
Restoring Pruned Large Language Models via Lost Component Compensation

Zijian Feng¹ Hanzhang Zhou¹ Zixiao Zhu¹ Tianjiao Li¹ Jia Jim Deryl Chua²
Lee Onn Mak² Gee Wah Ng² Kezhi Mao^{1,*}

¹School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore

²Home Team Science and Technology Agency (HTX), Singapore

{feng0119, hanzhang001, zixiao001}@e.ntu.edu.sg

{tianjiao.li, ekzmao}@ntu.edu.sg

{deryl_chua, mak_lee_onn, ng_gee_wah}@htx.gov.sg

Abstract

Pruning is a widely used technique to reduce the size and inference cost of large language models (LLMs), but it often causes performance degradation. To mitigate this, existing restoration methods typically employ parameter-efficient fine-tuning (PEFT), such as LoRA, to recover the pruned model’s performance. However, most PEFT methods are designed for dense models and overlook the distinct properties of pruned models, often resulting in suboptimal recovery. In this work, we propose a targeted restoration strategy for pruned models that restores performance while preserving their low cost and high efficiency. We observe that pruning-induced information loss is reflected in attention activations, and selectively reintroducing components of this information can significantly recover model performance. Based on this insight, we introduce RestoreLCC (Restoring Pruned LLMs via Lost Component Compensation), a plug-and-play method that contrastively probes critical attention heads via activation editing, extracts lost components from activation differences, and finally injects them back into the corresponding pruned heads for compensation and recovery. RestoreLCC is compatible with structured, semi-structured, and unstructured pruning schemes. Extensive experiments demonstrate that RestoreLCC consistently outperforms state-of-the-art baselines in both general and task-specific performance recovery, without compromising the sparsity or inference efficiency of pruned models ².

1 Introduction

Large language models (LLMs) have achieved remarkable success in various natural language processing (NLP) tasks like commonsense reasoning, math problem solving, and text completion [1, 2, 3]. However, their large parameter counts demand substantial computational resources for deployment and inference. To democratize LLMs, pruning has emerged as a key technique to reduce model size and accelerate inference [4]. Pruning typically involves two steps: weight pruning and performance restoration. Weight pruning methods can be roughly categorized into structured pruning, such as LLM-Pruner [5] and SlimGPT [6], semi-structured pruning, and unstructured pruning, such as SparseGPT [7] and Wanda [4]. These methods estimate the importance of parameters and zero out unimportant ones to reduce model size. Performance restoration is a critical step in mitigating the

* Corresponding author.

²Code: <https://github.com/zijian678/restorelcc/>

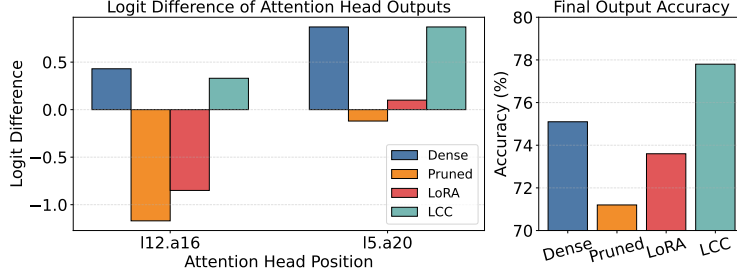


Figure 1: These figures compare the performance of the original **dense** model, the **pruned** model, and two recovery methods applied to the pruned model: **LoRA** and **Lost Component Compensation (LCC, ours)**, which directly adds back components lost due to pruning. The used dataset is BoolQ, and please see § 3 for experimental details. The left figure shows the logit difference (correct minus incorrect) of attention heads with/without recovery, and the right figure shows the accuracies of final model outputs for each method. $l[\cdot]$ and $a[\cdot]$ indicate the layer index and the head index, respectively. For instance, 12.a16 means the 16-th head at the 12-th layer.

performance degradation caused by weight pruning, aiming to recover model capability by adjusting weights through language modeling or instruction tuning datasets [8, 9, 10].

As full fine-tuning (FT) still requires substantial computational resources, parameter-efficient fine-tuning (PEFT) has become the mainstream approach for restoring pruned models. Recent pruning methods, including LLM-Pruner, SparseGPT, Wanda, and SlimGPT, all adopt LoRA [11] to recover performance. Although seemingly straightforward, applying existing PEFT methods, such as LoRA-based approaches [11, 12, 13] and representation engineering [14, 15, 16], to pruned LLMs raises concerns.

These PEFT methods are originally designed for dense models, where they adapt LLMs to downstream tasks by training a small subset of parameters. When applied to pruned models, they often overlook pruning-specific characteristics, such as the need to account for lost information, leading to inefficient parameter search and suboptimal restoration. As shown in Figure 1, attention heads recovered using LoRA achieve only limited improvement in logit predictions, resulting in suboptimal performance on the final task accuracy. In contrast, directly reintroducing the pruned components into the pruned attention heads (**LCC**) significantly improves both the outputs of the attention heads and the overall model accuracy. Built on this insight, we propose RestoreLCC (Restoring Pruned LLMs via Lost Component Compensation), which explicitly reconstructs critical information lost during pruning to bridge the performance gap between pruned and dense models.

Specifically, our study in § 3 reveals that important information removed by pruning can be captured in attention head activations, and reintroducing components of this lost information can substantially restore the performance of the pruned model. However, this insight also presents challenges, such as determining which attention heads to select and how to estimate component vectors that encode critical information for compensation. To overcome these challenges, RestoreLCC incorporates two main mechanisms: (1) **contrastive probing**, a general approach that leverages activation editing to construct contrastive sample pairs and probes a subset of attention heads critical for performance recovery; and (2) **lost component compensation (LCC)**, which retrieves pruning-induced lost information from these key heads. This information is decomposed into components, represented as vectors that capture the lost information directions. We optimize their magnitudes to enable targeted restoration along these directions, aggregate them into a single informative component, and finally inject it into the pruned model to recover performance. Unlike existing PEFT methods that restore pruned LLMs in an unguided manner, RestoreLCC explicitly compensates for key components lost during pruning, offering a targeted and effective restoration strategy.

We empirically evaluate RestoreLCC against other performance restoration methods across all three types of pruned models (e.g., structured, semi-structured, and unstructured), on both general recovery and task-specific settings across a wide range of LLMs of different sizes. Our results show that RestoreLCC significantly improves pruned model performance under general recovery settings while maintaining similar inference speed and sparsity ratios, outperforming existing PEFT methods. Furthermore, under task-specific recovery settings, RestoreLCC successfully recovers task-specific information and enables higher pruning ratios, where other PEFT methods often fail.

Contributions. Our main contributions are: (1) We observe that pruning-induced information loss is reflected in attention activations, and that selectively restoring key components can significantly recover model performance (§ 3); (2) We propose RestoreLCC, a method that learns the magnitudes of important component directions lost during pruning and reintroduces them to restore pruned models effectively (§ 4); (3) Extensive experiments across various LLMs and pruning schemes demonstrate that RestoreLCC consistently outperforms existing restoration baselines (§ 5).

2 Related work

LLM Pruning. Pruning reduces model size and speeds up inference by removing less important weights. Unstructured pruning removes individual weights irrespective of position, as in SparseGPT [7], which uses approximate Hessian-based reconstruction, and Wanda [4], which ranks weights by magnitude and activation norms. DSOT [17] and ALPS [18] further refine sparsity via optimization. Semi-structured pruning enforces patterns like N:M sparsity [19], and many unstructured methods adapt to this format [4, 7, 18]. Structured pruning removes entire blocks (e.g., rows or columns) to enhance hardware efficiency. Recent methods include LLM-Pruner [5], Compresso [20], LoRAPrune [21], and SlimGPT [6].

Performance Restoration and PEFT. Pruning methods such as LLM-Pruner [5], LoRAPrune [21], SparseGPT [7], Wanda [4], and SlimGPT [6] consistently benefit from a performance restoration step, typically involving fine-tuning on language modeling (e.g., C4 [8], WikiText [9]) or instruction-tuning datasets (e.g., Alpaca [10]). While full fine-tuning (FT) is effective, it remains computationally expensive, often requiring days on large GPU clusters. Parameter-efficient fine-tuning (PEFT) offers a cheaper alternative by updating a small subset of parameters. PEFT techniques include adapter-based [22, 23, 24], prompt-based [25, 26], LoRA-based (e.g., LoRA [11], ALoRA [27], VERA [12], DoRA [13]), and representation-engineering methods (e.g., RED [14], ReFT [15], LoFit [16]). Among them, LoRA is the most widely adopted in modern pruning pipelines. In addition to PEFT methods, FLAP [28] proposes a bias compensation strategy to reduce the pruning loss. EoRA [29] provides a fine-tuning-free approach to recover pruned models by searching low-rank spaces in a task-specific eigenspace to minimize compression loss.

3 Insight: injecting pruned components back effectively restores models

Preliminaries. We begin by clarifying several key terms and the scope of this study. Current LLM pruning methods primarily target the weight matrices of both attention and feed-forward (FFN) modules. For example, Wanda removes 50% of the weights in each matrix within both the attention and FFN modules (setting them to zero), resulting in an overall sparsity of 50%. Accordingly, in our work, **all weight matrices within the LLM are pruned, and we focus on restoring the pruned model by compensating through attention heads.** An *activation* refers to the output of a specific module within the Transformer architecture. A *pruned activation* is the output produced by a pruned module. Although its dimensionality remains identical to that of the original activation, it typically contains less information because the underlying weight matrices have been pruned. Importantly, the activations themselves are not pruned—only the associated weight matrices are. *Sparsity* denotes the proportion of weight parameters that have been pruned.

