DenseLoRA: Dense Low-Rank Adaptation of Large Language Models

Lin Mu[1], **Xiaoyu Wang**[1], **Li Ni**[1], **Yang Li**[1], **Zhize Wu**[2], **Peiquan Jin**[3], **Yiwen Zhang**[1]*
[1]Anhui University, [2]Hefei University,
[3]University of Science and Technology of China,
{mulin, nili, zhangyiwen}@ahu.edu.cn {wangxiaoyu,g12114008}@stu.ahu.edu.cn
wuzz@hfuu.edu.cn jpq@ustc.edu.cn

Abstract

Low-rank adaptation (LoRA) has been developed as an efficient approach for adapting large language models (LLMs) by fine-tuning two low-rank matrices, thereby reducing the number of trainable parameters. However, prior research indicates that many of the weights in these matrices are redundant, leading to inefficiencies in parameter utilization. To address this limitation, we introduce **Dense Lo**w-**R**ank **A**daptation (**DenseLoRA**), a novel approach that enhances parameter efficiency while achieving superior performance compared to LoRA. DenseLoRA builds upon the concept of representation fine-tuning, incorporating a single $Encoder$-$Decoder$ to refine and compress hidden representations across all adaptation layers before applying adaptation. Instead of relying on two redundant low-rank matrices as in LoRA, DenseLoRA adapts LLMs through a dense low-rank matrix, improving parameter utilization and adaptation efficiency. We evaluate DenseLoRA on various benchmarks, showing that it achieves 83.8% accuracy with only 0.01% of trainable parameters, compared to LoRA's 80.8% accuracy with 0.70% of trainable parameters on LLaMA3-8B. Additionally, we conduct extensive experiments to systematically assess the impact of DenseLoRA's components on overall model performance. Code is available at https://github.com/mulin-ahu/DenseLoRA.

1 Introduction

Large language models (LLMs) (Brown et al., 2020; Touvron et al., 2023), pre-trained on vast general-domain datasets, have shown remarkable generalization capabilities across a diverse range of downstream tasks (Ruiz et al., 2023; Thirunavukarasu et al., 2023). A common approach for adapting LLMs to new tasks is full fine-tuning, which involves retraining all model parameters. However, as LLMs continue to scale, fully fine-tuning all parameters becomes increasingly impractical due to escalating computational and memory costs, especially in resource-constrained settings.

To address this challenge, researchers have explored parameter-efficient fine-tuning (PEFT) (Houlsby et al., 2019; Lialin et al., 2023), a class of methods that adapt LLMs by fine-tuning only a small subset of task-specific parameters while keeping the rest of the model frozen. PEFT achieves performance comparable to full fine-tuning while significantly reducing the trainable parameter. Among these methods, low-rank adaptation (LoRA) (Hu et al., 2022) has emerged as a particularly effective technique, as it maintains the architecture of the LLMs and preserves inference efficiency. LoRA draws on the assumption that the adaptation weights of LLMs exhibit a low "intrinsic rank," allowing updates to be approximated efficiently with low-rank matrices. Despite its efficiency, LoRA has limitations. Empirical studies reveal that a significant portion of the weights within LoRA's low-rank matrices remain inactive during adaptation, leading to redundancy. As shown in Figure 1, many parameters in these matrices exhibit near-zero increments, indicating that these weights do not contribute meaningfully to the adaptation. While recent LoRA variants (Zhang et al., 2023; Wang et al., 2024; Ren et al., 2024; Yin et al., 2024) attempt to address these inefficiencies by selectively identifying impactful weights, they remain constraints of traditional low-rank adaptation framework. This raises a fundamental research question:

"Can we develop a low-rank adaptation method that enhances model performance while leverages a denser structure to achieve greater efficiency with fewer trainable parameters?"

