Neural Networks 200 (2026) 108840

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

WLR: Well-conditioned linear reconstruction for retraining-free pruning of LLMs

Siqi Li [a], Jingyang Xiang [a], Jiateng Wei [a], Chengrui Zhu [a], Jiandang Yang [a], Jun Chen [b,c,*], Jian Yang [d], Xiaobin Wei [e], Yunliang Jiang [c,f], Yong Liu [a,*]

[a] Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, 310027, China
[b] National Special Education Resource Center for Children with Autism, Zhejiang Normal University, Hangzhou, 311231, China
[c] School of Computer Science and Technology, Zhejiang Normal University, Jinhua, 321004, China
[d] China Research and Development Academy of Machinery Equipment, Beijing, China
[e] Wasu Media & Network CO., Ltd, China
[f] School of Information Engineering, Huzhou University, Huzhou, 313000, China

ARTICLE INFO

Keywords:
Deep neural networks
Large language model
Model compression
Network pruning

ABSTRACT

Structured pruning is widely recognized as an effective method for reducing the size and computational demands of large language models (LLMs). Traditionally, structured pruning follows a pretrain-prune-retrain framework. Given the high cost of retraining LLMs, recent research has focused on efficient compensation methods to replace the retraining phase. However, many compensation methods introduce additional parameters into the pruned model, which affect deployment and inference. Additionally, many compensation techniques overlook the issue of ill-conditioning caused by outliers in LLMs during the process of solving optimization problems, leading to numerical instability and inefficiency in solutions. To overcome these challenges, we propose Well-conditioned Linear Reconstruction (WLR), a structured pruning compensation method that avoids introducing extra parameters. WLR reconstructs pruned layers using a linear combination of preserved channels while addressing the issue of ill-conditioning during the process. We evaluated our method on LLaMA-V1/V2/V3 and OPT families across multiple language tasks, achieving performance that surpasses the state-of-the-art methods.

1. Introduction

Large Language Models (LLMs) have achieved remarkable performance across various domains. A number of cutting-edge models Brown et al. (2020), Bubeck et al. (2023), Dubey et al. (2024), Touvron et al. (2023a,b), Zhang et al. (2022) are widely applied to various language tasks, including natural language understanding, machine translation, and content generation Mihaylov et al. (2018), Sakaguchi et al. (2021), Zellers et al. (2019), Zhang et al. (2023). However, the success of LLMs comes at the cost of extensive computational resources. Typically, LLMs contain billions of parameters, which means they not only require substantial resources for training but deploying pre-trained models also incurs significant costs. In addition, LLMs also have high memory and processing requirements, which not only increase hardware costs but also raise energy consumption.

To address the above challenges, researchers are exploring model compression techniques to reduce the size and computational requirements of LLMs. A large number of compression methods for LLMs have been proposed, including pruning, quantization, and knowledge distillation Ashkboos et al. (2024), Frantar and Alistarh (2023), Frantar et al. (2022), Hu et al. (2021), Lee et al. (2023), Lin et al. (2023), Xiao et al. (2023). Among the various methods, pruning is considered an effective approach to reduce the number of model parameters and computations. Pruning reduces the size of a model by removing its parameters. Based on the granularity of parameter removal, it can be categorized into two types: structured pruning and unstructured pruning. Since unstructured pruning usually leads to irregular sparse patterns and requires specialized hardware support, we mainly focus on structured pruning. Retraining is the mainstream approach for restoring model performance after structured pruning He et al. (2018), Ma et al. (2023). However, the pretrain-prune-retrain framework typically requires significant computational overhead making it unsuitable for large-scale LLMs. Consequently, much research focuses on retraining-free pruning for LLMs An et al. (2024), Ashkboos et al. (2024), Frantar and Alistarh (2023), Wang et al. (2024a,b).

Retraining-free pruning methods utilize compensation processes instead of retraining to restore the LLMs' performance after pruning. Unlike retraining, the compensation process incurs minimal computational

* Corresponding authors.
E-mail addresses: junc.change@zjnu.edu.cn (J. Chen), yongliu@iipc.zju.edu.cn (Y. Liu).

https://doi.org/10.1016/j.neunet.2026.108840

Available online 15 March 2026

overhead and only requires a small number of samples. Due to its high efficiency, this approach is more suitable for large-scale LLMs. However, the current compensation methods of LLMs pruning have certain problems. First, while recent SoTA methods like SlimGPT Ling et al. (2024) successfully maintain the original parameter count, other representative compensation techniques introduce additional parameters. For example, bias compensation in such as FLuctuation-based Adaptive Structured Pruning (FLAP) An et al. (2024) and Linear Interpolation-based Adaptive Recovery (LIAR) Wang et al. (2024a) introduces a bias to a linear layer that originally had no bias, while SliceGPT Ashkboos et al. (2024) takes this further by adding a new linear layer to each skip connection. Compensation methods that introduce extra parameters go against the core objective of pruning to reduce the number of parameters, and they may also affect subsequent deployment and inference. Secondly, due to the presence of outliers in specific channels of LLMs' activations, existing compensation methods often encounter ill-conditioned matrices when solving the optimization problem, leading to numerical instability and low computational efficiency.

To better address the problems in existing compensation techniques, we propose Well-conditioned Linear Reconstruction (WLR), a compensation method for retraining-free structured pruning of LLMs that does not introduce additional parameters. The main contributions are summarized as follows:

- We propose a compensation method for LLM pruning that does not introduce additional parameters, and we provide a detailed mathematical derivation. Specifically, we compensate each pruned channel by a linear combination of the preserved channels.
- We improve the optimization problem in the compensation method by transforming the ill-conditioning issue problem by outliers into a well-conditioned one, making it easier to solve. We provide a mathematical proof of the existence of solutions to the optimization problem.
- We conduct multiple experiments on LLMs including OPT Zhang et al. (2022), LLaMA-V1 Touvron et al. (2023a), LLaMA-V2 Touvron et al. (2023b) and LLaMA-V3 Dubey et al. (2024) families, covering various language benchmarks Bisk et al. (2020), Clark et al. (2018), Mihaylov et al. (2018), Sakaguchi et al. (2021), Wang et al. (2019), Zellers et al. (2019). Compared to state-of-the-art (SOTA) retraining-free methods, our approach demonstrates superior performance.

2. Related works

2.1. LLM Pruning

Many studies indicate that pruning is an effective way to reduce the size of large language models An et al. (2024), Ashkboos et al. (2024), Frantar and Alistarh (2023), Ma et al. (2023), Wang et al. (2024a). SparseGPT proposed by Frantar and Alistarh (2023) uses the inverse Hessian matrix for unstructured pruning, followed by weight updates. LLM-Pruner Ma et al. (2023) applies structured pruning based on gradient information and incorporates fine-tuning through Low-Rank Adaptation (LoRA) Hu et al. (2021). SliceGPT Ashkboos et al. (2024) modifies LayerNorm to RMSNorm and performs transformations on each block using computational invariance before executing pruning. FLAP An et al. (2024) introduces a framework with global structure search and baseline bias compensation, while LIAR Wang et al. (2024a) enhances FLAP's compensation method and generalizes it across various pruning criteria.

However, some methods tend to introduce additional parameters. For instance, FLAP and LIAR's baseline bias compensation adds bias to linear layers that initially have none, and SliceGPT further complicates the model by introducing a new linear layer to each skip connection. We hold the opinion that these compensation techniques, which require the introduction of new parameters, undermine the core objective of pruning and may complicate deployment and inference. This is one challenge. Separately, methods like LIAR overlook the issue of ill-conditioned problems during reconstruction, resulting in numerical instability and decreased computational efficiency. Our approach addresses both of these limitations.