Pruning inherently leads to information loss, which becomes more pronounced at higher pruning ratios. In this section, we demonstrate that the pruned activations contain critical information, including discriminative components essential for downstream NLP tasks. By reintroducing the components, the performance of the pruned model can be substantially restored.

Recent studies have shown that different attention heads specialize in distinct functions when performing NLP tasks [30, 31, 32]. Motivated by this, we investigate how pruning-induced loss of head activations affects the model’s retained functional capacity and performance. Let MultiHead denote the multi-head attention output in a Transformer block with H attention heads. The output can be expressed as:

$$\text{MultiHead}(x^l) = \text{concat}\left(z^{(l,0)}, \dots, z^{(l,H-1)}\right) W^O, \quad (1)$$

where x^l is the input to the l -th layer, $z^{(l,h)}$ is the output of the h -th head, and W^O is a shared output projection matrix. Denote by $z_d^{(l,h)}$ and $z_p^{(l,h)}$ the activations of the h -th head in the dense and pruned

models, respectively, for the same input. The lost activation can be computed as Eq. 2. Our objective is to analyze whether $\delta z^{(l,h)}$ carries critical information that could aid in model recovery.

$$\delta z_{d-p}^{(l,h)} = z_d^{(l,h)} - z_p^{(l,h)}. \quad (2)$$

Given N samples, we denote the activation loss matrix for all samples as $\Delta \mathbf{Z}^{(l,h)} = [\delta z_{0,d-p}^{(l,i)}; \dots; \delta z_{N-1,d-p}^{(l,h)}] \in \mathbb{R}^{N \times d_h}$, where d_h is the dimensionality of the head activation. To better characterize the structure of the lost information, we apply singular value decomposition (SVD) to $\Delta \mathbf{Z}^{(l,h)}$, as shown in Eq. 3, which decomposes the activation loss matrix into a set of orthogonal latent components. This formulation enables us to identify the dominant directions of lost activation information and quantify their impact on each sample.

$$\Delta \mathbf{Z}^{(l,h)} = \mathbf{U}^{(l,h)} \Sigma^{l,h} \mathbf{V}^{(l,h)\top} = \sum_{i=1}^{d_h} \sigma_i u_i^{(l,h)} v_i^{(l,h)} \approx \sum_{i=1}^K \sigma_i u_i^{(l,h)} v_i^{(l,h)}. \quad (3)$$

For the output $z_p^{(l,h)}$ of a pruned attention head, the lost principal components can be approximated by

$$c^{(l,h)} = \sum_{i=1}^K \alpha_i^{(l,h)} v_i^{(l,h)}, \quad (4)$$

where $c^{(l,h)} \in \mathbb{R}^{d_h}$, and $\alpha_i^{(l,h)}$ denotes the average of $\sigma_i u_i^{(l,h)}$, representing the mean projection coefficients across samples. Finally, the activation output of the pruned attention head $z_p^{(l,h)}$ can be compensated and recovered by injecting the estimated lost components $c^{(l,h)}$ back:

$$z_c^{(l,h)} = z_p^{(l,h)} + c^{(l,h)}, \quad (5)$$

where we omit the sample index i in $z_{i,c}^{(l,h)}$ and $z_{i,p}^{(l,h)}$ for simplicity. To evaluate the predictive behavior of an attention head’s activation, we follow the theories of LogitLens [33] and prior work on interpreting LLMs in embedding space [34]. Specifically, we project the activation into the embedding space and compute its prediction over the vocabulary space:

$$p_{z_c}^{(l,h)} = \text{LM_Head} \left[\phi \left(z_c^{(l,h)} W^{O,h} \right) \right], \quad (6)$$

where LM_Head denotes the LLM’s prediction head, $\phi(\cdot)$ is the layer normalization function, $W^{O,h}$ is the output projection matrix in W^O corresponding to the h -th head, $p_{z_c}^{(l,h)} \in \mathbb{R}^{|\mathcal{V}|}$, $\mathcal{V}$ represents the model’s vocabulary, and $|\mathcal{V}|$ is the vocabulary size. Following IOI [35], we compute the **logit difference**, defined as the logit assigned to the correct token minus the logit assigned to the wrong token. This difference directly reflects the model’s confidence and faithfulness in predicting the correct token over an incorrect alternative.

Empirical Study. To evaluate whether reintroducing lost components can aid in performance recovery, we conduct an empirical study using BoolQ [36], a widely used commonsense reasoning dataset. We adopt LLaMA-7B [2] as the dense backbone model and apply Wanda [4], a state-of-the-art pruning method, to prune the model to 50% sparsity. We randomly sample 1,000 examples from BoolQ, where both the dense and pruned models are tasked with answering “yes” or “no” questions. We extract the attention head activations from the pruned model and reconstruct the compensated activations using the main components, as described in Eqs. 2-5. To quantify model confidence, we compute the **logit difference** λ using:

$$\lambda = p_{z_c}^{(l,h)}[\text{yes}] - p_{z_c}^{(l,h)}[\text{no}] \quad \text{or} \quad p_{z_c}^{(l,h)}[\text{no}] - p_{z_c}^{(l,h)}[\text{yes}],$$

We also report final **accuracy**, i.e., the predictions from the last layer, to assess the effect of component compensation on the model’s output.

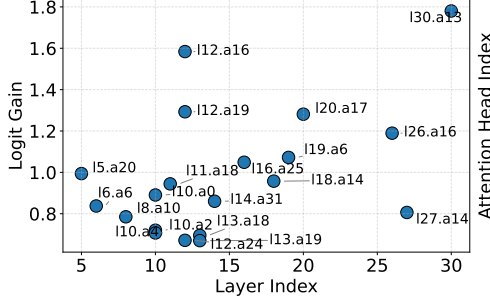


Figure 2: Logit gain of different attention heads recovered with principal components.

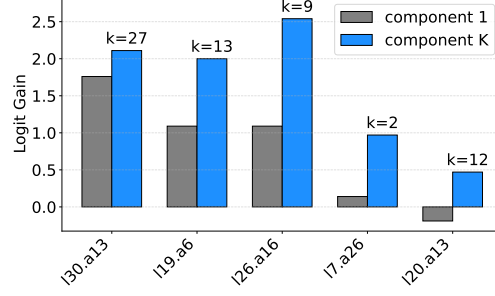


Figure 3: Logit gain of attention heads recovered with different components.

Finding 1. Reintroducing lost components to selected pruned attention heads can significantly restore model performance. As shown in Figure 1, we use the top-10 ($K = 10$ in Eq. 4) lost components to reconstruct the pruned activations. The left figure demonstrates that this compensation substantially recovers the pruned activations. Specifically, the logit difference is restored to a level comparable to that of the original dense model. Furthermore, compensating these attention heads also leads to a notable improvement in the model’s final output accuracy.

Finding 2. Different attention heads vary in importance and exhibit distinct recovery behaviors. We manually select and examine the outputs of several attention heads, along with their restoration using the top-10 lost components. To quantify the direct effect of compensation, we calculate the **logit gains** as $\delta\lambda = \lambda_{\text{recovered}} - \lambda_{\text{pruned}}$, which directly captures the improvement brought by reintroducing the lost components. Figure 2 shows the recovered logit gains for these heads. The results reveal that attention heads respond differently to the compensation process, indicating that not all can be effectively restored using the lost principal components.

Finding 3. Discriminative information may reside in minor components rather than in the principal ones. Figure 3 illustrates the recovery performance using either the top principal component or a selected minor component. Notably, since the coefficients of minor components are extremely small, we scale them by a factor of 1000 for visualization and evaluation. Surprisingly, incorporating certain minor components can lead to substantially better restoration performance compared to using the leading principal component.

Extension to FFN. Each Transformer layer comprises a multi-head attention (MHA) module and a feed-forward network (FFN) module. In the above analysis, we focus on compensating for the MHA rather than the FFN. To further justify this design choice, that is, using MHA instead of FFN, we provide additional theoretical and empirical analyses in Appendix A.

To summarize, the above analysis highlights not only the strong potential of leveraging lost components for performance restoration, but also several key challenges: (1) How to select pruned attention heads that are important for recovery? (2) How to determine the positions and coefficients of key components? and (3) How to apply the findings to more general tasks beyond BoolQ?

4 RestoreLCC: restoring pruned LLMs via lost component compensation

To address these challenges and enable universal performance restoration for pruned LLMs, we propose RestoreLCC, shown as Figure 4, which consists of two key mechanisms: (1) contrastive probing, a general method that uses activation editing to create contrastive sample pairs and identify critical attention heads; and (2) lost component compensation (LCC), which optimizes the magnitudes of directional components for the lost information. The optimized components are then injected back into the model to restore performance.

4.1 Contrastive probing

Contrastive Sample Construction. We construct contrastive samples for general NLP tasks to support activation editing and probing to localize important heads. Given a dataset, either task-

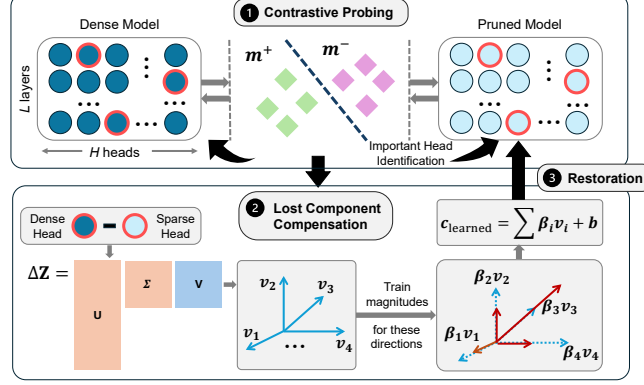


Figure 4: Overview of the RestoreLCC framework. It integrates two key modules, (1) contrastive probing and (2) lost component compensation (LCC), to restore the performance of pruned LLMs.

specific (e.g., BoolQ) or general (e.g., the Alpaca dataset), every sample is organized as (q, r^+) , where q is the question and r^+ is the correct (positive) response. All responses are collected into a set $[r_0^+, \dots, r_{N-1}^+]$. We use a sentence encoder, such as MiniLM-L6 [37], to encode all responses. For each sample, we select the most similar response (excluding the correct one) based on cosine similarity as the negative response r^- . This process converts each sample into a contrastive tuple (q, r^+, r^-) . Note that this method is universal and can be applied to any dataset. We provide examples of constructed samples from the BoolQ and Alpaca datasets in Appendix B.