To answer this, we propose a paradigm shift:

*Corresponding author

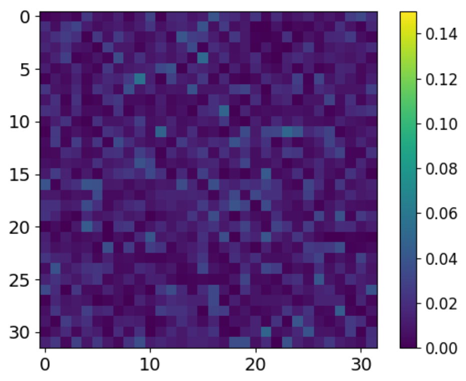


Figure 1: Increments of a low-rank matrix of LoRA during training. We randomly select the slices of a small set of low-rank matrix for demonstration.

rather than relying solely on modifying weight matrices, we explore refining the hidden representations themselves. Drawing inspiration from representation fine-tuning (Wu et al., 2024c,a) techniques, we aim to enhance expressivity while maintaining efficiency. Building on this insight, we introduce **Dense Low-Rank Adaptation (DenseLoRA)**, a novel framework that integrates low-rank adaptation with representation fine-tuning to improve the adaptation efficiency of LLMs. Specifically, DenseLoRA refines and compresses hidden representations before adaptation, allowing for a more efficient adaptation process with a dense low-rank matrix.

In DenseLoRA, an $Encoder$ module first refines and compresses the hidden representations across all adaptation layers, preserving essential task-relevant information. A dense low-rank matrix then adapts the compressed representations at each layer. Finally, a $Decoder$ module reconstructs the refined representations, ensuring seamless integration with the pre-trained model. Notably, the $Encoder$ and $Decoder$ are shared across all adaptation layers, which enhances efficiency and reduces redundancy. Unlike LoRA, which relies on two redundant matrices, DenseLoRA leverages a dense and small matrix to achieve a more efficient and compact adaptation process. This results in a significant reduction in trainable parameters while maintaining effective low-rank adaptation of the pre-trained weight matrix W_0.

Our main contributions can be summarized as follows:

- We introduce DenseLoRA, a novel PEFT method that enhances low-rank adaptation by utilizing a dense and small matrix. This approach leads to more efficient parameter updates, reducing redundancy.
- We integrate low-rank adaptation with representation fine-tuning, enabling DenseLoRA to enhance the expressivity of the model while maintaining computational efficiency.
- We conduct extensive experiments to evaluate DenseLoRA performance on various tasks. Notably, DenseLoRA adapts only 0.01% of trainable parameters while achieving 83.8% accuracy, surpassing LoRA's 80.8% accuracy with 0.70% trainable parameters on commonsense reasoning tasks. Furthermore, we provide an in-depth analysis of its components and their impact on performance.