2.2. Compensation for retraining-free pruning

In the field of convolutional neural network (CNN) pruning, several methods aim to reduce dependency on fine-tuning and dataset, thus eliminating the need for extensive retraining He et al. (2017), Luo et al. (2017), Mussay et al. (2020). ThiNet Luo et al. (2017) frames channel pruning as an optimization problem focused on minimizing layer-wise reconstruction error and introduces a solution that minimizes dataset usage. *He* et al. He et al. (2017) propose a channel selection method using Least Absolute Shrinkage and Selection Operator (LASSO) regression, followed by reconstructing the output feature maps via least squares. Similarly, *Mussay* et al. Mussay et al. (2020) introduce pruning criteria based on Coreset, aiming at reducing the high costs associated with retraining. There are also approaches that enable network pruning without relying on any data or fine-tuning Bai et al. (2023), Kim et al. (2020), Srinivas and Babu (2015), Yvinec et al. (2021). The method introduced by *Srinivas* et al. Srinivas and Babu (2015) is the first to present data-free techniques for eliminating redundant neurons. Neuron Merging Kim et al. (2020) proposes to replace pruned channels with similar preserved ones. *Yvinec* et al. Yvinec et al. (2021) identifies redundancies among neurons using adaptive data-free scalar hashing, allowing for both pruning and recovery. Unified Data-Free Compression (UFDC) Bai et al. (2023) offers a unified data-free compression framework that simultaneously handles pruning, quantization, and their associated compensations.

The above methods are designed for CNN compression, considering only the structure of convolutional layers followed by batch normalization layers, and they do not support the complex multi-head self-attention blocks in LLMs. However, some of these ideas are insightful for retraining-free compensation in LLM pruning. For instance, the assumption proposed in UFDC suggests that a pruned channel can be compensated by a linear combination of the preserved channels.

3. Problem formulation

3.1. Background and notation

Consider a Large Language Model (LLM) consisting of L layers. Each layer consists of a multi-head self-attention (MHA) block and a feed-forward network (FFN) block, which are connected by a layer normalization (LayerNorm) layer. As shown in the left part of Fig. 1.

For the ℓ-th layer, the MHA block can be represented as

$$\begin{cases} X^{\ell}_{head} = \mathrm{Attn}\left(W^{q}X^{\ell}, W^{k}X^{\ell}, W^{v}X^{\ell}\right) \\ X^{\ell+1} = W^{out}X^{\ell}_{head} + b^{out} \end{cases} \tag{1}$$

where $X^{\ell} \in \mathbb{R}^{D\times N}$ is the input tensor, D is the embedding dimension, and N is the sequence length. Attn represents the multi-head self-attention operation. W^{q}, W^{k}, and W^{v} denote the weight of the Query, Key, and Value. W^{out} and b^{out} are the weights and bias of the output projection layer, respectively.

In addition, the FFN block of the ℓ-th layer can be represented as:

$$\begin{cases} X^{\ell}_{1} = W^{1}X^{\ell} + b^{1} \\ X^{\ell+1} = W^{2}\sigma\left(X^{\ell}_{1}\right) + b^{2} \end{cases} \tag{2}$$

where σ represents the activation function, W^{1} and W^{2} denote the weights of the two linear layers, and b^{1} and b^{2} denote the bias of the two linear layers. Some LLMs adopt a gated structure at the FFN layer, which can be represented as:

$$\begin{cases} X^{\ell}_{1} = W^{1}X^{\ell} + b^{1} \\ X^{\ell}_{2} = W^{2}X^{\ell} \\ X^{\ell+1} = W^{3}\sigma\left(X^{\ell}_{1} \circ X^{\ell}_{2}\right) + b^{3} \end{cases} \tag{3}$$

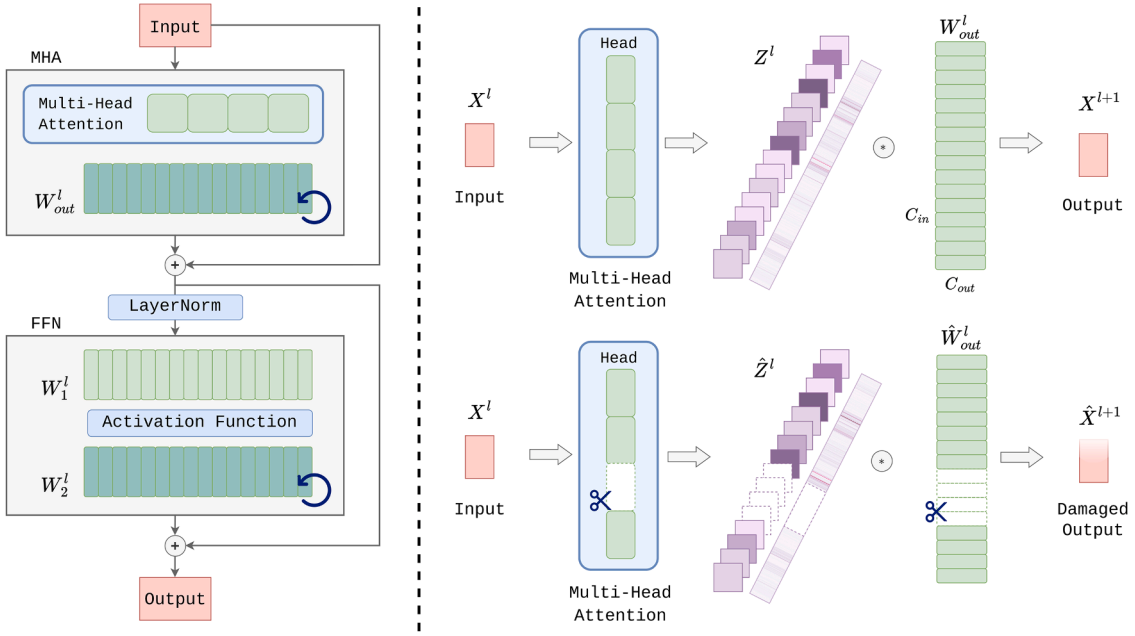

Fig. 1. Left: The ℓ-th layer of the large language model. A layer consists of a MHA block and a FFN block. Right: Comparison before and after pruning an entire head in the MHA block. After pruning one head, the corresponding input channels of the output projection layer are pruned, leading to a damaged output $\hat{X}^{(\ell+1)}$.

where $\circ$ denotes the element-wise product.

Combining Eq. (1)–Eq. (3), we can summarize both the MHA and FFN in the following:

$$\begin{cases} Z^{\ell} = sublayer(X^{\ell}) \\ X^{\ell+1} = W^{out}\ Z^{\ell} + b^{out} \end{cases}. \tag{4}$$

The $sublayer$ denotes the operation before the last linear layer $W^{out} \in \mathbb{R}^{C_{out} \times C_{in}}$ in the block, in the case of the MHA layer it is a multi-head self-attention operation, and in the case of the FFN layer it is a linear layer or a gated operation. Z^{ℓ} has the dimensions $Z^{\ell} \in \mathbb{R}^{C_{in} \times N}$.

To perform structured pruning on the layer represented by Eq. (4), specifically on the output channels of $sublayer$ and the input channels of W^{out}, the pruned layer can be represented as follows:

$$\begin{cases} \hat{Z}^{\ell} = sublayer_{prune}(X^{\ell}) \\ \hat{X}^{\ell+1} = \hat{W}^{out}\hat{Z}^{\ell} + b^{out} \approx X^{\ell+1} \end{cases} \tag{5}$$

After pruning, the output of pruned sublayer $\hat{Z}^{\ell}$ has the dimensions $\hat{Z}^{\ell} \in \mathbb{R}^{\hat{C}_{in} \times N}$, where $\hat{C}_{in}$ denotes the number of preserved channels. At this time, the weights of the last linear layer are represented as $\hat{W}^{out} \in \mathbb{R}^{C_{out} \times \hat{C}_{in}}$. As shown in the right part of Fig. 1.