Activation Editing. Recent studies suggest that high-dimensional activations in LLMs are approximately orthogonal with high probability [38]. An activation can be guided toward a desired generation space by adding a steering vector or a task-specific function vector [39, 32]. For pruned LLMs, we assume the lost principal components can be reintroduced to recover the activation and restore the correct response. Formally: recovered question activation $\approx$ pruned question activation + lost important component. Based on this, the recovered activation for a question is defined as

$$z_c^q = z_p^q + c^q, \quad (7)$$

where we omit the layer and head indices (l, h) for simplicity. Here, q indicates the activation is taken from the last token of the question, and the other symbols follow the notation in Eq. 5.

Attention Head Probing. Ideally, the recovered activation z_c^q should contain sufficient information to generate the correct response. It should be consistent with the activation of the complete and correct sample, denoted as $z_d^{q+r^+}$, which is obtained from the last token of the full sequence (including both the question q and the positive response r^+) in the original dense model. Additionally, it should be contrastive with respect to the negative sequence activation $z_d^{q+r^-}$. However, some attention heads may be inactive, and certain components may contribute little to performance restoration. In such cases, the recovered activation z_c^q may not adequately distinguish between $z_d^{q+r^+}$ and $z_d^{q+r^-}$.

In this way, identifying important attention heads can be formulated as a natural language inference (NLI) task. If an attention head is important and the corresponding component is useful, then the recovered activation z_c^q should entail the correct sequence activation $z_d^{q+r^+}$ and contradict the negative sequence activation $z_d^{q+r^-}$. Specifically, activation pairs are constructed as $m^+ = [z_c^q, z_d^{q+r^+}]$ with label 1 (entailment) and $m^- = [z_c^q, z_d^{q+r^-}]$ with label 0 (contradiction). A probing classifier, composed of a linear layer followed by a sigmoid activation, is trained to assess the discriminative power of each recovered head activation. The importance of attention heads is then ranked based on the accuracy of their corresponding probing classifiers.

4.2 Lost component compensation

We now have a list of important attention heads along with their corresponding lost components v_i based on Eq. 3. The goal is to use these components to approximate the missing information and recover the performance of pruned LLMs.

Lost Component Estimation. As observed in § 3, the usefulness of a component does not necessarily correlate with its coefficient. Minor components may also carry critical information and enable more effective recovery of pruned LLMs. To address this, we consider all possible components. Since the vectors v_i are orthogonal and unit-normalized, we treat each v_i as a potential direction of lost information. We keep these directions fixed and learn a scalar magnitude for each, representing its importance. Formally, for an attention head, we model its lost component containing important information as:

$$c_{\text{learned}} = \sum_{i=1}^{d_h} \beta_i v_i + b. \quad (8)$$

Here, v_i is obtained from Eq. 3 and remains fixed during training, while β_i is a trainable scalar indicating the importance of each direction. The term $b \in \mathbb{R}^{d_h}$ is a trainable bias vector for the attention head, acting as a hedging vector to provide flexibility in cases where important information lies outside the span of the predefined directions.

Component Compensation. For each pruned attention head, its output activation is recovered as $\tilde{z}_p = z_p + c_{\text{learned}}$. It is worth noting that the final learned component c_{learned} is a constant bias vector, capturing important information that was lost for all samples as a result of pruning. The recovered activation $\tilde{z}_p$ is then passed to the subsequent computations, such as those described in Eq. 1, allowing the model to better approximate the behavior of the original dense LLM.

Extension of RestoreLCC to FFNs. It is worth noting that RestoreLCC can be seamlessly extended to FFN modules. Our objective is to restore pruned models in which all weight matrices—both in the attention and FFN modules—have been pruned. We provide further theoretical and empirical analyses in Appendix A to justify why we apply RestoreLCC to attention heads instead of FFNs.

4.3 Overhead analysis of sparsity and inference speed

LLM pruning aims to produce a sparse model, while performance restoration seeks to close the performance gap between the pruned and dense models. It is important to ensure that performance restoration improves the pruned model’s accuracy without compromising its sparsity or inference efficiency.

Sparsity Analysis. Each compensated attention head introduces a learned vector $c_{\text{learned}} \in \mathbb{R}^{d_h}$. In the worst case, where all heads in a layer are compensated, the total number of additional parameters introduced across the projection matrices (q_proj, k_proj, v_proj, o_proj) within the multi-head attention mechanism is:

$$\text{Parameter Overhead} = \frac{2d_l}{4d_l^2} = \frac{1}{2d_l},$$

where d_l is the hidden size of the layer and $d_h = d_l/H$. Since d_l is typically larger than 1000, the increase in parameters is less than 0.05%, which has minimal impact on sparsity.

Inference Speed. To maintain inference efficiency, the learned vector c_{learned} is absorbed as the constant bias vector in the multi-head attention block. This modification introduces almost no additional computation during inference and preserves the speed of the pruned model. Therefore, RestoreLCC effectively preserves the pruned model’s sparsity and inference speed while recovering its performance. Additional empirical evidence is provided in Appendix D.

5 Experiments

5.1 Experimental settings

Metrics. To evaluate the restoration effectiveness of RestoreLCC, we experiment with representative pruned LLMs from diverse pruning strategies. Specifically, we use Wanda [4] for unstructured pruning, SparserGPT [7] for semi-structured pruning, and SlimGPT [6] for structured pruning. The pruning ratio is set to 50% for unstructured and semi-structured methods, and 20% for structured pruning, with C4 [8] as the calibration dataset. Following these works, we assess **perplexity (PPL)** of language modeling on the held-out WikiText [9] and **accuracy** of several commonsense reasoning

benchmarks, including BoolQ [36], HellaSwag [40], WinoGrande [41], ARC-easy [42], ARC-challenge [42], RTE [43], and OpenBookQA [44], all evaluated using the lm-eval-harness framework [45]. We consider two restoration settings:

- **General Recovery.** It is widely adopted in previous work. We follow SlimGPT and use the Alpaca instruction dataset [10] for tuning. Evaluations are performed in a zero-shot setting across language modeling and commonsense reasoning tasks.
- **Task-Specific Recovery.** While prior work focuses on general recovery, we highlight the importance of task-specific restoration, as pruned LLMs should also perform effectively when deployed in specialized domains. To evaluate this, we increase the sparsity ratio and restore pruned models using 100^3 training examples (e.g. 100 BoolQ samples) from the target task (e.g., BoolQ).

Models and Implementations. We conduct our main experiments using LLaMA-7B/13B models [2]. To evaluate the universality and scalability of RestoreLCC, we provide additional results on LLMs of varying sizes and families, including LLaMA-30B, LLaMA-2-7B/13B, LLaMA-3-8B [2], Vicuna-7b-v1.5 [46], Tulu-2-7B [47], Qwen-3-8B/14B [48], and DeepSeek-R1-Qwen3-8B [49] in Appendix K. For RestoreLCC, the number of components K is set to 1 to identify important attention heads, and we select 10%–25% of them for recovery. Additional implementation details are also provided in Appendix K.

Baselines. To ensure a comprehensive evaluation, we compare RestoreLCC with comprehensive baselines methods as follows:

- **LoRA** [11], which fine-tunes low-rank adapters to update the parameters of pruned LLMs. It is widely used for performance restoration in recent SOTA pruning methods [4, 5, 6, 7].
- **DORA** [13], which enhances LoRA by decomposing low-rank adaptation.
- **FLAP** [28], which recovers the pruned model by bias compensation.
- **EOA** [29], which searches low-rank spaces in the eigenspace to minimize compression loss.
- **LoFiT** [16], which is a SOTA representation engineering method that intervenes in attention activations to adapt LLMs to downstream tasks.

5.2 Performance of general recovery

Table 1 reports results across three pruning regimes on LLaMA-7B. Under unstructured pruning at 50% sparsity, RestoreLCC achieves 58.83% mean accuracy, outperforming the best baseline, LoFiT (56.82%), by **+2.01%**, and reduces PPL to 6.93. In the semi-structured pruning setting, RestoreLCC improves mean accuracy to 55.00%, yielding a **+2.65%** gain over the best baseline DoRA (52.35%), while also lowering PPL from 9.16 to 8.99. Under structured pruning at 20% sparsity, RestoreLCC reaches 59.76% accuracy, improving over DoRA (58.51%) by **+1.25%**. Across all settings, RestoreLCC consistently outperforms prior recovery methods in both accuracy and perplexity, recovering performance close to the dense model (59.99%).

Table 1: Performance of general recovery on zero-shot language modeling (PPL) and commonsense reasoning tasks (accuracy) using **LLaMA-7B**. Best scores are **bolded**.