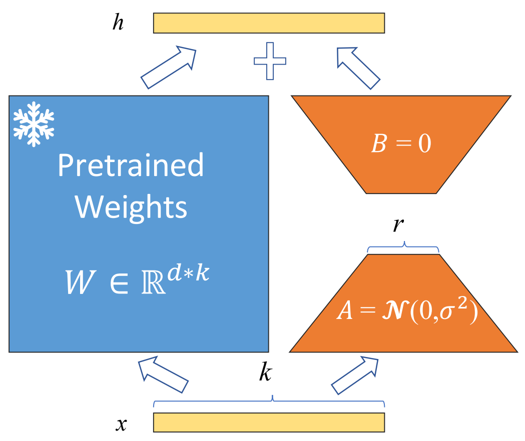
2 Background

2.1 Low-Rank Adaptation (LoRA)

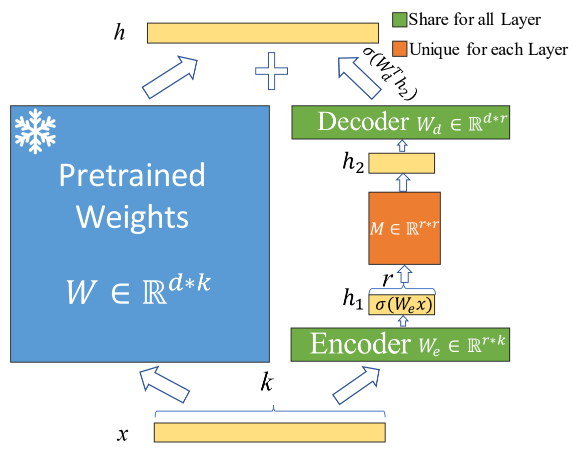
The core hypothesis of LoRA (Hu et al., 2022) is that during the fine-tuning, the weight updates exhibit a low "intrinsic rank". Building on this observation, LoRA freezes the pre-trained weights of LLMs and incrementally updates these weights by utilizing the product of two trainable low-rank matrices. This approach has been demonstrated to achieve performance comparable to full fine-tuning across numerous benchmarks while significantly reducing training parameters. To formally describe LoRA's adaptation process, let $W_0 \in \mathbb{R}^{d\times k}$ denote the pre-trained weight matrix. Instead of directly updating W_0, LoRA applies incremental updates using two low-rank matrices, expressed as $\Delta W = BA$, where $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$. The rank r is significantly smaller than both d and k (i.e. $r << min(d, k)$). This approach substantially reduces the number of trainable parameters compared to full fine-tuning. By incorporating these incremental updates, the adapted hidden representation $\hat{h} \in \mathbb{R}^d$ represented as:

$$\hat{h} = (W_0 + \Delta W)h = W_0 h + BAh \qquad (1)$$

Here, $h \in \mathbb{R}^k$ indicates the hidden representation before adaptation. W_0 remains frozen during fine-tuning. To ensure stability during fine-tuning, matrix A is initialized using a uniform Kamining distribution (He et al., 2015), while B is set to zero, ensuring that $\Delta W = 0$ at the beginning of fine-tuning.



(a) (a) LoRA



(b)
(b) DenseLoRA

Figure 2: Framework comparison of LoRA(left) and DenseLoRA(right).

As shown in Eq.(1), during inference, LoRA can merge the W_0 and ΔW (i.e. $W' = W_0 + \Delta W$). This property ensures that LoRA does not introduce any additional inference latency compared to the original model.

2.2 Representation Fine-tuning

Recently, several studies have explored the representation fine-tuning (Wu et al., 2024c,a) technique. Instead of adapting model weights, these methods focus on refining hidden representations directly, enabling task-specific adjustments without modifying weight matrices. For example, Red (Wu et al., 2024a) refined hidden representations by introducing two learnable components: scaling vector $l_{scaling} \in \mathbb{R}^d$ and bias vector $l_{bias} \in \mathbb{R}^d$. This process can be mathematically expressed as follows:

$$\hat{h} = l_{scaling} \odot h + l_{bias} \tag{2}$$

Where $\odot$ represents Hadamard product, $h \in \mathbb{R}^d$ indicates the hidden representation.

3 Methodology

To address the inefficiencies of existing low-rank adaptation methods, we propose Dense Low-Rank Adaptation (DenseLoRA) a novel framework that integrates low-rank adaptation with representation fine-tuning. DenseLoRA overcomes the redundancy issues observed in vanilla LoRA approaches by adapting a dense, small low-rank matrix, ensuring a more efficient use of parameters. Specifically, DenseLoRA introduces a structured three-stage process: (1) An *Encoder* refines and compresses hidden representations; (2) A denser low-rank adaptation module adapts the model; and (3) A *Decoder* reconstructs the refined representations to ensure seamless integration with the pre-trained model. This structured approach distinguishes DenseLoRA from vanilla LoRA, which relies on two low-rank matrices.