During structured pruning, the preserved features $\hat{Z}^{\ell}$ often exhibit strong linear dependence, which makes the subsequent reconstruction problem ill-conditioned. For a basic linear problem $Ax = b$, if small perturbations in A or b result in large changes in the solution x, the problem is considered ill-conditioned. Ill-conditioned systems often lead to numerical instability and difficulties in solving. One key cause of ill-conditioning is the high linear dependence among the column vectors of the coefficient matrix, which means that the features represented by these vectors are nearly redundant, causing ambiguity in determining unique solutions. To make this more precise, let $\hat{Z}^{\ell} = Z \in \mathbb{R}^{N \times m}$ be the matrix whose columns correspond to the features. Using the singular value decomposition (SVD), we write

$$Z = U\Sigma V^{\top}, \qquad \Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_r), \quad \sigma_1 \geq \cdots \geq \sigma_r > 0,$$

where $r = \mathrm{rank}(Z)$. The spectral condition number is

$$\mathrm{cond}(Z) = \frac{\sigma_1}{\sigma_r}.$$

When the columns of Z are highly linearly dependent (e.g., nearly collinear), the smallest singular value σ_r approaches zero, making $\mathrm{cond}(Z)$ very large. Geometrically, this means that certain directions in $\mathbb{R}^m$ are mapped almost to zero, so different coefficient vectors produce nearly identical linear combinations of the columns. In least-squares formulations, the Gram matrix $G = Z^{\top}Z$ has eigenvalues $\{\sigma_i^2\}$, giving

$$\mathrm{cond}(G) = \frac{\sigma_1^2}{\sigma_r^2} = \mathrm{cond}(Z)^2,$$

which further amplifies the ill-conditioning. Moreover, standard perturbation bounds for linear systems,

$$\frac{\|\Delta x\|_2}{\|x\|_2} \lesssim \mathrm{cond}(A)\left(\frac{\|\Delta A\|_2}{\|A\|_2} + \frac{\|\Delta b\|_2}{\|b\|_2}\right),$$

show that the relative error in the solution grows proportionally with the condition number. Thus, strong column dependence leads to a small σ_r, which in turn results in a large $\mathrm{cond}(G)$, implying that the estimated coefficients are extremely sensitive to perturbations and the system is ill-conditioned.

3.2. Reconstruction assumption

Drawing inspiration from the data-free compression technique UDFC Bai et al. (2023), our assumption is that channels damaged by pruning can be restored through a linear combination of the preserved channels. Let $\mathcal{P}$ represent the set of indexes corresponding to the pruned input channels of $W^{out}_{:,:}$. Our assumptions can be expressed as follows, for $\forall p \in \mathcal{P}$, it holds that:

$$W^{out}_{:,p} \approx \sum_{j=1, j \notin \mathcal{P}}^{C_{in}} s_{p,j} \times W^{out}_{:,j}, \quad \forall p \in \mathcal{P} \tag{6}$$

where $s_{p,j}$ is the scaling factor between the j-th channel and the p-th channel.

Through such a reconstruction, the weight of each preserved channel $\tilde{W}^{out}_{:,j}$ consists of its original weight $W^{out}_{:,j}$ plus the sum of the product of each pruned channel weight $W^{out}_{:,p}$ and corresponding scaling factor $s_{p,j}$. This can be expressed as follows:

$$\tilde{W}^{out}_{:,j} = W^{out}_{:,j} + \sum_{p \in \mathcal{P}} s_{p,j} \times W^{out}_{:,p}, \quad \forall j \in [1, C_{in}], \quad j \notin \mathcal{P} \tag{7}$$

After reconstructing the weights of the last linear layer, we can obtain the block output $\tilde{X}^{\ell+1}$ as follows:

$$\begin{cases} \hat{Z}^{\ell} = sublayer_{prune}\left(X^{\ell}\right) \\ \tilde{X}^{\ell+1} = \tilde{W}^{out}\hat{Z}^{\ell} + b^{out} \end{cases} \tag{8}$$

For the reconstruction problem, our objective is to ensure that the output of the reconstructed block closely approximates the original output $X^{\ell+1}$. Under the assumption proposed, the recovery of the model's performance can be formulated as an optimization problem, which can be expressed as follows:

$$\min_{\mathbf{s}} \left\| X^{\ell+1} - \tilde{X}^{\ell+1} \right\|_2^2 \tag{9}$$

Our objective is to find an appropriate matrix $\mathbf{s} = [s_{p,j}]$ that minimizes the error between the output of the reconstructed output and the original output.

4. Methodology

4.1. Layer-wise linear reconstruction

Before the pruning-reconstruction process, the k-th output channel of the ℓ-th block can be represented as:

$$X_k^{\ell+1} = \sum_{j=1}^{C_{in}} W_{k,j}^{out} Z_j^{\ell} + b_k^{out} \quad \forall k \in [1, C_{out}] \tag{10}$$

where $Z_j^{\ell} \in \mathbb{R}^N$ represents the input corresponding to the j-th input channel.

After pruning the input channels in set $\mathcal{P}$ and reconstructing with Eq. (7), the k-th output channel can be represented as:

$$\tilde{X}_k^{\ell+1} = \sum_{j=1, j\notin\mathcal{P}}^{C_{in}} \left(W_{k,j}^{out} + \sum_{p\in\mathcal{P}} s_{p,j} \times W_{k,p}^{out} \right) Z_j^{\ell} + b_k^{out} \quad \forall k \in [1, C_{out}] \tag{11}$$

The reconstruction error ℓ_{re} of the k-th channel can be defined as:

$$\begin{aligned} \ell_{re} &= \left\| X_k^{\ell+1} - \tilde{X}_k^{\ell+1} \right\|_2^2 = \left\| \sum_{p\in\mathcal{P}} W_{k,p}^{out} Z_p^{\ell} - \sum_{j=1, j\notin\mathcal{P}}^{C_{in}} \sum_{p\in\mathcal{P}} s_{p,j} \times W_{k,p}^{out} Z_j^{\ell} \right\|_2^2 \\ &= \left\| \sum_{p\in\mathcal{P}} W_{k,p}^{out} Z_p^{\ell} - \sum_{p\in\mathcal{P}} W_{k,p}^{out} \sum_{j=1, j\notin\mathcal{P}}^{C_{in}} s_{p,j} \times Z_j^{\ell} \right\|_2^2 \\ &= \left\| \sum_{p\in\mathcal{P}} W_{k,p}^{out} \left(Z_p^{\ell} - \sum_{j=1, j\notin\mathcal{P}}^{C_{in}} s_{p,j} \times Z_j^{\ell} \right) \right\|_2^2 \end{aligned} \tag{12}$$

Our objective is to minimize ℓ_{re}, where minimizing ℓ_{re} is equivalent to minimizing ℓ_{re}^p for each $p \in \mathcal{P}$. The term ℓ_{re}^p is defined as follows:

$$\ell_{re}^p = \left\| W_{k,p}^{out} \left(Z_p^{\ell} - \sum_{j=1, j\notin\mathcal{P}}^{C_{in}} s_{p,j} \times Z_j^{\ell} \right) \right\|_2^2 \tag{13}$$

Note that pruning does not change $W_{k,p}^{out}$. Therefore, we can further define the reconstruction error as:

$$\ell_{re}^p = \left\| Z_p^{\ell} - \sum_{j=1, j\notin\mathcal{P}}^{C_{in}} s_{p,j} \times Z_j^{\ell} \right\|_2^2 \tag{14}$$

For simplicity, we define:

$$\begin{cases} \mathbf{Z}_p = [Z_p^{\ell}] & p \in \mathcal{P} \\ \mathbf{Z} = [Z_j^{\ell}] & j \in [1, C_{in}], j \notin \mathcal{P} \\ \mathbf{s}_p = [s_{p,j}] & p \in \mathcal{P}, j \in [1, C_{in}], j \notin \mathcal{P} \end{cases} \tag{15}$$

At this point, the optimization problem in Eq. (9) transforms into the problem of minimizing the error ℓ_{re}^p in Eq. (14), which can be expressed as:

$$\min_{\mathbf{s}_p} \left\| \mathbf{Z}_p - \mathbf{s}_p \mathbf{Z} \right\|_2^2 \tag{16}$$

Given that the activation of the LLM exhibits outliers in a small number of specific channels, which have a relatively large impact on the performance of the model Sun et al. (2023). Therefore, we choose the $\ell 2$-norm of the input activation as pruning criteria to retain more outliers. At this point, the index set of the pruned channels can be represented as:

$$\mathcal{P} = [1, C_{in}] - \left\{ \arg\operatorname*{Top}_j K \left(\| Z_j^{\ell} \|_2, \hat{C}_{in} \right) \right\} \tag{17}$$

Solving Eq. (16) presents a challenge. As shown in Fig. 2, the matrix $\mathbf{Z}$ contains outliers with large magnitudes in a few specific channels. This causes the column vectors in $\mathbf{Z}$ to form very small angles in the multi-dimensional space, indicating high linear dependence among the column vectors, as explained in Fig. 3. As mentioned in Section 3.1, the high linear dependence among the column vectors leads to ill-conditioning. Consequently, the matrix $\mathbf{Z}$ is ill-conditioned, making it difficult to effectively solve the optimization problem presented in Eq. (16).