Method	PPL ↓	BoolQ ↑	RTE ↑	HellaSwag ↑	WinoGrande ↑	ARC-e ↑	ARC-c ↑	OBQA ↑	Mean ↑
Dense Model	5.68	75.02	66.79	56.95	69.93	75.29	41.72	34.20	59.99
<i>Unstructured Pruning at 50% Sparsity</i>									
Wanda (Base Model for Recovery)	7.26	71.07	54.87	51.86	65.98	69.19	37.03	28.60	54.09
LoRA	7.09	72.05	58.84	52.93	66.85	71.68	39.16	32.40	56.27
DoRA	7.11	71.83	62.45	54.09	65.19	71.13	39.76	31.60	56.58
EOA	7.14	74.16	60.29	51.27	68.27	70.96	37.63	28.60	55.88
LoFiT	7.35	72.29	64.26	54.79	65.19	69.99	38.40	32.80	56.82
RestoreLCC (Ours)	6.93	72.84	69.68	56.34	65.98	71.80	40.96	34.20	58.83 (+2.01)
<i>Semi-Structured Pruning (N:M=2:4) at 50% Sparsity</i>									
SparseGPT (Base Model for Recovery)	11.04	69.45	54.51	43.12	60.93	60.90	30.20	23.80	48.99
LoRA	9.32	70.89	58.12	48.81	63.77	64.60	31.06	24.40	51.66
DoRA	9.16	71.41	59.21	49.16	62.04	65.99	33.45	25.20	52.35
FLAP	10.57	68.81	54.15	44.48	64.40	63.97	30.03	24.00	49.98
EOA	9.87	71.68	59.93	44.65	64.33	63.72	30.38	23.80	51.21
LoFiT	10.02	70.24	58.48	49.41	63.06	64.06	32.51	27.00	52.11
RestoreLCC (Ours)	8.99	73.61	63.90	51.71	65.11	68.14	33.53	29.00	55.00 (+2.65)
<i>Structured Pruning at 20% Sparsity</i>									
SlimGPT (Base Model for Recovery)	7.46	75.99	62.09	53.73	67.72	72.14	39.33	31.80	57.54
LoRA	7.66	76.48	65.70	55.25	66.77	72.01	40.36	32.60	58.45
DoRA	7.54	76.79	64.62	55.40	66.77	72.35	40.61	33.00	58.51
LoFiT	7.86	75.90	63.54	56.26	67.80	71.34	40.53	33.60	58.42
RestoreLCC (Ours)	7.53	76.48	68.59	57.05	69.46	72.01	40.53	34.20	59.76 (+1.25)

³We found that 100 samples are sufficient for effective restoration.

Table 2: General recovery on zero-shot language modeling (PPL) and commonsense reasoning tasks (mean accuracy) with LLaMA-13B.

Method	PPL ↓	Acc. ↑
Dense Model	5.09	62.57
<i>Unstructured Pruning at 50% Sparsity</i>		
Wanda	6.15	59.43
LoRA	6.08	61.13
DoRA	6.11	61.42
LoFiT	6.29	60.98
RestoreLCC (Ours)	6.08	62.46 (+1.04)
<i>Semi-Structured Pruning (N:M=2:4) at 50% Sparsity</i>		
SparseGPT	9.08	53.32
LoRA	7.75	55.22
DoRA	7.72	55.18
LoFiT	8.10	55.94
RestoreLCC (Ours)	7.61	57.67 (+1.73)
<i>Structured Pruning at 20% Sparsity</i>		
SlimGPT	5.99	60.21
LoRA	6.29	61.12
DoRA	6.10	61.96
LoFiT	6.51	61.22
RestoreLCC (Ours)	6.21	63.41 (+1.45)

Table 2 reports the recovery performance of LLaMA-13B; detailed results are provided in Table 17. Under unstructured pruning, RestoreLCC achieves 62.46% mean accuracy with a PPL of 6.08, corresponding to a **+1.04%** gain over the best baseline. In the semi-structured setting, RestoreLCC outperforms LoFiT by **+1.73%**. For structured pruning, RestoreLCC reaches 63.41% accuracy, surpassing DoRA by **+1.45%**. These results verify that RestoreLCC generalizes well to larger models, and consistently provides superior recovery across all pruning types.

5.3 Task-specific recovery

Table 3 evaluates task-specific recovery on LLaMA-7B under three sparsity settings. For unstructured pruning at 60% sparsity, RestoreLCC achieves a mean accuracy of 61.77%, improving over LoFiT (58.21%) by **+3.56%**. In the semi-structured pruning, RestoreLCC achieves 63.74% mean accuracy, outperforming LoFiT by **+3.06%**. For structured pruning at 40% sparsity, RestoreLCC again yields the best mean accuracy (59.55%), outperforming LoRA by **+3.4%**. In addition, we increase the pruning ratio by a challenging **10–20%**. RestoreLCC successfully recovers performance, demonstrating its effectiveness in **enabling higher sparsity** ratios for LLM pruning.

5.4 Ablation study

In this subsection, we analyze the impact of different mechanisms in RestoreLCC.

Effects of Contrastive Probing on Identifying Important Attention Heads. We apply contrastive probing to identify attention heads critical for performance recovery. Table 4 compares performance using heads selected by contrastive probing (RestoreLCC) versus randomly selected heads (w/o probing). The 1.26% degradation in the latter case confirms that contrastive probing effectively identifies heads essential for recovery. We also conduct experiments with two alternative head-selection strategies: (1) MSE-selected heads: selecting heads with the smallest MSE between the outputs of the dense and pruned models. (2) KL-selected heads: selecting heads with the smallest KL divergence. It can be observed that our probing-based selection consistently identifies important heads and is more effective than other metric-based approaches.

Effects of $\sum_{i=1}^{d_h} \beta_i v_i$. We estimate the lost information from pruning using $\sum_{i=1}^{d_h} \beta_i v_i$, which captures both direction and magnitude. To evaluate its impact, we remove this term and tune c_{learned} using only the bias vector in Eq. 8. This results in a 1.70% performance drop, underscoring the importance of recovering both directional and magnitude information in pruned models.

Table 3: Performance of task-specific recovery on individual commonsense reasoning tasks with LLaMA-7B.

Method	BoolQ ↑	RTE ↑	ARC-e ↑	ARC-c ↑	Mean ↑
Dense Model	75.02	66.79	75.29	41.72	64.71
<i>Unstructured Pruning at 60% Sparsity</i>					
Wanda	68.87	59.21	62.67	30.29	55.26
LoRA	69.14	67.51	62.08	33.36	58.02
DoRA	69.82	67.51	61.03	33.19	57.89
LoFiT	69.97	67.87	60.10	34.90	58.21
RestoreLCC (Ours)	75.32	70.40	63.89	37.46	61.77 (+3.56)
<i>Semi-Structured Pruning (N:M=2:4) at 50% Sparsity</i>					
SparseGPT	69.45	54.51	60.90	30.20	53.77
LoRA	69.76	68.59	62.46	35.92	59.18
DoRA	69.72	68.95	64.27	35.67	59.65
LoFiT	70.92	68.95	66.58	36.26	60.68
RestoreLCC (Ours)	77.74	69.31	69.32	38.57	63.74 (+3.06)
<i>Structured Pruning at 40% Sparsity</i>					
SlimGPT	69.57	64.26	55.98	32.94	55.69
LoRA	69.02	66.06	56.31	33.19	56.15
DoRA	69.20	65.34	56.10	33.19	55.96
LoFiT	67.68	64.26	57.41	33.45	55.70
RestoreLCC (Ours)	70.95	68.23	62.08	36.95	59.55 (+3.40)

Table 4: Ablation study results on a 50% pruned LLaMA-7B model using Wanda.

Method	Mean Accuracy
RestoreLCC	58.83
w/o probing	57.57 (-1.26)
MSE-selected	58.14 (-0.69)
KL-selected	57.92 (-0.91)
w/o $\sum_{i=1}^{d_h} \beta_i v_i$	57.13 (-1.70)
w/o b	58.26 (-0.57)

Effects of Bias Vector. We introduce a bias vector in Eq. 8 to serve as a hedging term, allowing flexibility when important information lies outside the span of predefined directions. Table 4 reports the results of removing the bias vector (w/o b), showing a 0.57% performance drop, which confirms its role in optimizing the final learned component.

5.5 Interpreting the learned component

To examine the information encoded in the learned component c_{learned} and its role in performance recovery, we illustrate task-specific recovery on BoolQ (binary yes/no answering). Following § 3, we use LogitLens to project c_{learned} into the embedding space and show the top-5 decoded tokens in Figure 5. The results suggest that c_{learned} captures task-relevant signals, such as indicating “yes” or “no” answers for BoolQ examples.

Heads	Top-5 Decoded Tokens
l30a13	'_no', 'no', '_yes', 'No', 'yes'
l29a10	'_yes', '_young', 'Young', 'no', 'yes'
l26a16	'yes', '_yes', '_off', 'no', 'YES'

Figure 5: Visualization of learned components for different attention heads.

5.6 Further analysis

We provide additional analysis to verify the universality and efficiency of RestoreLCC as follows:

- **Hyperparameter Sensitivity** (Appendix C): We evaluate RestoreLCC under varying numbers of attention heads and components in Eq. 3. Results demonstrate its stability.
- **Overhead and Efficiency Analysis** (§ 4.3 and Appendix D): We compare the trainable parameters, inference speed and overhead sensitivity of RestoreLCC with other baselines. The results verify that RestoreLCC restores pruned LLMs without compromising sparsity or inference efficiency.
- **Parameter Visualization** (Appendix E): We visualize the trained directions, magnitudes, and biases, offering deeper insight into the internal mechanisms of RestoreLCC.
- **Comparison with Full-Parameter Tuning** (Appendix F): We present experimental results that compare RestoreLCC with full-parameter tuning (FT).
- **Cross-Task Portability and Generalization of Probing** (Appendix G): We discuss the cross-task portability and generalization of the contrastive probing module in Appendix G.
- **Efficiency at Scale** (Appendix H): We demonstrate the efficiency of RestoreLCC on a larger LLM (LLaMA-70B) and evaluate its latency.
- **Compatibility with Quantized Models.** (Appendix H): We conducted experiments with 4-bit quantization on the pruned model and verify RestoreLCC’s compatibility with heavily quantized models.
- **Effect of Probing Samples.** (Appendix J): We study the effect of the number of probing samples on RestoreLCC.
- **Evaluation on More LLMs** (Appendix K): Experiments on LLMs including LLaMA-30B, LLaMA-2-7B/13B, LLaMA-3-8B [2], Vicuna-7b-v1.5 [46], Tulu-2-7B [47], Qwen-3-8B/14B [48], and DeepSeek-R1-Qwen3-8B [49] further validate the universality and scalability of RestoreLCC.

