Revisiting LoRA through the Lens of Parameter Redundancy: Spectral Encoding Helps

Jiashun Cheng[*], **Aochuan Chen**[*], **Nuo Chen, Ziqi Gao, Yuhan Li**
Jia Li[†], **Fugee Tsung**

The Hong Kong University of Science and Technology (Guangzhou)
The Hong Kong University of Science and Technology
jchengak@connect.ust.hk, jialee@ust.hk

Abstract

Low-Rank Adaptation (LoRA) has emerged as a prominent technique for fine-tuning large foundation models. Despite its successes, the substantial parameter redundancy, which limits the capacity and efficiency of LoRA, has been recognized as a bottleneck. In this work, we systematically investigate the impact of redundancy in fine-tuning LoRA and reveal that reducing density redundancy does not degrade expressiveness. Based on this insight, we introduce Spectral-encoding Low-Rank Adaptation (SeLoRA), which harnesses the robust expressiveness of spectral bases to re-parameterize LoRA from a sparse spectral subspace. Designed with simplicity, SeLoRA enables seamless integration with various LoRA variants for performance boosting, serving as a scalable plug-and-play framework. Extensive experiments substantiate that SeLoRA achieves greater efficiency with fewer parameters, delivering superior performance enhancements over strong baselines on various downstream tasks, including commonsense reasoning, math reasoning, and code generation.

1 Introduction

In recent years, Large Foundation Models (LFMs), have showcased exceptional generalization capabilities, greatly improving performance in a wide array of tasks across natural language processing (NLP) (Brown et al., 2020; Touvron et al., 2023a), computer vision (CV) (Radford et al., 2021; Kirillov et al., 2023), and other fields (Azad et al., 2023; Li et al., 2024c, 2025). Typically, adapting these general models for specific downstream tasks requires full fine-tuning, which involves retraining all model parameters and can pose significant challenges, particularly in resource-limited environments. To address this issue, Parameter-efficient fine-tuning (PEFT) techniques (Mangrulkar et al., 2022), have been developed, offering more feasible alternatives. Among these, Low-Rank Adaptation (LoRA) (Hu et al., 2021), which decomposes the weight changes into the product of two low-rank matrices, has stood out for its effectiveness and simplicity.

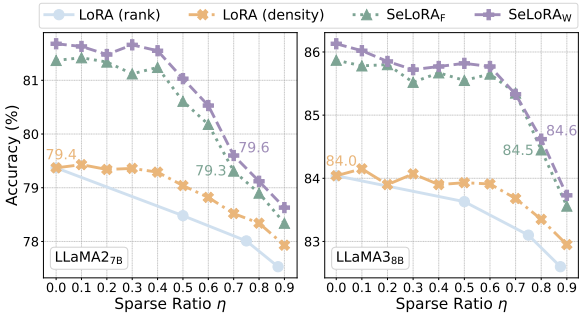


Figure 1: Average accuracy of commonsense reasoning tasks (**y-axis**) across various sparse ratios η (**x-axis**) on trainable parameters. Masking a significant proportion of LoRA's parameters still retains comparable performance, while naively learning with reduced ranks leads to clear degradation.

Despite its success, recent studies still indicate redundancy in LoRA's parameters. Early work (He et al., 2022) conducts a systematic exploration into such redundancy in encoder-only models. For decoder-only LFMs, subsequent approaches (Kopiczko et al., 2023; Renduchintala et al., 2023) reduce the parameters by sharing frozen LoRA matrices across layers and modules while learning small scaling vectors with only slight performance losses. More recent efforts (Bałazy et al., 2024; Li et al., 2024a; Sehanobish et al., 2024) further minimized the parameter usage by re-parameterizing LoRA via matrix decomposition. Concurrently, another line of research (Ding et al., 2023; Jiang et al., 2024a, 2025) unveils that removing the redundant components from the fine-tuned LoRA's parameters can yield further performance gains. However, despite these

[*]Equal contribution.
[†]Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2025, pages 2701–2718
July 27 - August 1, 2025

advancements, a systematic understanding of how redundancy affects LoRA in the context of decoder-only LFMs during the fine-tuning phase remains absent.

To address this gap, drawing inspiration from prior works (He et al., 2022; Yu et al., 2024), we conduct a comprehensive investigation into the impacts of redundancy in LoRA from two perspectives: (1) **Rank redundancy** - Fine-tuning LoRA with reduced ranks; (2) **Density redundancy** - Masking a portion of LoRA's parameters to zero while fine-tuning the remainder at fixed rank. As presented in Figure 1, we observe that reducing the rank alone leads to notable performance degradation. Conversely, when fine-tuning LoRA at an appropriately chosen rank, the introduction of sparsity, achieved by masking up to 60% of the parameters, demonstrates performance on par with that of the fully parameterized LoRA. It highlights that adequately reducing density redundancy does not compromise its expressiveness and we term this phenomenon the *sparsity property* of LoRA. These findings suggest that LoRA's parameters are not fully utilized, leaving room for further enhancement. In light of these observations, a question is naturally raised:

How can we unleash the potential of LoRA utilizing its sparsity property?

This question aligns closely with the principles of sparse learning (Han et al., 2015a), which seeks to acquire expressive information while necessitating fewer learnable parameters. Despite the success of the predominant pruning techniques (Han et al., 2015b; Frankle and Carbin, 2018), recent studies (Zhao et al., 2024; Gu et al., 2024) demonstrate effective pruning during LoRA fine-tuning often requires intricate strategies. In contrast, spectral encoding of weight matrices (Koutnik et al., 2010; Van Steenkiste et al., 2016), which enables expressive representation learning with sparse spectral entries (Wolter et al., 2020; Irie and Schmidhuber, 2021), offers a more straightforward yet powerful alternative.

Motivated by these findings, we introduce Spectral-encoding Low-Rank Adaptation (SeLoRA), a novel approach that harness spectral transformations to re-parameterize low-rank matrices as the spatial equivalents of spectral components. Essentially, SeLoRA selectively learns only a sparse set of spectral components at the predefined globally shared spectral locations, where inverse spectral transformation is then applied to derive the adaptation matrices in the spatial domain. Designed with simplicity and flexibility, SeLoRA naturally accommodates various choices of spectral bases, making it highly flexible. Furthermore, its lightweight and modular nature enables seamless integration as a plug-and-play framework compatible with a variety of LoRA variants. Our evaluation of SeLoRA on advanced LFMs like LLaMA families across diverse instruction-tuning tasks demonstrates its enhanced capacity and efficiency, as exemplified in Figure 1. Extensive in-depth analyses are further conducted to substantiate the robustness of SeLoRA, confirming its advantages and practical utility.

In summary, our contributions are as follows:

- Our investigation into the impact of redundancy in LoRA reveals the *sparsity property*, where optimizing only a sparse subset of tunable parameters preserves comparable expressive power.
- Based on this insight, we introduce SeLoRA, a novel extension of LoRA with expressive spectral re-parameterization, enhancing performance while reducing parameter overhead.
- We rigorously evaluate SeLoRA across multiple domains, validating its effectiveness and efficiency. A comprehensive analysis further elucidates the impact of its designs.

2 Methodology

In this section, we first outline the basic properties of LoRA fine-tuning. We then present our proposed Spectral-encoding Low-Rank Adaptation (SeLoRA), which takes advantage of the *sparsity property* along with spectral transformations for effective representation learning. The overall framework is presented in Figure 2.