3.1 DenseLoRA Architecture

To realize this three-stage adaptation process, DenseLoRA introduces a novel architecture (illustrated in Figure 2) that integrates an *Encoder-Decoder* mechanism with low-rank adaptation. This architecture refines and compresses hidden representations before adaptation, reducing redundancy and significantly lowering the number of trainable parameters while maintaining the model's expressiveness. At the core of this architecture is the *Encoder-Decoder* mechanism, implemented as fully connected neural networks, which refines representations and adapts LLMs through the following three stages:

- **Compression:** The *Encoder* applies a transformation using weights $W_e \in \mathbb{R}^{r \times k}$ to refine and compress the hidden representation $h \in \mathbb{R}^k$ into a lower-dimensional representation. This is followed by an activation function $\sigma(\cdot)$, producing a compressed representation $h' \in \mathbb{R}^r$.

- **Adaptation:** To fine-tune the model for downstream tasks, a dense low-rank matrix $\boldsymbol{M} \in \mathbb{R}^{r \times r}$ is applied to the compressed representation. This step adapts the model while keeping the pre-trained weight matrix W_0 frozen.

- **Reconstruction:** The *Decoder* then reconstructs the adapted representation back to the original hidden dimension using weights $W_d \in \mathbb{R}^{d \times r}$, followed by an activation function. This reconstruction step ensures that the adapted representation seamlessly integrates with the frozen pre-trained model, preserving expressivity while maintaining efficiency.

The overall adaptation process in DenseLoRA can be mathematically formulated as follows, combining the frozen pre-trained weights W_0 with the refined and adapted hidden representations:

$$\hat{h} = W_0 h + Decoder(\boldsymbol{M} Encoder(h)) \quad (3)$$

To enhance parameter efficiency, DenseLoRA shares a single *Encoder-Decoder* across all adaptation layers. This strategy reduces redundancy and significantly lowers the number of trainable parameters. To maintain layer-specific adaptability, DenseLoRA fine-tunes unique low-rank matrices $\boldsymbol{M} \in \mathbb{R}^{r \times r}$ for each adaptation layer, ensuring that adaptation remains flexible and effective across different layers. The transformations applied by the *Encoder* and *Decoder* can be mathematically expressed as follows, capturing how hidden representations are refined and reconstructed:

$$h^{'} = Encoder(h) = \sigma(W_e h) \quad (4)$$

$$\hat{h} = Decoder(h^{'}) = \sigma(W_d^T h^{'}) \quad (5)$$

3.2 Parameter Analysis

Initialization Strategies: DenseLoRA employs two categories of matrices: Shared Matrices and Unique Matrices, each initialized to ensure stability and efficiency during training. Shared Matrices: These include $W_e \in \mathbb{R}^{r \times k}$ and $W_d \in \mathbb{R}^{d \times r}$, which are shared across different adaptation layers. The W_e matrix is initialized using Kaiming initialization (He et al., 2015), while the W_d matrix is initialized to zeros. This ensures that the W_d matrix does not interfere with the output during the first forward pass. Unique Matrices: These include $\boldsymbol{M} \in \mathbb{R}^{r \times r}$, which is unique to each adaptation layer. Like the matrices W_e, $\boldsymbol{M}$ is also initialized using Kaiming initialization (He et al., 2015) to promote efficient and stable training.

By combining shared matrices for parameter efficiency and unique matrices for layer-specific adaptability, DenseLoRA achieves an effective balance between fine-tuning flexibility and computational cost.

Parameter Count: To effectively fine-tune large language models (LLMs) while maintaining parameter efficiency, DenseLoRA significantly reduces the number of trainable parameters compared to existing LoRA variants. Let l represent the number of adaptation layers in LLMs, and k and d represent the input and output dimensions, respectively. The parameter count for different methods is summarized as follows:

- **Full Fine-Tuning (FFT)**: The total number of trainable parameters is $|\Theta| = l \times d \times k$.

- **LoRA**: LoRA reduces the parameter count to $|\Theta| = l \times (d + k) \times r$, here rank $r << min(d, k)$.

- **DenseLoRA**: Unlike LoRA, which relies on two redundant low-rank matrices, DenseLoRA optimizes efficiency by leveraging a denser structure, resulting in the following parameter count: $|\Theta| = (d+k+l \times r) \times r$, where rank $r << min(d, k)$.