4.2. Reconstruct without outliers

To address the issue of $\mathbf{Z}$ being ill-conditioned, we propose to eliminate outliers from the matrix $\mathbf{Z}$. As shown in Fig. 3, once the outliers are removed from $\mathbf{Z}$, the angles between the column vectors in $\mathbf{Z}$ increase, thereby reducing the linear dependence between the column vectors and improving the condition of the matrix. Specifically, we reconstruct the pruned channels with a linear combination of the preserved non-outlier channels. Since outliers exist only in a small number of specific channels, the number of non-outlier channels is sufficient for reconstruction. At this point, Eq. (6) transforms into the following expression:

$$W_{:,p}^{out} \approx \sum_{j=1, j\notin\mathcal{P}\cup\mathcal{O}}^{C_{in}} s_{p,j} \times W_{:,j}^{out}, \quad \forall p \in \mathcal{P} \tag{18}$$

where $\mathcal{O}$ is a set of indexes corresponding to the outliers. $\mathcal{O}$ is defined as:

$$\mathcal{O} = \left\{ j \;\middle|\; \| Z_j \|_2 > \; M \times \frac{\sum_{j=1}^{C_{in}} \| Z_j \|_2}{C_{in}} \right\} \tag{19}$$

where $M > 0$ is a hyperparameter for identifying outliers. We recognize a channel as an outlier channel when its $\ell 2$-norm M times is greater than the mean value. Since our pruning criteria retain the outliers within the channels, we have $\mathcal{P} \cap \mathcal{O} = \emptyset$.

The weights of the preserved channels after reconstruction can be expressed as follows:

$$\tilde{W}_{:,j}^{out} = \begin{cases} W_{:,j}^{out} + \sum_{p\in\mathcal{P}} s_{p,j} \times W_{:,p}^{out} & j \notin \mathcal{O} \\ W_{:,j}^{out} & j \in \mathcal{O} \end{cases}, \quad \forall j \in [1, C_{in}], \quad j \notin \mathcal{P} \tag{20}$$

After reconstructing with Eq. (18), we can obtain the k-th output channel as follow:

$$\bar{X}_k^{\ell+1} = \sum_{j\in\mathcal{O}} W_{k,j}^{out} Z_j^{\ell} + \sum_{j=1, j\notin\mathcal{P}\cup\mathcal{O}}^{C_{in}} \left(W_{k,j}^{out} + \sum_{p\in\mathcal{P}} s_{p,j} \times W_{k,p}^{out} \right) Z_j^{\ell} + b_k^{out} \tag{21}$$

The reconstruction error $\bar{\ell}_{re}$ of the k-th channel can be defined as:

$$\bar{\ell}_{re} = \left\| X_k^{\ell+1} - \bar{X}_k^{\ell+1} \right\|_2^2 = \left\| \sum_{p\in\mathcal{P}} W_{k,p}^{out} \left(Z_p^{\ell} - \sum_{j=1, j\notin\mathcal{P}\cup\mathcal{O}}^{C_{in}} s_{p,j} \times Z_j^{\ell} \right) \right\|_2^2 \tag{22}$$

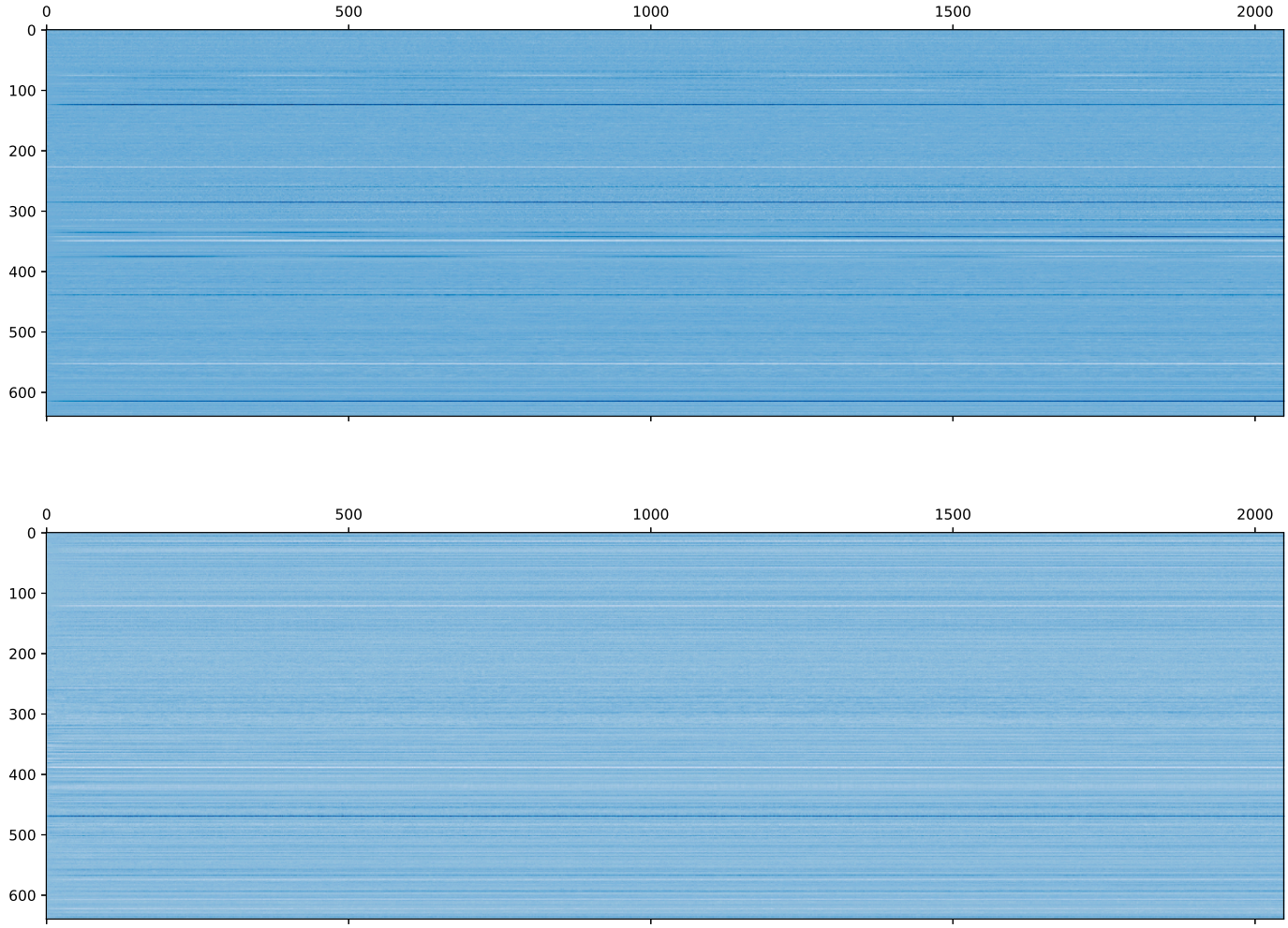

Fig. 2. The matrix **Z** for specific MHA blocks of the OPT-125M model using 64 calibration samples, under a uniform pruning rate of 20%. Top: Matrix **Z** for the MHA block in the 8-th layer. Bottom: Matrix **Z** for the MHA block in the 11-th layer.