2.1 Background

Low-Rank Adaptation. LoRA (Hu et al., 2021) assumes parameter changes typically occur within a low-rank space (Aghajanyan et al., 2021) and proposes to use the product of two low-rank matrices $\mathbf{B} \in \mathbb{R}^{d_1 \times r}$ and $\mathbf{A} \in \mathbb{R}^{r \times d_2}$ as the incremental weight update $\Delta\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$. For pre-trained weight $\mathbf{W}_0 \in \mathbb{R}^{d_1 \times d_2}$, LoRA is expressed as

$$\mathbf{W}' = \mathbf{W}_0 + \Delta\mathbf{W} = \mathbf{W}_0 + \mathbf{BA}, \quad (1)$$

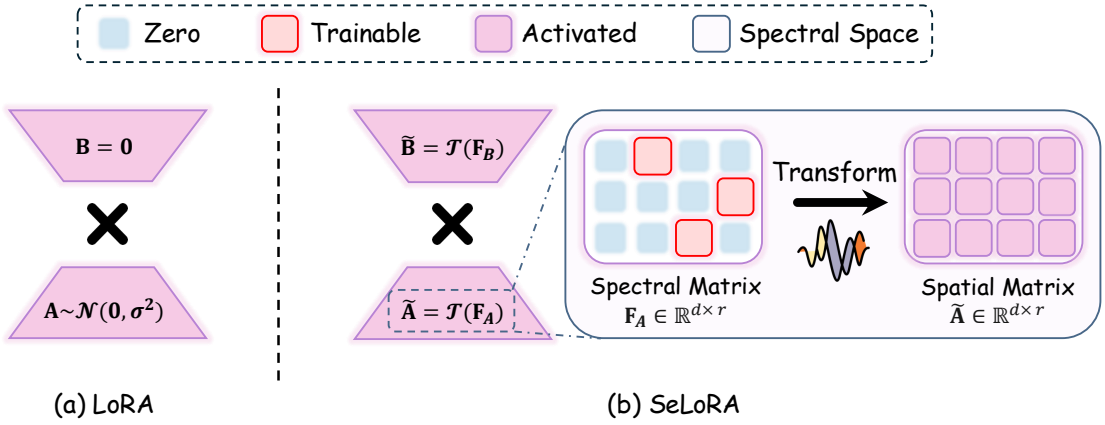


Figure 2: An overview of the schematic comparison between LoRA and our proposed SeLoRA. In contrast to fully parameterized LoRA, SeLoRA employs spectral transformations on sparse spectral components to represent weight matrices.

where $\mathbf{A}, \mathbf{B}$ are trainable with the rank $r \ll \{d_1, d_2\}$ while $\mathbf{W}_0$ is frozen during fine-tuning. Without loss of generality, we assume $d = d_1 = d_2$ for notation-wise simplicity.

2.2 LoRA with Spectral Encoding

As previously discussed, our objective is to enhance the expressiveness of $\mathbf{A}$ and $\mathbf{B}$ while learning with reduced density redundancy, which aligns closely with the foundational principle of sparse learning. Upon revisiting prior successes, we adopt the spectral encoding for weight matrices, a well-established approach that balances simplicity and expressiveness (Wolter et al., 2020; Irie and Schmidhuber, 2021).

Essentially, our approach centers on re-parameterizing the adaptation matrices, termed $\tilde{\mathbf{A}} \in \mathbb{R}^{r \times d}$ and $\tilde{\mathbf{B}} \in \mathbb{R}^{d \times r}$, as the spatial recovery of sparse spectral components, while retaining LoRA's update schema:

$$\mathbf{W}' = \mathbf{W}_0 + \Delta\mathbf{W} = \mathbf{W}_0 + \tilde{\mathbf{B}}\tilde{\mathbf{A}}. \quad (2)$$

To achieve this, we first introduce the sparse ratio $\eta \in (0, 1)$, representing the proportion of masking parameters relative to the total elements in the low-rank matrix. We then randomly initialize an index set Ω, where $|\Omega| = \lfloor (1-\eta) \cdot rd \rfloor$, specifying the locations of the learnable spectral components shared across the low-rank matrices. Subsequently, we define the corresponding sparse spectral matrices as $\mathbf{F}_A \in \mathbb{R}^{r \times d}$ and $\mathbf{F}_B \in \mathbb{R}^{d \times r}$. For clarity, we denote $\mathbf{F}_A(u, v)$ as the element at index (u, v). The matrix $\mathbf{F}_A$ is constructed such that its entries $\mathbf{F}_A(u, v)$ are learnable if $(u, v) \in \Omega$, while all other entries are fixed to zero. The same principle applies to the construction of $\mathbf{F}_B$. Given such, the low-rank matrices are derived from their spatial counterpart by applying various inverse spectral transformations:

$$\tilde{\mathbf{A}} = \mathcal{T}(\mathbf{F}_A), \tilde{\mathbf{B}} = \mathcal{T}(\mathbf{F}_B), \quad (3)$$

where $\mathcal{T}(\cdot)$ represents transformation function.

As demonstrated, the core of this framework is leveraging efficient and expressive spectral bases to enhance learning capacity. To this end, we explore two widely adopted and well-established spectral transformations as representative instantiations of our approach. We exemplify these two variants via the computation of $\tilde{\mathbf{A}}$, and $\tilde{\mathbf{B}}$ is computed by applying the identical procedure.

Fourier Encoding. Known for the capacity to capture high-fidelity information from sparse spectral components (Duarte and Baraniuk, 2013; Vlaardingerbroek and Boer, 2013), the Fourier basis emerges as an indispensable choice. In light of that, we employ the discrete inverse 2D Fourier transform, denoted as $\mathcal{F}^{-1}(\cdot)$, and retain only the real part of the transformed results to simplify computations. Accordingly, the resulting method is

defined as the following:

$$\begin{aligned}\tilde{\mathbf{A}}(j,k) &= \text{Re}[\mathcal{F}^{-1}(\mathbf{F}_A)] \\ &= \text{Re}\big[\sum_{u,v} \mathbf{F}_A(u,v)\, e^{i2\pi(\frac{u}{r}j+\frac{v}{d}k)}\big],\end{aligned} \tag{4}$$

where i denotes the imaginary unit and the transformation function is formulated as $\mathcal{T}(\cdot) = \text{Re}[\mathcal{F}^{-1}(\cdot)]$.

Wavelet Encoding. Upon revisiting prior successes, the Wavelet basis also presents as a promising choice (Van Steenkiste et al., 2016). Unlike the Fourier transform, which primarily captures global frequency information, the Wavelet transform provides a more localized and hierarchical reconstruction of information with flexible filter options. In this paper, we adopt the discrete inverse 2D Wavelet transform, denoted as $\mathcal{T}(\cdot) = \mathcal{W}^{-1}(\cdot)$, that is widely utilized in image restoration.

Following the standard practice, we first decompose the spectral matrix $\mathbf{F}_A$ into four spectral components:

$$\mathbf{F}_A = \begin{bmatrix} \mathbf{F}_A^a & \mathbf{F}_A^h \\ \mathbf{F}_A^v & \mathbf{F}_A^d \end{bmatrix}, \tag{5}$$

where $\mathbf{F}_A^a \in \mathbb{R}^{\frac{r}{2}\times\frac{d}{2}}$ denotes the approximation coefficients, while the remaining $\mathbf{F}_A^o \in \mathbb{R}^{\frac{r}{2}\times\frac{d}{2}}$, $o \in \{h,v,d\}$ correspond to detail coefficients of different directions. Let $O = \{a,h,v,d\}$ denote the set of coefficient types, the transformation is then expressed as:

$$\begin{aligned}\tilde{\mathbf{A}}(j,k) &= \mathcal{W}^{-1}(\mathbf{F}_A) \\ &= \sum_{o\in O}\sum_{u,v} \mathbf{F}_A^o(u,v)\, \psi_{u,v}^o(j,k),\end{aligned} \tag{6}$$

with

$$\psi_{u,v}^o(j,k) = \frac{1}{\sqrt{2}}\psi^o(\frac{j}{2}-u, \frac{k}{2}-v), \tag{7}$$

where $\psi^o(\cdot,\cdot)$ denotes the basis function of wavelet filters. For instance, when employing the Haar wavelet, $\psi^o(\cdot,\cdot)$ reduces to an indicator function taking values in $\{-1,1\}$, active within the range $2u \le j \le 2u+1$, $2v \le k \le 2v+1$. The choice of wavelet filter directly influences the balance between smoothness and detail preservation in the transformed representation. The specific construction details of wavelet filters are provided in the Appendix B. Unless otherwise specified, we use the Haar wavelet as the default basis in this study.

2.3 Discussions

Initialization Strategies. Matrix initialization with consistent variance (Glorot and Bengio, 2010) is crucial for maintaining numerical stability and accelerating convergence. However, unlike LoRA, directly initializing the spectral space in SeLoRA can lead to suboptimal variance in spatial space due to the involvement of spectral transformations. For matrix $\tilde{\mathbf{A}}$, we first employ Xavier (Glorot and Bengio, 2010) or Kaiming initialization (He et al., 2015) to the spectral matrix $\mathbf{F}_A$ and an auxiliary matrix $\mathbf{A}' \in \mathbb{R}^{r\times d}$. Next, we scale $\mathbf{F}_A$ to ensure $\text{Var}(\tilde{\mathbf{A}}) = \text{Var}(\mathcal{T}(\mathbf{F}_A)) = \text{Var}(\mathbf{A}')$. In contrast, matrix $\tilde{\mathbf{B}}$ is initialized to zeros following the standard practice of LoRA (Hu et al., 2021). We employ Kaiming initialization by default unless specially stated.