To illustrate this advantage, we compare trainable parameters in real-world LLM settings:

- In LLaMA2-7B with $r = 16$, LoRA require approximate $28M$ trainable parameters, while DenseLoRA reduces this to $0.9M$, achieving a 30× reduction

- In LLaMA3-8B with $r = 16$, DenseLoRA maintains a similar reduction relative to LoRA, confirming its scalability across models.

These results demonstrate that DenseLoRA has significantly fewer trainable parameters, making it a highly scalable and computationally efficient solution for fine-tuning LLMs.

4 Experiments

In this section, we conduct extensive experiments to evaluate the effectiveness of DenseLoRA across various tasks. Our evaluation follows a structured approach to ensure a comprehensive analysis.

First, we compare DenseLoRA with LoRA and its variants by fine-tuning LLaMA2-7B and LLaMA3-8B on commonsense reasoning tasks.

LLM	Method	Param(%)	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	Avg.
ChatGPT	-	-	73.1	85.4	68.5	78.5	66.1	89.8	79.9	74.8	77.0
LLaMA2-7B	LoRA	0.83	69.8	79.9	79.5	83.6	82.6	79.8	64.7	81.0	77.6
	DenseLoRA*	**0.01**	69.9	79.5	78.2	83.0	78.0	81.5	63.8	76.6	76.3
	DenseLoRA**	**0.03**	71.3	81.0	78.9	85.0	79.5	82.4	65.4	76.2	77.5
	DenseLoRA***	**0.06**	70.2	81.8	78.8	90.0	81.9	66.2	82.6	79.2	78.8
LLaMA3-8B	LoKr	0.01	65.1	81.6	78.7	92.0	82.1	89.2	76.7	80.9	80.9
	NoRA	0.10	73.3	86.4	79.1	94.1	84.3	88.2	77.5	85.0	83.1
	VeRA‡	0.01	62.2	81.6	64.8	54.5	6.18	84.4	67.2	64.6	67.7
	AdaLoRA	0.35	75.1	86.4	76.7	75.4	83.3	90.4	79.1	81.4	81.4
	LoRA	0.70	70.8	85.2	79.9	91.7	84.3	84.2	71.2	79.0	80.8
	DoRA†	0.35	74.5	88.8	80.3	95.5	84.7	90.1	79.1	87.2	85.0
	DoRA	0.71	74.6	89.3	79.9	95.5	85.6	90.5	80.4	85.8	85.2
	DenseLoRA*	**0.01**	72.3	87.5	79.8	93.5	85.2	89.8	78.2	84.0	83.8
	DenseLoRA**	**0.02**	74.3	88.0	80.3	94.5	86.0	89.7	78.7	85.6	84.6
	DenseLoRA***	**0.06**	74.1	88.9	80.3	95.0	87.0	90.0	79.2	85.6	85.0

Table 1: Accuracy(%) comparison of various methods fine-tuning LLaMA2-7B and LLaMA3-8B on the commonsense reasoning tasks. † denotes that the rank is equal to 32 in DoRA (Liu et al., 2024). ‡ denotes that we experiment on the commonsense reasoning task using the same settings in VeRA (Liu et al., 2024), with a rank equal to 256. * denotes $r = 16$. ** denotes $r = 32$. *** denotes $r = 64$.

Next, we extend our analysis to arithmetic reasoning, focusing on fine-tuning LLaMA3-8B models. To assess the performance of DenseLoRA under limited data availability, we conduct experiments by sampling a subset of the original training data and analyzing its impact on model performance. Additionally, we investigate the optimal tuning granularity by analyzing which weight matrices within the transformer architecture benefit the most from adaptation using DenseLoRA. Finally, we delve deeper into the mechanics of DenseLoRA by investigating the effects of fine-tuning the $Encoder$ and $Decoder$ components and comparing the adaptation matrix $\boldsymbol{M}$ to A and B used in LoRA.