6 Conclusion

In this work, we propose RestoreLCC, a targeted strategy for restoring the performance of pruned LLMs without compromising their sparsity or inference speed. RestoreLCC integrates two key mechanisms: (1) contrastive probing, which leverages activation editing to probe critical attention heads, and (2) lost component compensation (LCC), which estimates and restores the lost directional information in pruned heads. Extensive experiments across diverse pruning settings and LLMs demonstrate the effectiveness of RestoreLCC in recovering model performance.

Limitation. We assign learnable magnitudes to all components to compensate for pruned attention heads, allowing less important ones to be down-weighted. However, this still may cause overfitting of unimportant components. In future work, we plan to pre-select relevant components before learning their magnitudes, reducing both overfitting and the number of trainable parameters. In addition, Moore and Chaudhuri [50] utilize activation noise to probe network structure and identify redundant neurons. They also discuss a novel way of leveraging activation noise for neuron identification. We believe activation noise could similarly be used to enhance contrastive probing, and we plan to explore this direction in future work.

Acknowledgments

We extend our heartfelt gratitude to the reviewers for their insightful and constructive feedback. This research was supported by the Home Team Science and Technology Agency (HTX), Singapore under the NTU-HTX collaboration project: *Parsimonious Domain Specific Large Language Model Enabled Multimodality Sensemaking*. We express our sincere appreciation to HTX for their continued support and collaboration.

References

- [1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- [2] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [3] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.
- [4] Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *The Twelfth International Conference on Learning Representations*, 2023.
- [5] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 21702–21720. Curran Associates, Inc., 2023.
- [6] Gui Ling, Ziyang Wang, Yuliang Yan, and Qingwen Liu. Slimgpt: Layer-wise structured pruning for large language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 107112–107137. Curran Associates, Inc., 2024.
- [7] Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10323–10337. PMLR, 23–29 Jul 2023.
- [8] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.
- [9] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.
- [10] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.
- [11] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.
- [12] Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M Asano. Vera: Vector-based random matrix adaptation. In *The Twelfth International Conference on Learning Representations*, 2024.
- [13] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: weight-decomposed low-rank adaptation. In *Proceedings of the 41st International Conference on Machine Learning, ICML’24. JMLR.org*, 2024.

- [14] Muling Wu, Wenhao Liu, Xiaohua Wang, Tianlong Li, Changze Lv, Zixuan Ling, Zhu JianHao, Cenyuan Zhang, Xiaoqing Zheng, and Xuanjing Huang. Advancing parameter efficiency in fine-tuning via representation editing. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13445–13464, Bangkok, Thailand, August 2024. Association for Computational Linguistics.
- [15] Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. Reft: Representation finetuning for language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 63908–63962. Curran Associates, Inc., 2024.
- [16] Fangcong Yin, Xi Ye, and Greg Durrett. Lofit: Localized fine-tuning on llm representations. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 9474–9506. Curran Associates, Inc., 2024.
- [17] Yuxin Zhang, Lirui Zhao, Mingbao Lin, Sun Yunyun, Yiwu Yao, Xingjia Han, Jared Tanner, Shiwei Liu, and Rongrong Ji. Dynamic sparse no training: Training-free fine-tuning for sparse llms. *The Twelfth International Conference on Learning Representations*, 2024.
- [18] Xiang Meng, Kayhan Behdin, Haoyue Wang, and Rahul Mazumder. Alps: Improved optimization for highly sparse one-shot pruning for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- [19] Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*, 2021.
- [20] Song Guo, Jiahang Xu, Li Lyna Zhang, and Mao Yang. Compresso: Structured pruning with collaborative prompting learns compact large language models. *arXiv preprint arXiv:2310.05015*, 2023.
- [21] Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. LoRAPrune: Structured pruning meets low-rank parameter-efficient fine-tuning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3013–3026, Bangkok, Thailand, August 2024. Association for Computational Linguistics.
- [22] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 09–15 Jun 2019.
- [23] Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 565–576, Online, August 2021. Association for Computational Linguistics.
- [24] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: efficient low-rank hypercomplex adapter layers. In *Proceedings of the 35th International Conference on Neural Information Processing Systems, NIPS ’21*, Red Hook, NY, USA, 2021. Curran Associates Inc.
- [25] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

- [26] Anastasiia Razdaibiedina, Yuning Mao, Madian Khabsa, Mike Lewis, Rui Hou, Jimmy Ba, and Amjad Almahairi. Residual prompt tuning: improving prompt tuning with residual reparameterization. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6740–6757, Toronto, Canada, July 2023. Association for Computational Linguistics.
- [27] Zeyu Liu, Souvik Kundu, Anni Li, Junrui Wan, Lianghao Jiang, and Peter Beerel. AFLoRA: Adaptive freezing of low rank adaptation in parameter efficient fine-tuning of large models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 161–167, Bangkok, Thailand, August 2024. Association for Computational Linguistics.
- [28] Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence, AAAI’24/IAAI’24/EAAI’24*. AAAI Press, 2024.
- [29] Shih-Yang Liu, Huck Yang, Chein-Yi Wang, Nai Chit Fung, Hongxu Yin, Charbel Sakr, Saurav Muralidharan, Kwang-Ting Cheng, Jan Kautz, Yu-Chiang Frank Wang, et al. Eora: Training-free compensation for compressed llm with eigenspace low-rank approximation. *arXiv e-prints*, pages arXiv–2410, 2024.
- [30] Wei Zhang, Chaoqun Wan, Yonggang Zhang, Yiu-Ming Cheung, Xinmei Tian, Xu Shen, and Jieping Ye. Interpreting and improving large language models in arithmetic calculation. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 59932–59950. PMLR, 21–27 Jul 2024.
- [31] Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for llms. In *12th International Conference on Learning Representations, ICLR 2024*, 2024.
- [32] Xinyu Ma, Yifeng Xu, Yang Lin, Tianlong Wang, Xu Chu, Xin Gao, Junfeng Zhao, and Yasha Wang. Dressing up llm: Efficient stylized question-answering via style subspace editing. In *The Thirteenth International Conference on Learning Representations*, 2025.
- [33] nostalgebraist. Interpreting gpt: the logit lens, 2020. Accessed: 2025-05-03.
- [34] Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. Analyzing transformers in embedding space. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16124–16170, Toronto, Canada, July 2023. Association for Computational Linguistics.
- [35] Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. In *The Eleventh International Conference on Learning Representations*, 2023.
- [36] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
- [37] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5776–5788. Curran Associates, Inc., 2020.

- [38] Zhichao Wang and Yizhe Zhu. Overparameterized random feature regression with nearly orthogonal data. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent, editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 8463–8493. PMLR, 25–27 Apr 2023.
- [39] Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. Function vectors in large language models. In *The Twelfth International Conference on Learning Representations*, 2024.
- [40] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics.
- [41] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, August 2021.
- [42] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.
- [43] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.
- [44] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun’ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium, October–November 2018. Association for Computational Linguistics.
- [45] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024.
- [46] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc., 2023.
- [47] Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A Smith, Iz Beltagy, et al. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702*, 2023.
- [48] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.
- [49] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.
- [50] Eli Moore and Rishidev Chaudhuri. Using noise to probe recurrent neural network structure and prune synapses. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS ’20, Red Hook, NY, USA, 2020. Curran Associates Inc.
- [51] Wei Zhang, Chaoqun Wan, Yonggang Zhang, Yiu-ming Cheung, Xinmei Tian, Xu Shen, and Jieping Ye. Interpreting and improving large language models in arithmetic calculation. In *Proceedings of the 41st International Conference on Machine Learning, ICML’24*. JMLR.org, 2024.

- [52] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc.
- [53] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: The abstract and introduction provide an accurate summary of our contributions and the scope of the paper.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: Please see section 6.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[Yes\]](#)

Justification: We have provided assumptions and proof in section 3.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: Please refer to sections 4 and 5.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will submit our source code as an anonymized zip file in the supplemental material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Please see section 5.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: The paper does not include error bars because of limited computational resources that prevented extensive experimentation.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Please see details of implementations in section 5.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: We have read the NeurIPS Code of Ethics and ensure that our paper strictly adheres to these guidelines.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Please see section 6.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks as we only used open-sourced models.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All of these are properly credited.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset’s creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: The paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

A MHA Compensation versus FFN Compensation

Each Transformer layer comprises a multi-head attention (MHA) module and a feed-forward network (FFN) module. In this study, we restore pruned LLMs via attention head compensation instead of FFN compensation, as justified by both theoretical analysis and empirical results.