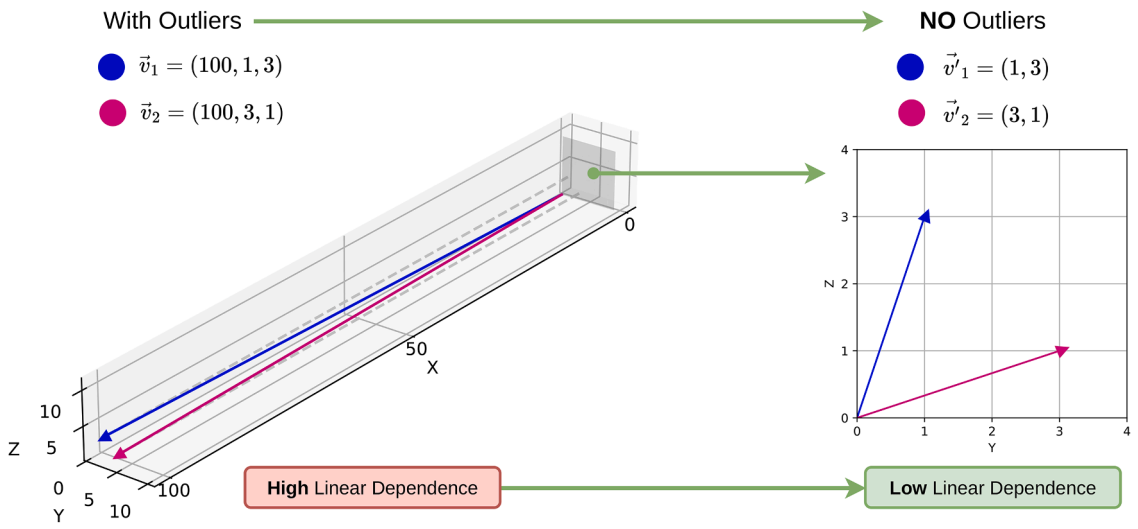

Fig. 3. The impact of outliers on the degree of linear dependence between vectors. As shown in the left, with outliers, the two vectors $\vec{v}_1$ and $\vec{v}_2$ exhibit a high degree of linear dependence. When the outliers are removed, as shown on the right, the linear dependence between vectors $\vec{v'}_1$ and $\vec{v'}_2$ decreases.

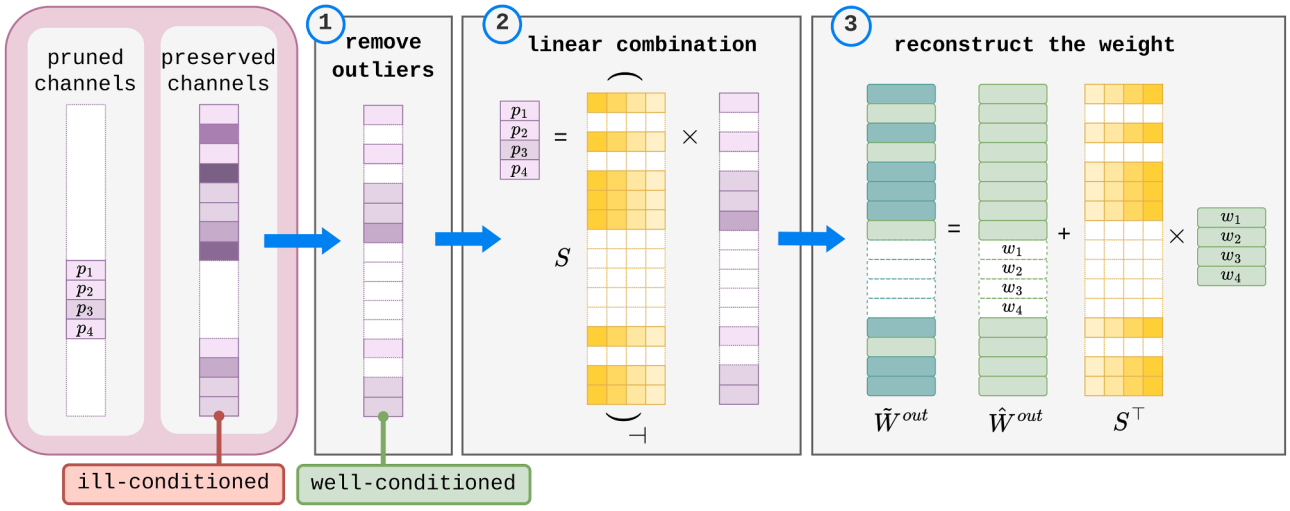

Fig. 4. Overview of the WLR method. During the pruning phase, we apply a pruning criteria that retains more outliers. The activations of the preserved channels form an ill-conditioned matrix, so we first remove the outliers to transform it into a well-conditioned matrix. Next, for each pruned channel (p_1-p_4), we obtain a scaling factor by solving the optimization problem in Eq. (26) (resulting in the four columns of S). At this point, each pruned channel can be represented as a linear combination of the preserved non-outlier channels, with the coefficients given by the scaling factors. Finally, we reconstruct the weights of the last linear layer with the scale factors S. The reconstructed weights $\tilde{W}^{out}$ consist of the original pruned weights $\hat{W}^{out}$ plus the weights of each pruned channel (w_1-w_4) multiplied by the corresponding scale factor.

At this point, minimizing $\bar{\ell}_{re}$ is equivalent to minimizing $\bar{\ell}^p_{re}$ for each $p \in \mathcal{P}$. $\bar{\ell}^p_{re}$ is defined as:

$$\bar{\ell}^p_{re} = \left\| W^{out}_{k,p} \left(Z^{\ell}_p - \sum_{j=1, j \notin \mathcal{P} \cup \mathcal{O}}^{C_{in}} s_{p,j} \times Z^{\ell}_j \right) \right\|_2^2 \tag{23}$$

Since pruning does not change $W^{out}_{k,p}$, we can further define the reconstruction error as:

$$\bar{\ell}^p_{re} = \left\| Z^{\ell}_p - \sum_{j=1, j \notin \mathcal{P} \cup \mathcal{O}}^{C_{in}} s_{p,j} \times Z^{\ell}_j \right\|_2^2 \tag{24}$$

For simplicity, we define $\bar{\mathbf{Z}} = [Z^{\ell}_j]$, where $j \in [1, C_{in}], j \notin \mathcal{P} \cup \mathcal{O}$. At this point, the optimization problem can be represented as:

$$\min_{\mathbf{s}_p} \left\| \mathbf{Z}_p - \mathbf{s}_p \bar{\mathbf{Z}} \right\|_2^2 \tag{25}$$

Since we remove the outliers in $\mathbf{Z}$, the matrix $\bar{\mathbf{Z}}$ is well-conditioned at this point. Therefore, we can solve the optimization problem in Eq. (25) by Ridge regression:

$$\min_{\mathbf{s}_p} \left\| \mathbf{Z}_p - \mathbf{s}_p \bar{\mathbf{Z}} \right\|_2^2 + \lambda \left\| \mathbf{s}_p \right\|_2^2 \tag{26}$$

where λ is a non-negative penalty coefficient.

The reconstruction error can be simplified as $\bar{\ell}^p_{re} = (\mathbf{Z}_p - \mathbf{s}_p \bar{\mathbf{Z}})^\top (\mathbf{Z}_p - \mathbf{s}_p \bar{\mathbf{Z}}) + \lambda \mathbf{s}_p^\top \mathbf{s}_p$. The first and second derivative of the $\mathbf{s}_p$ is:

$$\begin{cases} \dfrac{\partial \bar{\ell}^p_{re}}{\partial \mathbf{s}_p} = -2\bar{\mathbf{Z}}^\top \mathbf{Z}_p + 2\mathbf{s}_p \left(\bar{\mathbf{Z}}^\top \bar{\mathbf{Z}} + \lambda \mathbf{I} \right) \\ \dfrac{\partial^2 \bar{\ell}^p_{re}}{\partial^2 \mathbf{s}_p} = 2\bar{\mathbf{Z}}^\top \bar{\mathbf{Z}} + 2\lambda \mathbf{I} \end{cases} \tag{27}$$

It can be seen that ℓ_{re} is a convex function as $\frac{\partial^2 \bar{\ell}^p_{re}}{\partial^2 \mathbf{s}_p} > 0$. And there exists a unique optimal solution s_p such that $\frac{\partial \bar{\ell}^p_{re}}{\partial \mathbf{s}_p} = 0$.

The overview of our method is illustrated in Fig. 4. Before performing compensation, we compute the input activations Z^{ℓ} of the ℓ-th layer with a reconstruction set $\mathcal{R}^{\ell} = \{Z^m\}_{m=1}^{M}$. When the model is relatively small (less than 3B), Z^{ℓ} is defined as $Z^{\ell} = \text{Concat}[Z^1, Z^2, \cdots, Z^M]$, otherwise Z^{ℓ} is the average of $\mathcal{R}^{\ell}$. The Well-conditioned Linear Reconstruction (WLR) can be divided into three steps. First, we achieve a well-conditioned optimization problem in Eq. (26) by eliminating outliers from the activation of the preserved channels. Second, we solve the optimization problem in Eq. (26) using Ridge regression to obtain a scale factor $\mathbf{s}_p$ for each pruned channel. Finally, we reconstruct the weights using the scale factors $\mathbf{s}_p$ according to Eq. (20).