Extension to LoRA Variants. Unlike introducing an entirely new learning paradigm, our method capitalizes on the modular nature of spectral encoding for weight matrices, enabling seamless integration as a plug-in within various LoRA variants, including DoRA (Liu et al., 2024b), X-LoRA (Buehler and Buehler, 2024) and HiRA (Huang et al., 2025). Moreover, by leveraging fast spectral transformation (Nussbaumer and Nussbaumer, 1982; Wolter et al., 2024), our approach introduces only minimal additional computational cost during training while incurring no extra overhead during inference, making it a highly efficient and scalable solution. A more detailed empirical evaluation is provided in Section 3.

3 Experiments

3.1 Setup

Tasks. Our goal is to provide a rich picture of how our proposed approach performs in different scenarios. Our experiments generally align with those reported by Liu et al. (2024b) and Biderman et al. (2024). We apply all the methods to instruction fine-tuning and evaluated their performance on conventional commonsense reasoning and two challenging tasks - mathematical reasoning and code generation. To provide a more rigorous evaluation, we adopt `alphaca-chat` prompt template throughout the training and assessments, more details are provided in Appendix D.

• **Commonsense Reasoning.** We utilize Commonsense170K (Hu et al., 2023) as the training data, a collection of multiple-choice question-

Methods	Params (%)	Time	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	Avg.
GPT-3.5-turbo											
Zero-shot	-	-	73.1	85.4	68.5	78.5	66.1	89.8	79.9	74.8	77.0
LLaMA2$_{7B}$											
$\mathcal{LS}$-LoRA	0.50	7.7h	72.2	82.6	80.2	89.7	83.2	84.2	69.6	82.7	80.6
LoRETTA$_{rep}$	0.53	8.5h	71.9	82.4	80.0	90.1	83.3	84.4	69.1	82.3	80.4
LoRA	0.83	7.4h	71.4	81.4	79.6	87.8	83.2	82.6	67.5	81.5	79.4
SeLoRA$_F$	0.50	7.6h	72.8	83.4	80.0	90.9	83.7	85.4	70.5	**83.4**	81.3 (↑ 1.9)
SeLoRA$_W$	0.50	7.5h	72.9	83.3	80.5	**92.1**	83.5	85.3	**71.9**	83.2	81.6 (↑ 2.2)
DoRA	0.84	12.2h	71.8	83.1	77.1	90.1	82.8	84.1	69.5	82.4	80.1
SeDoRA$_F$	0.51	12.7h	72.5	83.4	80.2	91.1	84.2	85.3	71.7	83.2	81.5 (↑ 1.4)
SeDoRA$_W$	0.51	12.4h	**73.7**	83.8	**80.6**	92.0	**84.6**	**86.0**	71.6	83.0	**81.9** (↑ **1.8**)
HiRA	0.83	11.7h	72.0	82.1	78.7	86.7	79.6	84.3	70.1	79.6	79.1
SeHiRA$_F$	0.50	12.1h	71.8	83.1	79.3	89.9	81.2	84.5	70.0	80.8	80.1 (↑ 1.0)
SeHiRA$_W$	0.50	11.9h	73.2	**84.2**	79.9	90.8	83.2	**86.0**	70.8	81.6	81.2 (↑ 2.1)
LLaMA3$_{8B}$											
$\mathcal{LS}$-LoRA	0.28	8.1h	74.2	88.0	79.6	94.7	85.2	90.1	79.1	85.4	84.5
LoRETTA$_{rep}$	0.30	9.0h	74.5	87.8	79.7	94.6	85.4	89.7	78.2	87.0	84.6
LoRA	0.70	7.8h	74.0	88.2	80.4	94.0	85.5	87.5	78.1	84.0	84.0
SeLoRA$_F$	0.28	8.1h	74.4	89.0	81.3	95.6	**87.5**	90.6	80.3	87.0	85.7 (↑ 1.7)
SeLoRA$_W$	0.28	8.0h	76.0	89.3	80.6	95.9	86.7	91.0	81.4	86.6	85.9 (↑ 1.9)
DoRA	0.71	12.8h	74.9	88.9	80.2	95.5	85.6	90.5	80.4	85.8	85.2
SeDoRA$_F$	0.28	13.3h	75.7	89.3	81.2	95.8	86.9	91.8	81.2	86.8	86.1 (↑ 0.9)
SeDoRA$_W$	0.28	13.0h	**76.2**	**89.7**	**81.4**	**96.0**	**87.5**	91.2	**82.0**	87.8	**86.5** (↑ **1.3**)
HiRA	0.70	12.3h	74.6	88.3	79.7	95.3	85.3	90.8	80.3	88.0	85.3
SeHiRA$_F$	0.28	12.8h	75.7	89.4	81.1	95.5	86.5	91.6	80.8	87.0	86.0 (↑ 0.7)
SeHiRA$_W$	0.28	12.5h	76.0	89.1	80.8	95.9	87.3	**92.2**	81.8	**88.8**	**86.5** (↑ **1.2**)

Table 1: Comparison of LLaMA2$_{7B}$ and LLaMA3$_{8B}$ against various methods on eight commonsense datasets. The results of all baseline methods are reproduced by implementing their official codebase. The highest scores within each baseline method are highlighted in blue and the best results of each LLM are marked in **bold**. Training time is measured on 1x A100 GPU.

answer (QA) pairs derived from the training sets of eight sub-tasks: BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-e, ARC-c (Clark et al., 2018), and OBQA (Mihaylov et al., 2018). For evaluation, we employ greedy search for answer generation and assess the model's performance on the test sets of each dataset. Consistent with the protocols in Hu et al. (2023) and Liu et al. (2024b), the model's response is recorded as the first occurrence of the answer keywords in the generated output.

• **Mathematical Reasoning.** We employ MetaMathQA (Yu et al., 2023) for model fine-tuning, which consists of 395K mathematical QA pairs evolved from GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2020). The evaluation is performed on the respective test sets of GSM8K and MATH, both of which require chain-of-thought reasoning (Wei et al., 2022) to reach the final answer. Following the evaluation protocol outlined in Yu et al. (2023), we assess performance by measuring the accuracy of the final numeric answer generated through greedy search.

• **Code Generation.** We utilize Magicoder-Evol-Instruct-110k (Wei et al., 2024) as the training dataset, a programming QA collection that has been reproduced and decontaminated from WizardCoder (Luo et al., 2024). The fine-tuned models are evaluated on the HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) benchmarks. To ensure comprehensive evaluation, we also assess the models on HumanEval+ and MBPP+ using the evaluation protocol outlined in EvalPlus (Liu et al., 2024a) and report the Pass@1 metric for each benchmark.

Baselines. We choose LoRA (Hu et al., 2021), DoRA (Liu et al., 2024b) and HiRA (Huang et al., 2025) as the integrable baselines, while $\mathcal{LS}$-

Methods	Params (%)	GPU (GB)	Math			Code				
			GSM8k	MATH	Avg.	HumanEval	HumanEval+	MBPP	MBPP+	Avg.
***LLaMA2*$_{7B}$**										
Zero-shot	-	-	7.3	1.1	4.2	11.0	9.8	30.2	24.1	18.8
FourierFT	0.66	47.8	61.5	10.9	36.4	31.6	27.6	37.4	32.4	32.2
LoRA	0.83	42.2	60.5	11.7	36.1	32.1	28.4	35.8	30.8	31.8
SeLoRA$_F$	0.66	42.9	61.4	12.5	37.0 (↑ 0.9)	31.2	26.9	38.9	32.5	32.4 (↑ 0.6)
SeLoRA$_W$	0.66	42.9	62.4	13.7	38.1 (↑ 2.0)	**35.2**	29.1	40.1	34.8	**34.8** (↑ **3.0**)
DoRA	0.84	56.4	61.2	12.1	36.7	32.9	28.7	39.9	33.1	33.7
SeDoRA$_F$	0.67	57.1	62.0	12.8	37.4 (↑ 0.9)	31.5	27.8	**41.6**	**35.8**	34.2 (↑ 0.5)
SeDoRA$_W$	0.67	57.0	**63.0**	**14.1**	**38.6** (↑ **1.9**)	33.5	**29.9**	41.0	34.7	**34.8** (↑ **1.1**)
***LLaMA3*$_{8B}$**										
Zero-shot	-	-	33.1	5.3	19.2	33.5	29.3	61.4	51.6	44.0
FourierFT	0.42	55.9	77.8	28.9	53.3	62.9	56.1	60.8	53.1	58.2
LoRA	0.70	51.6	77.2	28.2	52.7	57.9	52.8	64.8	55.3	57.7
SeLoRA$_F$	0.42	52.1	77.9	29.4	53.7 (↑ 1.0)	63.4	56.7	61.4	53.7	58.8 (↑ 1.1)
SeLoRA$_W$	0.42	52.0	80.3	29.8	55.1 (↑ 2.4)	59.3	55.6	**66.1**	**56.6**	59.4 (↑ 1.7)
DoRA	0.71	64.3	78.0	28.7	53.4	60.4	56.1	61.9	53.7	58.0
SeDoRA$_F$	0.42	64.7	78.9	29.2	54.1 (↑ 0.7)	62.8	**57.3**	61.6	53.3	58.8 (↑ 0.8)
SeDoRA$_W$	0.42	64.7	**80.4**	**30.3**	**55.4** (↑ **2.0**)	**63.4**	56.1	63.5	55.3	**59.6** (↑ **1.6**)