Theoretical Analysis. Studies on mechanistic interpretability show that attention heads specialize in distinct functions for different tasks, while FFNs mainly store knowledge and map inputs to outputs. For example, [51] finds that in arithmetic tasks, attention heads process key information and FFNs then produce the final answer. They also show that fine-tuning only important heads outperforms tuning all parameters (including FFNs). Similarly, [31] demonstrates that different heads play distinct roles, and [32] also supports this observation. By contrast, FFNs store factual knowledge [52] and refine output logits [53]. Hence, compensating attention heads is more appropriate than compensating FFNs.

Moreover, attention heads provide finer information, whereas FFNs produce high-dimensional, aggregated outputs. In LLaMA-7B, each head outputs 128 dimensions (32 heads per layer), while an FFN outputs 4096-dimensional vectors. Smaller head outputs allow finer analysis. Moreover, choosing 32 heads can span multiple layers, while the same size in FFN covers only one layer.

Empirical Evidence. It is worth noting that our method can also be directly applied to the FFN modules. Accordingly, we applied RestoredLCC to the FFNs and report the best results in Table 5. The results obtained with different numbers of FFN layers are presented in Table 6. Compensating the attention modules achieves a score of 58.83, whereas compensating only the FFNs yields 57.00.

Table 5: Performance (**accuracy**) of general recovery on zero-shot commonsense reasoning tasks using **LLaMA-7B** pruned by Wanda at 50% sparsity.

Method	BoolQ $\uparrow$	RTE $\uparrow$	HellaSwag $\uparrow$	WinoGrande $\uparrow$	ARC-e $\uparrow$	ARC-c $\uparrow$	OBQA $\uparrow$	Mean $\uparrow$
Recover attn.	72.84	69.68	56.34	65.98	71.80	40.96	34.20	58.83
Recover FFN.	69.42	63.54	55.54	67.17	70.96	39.59	32.80	57.00

Table 6: Average accuracy performance with different recovered FFN ratios.

FFN ratio	0.1	0.3	0.5	0.7	1.0
Mean	55.92	57.00	56.24	56.68	56.80

B Examples of Contrastive Samples

Figure 6 and Figure 7 show constructed contrastive sample pairs for the Alpaca dataset and the BoolQ dataset, respectively.

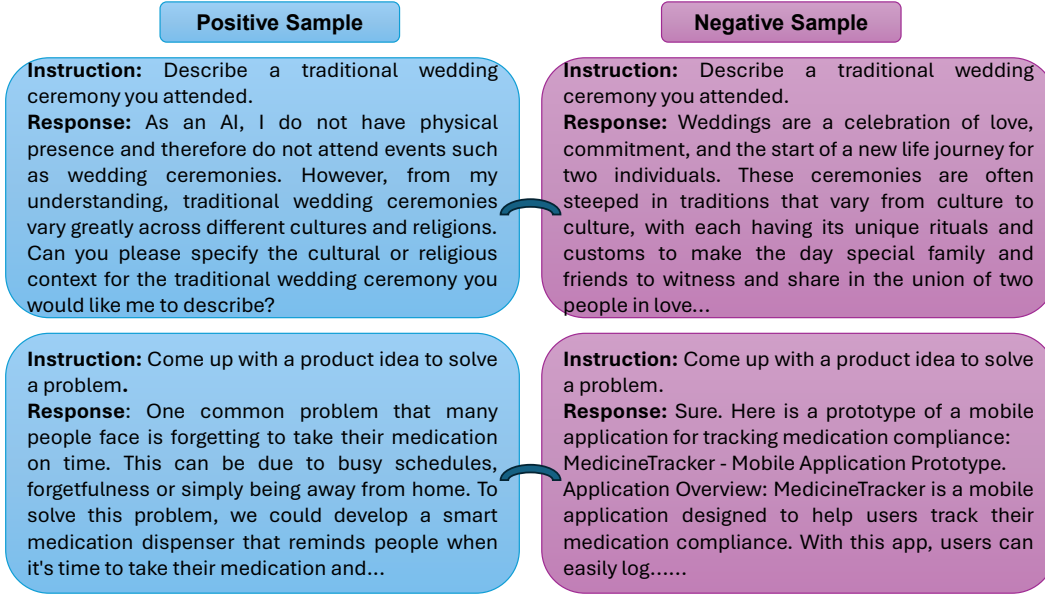


Figure 6: Constructed contrastive sample pairs for the Alpaca dataset.

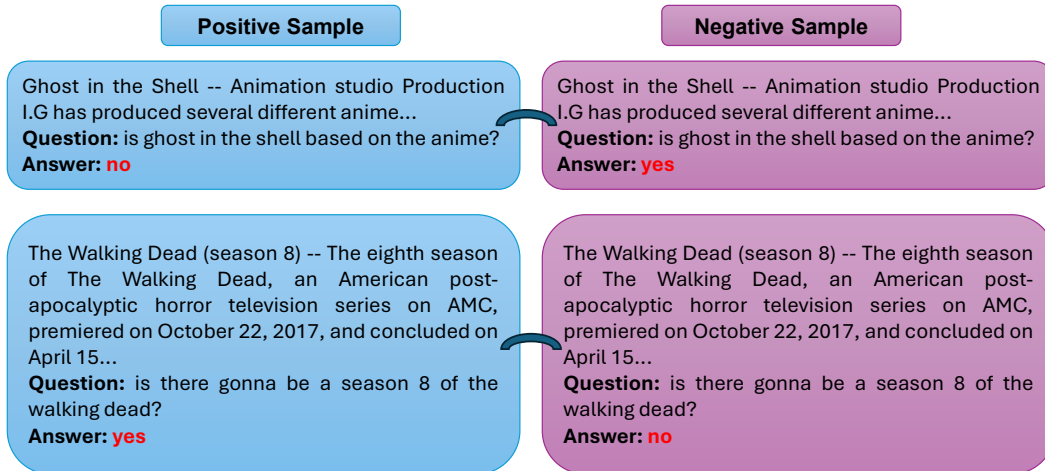


Figure 7: Constructed contrastive sample pairs for the BoolQ dataset.

C Hyperparameter Sensitivity

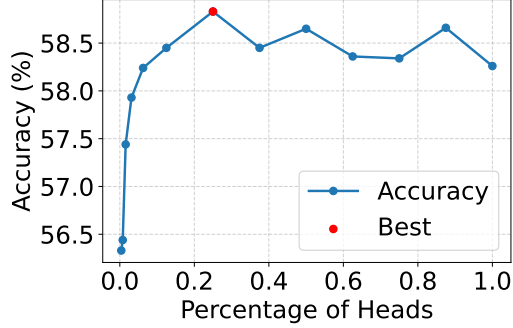


Figure 8: Average accuracy on commonsense reasoning tasks for different numbers of attention heads.

Number of Attention Heads. Figure 8 shows the average accuracy on commonsense reasoning tasks for different numbers of attention heads. We observe that using more than 5% of attention heads allows RestoreLCC to achieve around 58% mean accuracy. Based on the results, we recommend using 5%–25% of attention heads, striking a balance between strong performance and parameter efficiency.

Number of Components. The number of components K in Eq. 3 is used to identify important attention heads.

Since the component coefficients from SVD are ordered by magnitude, and the top components dominate, the selected heads remain consistent for $K \geq 3$. As a result, RestoreLCC achieves similar performance across these settings, as shown in Table 7. Additionally, RestoreLCC is highly stable across different K values, with only around 0.3% difference between $K = 1$, $K = 3$, and $K = 10$. Based on these observations, we recommend using $K \leq 3$, which already captures the most informative components.

Table 7: Experimental results on different numbers of components in Eq. 3

K	Mean Accuracy
1	58.83
3	58.53
10	58.51

D Overhead Analysis

Table 8: Comparison of trainable parameters and inference speed across different restoration methods. The base model is LLaMA-7B at 50% sparsity pruned by Wanda. The inference delay is computed by comparing the inference speed of the restored model to that of the original pruned model. A value of 1.0× indicates that the inference is performed at the same speed as the original pruned model.

Method	Number of Parameters	Inference Delay
Pruned Model (Base Speed)	–	1.0×
LoRA (w/ mask)	0.0600%	1.0×
LoRA (w/o mask)	0.0600%	1.1×
LoFiT	0.0005%	1.0×
RestoreLCC	0.0010%	1.0×

As discussed in § 4.3, RestoreLCC introduces negligible additional parameters that would impact the sparsity or inference speed of the pruned model. The representative engineering baseline, LoFiT, maintains a similar number of trainable parameters and can be integrated into the pruned LLM in the same manner as RestoreLCC, thus incurring no overhead. In contrast, LoRA and DoRA have two implementation options: (1) training with a masking constraint, where the low-rank matrices A and B in LoRA are constrained to have zeros in the same pruned positions. This allows the product BA

Table 9: Comparison of training time and GPU memory usage on LLaMA-7B at 50% sparsity pruned by Wanda.

Method	Time	Memory
LoRA	4h20min	61GB
DoRA	6h08min	71GB
LoFiT	8h11min	65GB
RestoreLCC	4h13min	65GB

Table 10: GPU memory usage and training time with different head ratios.

Head Ratio	0.1	0.2	0.3	0.5	1.0
GPU Memory	65GB	65GB	65GB	66GB	68GB
Time	3h40min	4h13min	4h40min	5h30min	7h42min

to be merged into the pruned LLM without affecting its sparsity or inference speed; and (2) direct training without masking, which treats A and B as external adapters. In this case, merging them would reintroduce non-zero values in pruned locations, compromising sparsity.

Furthermore, **the overhead introduced by the intermediate activations (v_i) is negligible and can be ignored.** These activations remain fixed (frozen) during training. Taking LLaMA-7B as an example, each attention head has 128 dimensions; by the property of singular value decomposition (SVD), this corresponds to at most 128 components, resulting in $128 \times 128 = 16,384$ parameters per head. Even in the extreme case where all 1,024 heads are used, this adds only $1,024 \times 16,384 \approx 16.8\text{M}$ parameters—merely about 0.24% of a 7B model. Moreover, as shown in Figure 8, only 10–30% of the heads are required for recovery, making the actual overhead substantially smaller.