5. Experiments

5.1. Experimental configurations

5.1.1. Models and datasets

We evaluate the performance of WLR across a series of LLMs, including the OPT Zhang et al. (2022), LLaMA-V1 Touvron et al. (2023a), LLaMA-V2 Touvron et al. (2023b), and LLaMA-V3 Dubey et al. (2024) model families. Our evaluation aligns with the existing LLM pruning methods An et al. (2024), Ashkboos et al. (2024), Wang et al. (2024a), including perplexity (PPL) assessment on the WikiText Merity et al. (2016) validation and evaluation on seven common-sense benchmarks (BoolQ Wang et al. (2019), OpenbookQA Mihaylov et al. (2018), WinoGrande Sakaguchi et al. (2021), HellaSwag Zellers et al. (2019), PIQA Bisk et al. (2020), ARC-e, and ARC-c Clark et al. (2018)) in zero-shot setting consistent with the LM-Evaluation-Harness Gao et al. (2021). To ensure fairness and consistency, all evaluations were conducted under the same environment using LM-Evaluation-Harness v0.0.1.

5.1.2. Comparison method baselines

To validate the effectiveness of our compensation method, we compare WLR with various SOTA retraining-free, structured pruning techniques, including Wanda-sp Sun et al. (2023), SliceGPT Ashkboos et al. (2024), FLAP An et al. (2024) and LIAR Wang et al. (2024a). In addition, we also compare the singular value decomposition-based method SVD-LLM Wang et al. (2024b). For all these baselines, we used their official open-source implementations and followed the hyperparameters provided in the original papers to ensure a fair comparison under our unified evaluation setup.

Table 1
WikiText perplexity of pruning methods for LLaMA-V2-7B/13B/70B at 20% and 50% sparsity. * indicates the experimental results we reproduced using the open-source code. The bolded results indicate the best performance.

Methods	LLaMA-V2-20%			LLaMA-V2-30%		LLaMA-V2-50%	
	7B	13B	70B	7B	13B	7B	13B
Dense	5.47	4.88	3.32	5.47	4.88	5.47	4.88
Wanda-sp * Sun et al. (2023)	9.07	6.05	5.02	9.14	12.78	71.44	24.57
SliceGPT * Ashkboos et al. (2024)	6.84	6.06	4.25	**7.55**	7.44	**21.11**	17.57
SliceGPT(EQP) * Ashkboos et al. (2024)	8.18	7.12	5.15	10.80	9.17	35.74	29.20
FLAP * An et al. (2024)	7.15	6.31	4.12	8.85	7.56	43.07	27.43
SVD-LLM * Wang et al. (2024b)	8.38	6.66	4.66	10.66	8.00	33.38	18.72
SlimGPT * Ling et al. (2024)	12.56	11.14	6.43	13.49	12.02	33.67	30.39
Ours	**6.80**	**5.96**	**4.04**	7.94	**7.26**	33.05	**14.93**

Table 2
WikiText perplexity of pruning methods for OPT-125M/1.3B/2.7B/6.7B/13B/ 30B at 20% and 30% sparsity. * indicates the experimental results we reproduced using the open-source code. The dash '-' represents results that could not be reproduced with the open-source code. The **bolded** results indicate the best performance, while the underlined ones signify the second-best performance.

Methods	Pruning Ratio	OPT					
		125M	1.3B	2.7B	6.7B	13B	30B
Dense	0%	27.64	14.61	12.46	10.85	10.12	9.56
Wanda-sp * Sun et al. (2023)	20%	38.10	21.29	17.92	15.52	14.51	12.23
SliceGPT * Ashkboos et al. (2024)		34.10	16.51	13.89	11.60	**10.71**	**9.92**
SliceGPT(EQP) * Ashkboos et al. (2024)		112.17	21.45	16.94	12.88	11.54	10.38
FLAP * An et al. (2024)		34.45	17.37	15.38	12.79	13.17	12.80
SVD-LLM * Wang et al. (2024b)		38.86	17.82	15.22	12.06	-	-
Ours		**31.17**	**16.42**	**13.66**	**11.27**	10.96	10.05
Wanda-sp * Sun et al. (2023)	30%	51.39	30.99	23.39	22.38	19.54	15.45
SliceGPT * Ashkboos et al. (2024)		44.23	19.58	16.31	12.79	**11.49**	10.39
SliceGPT(EQP) * Ashkboos et al. (2024)		175.02	28.87	22.31	14.87	12.66	11.02
FLAP * An et al. (2024)		40.05	20.77	18.31	15.42	16.01	16.49
SVD-LLM * Wang et al. (2024b)		49.13	20.68	17.79	13.06	-	-
Ours		**36.41**	**18.65**	**14.85**	**12.70**	12.11	**10.34**

5.1.3. Experimental setups

We utilize Pytorch Paszke et al. (2019) and Huggingface Transformers library Wolf et al. (2019) to implement our method. Specifically, we use the CuPy library to solve Ridge regression in Eq. (26), with the hyperparameter λ set to 0.9 and M set to 3. Moreover, we set the calibration set used to calculate the pruning criteria to contain 64 samples, while the reconstruction set used for compensation contains 32 samples. The samples are sampled from the original training set, with each sample containing 2048 tokens.

5.2. Language modeling performance

To evaluate the performance of WLR, we compare it with the SOTA methods on language modeling tasks. Table 1 and Table 2 present the pruning results on the LLaMA-2 and OPT families, respectively. Since SliceGPT introduces a substantial number of additional parameters, its final model ends up containing more parameters. For instance, with a 20% sparsity rate on LLaMA-2-70B, the final model pruned by other methods contains 55.7B parameters, while the SliceGPT result comprises 62.2B. Therefore, we also include a comparison of SliceGPT (EQP), in SliceGPT (EQP), we adjust the sparsity rate to ensure that the final number of parameters is the same as that of other methods. From Table 1, it can be seen that our method achieves better results on the LLaMA-V2 families compared to other methods, with lower PPL on WikiText. In Table 2, our method achieves the best results on most OPT models with various sparsity rates (20% and 30%). On a few models, our method yields the second-best performance, only trailing the SliceGPT method, which has more parameters.

5.3. Zero-shot tasks performance

We evaluate the zero-shot capabilities of our method on LLaMA-V1-7B and LLaMA-V2-7B on seven downstream tasks, as shown in Table 3 and Table 4. 'Ours' represents the experimental results obtained using calibration and reconstruction sets sampled from the downstream task dataset, while 'Ours (C4)' represents the results obtained using samples from the Colossal Clean Crawled Corpus (C4) Raffel et al. (2020) training set. The number of samples is the same as described in Section 5.1.3. As seen in Table 4 and Table 3, our method achieves better average accuracy than other methods under both calibration set selections. Besides, our method attains the best or second-best zero-shot performance in most downstream tasks.

5.4. Ablation study

5.4.1. Pruning criteria

We apply different pruning criteria to our method for ablation experiments, as shown in Table 5. The four pruning criteria listed from top to bottom in Table 5 are Wanda-sp, input activation, weight magnitude, and the fluctuation metric from FLAP. The fluctuation metric in FLAP is global pruning criteria. We employ the corresponding adaptive structural search from FLAP, resulting in a non-uniform pruning strategy. Although the FLAP method, with adaptive structural search, achieves good results before compensation, its performance after compensation is inferior to two pruning criteria that consider retaining outliers: Wanda-sp and input activation. Wanda-sp and input activation achieve the best and second-best results. Compared to simple pruning without compensation, our method significantly improved performance. This

Table 3
Zero-shot performance of the pruned LLaMA-V1-7B at 20% sparsity. * indicates the experimental results we reproduced with the open-source code. The **bolded** results indicate the best performance, while the underlined ones signify the second-best performance.