4.1 Commonsense Reasoning

To evaluate DenseLoRA's performance on commonsense reasoning tasks, we fine-tune LLaMA2-7B and LLaMA3-8B using a dataset comprising 170k training samples (Hu et al., 2023). These samples are drawn from the training sets of eight commonsense reasoning benchmarks (see Appendix A.1 for details): 1) BoolQ (Clark et al., 2019); 2) PIQA (Bisk et al., 2020); 3) SIQA (Sap et al., 2019); 4) HellaS. (HellaSwag) (Zellers et al., 2019); 5) WinoG. (WinoGrande) (Sakaguchi et al., 2021); 6) ARC-c and ARC-e (Clark et al., 2018); 7)OBQA (Mihaylov et al., 2018). All the experiments are conducted using 4 Nvidia 24GB 3090 GPUs, with the training hyperparameters detailed in Appendix A.2.

For comparative analysis, we evaluate DenseLoRA against several LoRA variant methods: 1) ChatGPT (Wei et al., 2022), which applies a zero-shot chain of thought approach (Kojima et al., 2022) on GPT-3.5-turbo; 2) LoRA (Hu et al., 2022), which updates a weight matrix via a low-rank matrices AB; 3) LoKr (YEH et al., 2024), which utilizes Kronecker products for weight matrix decomposition, significantly reducing trainable parameters; 4) NoRA (Lin et al., 2024), which adopts a dual-layer nested structure with singular value decomposition (SVD), effectively leveraging original matrix knowledge while reducing tunable parameters. 5) VeRA (Kopiczko et al., 2024), which employs a single pair of shared low-rank matrices across all layers and fine-tunes small scaling vectors; 6) AdaLoRA (Zhang et al., 2023), which fine-tunes pre-trained weights using SVD; 7) DoRA (Liu et al., 2024), which decomposes the pre-trained weight into magnitude and directional components for fine-tuning;

Main results. Table 1 presents the results of commonsense reasoning tasks on LLaMA2-7B and LLaMA3-8B across different ranks (16, 32, and 64). Key findings are:

1) **Excellent Performance**: DenseLoRA achieves an average accuracy of 85% on LLaMA3-

Method	Params(%)	GSM8K	AQUA	AddSub	SVAMP	Avg.
LoRA	0.70	47.1	18.1	90.6	71.9	56.9
DenseLoRA*	0.02	45.5	**20.5**	73.5	92.1	57.5
DenseLoRA**	0.06	**47.2**	19.7	**92.4**	**74.5**	**58.5**

Table 2: Accuracy(%) comparison of DenseLoRA and LoRA fine-tuning LLaMA3-8B on four arithmetic reasoning tasks. * denotes $r = 32$. ** denotes $r = 64$

8B, outperforming LoRA (80.8%) by 4.1%, while requiring only 10% of the trainable parameters used by LoRA. Notably, when fine-tunes just 0.01% of the parameters, DenseLoRA still maintains 83.8% accuracy, outperforming LoRA while reducing trainable parameters by a factor of 70. This demonstrates DenseLoRA's ability to achieve high accuracy with minimal adaptation overhead.

2) **Parameter Efficiency**: Compared to other parameter-efficient LoRA variants (VeRA, LoKr, and NoRA), DenseLoRA achieves higher accuracy. For instance, it achieves an accuracy of 83.8%, outperforming LoKr by 2.9%. While LoKr reduces training parameters via Kronecker products, it requires more computational resources (training time and GPU memory) than LoRA (Wu et al., 2024b). In contrast, DenseLoRA maintains similar computational costs to LoRA while offering superior performance, making it a more practical solution.

3) **Rank Robustness**: DenseLoRA demonstrates strong rank robustness, maintaining high accuracy across rank configurations (16, 32, 64). As the rank increases, its average accuracy consistently improves. For example, at rank 16, DenseLoRA fine-tunes only 1/70 of the trainable parameters used by LoRA, yet outperforms LoRA. At rank 64, DenseLoRA's trainable parameter increases to 0.06%, achieving 85% accuracy, surpassing LoRA (80.8%) and approaching DoRA (85.2%).

