + \boldsymbol{g})\right) + w_m = 0 \\ &\Rightarrow -\lambda\left(\boldsymbol{H}^{-1}\right)_{mm} - \mathbf{e}_m^\top \cdot \mathbf{H}^{-1} \cdot \boldsymbol{g} + w_m = 0 \\ &\Rightarrow \lambda = \frac{w_m - \mathbf{e}_m^\top \cdot \mathbf{H}^{-1} \cdot \boldsymbol{g}}{\left(\mathbf{H}^{-1}\right)_{mm}} \end{aligned} \tag{21}$$

From 20 and 21, we get the optimal weight change $\delta\boldsymbol{w}$ as:

$$\begin{aligned} \delta\boldsymbol{w} &= -\mathbf{H}^{-1} \cdot \left(\frac{w_m - \mathbf{e}_m^\top \cdot \mathbf{H}^{-1} \cdot \boldsymbol{g}}{\left(\mathbf{H}^{-1}\right)_{mm}} \cdot \mathbf{e}_m + \boldsymbol{g}\right) \\ &= -\frac{w_m}{\left(\mathbf{H}^{-1}\right)_{mm}}\mathbf{H}^{-1} \cdot \mathbf{e}_m + \frac{\mathbf{e}_m^\top \cdot \mathbf{H}^{-1} \cdot g}{\left(\mathbf{H}^{-1}\right)_{mm}}\mathbf{H}^{-1} \cdot \mathbf{e}_m - \mathbf{H}^{-1} \cdot \boldsymbol{g} \end{aligned} \tag{22}$$

The increase in error on changing weight w_m by $\delta\boldsymbol{w}$ is:

$$\delta\boldsymbol{E}_m = \boldsymbol{g}^\top \cdot \delta\boldsymbol{w} + \frac{1}{2}\delta\boldsymbol{w}^\top \cdot \mathbf{H} \cdot \delta\boldsymbol{w} \tag{23}$$

Substituting the optimal value of $\delta \boldsymbol{w}$ in Equation 23 gives:

$$\begin{aligned}\delta \boldsymbol{E}_m = & \frac{w_m^2}{2\left(\mathbf{H}^{-1}\right)_{mm}} - \frac{w_m\left(\boldsymbol{g}^\top \cdot \mathbf{H}^{-1} \cdot \mathbf{e}_m\right)}{\left(\mathbf{H}^{-1}\right)_{mm}} \\ & + \frac{\left(\mathbf{e}_m^\top \cdot \mathbf{H}^{-1} \cdot \boldsymbol{g}\right)^2}{2\left(\mathbf{H}^{-1}\right)_{mm}} - \frac{1}{2}\boldsymbol{g}^\top \cdot \mathbf{H}^{-1} \cdot \boldsymbol{g}\end{aligned} \tag{24}$$

J Different Pruning Metric

In the ablation Section 3.7, we present an analysis of our pruning metric. Table 16 enumerates all the pruning metrics we explored and serves as a comprehensive consolidation of our study.

K Higher Sparsity Evaluation

To evaluate the robustness of `GBLM-Pruner` at extreme sparsity levels, we conduct experiments at 0.7, 0.8, 0.9, and 0.95 sparsity ratios. Table 17 reports WikiText perplexity for Qwen3-14B and LLaMA-3.1-8B models. At all sparsity levels, we can observe that gradient information is extremely valuable: `GBLM-Pruner` achieves 24% lower perplexity than Wanda on Qwen3-14B at 0.8 sparsity, and 52% lower at 0.9 sparsity. Similar trends hold for LLaMA-3.1-8B, where our method achieves 43% and 41% lower perplexity at 0.8 and 0.9 sparsity, respectively. At extreme sparsity (0.95), models exhibit degraded performance, but `GBLM-Pruner` maintains a substantial advantage (30% lower perplexity). These results demonstrate that gradient-based pruning becomes particularly important at higher sparsity levels where pruning decisions are most challenging.

Table 17: WikiText perplexity at higher sparsity levels. Bold indicates best performance.

Model	Sparsity	Wanda	GBLM-Pruner
Qwen3-14B	0.7	33.12	**32.13**
	0.8	254.90	**192.85**
	0.9	10902.73	**5225.87**
	0.95	16872.77	**11845.89**
LLaMA-3.1-8B	0.7	113.59	**111.48**
	0.8	1480.43	**835.78**
	0.9	13155.59	**7726.91**

L Detailed Hessian Derivation

This appendix provides the complete derivation connecting our Hessian approximation $\mathbf{H} \approx \mathbf{X}^\top \mathbf{X}$ to the original Optimal Brain Surgeon framework. We show how layer-wise analysis combined with diagonal approximations yields the form used in our pruning metric.

Following Hassibi and Stork (Hassibi et al., 1993), for the squared error loss function under the $(t - o) \to 0$ approximation for well-trained networks, the Hessian of the global loss function can be expressed as (OBS Equation 11):

$$\mathbf{H} \approx \frac{1}{N}\sum_{k=1}^{N} \mathbf{J}^{[k]\top}\mathbf{J}^{[k]} \tag{25}$$

where $\mathbf{J}^{[k]} = \frac{\partial \mathbf{F}(\mathbf{w}, \mathbf{x}^{[k]})}{\partial \mathbf{w}}$ is the Jacobian of network outputs with respect to weights for sample k, and N is the number of calibration samples.