Method	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Ave.
Dense	75.02	79.16	76.20	70.09	72.85	44.62	44.40	66.05
Wanda-sp * Sun et al. (2023)	74.07	77.09	55.11	64.80	65.57	39.42	40.60	59.52
FLAP * An et al. (2024)	71.44	75.14	67.71	67.40	67.21	37.46	42.00	61.19
SliceGPT * Ashkboos et al. (2024)	58.99	69.86	59.45	68.43	62.37	36.60	37.40	56.15
LIAR Wang et al. (2024a)	74.37	77.69	70.00	67.01	68.52	40.61	42.40	62.94
SlimGPT * Ling et al. (2024)	**75.34**	77.09	**71.51**	67.32	68.69	40.11	42.60	63.23
Ours (C4)	74.84	77.58	70.08	**69.33**	70.16	39.08	41.60	63.23
Ours	74.14	**77.80**	70.04	68.09	**70.68**	**40.63**	**43.20**	**63.51**

Table 4
Zero-shot performance of the pruned LLaMA-V2-7B at 20% sparsity. * indicates the experimental results we reproduced with the open-source code. The **bolded** results indicate the best performance, while the underlined ones signify the second-best performance.

LLaMA-V2-7B	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Ave.
Dense	77.74	79.05	76.02	68.98	74.58	46.25	44.20	66.69
Wanda-sp * Sun et al. (2023)	69.60	77.09	72.07	63.38	65.61	42.15	40.40	61.47
FLAP * An et al. (2024)	63.06	72.96	64.77	63.93	57.49	33.45	40.00	56.52
SliceGPT * Ashkboos et al. (2024)	53.82	69.53	59.26	64.56	57.45	35.15	40.40	54.31
Ours (C4)	72.42	77.26	72.13	**66.38**	70.12	**42.32**	41.00	63.09
Ours	**73.61**	**77.80**	**73.06**	66.22	**72.98**	41.21	**41.40**	**63.75**

Table 5
Comparison of the PPL before and after WLR compensation for LLaMA-V2-7B under different pruning criteria. 'Uniform' indicates whether the pruning rate is uniform across layers. 'Outlier' indicates whether the pruning criteria preserves outliers in the activations. 'Prune' represents the results of pruning only, while 'Ours' represents the results after applying our compensation method based on pruning. The **bolded** results indicate the best performance, while the underlined ones signify the second-best performance.

Pruning criteria	Uniform	Outlier	20%		30%	
			prune	Ours	prune	Ours
$\sum_{i=1}^{C_{out}} \lvert W_{ij}\rvert \cdot \lVert X_j\rVert_2$ Sun et al. (2023)	✓	✓	9.07	6.86	13.16	**8.92**
$\lVert X_j\rVert_2$	✓	✓	8.57	**6.80**	10.58	9.26
$\sum_{i=1}^{C_{out}} \lvert W_{ij}\rvert$	✓	✗	Nan	877.92	Nan	3151.53
FLAP An et al. (2024)	✗	✗	7.26	13.77	9.45	52.86

Table 6
PPL of LLaMA-V2-7B and LLaMA-V3-8B under 20% sparsity for different hyperparameter M values. 'Outlier' indicates results where outliers are not removed during reconstruction.

	LLaMA-V2-7B	LLaMA-V3-8B
Dense	5.47	6.13
Outlier	7.14	10.54
$M = 3$	6.80	9.76
$M = 5$	6.83	9.94
$M = 7$	6.89	9.88

demonstrates the generalization capability of our method, as it consistently achieves superior results when pruning criteria preserve outliers.

5.4.2. Outliers and condition numbers

We conduct an ablation study on the hyperparameter M in Eq. (19), where M is used to identify outlier channels. A channel is considered an outlier when its $\ell 2$-norm is M times greater than the mean value of $\ell 2$-norm of all channels of this layer. Table 6 shows the PPL results for the LLaMA-V2-7B and LLaMA-V3-8B models at 20% sparsity under different M values. It can be seen that the reconstruction method that removes outliers consistently performs better compared to the method that does not remove outliers, regardless of the value of M. Different models exhibit varying performance across different hyperparameter M values. As shown in Table 6, for both LLaMA models, the performance with $M = 7$ and $M = 5$ is inferior to that with $M = 3$, indicating that using a threshold greater than five times the mean value is not suitable for identifying most outlier channels. Therefore, we select $M = 3$ as the default setting of our experiments.

To further investigate the impact of outliers on our compensation method, we study the condition number of the matrix $\mathbf{Z}$ in each layer, which reflects the degree of ill-conditioning in the optimization problem described in Eq. (26). The condition number serves as an important indicator of how ill-conditioned a problem is, the larger the condition number, the more ill-conditioned the problem becomes.

Fig. 5 presents a comparison of the condition numbers at different M values in specific intermediate layers for LLaMA-V2-7B and LLaMA-V3-8B. As seen in Fig. 5, the condition numbers for the FFN layers are generally smaller, while those for the MHA layers are larger. Additionally, compared to the method that does not remove outliers (represented by the red line), the method that removes outliers (represented by lines of other colors) significantly reduces the condition numbers. The effectiveness of reducing the condition numbers varies with different values of the hyperparameter M across different layers, but a clear reduction is

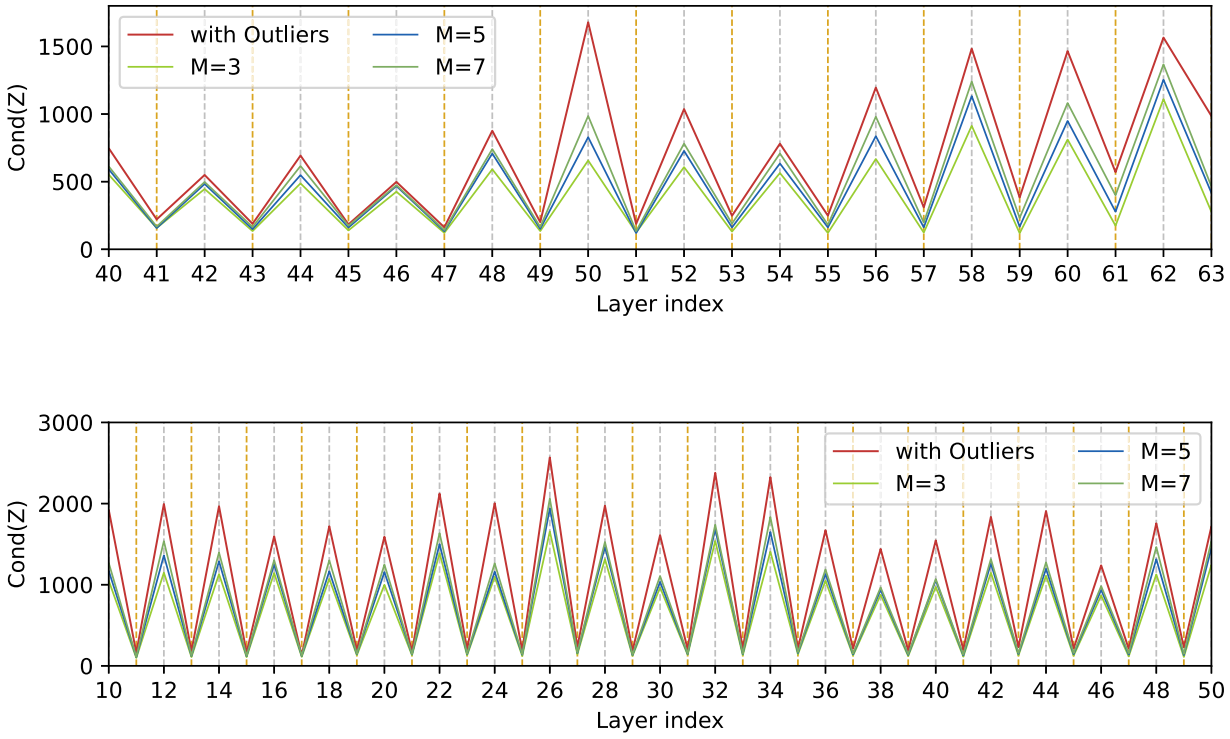

Fig. 5. Comparison of the condition numbers of matrices **Z** before and after removing outliers at a sparsity of 20%. Top: Condition number comparison of specific layers in LLaMA-V2-7B. Bottom: Condition number comparison of specific layers in LLaMA-V3-8B. The red line represents the condition number of matrix **Z** before outlier removal, while the other lines represent the condition numbers of $\tilde{\mathbf{Z}}$ after outlier removal with different M. The yellow dashed line represents the MHA layer, and the gray dashed line represents the FFN layer.

Table 7
At a 20% sparsity, the average condition numbers of the matrix **Z** in the MHA and FFN layers at different hyperparameter M. 'Outlier' indicates results where outliers are not removed during reconstruction.