4.2 Arithmetic Reasoning

To evaluate the effectiveness of DenseLoRA on the arithmetic reasoning task, we fine-tuned LLaMA3-8B on the Math10K (Hu et al., 2023) dataset and evaluated it performance on four different datasets, including 1) GSM8K (Cobbe et al., 2021), 2)AQUA (Ling et al., 2017), 3) AddSub (Hosseini et al., 2014), 4) SVAMP (Patel et al., 2021). Details of datasets are provided in Appendix A.1. We conduct experiments using 1 Nvidia 24GB 3090 GPU, with hyperparameters listed in Appendix A.2.

Table 2 presents the result on 4 arithmetic reasoning benchmarks, demonstrating DenseLoRA's

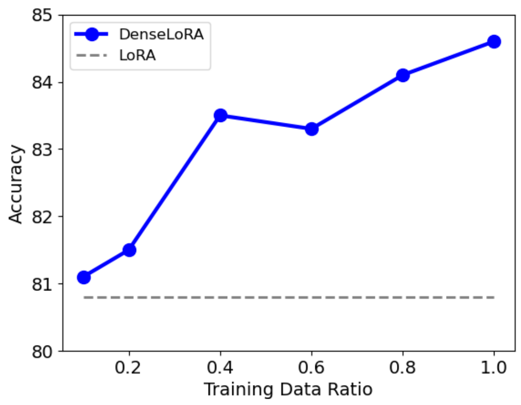


Figure 3: Accuracy(%) comparison of DenseLoRA and LoRA on the commonsense reasoning 170k dataset with fewer training samples.

strong performance with significantly fewer trainable parameters. Notably: at rank=32, DenseLoRA achieves superior performance using only 0.02% trainable parameters, a $35\times$ reduction compared to LoRA (0.7% trainable parameters). At rank=64, DenseLoRA attains 58.5% accuracy using only 0.06% trainable parameters, surpassing LoRA's 56.9% (0.7% trainable parameters). These results further validate DenseLoRA's effectiveness, demonstrating its parameter efficiency and adaptability across different reasoning tasks.

4.3 Low Resources Performance

Building on our evaluation of DenseLoRA's parameter efficiency, we now examine its performance under low-resource conditions. We randomly sampled 10%, 20%, 40%, 60%, and 80% of the original 170k commonsense reasoning training dataset and repeated the experiments using LLaMA3-8B with a rank of 32. Figure 3 illustrates the relationship between training sample size and performance, with detailed numbers presented in Appendix B.1. Notably, DenseLoRA consistently outperforms LoRA across all sample sizes, demonstrating its ability to generalize effectively even with limited data. For instance, DenseLoRA trained with just 10% of the

#Params(%)	LoRA	DenseLoRA	Avg.
0.70	QKVUD	-	80.8
0.25	QKV	UD	83.8
0.49	UD	QKV	83.2
0.01	-	QKV	82.3
0.02	-	UD	83.8
0.02	-	QKVUD	**84.6**

Table 3: Accuracy(%) comparison of DenseLoRA with several different tuning granularity fine-tuning LLaMA3-8B. Each module is represented by its first letter as follows: (Q)uery, (K)ey, (V)alue, (O)utput, (U)p, (D)own.

data achieves an accuracy of 81.1%, which is 0.3% higher than LoRA trained with the full dataset. As the number of training samples increases, the performance gap between DenseLoRA and LoRA widens, further highlighting DenseLoRA's ability to effectively capture complex patterns with reduced data requirements.