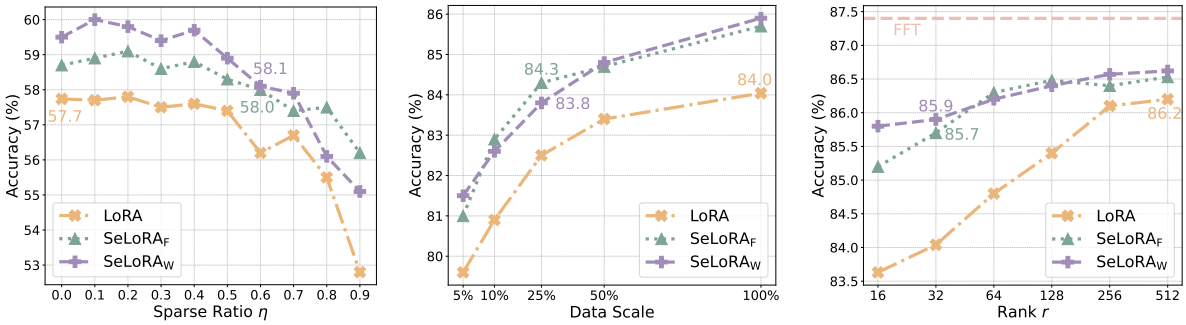
Table 2: Comparison of LLaMA2$_{7B}$ and LLaMA3$_{8B}$ against various methods on mathematical reasoning and code generation. We report the overall accuracy for mathematical reasoning and Pass@1 for code generation. The highest scores within each baseline method are highlighted in blue and the best results of each LLM are marked in **bold**. Memory is measured on 1x A100 GPU for mathematical reasoning per the micro-batch size in configurations.

LoRA (He et al., 2022), LoRETTA$_{rep}$ (Yang et al., 2024) and FourierFT (Gao et al., 2024) are selected as PEFT methods with sparse re-parameterization. All experiments are conducted using two open-source LFMs: LLaMA2$_{7B}$ (Touvron et al., 2023b) and LLaMA3$_{8B}$ (Dubey et al., 2024). Following common practice (Kopiczko et al., 2023), we used the base versions instead of the instruction-tuned ones.

Implementation Details. In line with the setup suggested in Hu et al. (2023), we fix all baseline models to a rank of $r = 32$ and set $\alpha = 64$ while conducting hyperparameter search on learning rates employing AdamW optimizer (Loshchilov and Hutter, 2019) during fine-tuning. To ensure fairness, we reproduce the results of LoRA, DoRA, FourierFT, $\mathcal{LS}$-LoRA and LoRETTA$_{rep}$ using their official implementations while implementing HiRA ourselves, as its official code is not publicly available, based on the optimal configurations reported in their original papers. We apply PEFTs to the query, key, and value modules in attention ($\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v$) and two feed-forward networks ($\mathbf{W}_{up}, \mathbf{W}_{down}$). For commonsense reasoning tasks, the LLMs are fine-tuned for 3 epochs, setting η to 0.4 for LLaMA2$_{7B}$ and 0.6 for LLaMA3$_{8B}$. For more complex tasks, we decrease η to 0.2 for LLaMA2$_{7B}$ and 0.4 for LLaMA3$_{8B}$ with 2 fine-tuned epochs. More details are provided in Appendix C.

3.2 Commonsense Reasoning

Table 1 provides a comprehensive overview of the general performance across different backbone architectures and baseline methods. In comparison with $\mathcal{LS}$-LoRA and LoRETTA$_{rep}$, which employ sparse re-parameterizations, both variants of SeLoRA achieve better performance under a similar parameter budget while consuming less training time. As evident from the results, our proposed methods consistently surpass all integrable baselines in terms of average accuracy. In particular, both SeLoRA$_F$ and SeLoRA$_W$ exhibit significant improvements over LoRA, achieving an average accuracy gain of approximately +2.0 across the LLaMA families. While integrating our methods into more advanced baselines such as DoRA and HiRA results in slightly reduced gains, our methods still deliver notable improvements, reaching up to +2.1 on LLaMA2$_{7B}$ and +1.3 on LLaMA3$_{8B}$. These results underscore the adaptability of our approach as a plug-and-play framework. Moreover, our methods enhance learning capacity while requiring significantly fewer trainable parameters, all without increasing training time, underscoring their efficiency. Additionally, we observe that pa-



(a) (b) (c)

Figure 3: (a) Average performance of code generation with varying sparse ratio η on LLaMA3$_{8B}$; (b) Average performance of commonsense reasoning using different scales of training dataset; (c) Average performance of LLaMA3$_{8B}$ when the rank r increases.

rameterizing the weight matrix using wavelet transformations generally leads to more pronounced performance gains. This can be attributed to the wavelet-based representation in better balancing smoothness with detail preservation.

3.3 More Challenging Tasks

Table 2 presents the overall results for mathematical reasoning and code generation. The zero-shot performances for code generation are in line with EvalPlus leaderboard[1]. Consistent with our findings in commonsense reasoning, our approaches consistently surpass LoRA and DoRA among LLaMA families for more challenging tasks. In comparison with FourierFT, which also leverages Fourier transformations, SeLoRAF achieves comparable performance under a similar parameter budget while exhibiting significantly lower memory consumption. This advantage arises because $\Delta\mathbf{W}$ of FourierFT has the same shape as the pre-trained weights, whereas SeLoRA$_F$ retains the low-rank update structure. Moreover, wavelet encoding continues to provide stable and substantial improvements, whereas fourier encoding exhibits greater performance fluctuations with a reduced level of enhancement. Particularly, wavelet-based variants achieve average improvements of +2.1 and +1.9 across all configurations for mathematical reasoning and code generation respectively. These results, in conjunction with our observations in commonsense reasoning, substantiate the effectiveness of our proposed method across diverse reasoning and generation tasks.

[1]https://evalplus.github.io/leaderboard.html

4 In-depth Analyses

In this section, we conduct a variety of quantitive analyses on our proposed approach to assess its robustness and generalizability. The experimental implementations adhere to the setup in Section 3.1 unless otherwise specified.

Sparsity Utilization. To further investigate the impact of spectral encoding in leveraging the *sparsity property*, we evaluate SeLoRA under two scenarios: **(1)** varying the sparse ratio η from 0.0 to 0.9 and **(2)** adjusting the rank r while keeping parameter counts fixed.

As presented in Figure 1 and Figure 3(a), the maximum sparse ratio that maintains comparable expressive power varies across tasks and backbone models. More challenging tasks and less expressive backbones have smaller values and exhibit lower redundancy. Nevertheless, both SeLoRA variants consistently outperform LoRA across all tasks and backbone architectures. Notably, SeLoRA maintains performance parity with LoRA even at relatively high sparse ratios, such as $\eta = 0.8$ for commonsense reasoning and 0.6 for code generation in LLaMA3$_{8B}$. Moreover, the performance gains are already pronounced at high sparse ratios, which can be attributed to the intrinsic properties of spectral transformations that allow high-quality reconstruction with extremely sparse elements.

Furthermore, as illustrated in Figure 4, pruning-based $\mathcal{LS}$-LoRA enhances performance by utilizing higher ranks through masked adaptation under fixed parameter budgets. SeLoRA further amplifies these gains by combining spectral encoding with masking, consistently delivering superior results. These findings jointly substantiate the significance

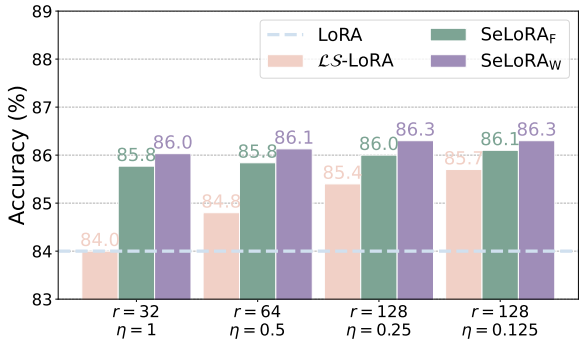


Figure 4: Performance distribution with different sparse learning mechanisms on LoRA modules for LLaMA3$_{8B}$ on commonsense reasoning. $\mathcal{LS}$-LoRA achieves a higher rank r while maintaining the same parameter budget against trivial LoRA via a proper sparse ratio η.