Consider a single linear layer within the network with transformation $\mathbf{a}^{[k]} = \mathbf{W}\mathbf{x}^{[k]}$, where the i-th output is $a_i^{[k]} = \sum_j W[i,j] \cdot x_j^{[k]}$. Applying the chain rule, the derivative of network output F_ℓ with respect to weight $W[i,j]$ is:

$$\frac{\partial F_\ell(\mathbf{w}, \mathbf{x}^{[k]})}{\partial W[i,j]} = \frac{\partial F_\ell}{\partial a_i^{[k]}} \cdot \frac{\partial a_i^{[k]}}{\partial W[i,j]} = B_{\ell i}^{[k]} \cdot x_j^{[k]} \tag{26}$$

where $B_{\ell i}^{[k]} := \frac{\partial F_\ell(\mathbf{w}, \mathbf{x}^{[k]})}{\partial a_i^{[k]}}$ represents the backpropagated sensitivity of output ℓ to the pre-activation a_i of our layer.

Substituting Equation 26 into the Hessian formula from Equation 25, the Hessian element corresponding to weights $W[i, j]$ and $W[i', j']$ becomes:

$$\begin{aligned} H_{(i,j),(i',j')} &\approx \frac{1}{N} \sum_{k=1}^{N} \sum_{\ell} \left(B_{\ell i}^{[k]} x_j^{[k]} \right) \left(B_{\ell i'}^{[k]} x_{j'}^{[k]} \right) \\ &= \frac{1}{N} \sum_{k=1}^{N} x_j^{[k]} x_{j'}^{[k]} \underbrace{\sum_{\ell} B_{\ell i}^{[k]} B_{\ell i'}^{[k]}}_{M_{ii'}^{[k]}} \end{aligned} \tag{27}$$

where $M_{ii'}^{[k]} = \sum_\ell B_{\ell i}^{[k]} B_{\ell i'}^{[k]}$ captures all backpropagation information and equals $\frac{\partial^2 E}{\partial a_i^{[k]} \partial a_{i'}^{[k]}}$, the second derivative of loss with respect to pre-activations.

We now introduce two simplifying assumptions to make the Hessian computation tractable. First, following LeCun et al. (LeCun et al., 1989) (Optimal Brain Damage), we assume that interactions between different output neurons in the same layer are negligible:

$$M_{ii'}^{[k]} \approx 0 \quad \text{for } i \neq i' \tag{28}$$

This diagonal approximation across neurons gives a block-diagonal Hessian structure in neuron index i, substantially reducing computational complexity.

Second, we assume that the curvature terms $M_{ii}^{[k]} = \frac{\partial^2 E}{\partial a_i^{[k]2}}$ are approximately constant across neurons and samples within a layer:

$$M_{ii}^{[k]} \approx c \quad \text{for all } i, k \tag{29}$$

where c is a constant. This is a strong assumption, as different neurons may have different curvature depending on their role in the network. However, computing per-neuron curvature would require expensive second-order computations. Moreover, this approximation is implicitly made in prior work such as Wanda (Sun et al., 2023), which uses $|\mathbf{W}[i, j]| \cdot \|\mathbf{X}[:, j]\|_2$ without any curvature terms.

Under these two assumptions, Equation 27 simplifies to:

$$H_{(i,j),(i',j')} \approx c \cdot \delta_{ii'} \left[\frac{1}{N} \sum_{k=1}^{N} x_j^{[k]} x_{j'}^{[k]} \right] = c \cdot \delta_{ii'} \cdot [\mathbf{X}^\top \mathbf{X}]_{jj'} \tag{30}$$

where $\delta_{ii'}$ is the Kronecker delta.

Further applying the diagonal approximation in the input dimension j (standard from Optimal Brain Damage), we consider only diagonal elements:

$$H_{(i,j),(i,j)} \approx c \cdot \sum_{k=1}^{N} (x_j^{[k]})^2 = c \cdot \|\mathbf{X}[:, j]\|_2^2 \tag{31}$$

In compact notation for the entire layer, this can be written as:

$$\mathbf{H} \approx c \cdot \text{diag}(\|\mathbf{X}[:, j]\|_2^2,\ 1 \leq j \leq d_{\text{in}}) \tag{32}$$

The constant c depends on the loss function and layer properties, but for weight ranking purposes, it can be absorbed into the overall scaling. The inverse Hessian diagonal element is proportional to $[\mathbf{H}^{-1}]_{mm} \propto \frac{1}{\|\mathbf{x}_m\|_2^2}$, which directly leads to the first term in our pruning metric derivation (Equation 12 in the main text). The

empirically determined scaling factor α in our final metric (Equation 15) accounts for both this constant and the relative magnitudes of gradient and activation contributions. Our extensive empirical results across 15+ model families (Tables 1-6 in the main text) demonstrate that the resulting pruning metric performs effectively in practice, suggesting that the diagonal and uniform curvature approximations provide sufficient signal for identifying important weights.