Table 8 presents the number of trainable parameters and inference speed relative to the original pruned model. RestoreLCC requires significantly fewer parameters than LoRA and does not degrade inference speed. These empirical results support our earlier analysis: RestoreLCC effectively restores pruned LLMs without introducing overhead in terms of sparsity or efficiency.

Table 9 compares the training time and GPU memory usage on same computing conditions across different methods for recovering LLaMA-7B (1 H100 GPU, torch.bfloat16 precision, max-length=512, batch-size=8, Alpaca Dataset). For training time, the methods rank as follows: RestoreLCC < LoRA < DoRA < LoFiT. For GPU memory usage, the order is LoRA < RestoreLCC = LoFiT < DoRA. Compared with DoRA and LoFiT, RestoreLCC exhibits the lowest overall overhead while providing superior performance.

Overhead Sensitivity to the Number of Trainable Parameters. The number of trainable parameters in RestoreLCC depends on the proportion of attention heads selected for tuning. Table 10 reports GPU memory usage and training time for different head ratios. The results show that: (i) GPU memory consumption remains nearly constant as the number of trainable parameters increases and is consistently lower than that of DoRA (71 GB); (ii) the training time is generally shorter than both DoRA and LoFiT. Importantly, increasing the number of trainable parameters does not necessarily improve performance, indicating that RestoreLCC is not sensitive to this factor. This observation also motivates our design choice to first contrastively identify the most informative heads.

Based on the theoretical and empirical analysis, the overhead of our proposed RestoreLCC is smaller than advanced PEFT methods while achieving much better performance.

E Analysis of Trained Components

Figure 9 and Figure 10 illustrate the training results for both magnitudes and directions. Figure 9 shows that the learned magnitudes diverge notably from the original SVD coefficients, which are ranked from largest to smallest, supporting Finding 3: minor components can retain valuable pruned information. Figure 10 visualizes the directions (v_i), the trained bias vector (b), the combined direction ($\sum_{i=1}^{d_h} \beta_i v_i$), and the final learned component (c_{learned}). The final component benefits

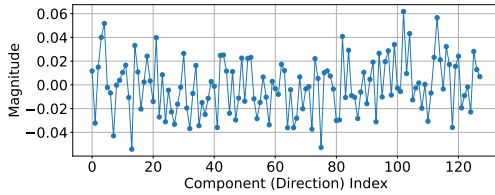


Figure 9: Visualization of trained magnitudes.

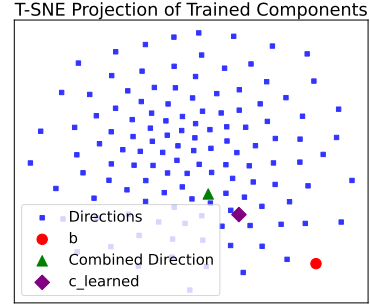


Figure 10: Visualization of trained directions.

Table 11: Comparison between RestoreLCC and full-parameter tuning (FT) on pruned LLaMA-7B models.

Method	Wanda (avg.)	SparseGPT (avg.)
LoFiT	56.82	52.11
FT	58.43	56.44
Ours (RestoreLCC)	58.83	55.00

from both magnitude adjustment and the bias vector, which collectively shift it slightly toward the lower-right.

F Comparison with Full-Parameter Tuning

We conduct experiments on LLaMA-7B. Table 11 summarizes the average results on commonsense reasoning tasks. On Wanda-pruned models, RestoreLCC achieves slightly better performance than full-parameter tuning (FT). On SparseGPT-pruned models, RestoreLCC performs slightly worse than FT. Overall, RestoreLCC achieves performance comparable to FT while requiring significantly fewer trainable parameters.

G Cross-Task Portability and Generalization of Probing

The probing module in **RestoreLCC** has (i) **negligible overhead**. For each attention head, we train only a simple linear layer as the probing classifier. This step is performed before the recovery tuning stage (Section 4.2, LCC) and can be completed even on a low-end GPU (e.g., NVIDIA RTX 1080) within minutes using minimal GPU memory. (ii) It is **easy to implement**. For new datasets or models, only head-wise activations need to be obtained. This can be achieved easily through three approaches: (a) using Python hook functions to capture the input/output of attention modules; (b) employing the open-source Pyvene⁴ package, which offers convenient APIs for activation extraction; or (c) leveraging the TransformerLens⁵ package, which supports modern architectures such as Qwen3. In all cases, only a few lines of Python code are required, and the classifiers can be quickly trained. (iii) **Cross-task portability**. When no dataset is available for a new task, the Alpaca instruction-tuning dataset can be used as a general-purpose probing dataset. As shown in Table 1, it works well for both language modeling and multiple commonsense reasoning tasks, enabling recovery without task-specific data. This makes the Alpaca dataset an effective universal fallback for probing and tuning, substantially improving portability. (iv) **Benefits outweigh the cost**. Probing focuses on identifying informative attention heads—typically only 10%–30% of all heads are restored—greatly improving efficiency by avoiding unnecessary tuning. Hence, the computational benefits of probing far exceed its minimal cost.

⁴<https://github.com/stanfordnlp/pyvene>

⁵<https://github.com/TransformerLensOrg/TransformerLens>

Table 12: Cross-task evaluation results using probing trained on BoolQ. RestoreLCC generalizes effectively without retraining.

Tasks	RTE	ARC-e	ARC-c
Pruned Model	59.21	62.67	30.29
RestoreLCC (w/o retraining probing)	68.23	62.92	35.41

Generalization Across Tasks. The probing module generalizes effectively across diverse downstream tasks without retraining, as evidenced by two observations:

(1) Probing on a general dataset, testing on diverse tasks. In our general recovery setup, attention heads are probed using the Alpaca dataset, while evaluations span distinct tasks including language modeling and seven commonsense reasoning benchmarks. As shown in Table 1, heads identified from Alpaca generalize well to diverse tasks such as BoolQ and ARC.

(2) Probing on a specific dataset, testing on others. To further verify scalability, we probe attention heads using the BoolQ dataset and then apply them to recover pruned models on other datasets, including RTE, ARC-e, and ARC-c. Table 12 presents the results. Even without retraining, RestoreLCC successfully restores performance beyond the pruned baseline.

H Efficiency at Scale

Scalability to Larger Models. We conduct task-specific recovery on the BoolQ dataset using LLaMA-70B. As shown in Table 13, our proposed RESTORELCC continues to outperform the LoRA baseline.

Table 13: Performance comparison on LLaMA-70B.

Method	BoolQ
Pruned Model	84.70
LoRA	86.21
RestoreLCC	87.25

Latency and Throughput. We also measure inference latency and throughput on the BoolQ dataset after recovery. The delay ratio is defined as the total inference time of RestoreLCC relative to that of the original pruned model. This ratio is approximately 1.03, corresponding to only a $\sim 3\%$ slowdown, which is negligible. Thus, RestoreLCC maintains nearly the same inference speed as the pruned baseline while delivering significantly better performance.

I Compatibility with Quantized Models

Our method can also be effectively applied to heavily quantized models. We conducted experiments with 4-bit quantization on the pruned model (using Wanda for task-specific recovery) and report the recovery results in Table 14. As shown, the proposed RestoreLCC successfully recovers the performance of heavily quantized LLMs.

Table 14: Performance of RestoreLCC on 4-bit quantized models.

Data	BoolQ	RTE	ARC-e	ARC-c
4-bit Quantized Model	68.20	57.76	59.89	29.95
RestoreLCC (Ours)	72.97	68.23	61.03	33.02

J Effect of Probing Samples

We investigate the effect of probing sample size on recovery performance. The results are shown in Tables 15 and 16. For general recovery, 1,000 samples are sufficient to achieve stable accuracy (≈ 58.8). For task-specific recovery on the BoolQ dataset, only 200 samples are sufficient to reach an accuracy of approximately 76.

Table 15: Effect of sample size on **general recovery**.

Samples	100	200	500	1000 (reported)	3000
Mean Accuracy	57.63	58.14	58.69	58.83	58.81

Table 16: Effect of sample size on **BoolQ task-specific recovery**.

Samples	10	20	50	100 (reported)	200	500	1000
Accuracy	70.03	73.09	74.98	75.32	76.15	76.12	76.51

K Experimental Results on More LLMs

Implementation Details. The batch size is set to 8. The learning rate is $\{1e-4, 1e-5\}$. The max sequence length is 512. All experiments are conducted on a single H100 GPU. For LoRA and DoRA, we use the same settings: $\alpha = 16$ and rank = 8. Regarding the applied modules, we try two configurations: (1) ["v_proj", "o_proj"], which tunes only the head output matrices; and (2) ["q_proj", "k_proj", "v_proj", "o_proj"], which tunes all head matrices. For LoFiT, we experiment with 10%, 20%, and 30% of the heads and report the best results.

For each probing classifier, we use the formulation of $y = \sigma(Wm + b)$, where m is either m^+ or m^- , W is a weight matrix (with input dimension equal to the size of m and output dimension 1), b is a bias term, and σ denotes the sigmoid function. The output $y \in [0, 1]$ represents the probability of contradiction or entailment. We employ a cross-entropy loss optimized with Adam (learning rate $= 1 \times 10^{-2}$). The dataset is divided into training and validation subsets in a 7:3 ratio, and models are trained for 100 epochs based on empirical observations. We report the probing accuracy on the validation set and **rank the importance of attention heads according to their probing accuracy**.