Methods	Bases	Task			
		Common	Math	Code	Avg.
LoRA	-	83.9	52.7	57.6	64.7
SeLoRA$_W$	Haar	85.9	55.1	59.4	66.8
	Db	85.9	**55.4**	59.1	66.8
	Bior	85.9	54.8	59.5	66.7
	Coif	**86.2**	55.2	**59.8**	**67.0**

Table 3: Peformance variations with different wavelet filters on LLaMA3$_{8B}$.

of employing spectral encoding in SeLoRA for effectively harnessing the *sparsity property*.

Data Scalability. We explore the influence of training data size on the performance of our approach. We experiment with the LLaMA3$_{8B}$ employing the rank of $r = 32$. For comprehensiveness, the examination involves random sampling of 5%, 10%, 25%, and 50% instances from the training data for the commonsense reasoning task. As illustrated in Figure 3(b) and Table 8, both variants of our approach exhibit steadily increased performance with an increase in training data volume. Impressively, with just 25% of the training data, SeLoRA outperforms LoRA even when the latter utilizes the entire dataset, highlighting SeLoRA's exceptional efficiency in leveraging training data for performance improvement.

Rank Scalability. We assess the sensitivity of our proposed approach against different ranks r with an increasing rank sequence [16, 32, 64, 128, 256, 512], as higher ranks are known to improve performance on complex tasks Biderman et al. (2024). In alignment with Section 3.1, we conduct evaluations on LLaMA3$_{8B}$ for commonsense reasoning

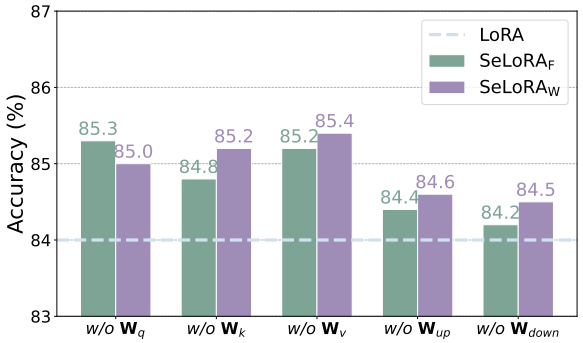


Figure 5: Performance distribution with the removal of spectral encoding among different modules for LLaMA3$_{8B}$ on commonsense reasoning.

while keeping $\eta = 0.6$ for SeLoRA. As illustrated in Figure 3(c) and Table 9, SeLoRA exhibits a similar trend to LoRA, benefiting from an expanded learning space and achieving progressively better performance as r increases while maintaining sparsity. Notably, we observe diminishing performance gains for LoRA beyond $r = 256$, whereas SeLoRA reaches comparable performance at just $r = 32$. This indicates that spectral encoding effectively enhances LoRA's efficiency, enabling it to realize the benefits of higher ranks with significantly fewer trainable parameters.

Module Sensitivity. Understanding which modules benefit most from spectral encoding is crucial for demystifying SeLoRA's effectiveness. Thereafter, we assess its impact by selectively removing spectral encoding from individual modules within $\{\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v, \mathbf{W}_{up}, \mathbf{W}_{down}\}$ yet keeping all other modules unchanged. As presented in Figure 5, the removal of spectral encoding from the feed-forward networks ($\mathbf{W}_{up}, \mathbf{W}_{down}$) leads to a substantial performance drop, whereas its absence in the attention components ($\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v$) results in relatively less degradation. This indicates parameters of feed-forward networks benefit more significantly from strategies of enhancing parameter utilization, which also aligns with insights from previous studies (Biderman et al., 2024; Jiang et al., 2024a).

Basis Expressiveness. We examine the effect of different wavelet bases on our approach, as they offer varying trade-offs between smoothness and detail preservation. In addition to the Haar wavelet, we explore three alternative bases: Daubechies-4 (Db), Biorthogonal (Bior), and Coiflets (Coif). As reported in Table 3, SeLoRA consistently outper-

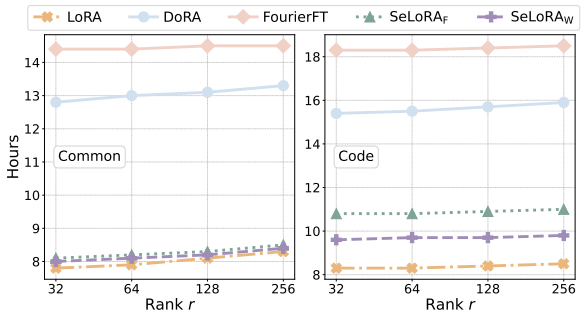


Figure 6: Comparison of training times across different methods adapted to LLaMA3$_{8B}$.

forms LoRA across various tasks, regardless of the specific wavelet transformation used, with only minor performance variations among the different wavelet bases. This demonstrates SeLoRA's robustness to wavelet selection.

Efficiency Comparison. We evaluate the training efficiency of our approach against different methods when adapting at varying ranks r. For consistency, all evaluations are conducted on 1x A100 GPU and FourierFT is assigned a comparable parameter budget to SeLoRA at each rank. As illustrated by Figure 6 and Table 1 and 2, SeLoRA achieves competitive performance compared to DoRA and FourierFT while requiring substantially less training time, and remains comparable in efficiency to LoRA. This highlights SeLoRA's advantage in balancing performance and computational cost.

Subspace Analysis. Inspired by Hu et al. (2021), we investigate the correlation between $\mathbf{W}$ and $\Delta\mathbf{W}$. We answer this question by projecting $\mathbf{W}$ onto the r-dimensional subspace of $\Delta\mathbf{W}$ by computing $\mathbf{U}_r^T\mathbf{W}\mathbf{V}_r$, with $\mathbf{U}/\mathbf{V}$ being the left-/right singular-vector matrices of $\Delta\mathbf{W}$. As defined in Jiang et al. (2025), the amplification factor (AF) $\frac{||\Delta\mathbf{W}||_F}{||\mathbf{U}_r^T\mathbf{W}\mathbf{V}_r||_F}$ measures the subspaces emphasized in the $\Delta\mathbf{W}$ when compared with $\mathbf{W}$, while the reverse amplification factor (RAF) $\frac{||\Delta\mathbf{W}||_F}{||\mathbf{U}_{d-r}^T\mathbf{W}\mathbf{V}_{d-r}||_F}$ indicates the already-amplified directions of $\mathbf{W}$ not being activated. From Table 4, we can draw the following conclusions: 1) Both LoRA and SeLoRA amplify the important features learned from the tasks but not emphasized in $\mathbf{W}$; 2) Compared to LoRA, SeLoRA further reduces the amplification of already-emphasized features in $\Delta\mathbf{W}$.

Methods	$\|\|\Delta\mathbf{W}\|\|_F$	$\|\|\mathbf{U}_r^T\mathbf{W}\mathbf{V}_r\|\|_F$	AF (↑)	RAF (↓)
LoRA	2.74	1.29	2.13	0.17
SeLoRA$_F$	1.98	0.99	2.01	0.09
SeLoRA$_W$	2.11	0.89	**2.37**	**0.04**

Table 4: The amplification factor and reverse amplification factor. The weight matrices are taken from the 24^{th} layer of LLaMA3$_{8B}$ trained for code generation.

5 Conclusion

In this paper, we first explore the impact of parameter redundancy in LoRA fine-tuning, revealing sparse tunable parameters are sufficient for expressive learning, a phenomenon termed as *sparsity property*. Built on these insights, we present SeLoRA, a novel extension of LoRA that leverages spectral transformations to re-parameterize adaptation matrices, enhancing parameter efficiency without compromising training or inference performance. With its flexible and modular nature, SeLoRA can be seamlessly integrated into various LoRA variants as a plug-and-play framework. Empirical evaluations demonstrate that SeLoRA delivers superior adaptability and improvements across diverse instruction tuning tasks. Further in-depth investigations validate its robustness and provide insights into its underlying mechanisms, highlighting its feasibility and practical utility. Future work may explore more effective strategies for leveraging parameter redundancy, further unlocking LoRA's potential for enhanced efficiency and scalability.