Avg(Cond(**Z**))	LLaMA-V2-7B		LLaMA-V3-8B	
Layer	MHA	FFN	MHA	FFN
Outlier	6446.34	419.57	2568.68	460.66
$M = 3$	1853.32	122.64	1403.62	131.05
$M = 5$	2114.81	143.50	1568.35	159.45
$M = 7$	2402.26	159.61	1699.25	178.01

Table 8
The average time consumption (in seconds) to reconstruct an MHA layer for LLaMA-V2-7B/13B/70B at 20% sparsity with different numbers of reconstruction samples. OOM represents out-of-memory.

Reconstruction Samples	LLaMA-V2		
	7B	13B	70B
32	104.37	95.34	339.64
64	120.29	94.02	646.35
128	85.62	95.45	OOM

consistently observed. This further demonstrates that LLMs are complex systems with varying outlier distributions across layers.

To further discussion, we compare the mean values of condition numbers of all MHA layers and all FFN layers in LLaMA-V2-7B and LLaMA-V3-8B, as shown in Table 7. It can be observed that by removing outliers, the mean values of condition numbers are reduced by 2–3 times, even in the FFN layers where the condition numbers are already relatively low. This further demonstrates that removing outliers in our method effectively mitigates the issue of ill-conditioning. Moreover, the condition numbers are the lowest when the hyperparameter M is 3, which aligns with the experimental results in Table 6, where our method demonstrated the best performance at $M = 3$.

5.4.3. Time consumption

Additionally, we evaluate the efficiency of our compensation method by focusing on the time consumption for the linear reconstruction calculations in WLR. This analysis excludes common steps shared by all compensation methods, such as pruning and obtaining the calibration set through inference. We concentrate on the time consumption specifically related to the layer-wise linear reconstruction process, which includes calculating Z^{ℓ}, removing outliers, and computing $\mathbf{s}_p$ for each pruned channel using the CuPy library. Table 8 presents the average time taken to reconstruct an MHA layer for LLaMA-V2 models with varying numbers of reconstruction samples. Since the MHA layer is more complex and has a larger condition number, it faces more severe ill-conditioning issues. The experiments are performed on one NVIDIA Tesla A800 80G GPU. As can be seen from Table 8, on the 7B and 13B models, we only need approximately or even less than two minutes of GPU time to reconstruct an MHA layer. For models as large as 70B, when the reconstruction set consists of 32 or 64 samples, our method can complete the reconstruction in approximately 5 to 10 minutes.

5.4.4. Latency reduction of structured pruning

Table 9 reports the inference time of OPT-6.7B after 20% sparsity pruning using WLR on two NVIDIA 4090 GPUs. It can be observed that our approach consistently reduces the inference time across different batch sizes. Notably, compared with 20%-SliceGPT, which introduces additional parameters, our method achieves better speedup while maintaining the same sparsity level.

Table 9
Inference time (ms) of OPT-6.7B after 20% sparsity pruning on two 4090 GPUs.

Batch Size	Dense	SliceGPT* Ashkboos et al. (2024)	Ours
1	295.3	377.9	**243.1**
2	307.7	361.9	**289.2**
4	315.1	372.2	**261.4**

6. Conclusion

In this work, we propose Well-Conditioned Linear Reconstruction (WLR), a compensation method for structured pruning for large language models. WLR eliminates the need for back-propagation and retraining while avoiding the introduction of additional parameters into the pruned model. The method reconstructs pruned layers through linear combinations and ensures that ill-conditioned problems are mitigated during the reconstruction process. We assessed the effectiveness and efficiency of WLR using the Wikitext dataset and various commonsense benchmarks. Experimental results demonstrate that WLR effectively resolves ill-conditioning issues and significantly improves the performance of the compressed models.

CRediT authorship contribution statement

Siqi Li: Software, Resources, Methodology; **Jingyang Xiang:** Formal analysis; **Jiateng Wei:** Data curation; **Chengrui Zhu:** Formal analysis; **Jiandang Yang:** Conceptualization; **Jun Chen:** Supervision; **Jian Yang:** Conceptualization, Funding acquisition; **Xiaobin Wei:** Investigation; **Yunliang Jiang:** Supervision; **Yong Liu:** Supervision.

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This research was supported by the State Key Laboratory of Industrial Control Technology, China (ICT2024A09).

References

An, Y., Zhao, X., Yu, T., Tang, M., & Wang, J. (2024). Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 10865–10873). (*vol. 38*).

Ashkboos, S., Croci, M. L., Nascimento, M. G. d., Hoefler, T., & Hensman, J. (2024). Slicegpt: Compress large language models by deleting rows and columns. arXiv preprint arXiv:2401.15024.

Bai, S., Chen, J., Shen, X., Qian, Y., & Liu, Y. (2023). Unified data-free compression: Pruning and quantization without fine-tuning. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)* (pp. 5876–5885).

Bisk, Y., Zellers, R., Gao, J., Choi, Y. et al. (2020). Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 7432–7439). (*vol. 34*).

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.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S. et al. (2023). Sparks of artificial general intelligence: Early experiments with gpt-4. arXiv preprint arXiv:2303.12712.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., & Tafjord, O. (2018). Think you have solved question answering? try arc, the ai2 reasoning challenge. arXiv preprint arXiv:1803.05457.

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

Frantar, E., & Alistarh, D. (2023). Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning* (pp. 10323–10337). PMLR.

Frantar, E., Ashkboos, S., Hoefler, T., & Alistarh, D. (2022). Gptq: Accurate post-training quantization for generative pre-trained transformers. arXiv preprint arXiv:2210.17323.

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., & Zou, A. (2021). A framework for few-shot language model evaluation. https://doi.org/10.5281/zenodo.5371628

He, Y., Lin, J., Liu, Z., Wang, H., Li, L.-J., & Han, S. (2018). Amc: Automl for model compression and acceleration on mobile devices. In *European conference on computer vision (ECCV)*.

He, Y., Zhang, X., & Sun, J. (2017). Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 1389–1397).

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

Kim, W., Kim, S., Park, M., & Jeon, G. (2020). Neuron merging: Compensating for pruned neurons. *Advances in Neural Information Processing Systems*, *33*, 585–595.

Lee, C., Jin, J., Kim, T., Kim, H., & Park, E. (2023). Owq: Lessons learned from activation outliers for weight quantization in large language models. arXiv preprint arXiv:2306.02272.

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

Ling, G., Wang, Z., & Liu, Q. (2024). Slimgpt: Layer-wise structured pruning for large language models. *Advances in Neural Information Processing Systems*, *37*, 107112–107137.

Luo, J.-H., Wu, J., & Lin, W. (2017). Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision* (pp. 5058–5066).

Ma, X., Fang, G., & 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., & Socher, R. (2016). Pointer sentinel mixture models. arXiv preprint arXiv:1609.07843.

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

Mussay, B., Osadchy, M., Braverman, V., Zhou, S., & Feldman, D. (2020). Data-independent neural pruning via coresets. In *International conference on learning representations*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, *32*.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, *21*(140), 1–67.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., & Choi, Y. (2021). Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, *64*(9), 99–106.

Srinivas, S., & Babu, R. V. (2015). Data-free parameter pruning for deep neural networks. arXiv preprint arXiv:1507.06149.

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

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.

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. (2019). SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d. Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc. (*vol. 32*). https://proceedings.neurips.cc/paper/2019/file/4496bf24afe7fab6f046bf4923da8de6-Paper.pdf.

Wang, P., Fan, Z., Hu, S., Chen, Z., Wang, Y., & Wang, Y. (2024a). Reconstruct the pruned model without any retraining. arXiv preprint arXiv:2407.13331.

Wang, X., Zheng, Y., Wan, Z., & Zhang, M. (2024b). Svd-llm: Truncation-aware singular value decomposition for large language model compression. arXiv preprint arXiv:2403.07378.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M. et al. (2019). Huggingface's transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771.

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

Yvinec, E., Dapogny, A., Cord, M., & Bailly, K. (2021). Red: Looking for redundancies for data-free structured compression of deep neural networks. *Advances in Neural Information Processing Systems*, *34*, 20863–20873.

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

Zhang, B., Haddow, B., & Birch, A. (2023). Prompting large language model for machine translation: A case study. In *International conference on machine learning* (pp. 41092–41110). PMLR.

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