Tables 17 – 24 present experimental results across a broader range of LLMs from various families and sizes, including LLaMA-13B/30B, LLaMA-2-7B/13B, LLaMA-3-8B [2], Vicuna-7B-v1.5 [46], Tulu-2-7B [47], Qwen-3-8B/14B [48], and DeepSeek-R1-Qwen3-8B [49]. Our proposed RestoreLCC consistently achieves the highest mean accuracy in all cases, demonstrating strong generalizability and scalability.

Table 17: Performance of general recovery on zero-shot language modeling (PPL) and commonsense reasoning tasks (**accuracy**) using **LLaMA-13B**.

Method	PPL ↓	BoolQ ↑	RTE ↑	HellaSwag ↑	WinoGrande ↑	ARC-e ↑	ARC-c ↑	OBQA ↑	Mean ↑
Dense Model	5.09	77.92	70.40	59.92	72.85	77.31	46.42	33.20	62.57
<i>Unstructured Pruning at 50% Sparsity</i>									
Wanda	6.15	75.90	63.18	55.73	71.90	73.36	43.77	32.20	59.43
LoRA	6.08	78.56	63.18	59.15	71.11	74.75	46.33	34.80	61.13
DoRA	6.11	78.59	64.62	59.06	71.43	74.79	46.42	35.00	61.42
LoFiT	6.29	73.52	68.59	59.41	71.11	75.04	45.22	34.00	60.98
RestoreLCC (Ours)	6.08	78.07	70.04	58.50	73.01	75.97	46.42	35.20	62.46
<i>Semi-Structured Pruning (N:M=2:4) at 50% Sparsity</i>									
SparseGPT	9.08	71.59	55.23	48.00	69.93	67.47	35.24	25.80	53.32
LoRA	7.75	73.21	57.04	53.57	67.09	68.27	38.57	28.80	55.22
DoRA	7.72	73.18	57.76	53.69	66.46	68.22	38.57	28.40	55.18
LoFiT	8.10	73.15	59.93	53.66	67.32	68.81	39.08	29.60	55.94
RestoreLCC (Ours)	7.61	74.98	61.37	55.51	68.43	70.33	40.87	32.20	57.67
<i>Structured Pruning at 20% Sparsity</i>									
SlimGPT	5.99	76.33	62.45	58.06	73.72	75.38	42.32	33.20	60.21
LoRA	6.29	76.39	64.26	60.20	72.14	75.72	45.31	33.80	61.12
DoRA	6.10	77.03	65.70	60.86	73.16	76.68	46.08	34.20	61.96
LoFiT	6.51	76.01	66.79	59.93	72.14	74.75	44.45	34.60	61.24
RestoreLCC (Ours)	6.21	78.47	71.84	61.88	72.77	75.34	46.16	37.40	63.41

Table 18: Performance (**accuracy**) of general recovery on zero-shot commonsense reasoning tasks using **LLaMA-30B**.

Method	BoolQ ↑	RTE ↑	HellaSwag ↑	WinoGrande ↑	ARC-e ↑	ARC-c ↑	OBQA ↑	Mean ↑
Dense Model	82.75	67.15	63.32	75.93	80.43	52.90	36.00	65.50
<i>Unstructured Pruning at 60% Sparsity</i>								
Wanda	76.85	51.99	56.65	72.22	76.43	46.25	32.00	58.91
LoRA	78.69	57.04	61.13	71.59	78.03	47.95	36.40	61.55
DoRA	77.65	64.26	61.55	71.98	78.11	49.06	36.40	62.72
LoFiT	82.23	61.37	60.52	70.64	76.30	45.99	37.20	62.04
RestoreLCC (Ours)	83.58	66.06	61.63	73.09	77.61	49.74	36.60	64.04

Table 19: Performance (**accuracy**) of general recovery on zero-shot commonsense reasoning tasks using **LLaMA-2-7B**.

Method	BoolQ ↑	RTE ↑	HellaSwag ↑	WinoGrande ↑	ARC-e ↑	ARC-c ↑	OBQA ↑	Mean ↑
Dense Model	77.74	62.82	57.14	69.14	76.30	43.52	31.40	59.72
<i>Unstructured Pruning at 50% Sparsity</i>								
Wanda	76.51	53.43	52.54	68.59	72.31	39.16	31.00	56.22
DoRA	76.67	57.40	53.78	66.54	73.06	40.78	31.60	57.12
LoFiT	74.34	59.57	54.24	67.72	72.81	40.44	32.40	57.36
RestoreLCC (Ours)	75.23	63.90	55.08	67.56	73.48	41.21	32.80	58.47

Table 20: Performance (**accuracy**) of general recovery on zero-shot commonsense reasoning tasks using **LLaMA-2-13B**.

Method	BoolQ ↑	RTE ↑	HellaSwag ↑	WinoGrande ↑	ARC-e ↑	ARC-c ↑	OBQA ↑	Mean ↑
Dense Model	80.55	65.34	60.05	72.06	79.42	48.38	35.20	63.00
<i>Unstructured Pruning at 50% Sparsity</i>								
Wanda	81.13	59.21	57.01	70.96	75.84	42.92	32.00	59.87
DoRA	79.97	63.54	59.64	71.11	75.84	43.17	36.20	61.35
LoFiT	79.72	64.98	59.00	70.80	75.72	44.11	35.20	61.36
RestoreLCC (Ours)	81.68	64.98	58.97	72.22	77.69	45.31	34.60	62.21

Table 21: Performance (**accuracy**) of general recovery on zero-shot commonsense reasoning tasks using **LLaMA-3-8B**.

Method	BoolQ ↑	RTE ↑	HellaSwag ↑	WinoGrande ↑	ARC-e ↑	ARC-c ↑	OBQA ↑	Mean ↑
Dense Model	81.38	69.68	60.19	72.69	80.09	50.43	34.80	64.18
<i>Unstructured Pruning at 50% Sparsity</i>								
Wanda	75.81	59.93	50.74	70.80	71.42	40.27	29.00	56.85
DoRA	74.46	55.96	55.52	70.01	75.63	45.05	30.80	58.20
LoFiT	78.04	61.01	55.89	67.88	73.61	43.26	31.40	58.73
RestoreLCC (Ours)	73.15	64.62	55.09	70.24	76.52	45.39	29.40	59.20

Table 22: Performance (**accuracy**) of general recovery on zero-shot commonsense reasoning tasks using **Vicuna-7B-v1.5**.

Method	BoolQ ↑	RTE ↑	HellaSwag ↑	WinoGrande ↑	ARC-e ↑	ARC-c ↑	OBQA ↑	Mean ↑
Dense Model	80.92	63.90	56.43	69.61	75.59	43.17	33.00	60.37
<i>Unstructured Pruning at 50% Sparsity</i>								
Wanda	80.00	54.87	53.20	68.35	71.46	40.44	29.20	56.79
DoRA	77.09	56.68	54.13	67.96	70.58	41.81	31.60	57.12
LoFiT	77.16	57.76	54.08	67.40	70.54	41.72	32.00	57.24
RestoreLCC (Ours)	80.83	55.96	53.86	68.90	74.41	41.72	32.40	58.30

Table 23: Performance (**accuracy**) of general recovery on zero-shot commonsense reasoning tasks using **Tulu-2-7B**.

Method	BoolQ ↑	RTE ↑	HellaSwag ↑	WinoGrande ↑	ARC-e ↑	ARC-c ↑	OBQA ↑	Mean ↑
Dense Model	82.48	65.34	58.90	69.85	80.39	48.81	33.40	62.74
<i>Unstructured Pruning at 50% Sparsity</i>								
Wanda	81.07	55.60	54.31	68.43	74.79	42.58	31.20	58.28
DoRA	78.65	67.15	54.09	67.72	71.38	43.09	32.00	59.15
LoFiT	78.59	68.23	54.06	68.19	71.30	43.43	31.80	59.37
RestoreLCC (Ours)	80.46	67.87	54.56	69.14	74.37	43.17	31.20	60.11

Table 24: Performance (**accuracy**) of general recovery on zero-shot commonsense reasoning tasks using **Qwen3-8B**, **Qwen3-14B**, and **DS-R1-8B**.

Model	BoolQ ↑	RTE ↑	HellaSwag ↑	WinoGrande ↑	ARC-e ↑	ARC-c ↑	OBQA ↑	Mean ↑
<i>Qwen3-8B Unstructured Pruning at 50% Sparsity</i>								
Dense	86.57	78.34	57.10	67.80	83.54	55.80	31.00	65.74
Wanda	84.86	70.04	50.12	69.46	80.22	50.85	28.20	61.96
DoRA	86.30	77.26	56.13	68.82	82.37	53.58	31.80	65.18
Ours	87.13	79.06	57.29	70.48	84.68	57.59	32.60	66.98
<i>Qwen3-14B Unstructured Pruning at 50% Sparsity</i>								
Dense	89.33	77.62	60.97	73.01	84.22	58.62	35.00	68.40
Wanda	87.43	72.92	57.69	69.77	83.50	55.97	33.60	65.84
DoRA	88.07	78.70	59.87	74.66	84.09	57.85	34.50	68.25
Ours	89.57	80.14	60.70	74.74	86.15	60.15	35.00	69.49
<i>DS-R1-8B Unstructured Pruning at 50% Sparsity</i>								
Dense	85.87	80.51	58.54	67.01	80.18	51.71	31.60	65.06
Wanda	82.48	77.26	51.53	67.17	75.67	48.12	28.40	61.52
DoRA	84.80	74.37	56.84	68.35	77.90	50.09	32.30	63.52
Ours	87.22	74.01	57.41	69.46	83.84	55.63	33.80	65.91