Limitations

We observe that the performance improvements of SeLoRA over LoRA gradually diminish as the rank increases, ultimately converging to a similar upper bound at higher ranks. This suggests that while SeLoRA effectively leverages spectral transformation to achieve the expressiveness of high-rank LoRA in a more parameter-efficient manner, it remains constrained by the inherent capacity limits of LoRA itself. Furthermore, SeLoRA is currently compatible with LoRA variants that follow the same update schema. However, its integration into alternative update strategies, such as SVD-based decomposition, remains underexplored. Lastly, due to computational constraints, we are unable to evaluate SeLoRA on models with 70 billion parameters. Exploring its scalability on larger models is an important direction for future work.

Acknowledgements

This work was supported by National Natural Science Foundation of China Grant No. 72371217, the Guangzhou Industrial Informatic and Intelligence Key Laboratory No. 2024A03J0628, the Nansha Key Area Science and Technology Project No. 2023ZD003, and Project No. 2021JC02X191.

References

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *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)*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Bobby Azad, Reza Azad, Sania Eskandari, Afshin Bozorgpour, Amirhossein Kazerouni, Islem Rekik, and Dorit Merhof. 2023. Foundational models in medical imaging: A comprehensive survey and future vision. *arXiv preprint arXiv:2310.18689*.

Klaudia Bałazy, Mohammadreza Banaei, Karl Aberer, and Jacek Tabor. 2024. Lora-xs: Low-rank adaptation with extremely small number of parameters. *arXiv preprint arXiv:2405.17604*.

Nadav Benedek and Lior Wolf. 2024. Prilora: Pruned and rank-increasing low-rank adaptation. *arXiv preprint arXiv:2401.11316*.

Srinadh Bhojanapalli, Ayan Chakrabarti, Andreas Veit, Michal Lukasik, Himanshu Jain, Frederick Liu, Yin-Wen Chang, and Sanjiv Kumar. 2021. Leveraging redundancy in attention with reuse transformers. *arXiv preprint arXiv:2110.06821*.

Dan Biderman, Jose Gonzalez Ortiz, Jacob Portes, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, et al. 2024. Lora learns less and forgets less. *arXiv preprint arXiv:2405.09673*.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Eric L Buehler and Markus J Buehler. 2024. X-lora: Mixture of low-rank adapter experts, a flexible framework for large language models with applications in protein mechanics and molecular design. *APL Machine Learning*, 2(2).

Aochuan Chen, Jiashun Cheng, Zijing Liu, Ziqi Gao, Fugee Tsung, Yu Li, and Jia Li. 2024a. Parameter-efficient fine-tuning via circular convolution. *arXiv preprint arXiv:2407.19342*.

Aochuan Chen, Yuguang Yao, Pin-Yu Chen, Yihua Zhang, and Sijia Liu. 2023a. Understanding and improving visual prompting: A label-mapping perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19133–19143.

Aochuan Chen, Yimeng Zhang, Jinghan Jia, James Diffenderfer, Jiancheng Liu, Konstantinos Parasyris, Yihua Zhang, Zheng Zhang, Bhavya Kailkhura, and Sijia Liu. 2023b. Deepzero: Scaling up zeroth-order optimization for deep model training. *arXiv preprint arXiv:2310.02025*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Nuo Chen, Yuhan Li, Jianheng Tang, and Jia Li. 2024b. Graphwiz: An instruction-following language model for graph computational problems. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 353–364.

Nuo Chen, Yan Wang, Haiyun Jiang, Deng Cai, Yuhan Li, Ziyang Chen, Longyue Wang, and Jia Li. 2023c. Large language models meet harry potter: A dataset for aligning dialogue agents with characters. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8506–8520.

Nuo Chen, Ning Wu, Jianhui Chang, and Jia Li. 2024c. Controlmath: Controllable data generation promotes math generalist models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12201–12217.

Jiashun Cheng, Man Li, Jia Li, and Fugee Tsung. 2023. Wiener graph deconvolutional network improves graph self-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 7131–7139.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind

Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Fahim Dalvi, Hassan Sajjad, Nadir Durrani, and Yonatan Belinkov. 2020. Analyzing redundancy in pretrained transformer models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. 2023. Sparse low-rank adaptation of pre-trained language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Marco F Duarte and Richard G Baraniuk. 2013. Spectral compressive sensing. *Applied and Computational Harmonic Analysis*, 35(1):111–129.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR.

Ziqi Gao, Qichao Wang, Aochuan Chen, Zijing Liu, Bingzhe Wu, Liang Chen, and Jia Li. 2024. Parameter-efficient fine-tuning with discrete fourier transform. *arXiv preprint arXiv:2405.03003*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

Naibin Gu, Peng Fu, Xiyu Liu, Bowen Shen, Zheng Lin, and Weiping Wang. 2024. Light-peft: Lightening parameter-efficient fine-tuning via early pruning. In *Findings of the Association for Computational Linguistics*.

Song Han, Huizi Mao, and William J Dally. 2015a. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

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

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Shwai He, Liang Ding, Daize Dong, Miao Zhang, and Dacheng Tao. 2022. Sparseadapter: An easy approach for improving the parameter-efficiency of adapters. *arXiv preprint arXiv:2210.04284*.

Shwai He, Guoheng Sun, Zheyu Shen, and Ang Li. 2024. What matters in transformers? not all attention is needed. *arXiv preprint arXiv:2406.15786*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*.

Qiushi Huang, Tom Ko, Zhan Zhuang, Lilian Tang, and Yu Zhang. 2025. HiRA: Parameter-efficient hadamard high-rank adaptation for large language models. In *The Thirteenth International Conference on Learning Representations*.

Kazuki Irie and Jürgen Schmidhuber. 2021. Training and generating neural networks in compressed weight space. *arXiv preprint arXiv:2112.15545*.

Shuyang Jiang, Yusheng Liao, Yanfeng Wang, Ya Zhang, and Yu Wang. 2025. Fine-tuning with reserved majority for noise reduction. In *The Thirteenth International Conference on Learning Representations*.

Shuyang Jiang, Yusheng Liao, Ya Zhang, Yanfeng Wang, and Yu Wang. 2024a. TAIA: Large language models are out-of-distribution data learners. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Ting Jiang, Shaohan Huang, Shengyue Luo, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, Qi Zhang, Deqing Wang, et al. 2024b. Mora: High-rank updating for parameter-efficient fine-tuning. *arXiv preprint arXiv:2405.12130*.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems*, 34:1022–1035.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. 2023. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki Markus Asano. 2023. Vera: Vector-based random matrix adaptation. *arXiv preprint arXiv:2310.11454*.

Jan Koutnik, Faustino Gomez, and Jürgen Schmidhuber. 2010. Evolving neural networks in compressed weight space. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 619–626.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. 2018. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *Preprint*, arXiv:2104.08691.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Yang Li, Shaobo Han, and Shihao Ji. 2024a. VB-loRA: Extreme parameter efficient fine-tuning with vector banks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Yang Li, Shaobo Han, and Shihao Ji. 2024b. Vb-lora: Extreme parameter efficient fine-tuning with vector banks. *arXiv preprint arXiv:2405.15179*.

Yuhan Li, Peisong Wang, Xiao Zhu, Aochuan Chen, Haiyun Jiang, Deng Cai, Victor W Chan, and Jia Li. 2024c. Glbench: A comprehensive benchmark for graph with large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 42349–42368.

Yuhan Li, Xinni Zhang, Linhao Luo, Heng Chang, Yuxiang Ren, Irwin King, and Jia Li. 2025. G-refer: Graph retrieval-augmented large language model for explainable recommendation. In *Proceedings of the ACM on Web Conference 2025*, pages 240–251.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2024a. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, 36.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024b. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*.

Shiwei Liu, Tianlong Chen, Xiaohan Chen, Li Shen, Decebal Constantin Mocanu, Zhangyang Wang, and Mykola Pechenizkiy. 2022. The unreasonable effectiveness of random pruning: Return of the most naive baseline for sparse training. *arXiv preprint arXiv:2202.02643*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2024. Wizardcoder: Empowering code large language models with evol-instruct. In *The Twelfth International Conference on Learning Representations*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2024. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*.

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

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

Mahdi Nikdan, Soroush Tabesh, and Dan Alistarh. 2024. Rosa: Accurate parameter-efficient fine-tuning via robust adaptation. *arXiv preprint arXiv:2401.04679*.

Henri J Nussbaumer and Henri J Nussbaumer. 1982. *The fast Fourier transform*. Springer.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2020. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Adithya Renduchintala, Tugrul Konuk, and Oleksii Kuchaiev. 2023. Tied-lora: Enhacing parameter efficiency of lora with weight tying. *arXiv preprint arXiv:2311.09578*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Arijit Sehanobish, Avinava Dubey, Krzysztof Choromanski, Somnath Basu Roy Chowdhury, Deepali Jain, Vikas Sindhwani, and Snigdha Chaturvedi. 2024. Structured unrestricted-rank matrices for parameter efficient fine-tuning. *arXiv preprint arXiv:2406.17740*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Sjoerd Van Steenkiste, Jan Koutník, Kurt Driessens, and Jürgen Schmidhuber. 2016. A wavelet-based encoding for neuroevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 517–524.

Marinus T Vlaardingerbroek and Jacques A Boer. 2013. *Magnetic resonance imaging: theory and practice*. Springer Science & Business Media.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. 2020. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. 2024. Magicoder: Empowering code generation with oss-instruct. In *Forty-first International Conference on Machine Learning*.

Moritz Wolter, Felix Blanke, Jochen Garcke, and Charles Tapley Hoyt. 2024. ptwt-the pytorch wavelet toolbox. *Journal of Machine Learning Research*, 25(80):1–7.

Moritz Wolter, Shaohui Lin, and Angela Yao. 2020. Neural network compression via learnable wavelet transforms. In *Artificial Neural Networks and Machine Learning–ICANN 2020: 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 15–18, 2020, Proceedings, Part II 29*, pages 39–51. Springer.

Yifan Yang, Jiajun Zhou, Ngai Wong, and Zheng Zhang. 2024. Loretta: Low-rank economic tensor-train adaptation for ultra-low-parameter fine-tuning of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3161–3176.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. 2023a. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning. *arXiv preprint arXiv:2308.03303*.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023b. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*.

Yihua Zhang, Yuguang Yao, Parikshit Ram, Pu Zhao, Tianlong Chen, Mingyi Hong, Yanzhi Wang, and Sijia Liu. 2022. Advancing model pruning via bi-level optimization. *Advances in Neural Information Processing Systems*, 35:18309–18326.

Bowen Zhao, Hannaneh Hajishirzi, and Qingqing Cao. 2024. APT: Adaptive pruning and tuning pretrained language models for efficient training and inference. In *Forty-first International Conference on Machine Learning*.

A Related Works

A.1 Parameter-Efficient Fine-Tuning

Fine-tuning large pre-trained language models is crucial for improving NLP tasks (Chen et al., 2023c, 2024c,b). However, updating all model parameters is computationally intensive and storage-demanding for models like GPT-3 (Brown et al., 2020) and LLaMA (Touvron et al., 2023a). Parameter-efficient fine-tuning (PEFT) methods address these issues by updating fewer parameters or adding lightweight modules.

One prominent approach in PEFT is the use of adapters, namely, small bottleneck layers inserted within each layer of a pre-trained model (Houlsby et al., 2019; Pfeiffer et al., 2020; Karimi Mahabadi et al., 2021; He et al., 2021). Houlsby et al. (2019) introduced adapters that enable task-specific adaptation while keeping the original model weights fixed. Building upon this, Pfeiffer et al. (2020) proposed a modular adapter framework that facilitates multi-task transfer. To further optimize parameter efficiency, Karimi Mahabadi et al. (2021) reduced the number of parameters by employing parameter sharing and low-rank approximations within adapters. Another line of research involves prompt tuning, which modifies the input embeddings to guide the model toward specific tasks (Lester et al., 2021; Liu et al., 2021; Li and Liang, 2021; Chen et al., 2023a). Lester et al. (2021) optimized continuous prompt embeddings while keeping the language model's parameters fixed, demonstrating the effectiveness of prompt tuning for task adaptation. Similarly, Prefix-Tuning (Li and Liang, 2021) prepends trainable vectors to the input of each transformer layer without altering the model architecture, effectively steering the model toward desired behaviors with minimal parameter updates.

While these methods exhibit high efficiency and preserve the originality of the pre-trained model, they inevitably introduce higher inference costs due to additional modules or modifications required during deployment. In contrast, LoRA (Hu et al., 2021) and its variants (Zhang et al., 2023a; Bałazy et al., 2024; Li et al., 2024b; Liu et al., 2024b; Nikdan et al., 2024; Gao et al., 2024; Jiang et al., 2024b; Huang et al., 2025) inject trainable low-rank matrix decomposition into transformer layers, have been widely used to adapt recent LFMs for various tasks. These approaches not only reduce the number of trainable parameters but also enable seamless merging with the original model weights, thereby avoiding increased inference burdens. Among these studies, Zhang et al. (2023b) and (Benedek and Wolf, 2024) further highlight that LoRA's intrinsic properties can be further exploited to enhance learning efficiency. In this work, we revisit the inherent parameter redundancy in LoRA and leverage this property to further unlock its potential.

A.2 Parameter Redundancy

Parameter redundancy has been extensively observed in pre-trained language models (Dalvi et al., 2020; Bhojanapalli et al., 2021; He et al., 2022), and recent studies have demonstrated that this redundancy can be exploited to accelerate inference speed (He et al., 2024; Men et al., 2024). Parallel findings reveal that parameter redundancy also exists within the context of LoRA for LFMs, presenting opportunities to enhance efficiency from two complementary perspectives: **(1)** Pre-processing: This research direction focuses on reducing the number of learnable parameters during the fine-tuning phase without sacrificing model performance. Various techniques have been proposed, including matrix projection (Kopiczko et al., 2023; Renduchintala et al., 2023), matrix decomposition (Li et al., 2024a), and matrix substitution (Sehanobish et al., 2024; Chen et al., 2024a). **(2)** Post-processing: This line of work aims to remove redundant parameters after fine-tuning to improve the model's capacity and adaptability. Notable applications include efficient model merging (Yu et al., 2024), out-of-domain adaptation (Jiang et al., 2024a), and enhancing learning capacity (Jiang et al., 2025).

Despite these advances, pre-processing approaches for parameter reduction often rely on heuristic designs, lacking a systematic understanding of the specific impact of parameter redundancy during the fine-tuning phase of LoRA for LFMs. In this work, we aim to bridge this gap by providing a comprehensive investigation of parameter redundancy in LoRA fine-tuning and leveraging it to achieve more efficient and expressive fine-tuning strategies.

A.3 Sparse Learning

Sparse neural networks exploit the fact that many weights in over-parameterized models can be pruned with minimal impact on performance (Han et al., 2015b; Lee et al., 2018; Frankle and Carbin, 2018; Wang et al., 2020; Liu et al., 2022; Fran-

tar and Alistarh, 2023). A common technique, magnitude pruning (Han et al., 2015a), eliminates weights with small magnitudes, significantly reducing model size while maintaining performance. Meanwhile, dynamic sparsity methods (Mocanu et al., 2018; Zhang et al., 2022; Chen et al., 2023b) iteratively adjust the sparsity patterns during training, allowing the network to discover efficient architectures on the fly adaptively.

Another promising direction involves learning in transformed domains, such as spectral space. By leveraging expressive spectral transformations (Cheng et al., 2023), parameterizing weight matrices with sparse, learnable components in the spectral domain has been shown to retain strong expressiveness in both traditional neuroevolution (Koutnik et al., 2010; Van Steenkiste et al., 2016) and modern neural networks (Wolter et al., 2020; Irie and Schmidhuber, 2021). In light of the inherent characteristics of LoRA, this work aims to investigate the potential of this simple yet powerful design to further enhance its efficiency.

B Wavelet Construction

Different wavelet bases strike a balance between smoothness and detail preservation. To illustrate the construction process, we use the Haar wavelet as an example. The 2D Haar wavelet filter is derived by extending the 1D Haar wavelet through tensor products.

In the 1D case, the Haar wavelet is defined by a scaling function, $\phi(\cdot)$, and a wavelet function, $\psi(\cdot)$, given by:

$$\phi(x) = \begin{cases} 1, & 0 \le x < 1, \\ 0, & \text{otherwise}, \end{cases} \tag{8}$$

with

$$\psi(x) = \begin{cases} 1, & 0 \le x < 0.5, \\ -1 & 0.5 \le x < 1, \\ 0, & \text{otherwise}. \end{cases} \tag{9}$$

The 2D Haar wavelet basis functions are then constructed via tensor products as follows:

$$\begin{aligned} \psi^a(x,y) &= \phi(x)\phi(y), \\ \psi^h(x,y) &= \psi(x)\phi(y), \\ \psi^v(x,y) &= \phi(x)\psi(y), \\ \psi^d(x,y) &= \psi(x)\psi(y), \end{aligned} \tag{10}$$

where the superscripts denote different coefficient types as defined in Section 2. This construction method can be generalized to other wavelets by applying the same procedure using their respective 1D scaling and wavelet functions.

C Additional Experimental Details

Training Configurations. All our experiments were carried out on Linux servers equipped with an AMD EPYC 7763 64-core CPU processor, 512GB RAM, and 8x NVIDIA A100 80G GPU with BFloat16 precision. The detailed configurations for all instruction-tuning tasks are illustrated in Table 5, Table 6, and Table 7.

Additional Results. The full experimental results in Section 4 are listed in Table 8 and Table 9.

D Prompt Template

Prompt 1: Training Prompt

Below is an instruction that describes a task. Write a response that appropriately completes the request.

Instruction:
{Question}

Response:
{Answer}

Prompt 2: Evaluation Prompt (Common)

Below is an instruction that describes a task. Write a response that appropriately completes the request.

Instruction:
{Question}

Response:

Prompt 3: Evaluation Prompt (Math)

Below is an instruction that describes a task. Write a response that appropriately completes the request.

Instruction:
{Question}

Response:
Let's think step by step.

Prompt 4: Evaluation Prompt (Code)

Below is an instruction that describes a task. Write a response that appropriately completes the request.

Instruction:
{Question}

Response:
{Import Section}

{Function Signature}
{Docstring}

Hyperparameter	LLaMA2$_{7B}$						LLaMA3$_{8B}$					
	SeLoRA$_F$	SeLoRA$_W$	SeDoRA$_F$	SeDoRA$_W$	SeHiRA$_F$	SeHiRA$_W$	SeLoRA$_F$	SeLoRA$_W$	SeDoRA$_F$	SeDoRA$_W$	SeHiRA$_F$	SeHiRA$_W$
Optimizer	AdamW											
LR Scheduler	Cosine											
Batch Size	16											
Warmup Steps	100											
Dropout	0.05											
Epochs	3											
Rank r	32											
Alpha α	64											
Sparse Ratio η	0.4						0.6					
Modules	[q_proj, k_proj, v_proj, up_proj, down_proj]											
Learning Rate	3e-4	2e-4	2e-4	2e-4	1e-3	1.5e-3	2e-4	2e-4	2e-4	2e-4	1.5e-3	1.5e-3

Table 5: Hyperparameter configurations for commonsense reasoning.

Hyperparameter	LLaMA2$_{7B}$				LLaMA3$_{8B}$			
	SeLoRA$_F$	SeLoRA$_W$	SeDoRA$_F$	SeDoRA$_W$	SeLoRA$_F$	SeLoRA$_W$	SeDoRA$_F$	SeDoRA$_W$
Optimizer	AdamW							
LR Scheduler	Cosine							
Micro Batch Size	32							
Batch Size	128							
Warmup Ratio	0.03							
Dropout	0.05							
Epochs	2							
Rank r	32							
Alpha α	64							
Sparse Ratio η	0.2				0.4			
Modules	[q_proj, k_proj, v_proj, up_proj, down_proj]							
Learning Rate	3e-4				4e-4			

Table 6: Hyperparameter configurations for mathematical reasoning.

E Generation Examples

Example E.1: Hellas.

Below is an instruction that describes a task. Write a response that appropriately completes the request.

Instruction:
Please choose the correct ending to complete the given sentence: Roof shingle removal: A man is sitting on a roof. he

Ending1: is using wrap to wrap a pair of skis. Ending2: is ripping level tiles off. Ending3: is holding a rubik's cube. Ending4: starts pulling up roofing on a roof.

Answer format: ending1/ending2/ending3/ending4

Response:
the correct answer is ending4.

Example E.2: MATH

Below is an instruction that describes a task. Write a response that appropriately completes the request.

Instruction:
How many inches are in the circumference of a great circle of a sphere which has a surface area of 196π sq in? Express your answer in terms of π.
Note: To measure the circumference, take the largest circle on the surface of the sphere.

Response:
Let's think step by step.
The surface area of a sphere is $4\pi r^2$, so $r^2 = 49$ and $r = 7$.
The circumference of the great circle is $2\pi r = \boxed{14\pi}$.
The final answer is: 14π.

	LLaMA2$_{7B}$				LLaMA3$_{8B}$			
Hyperparameter	SeLoRA$_F$	SeLoRA$_W$	SeDoRA$_F$	SeDoRA$_W$	SeLoRA$_F$	SeLoRA$_W$	SeDoRA$_F$	SeDoRA$_W$
Optimizer	AdamW							
LR Scheduler	Cosine							
Micro Batch Size	8							
Batch Size	128							
Warmup Ratio	0.03							
Dropout	0.05							
Epochs	2							
Rank r	32							
Alpha α	64							
Sparse Ratio η	0.2				0.4			
Modules	[q_proj, k_proj, v_proj, up_proj, down_proj]							
Learning Rate	4e-4	4e-4	4e-4	6e-4	4e-4	4e-4	6e-4	4e-4

Table 7: Hyperparameter configurations for code generation.

Train Ratio	Methods	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	Avg.
5%	LoRA	69.7	82.8	73.9	90.5	79.5	87.9	75.7	76.6	79.6
	SeLoRA$_F$	71.7	85.6	75.0	91.1	81.6	90.1	77.2	79.4	81.5
	SeLoRA$_W$	71.5	86.3	75.7	91.4	81.2	90.4	76.8	78.8	81.5
10%	LoRA	69.9	85.0	75.1	91.1	82.4	88.7	76.3	78.8	80.9
	SeLoRA$_F$	72.5	85.9	77.6	92.7	83.8	90.1	78.3	82.0	82.9
	SeLoRA$_W$	71.4	86.0	77.8	92.6	83.8	89.3	77.3	82.4	82.6
25%	LoRA	72.7	84.8	76.7	92.6	85.5	88.7	76.6	82.6	82.5
	SeLoRA$_F$	73.9	87.7	79.5	93.9	86.4	89.1	78.2	85.4	84.3
	SeLoRA$_W$	73.9	86.1	78.9	94.2	86.7	89.8	77.9	83.2	83.8
50%	LoRA	72.9	86.7	79.1	93.6	85.3	88.7	77.5	83.4	83.4
	SeLoRA$_F$	73.7	88.0	79.8	94.8	86.9	91.1	78.8	84.6	84.7
	SeLoRA$_W$	73.8	87.8	80.0	95.0	86.1	90.3	79.1	85.2	84.7
100%	LoRA	74.0	88.2	80.4	94.0	85.5	87.5	78.1	84.0	84.0
	SeLoRA$_F$	74.4	89.0	81.3	95.6	87.5	90.6	80.3	87.0	85.7
	SeLoRA$_W$	76.0	89.3	80.6	95.9	86.7	91.0	81.4	86.6	85.9

Table 8: Full experiment results on LLaMA3$_{8B}$ with various ratios of training data on eight commonsense datasets.

Rank	Methods	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	Avg.
16	LoRA	73.6	87.8	80.0	93.6	85.1	87.2	77.7	83.6	83.6
	SeLoRA$_F$	74.8	88.9	81.1	95.3	85.3	90.1	79.3	86.4	85.2
	SeLoRA$_W$	74.8	89.6	80.7	95.6	86.7	90.9	81.7	86.6	85.8
32	LoRA	74.0	88.2	80.4	94.0	85.5	87.5	78.1	84.0	84.0
	SeLoRA$_F$	74.4	89.0	81.3	95.6	87.5	90.6	80.3	87.0	85.7
	SeLoRA$_W$	76.0	89.3	80.6	95.9	86.7	91.0	81.4	86.6	85.9
64	LoRA	74.4	88.8	80.3	95.1	85.4	89.0	80.0	85.2	84.8
	SeLoRA$_F$	76.4	89.4	81.1	96.1	87.7	91.9	80.5	87.0	86.3
	SeLoRA$_W$	76.2	89.8	80.5	96.2	87.1	92.0	80.9	87.0	86.2
128	LoRA	76.0	88.4	80.5	95.0	86.0	90.2	80.3	86.4	85.4
	SeLoRA$_F$	76.6	88.8	81.5	96.1	88.2	91.6	81.4	87.6	86.5
	SeLoRA$_W$	76.3	89.0	80.8	96.0	86.7	91.7	81.5	87.6	86.2
256	LoRA	76.6	88.7	80.6	96.0	86.5	91.6	80.9	87.8	86.1
	SeLoRA$_F$	76.2	89.6	81.1	95.9	87.4	92.1	81.1	87.4	86.4
	SeLoRA$_W$	77.4	88.8	81.1	96.2	86.7	92.0	81.4	89.0	86.6
512	LoRA	76.4	89.2	81.1	96.1	87.5	91.9	80.5	87.0	86.2
	SeLoRA$_F$	76.4	89.0	81.4	96.2	87.8	92.0	81.2	87.8	86.5
	SeLoRA$_W$	77.0	89.2	81.0	96.3	87.2	92.1	81.8	88.2	86.6

Table 9: Full experiment results on LLaMA3$_{8B}$ with various ranks on eight commonsense datasets.