4.4 Tuning Granularity Analysis

This section evaluates the impact of adapting different weight modules using DenseLoRA. Each module is represented by its first letter as follows: (Q)uery, (K)ey, (V)alue, (O)utput, (G)ate, (U)p, (D)own. We conduct experiments using LLaMA3-8B with a rank of 32 on commonsense reasoning training samples. The result, shown Table 3, highlight several key observations:

Consistent with the original LoRA configuration suggested in (Hu et al., 2022), which requires tuning both the Multi-head Attention and MLP layers for optimal performance, DenseLoRA achieves superior accuracy when updating both components. Furthermore, adapting the MLP layers proves to be more effective than tuning Multi-head Attention layers. Notably, when DenseLoRA is applied to QKV modules, the model achieves an average of 82.3%. However, when the UD modules are adapted instead, accuracy improves to 83.8%, surpassing the QKV configuration. Interestingly, even when only the UD modules are tuned using DenseLoRA (without tuning QKV modules), the accuracy remains at 83.8%. These findings indicate that DenseLoRA is highly efficient than LoRA, achieving superior performance with significantly fewer trainable parameters.

Method	# Params(%)	Avg.
DenseLoRA	0.02	**84.6**
– Freeze	0.03	79.5
– Only Matrix	0.02	83.3

Table 4: Accuracy (%) comparison of several variants of DenseLoRA.

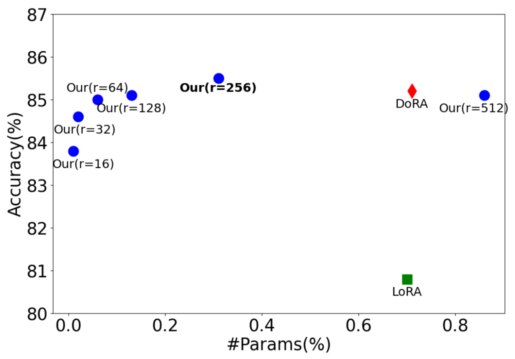


Figure 4: Evaluate DenseLoRA with rank from 16 to 512.

4.5 Robustness of Rank

We evaluated DenseLoRA with more rank configurations of 128, 256, and 512 on commonsense reasoning tasks using LLaMA3-8B. The corresponding performance is shown in the following Figure 4. Notably, with a rank of 256 and 0.31% of the parameters fine-tuned, DenseLoRA outperforms LoRA (80.8%) and DoRA (85.2%) with a performance of 85.5%. Therefore, DenseLoRA performance can be improved by increasing the number of parameters fine-tuned.

4.6 Understanding the DenseLoRA

Having established DenseLoRA's empirical advantages, we now examine its internal mechanisms to better understand the advantages of DenseLoRA. To this end, we conduct a focused investigation on commonsense reasoning tasks using LLaMA3-8B (rank = 32), addressing two key questions: 1) What is the effectiveness of the representation fine-tuning module that is the $Encoder$ and $Decoder$ modules? 2) How does the adaptation matrix $\boldsymbol{M}$ compare to matrices A and B used in LoRA?

Effectiveness of *Encoder* and *Decoder*: To explore the role of the representation fine-tuning module in DenseLoRA, we evaluate two key variants of the $Encoder$ and $Decoder$ components: 1) **Freeze**: In this setting, we freeze the parameters of both the $Encoder$ and $Decoder$, keeping them

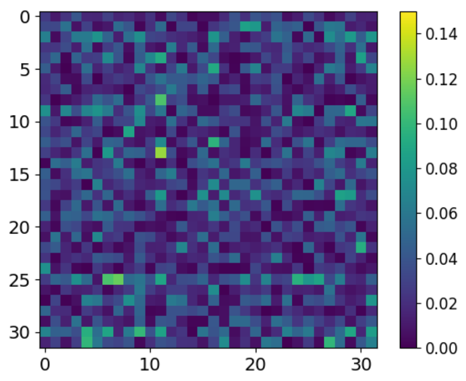


Figure 5: Increments of matrix $\boldsymbol{M}$ during training.

fixed during training. This significantly reduces the number of trainable parameters and is conceptually similar to VeRA (Kopiczko et al., 2024). To maintain a comparable number of trainable parameters to DenseLoRA, we compensate by setting the rank to 128. 2) **Only Matrix**: This variant isolates the effect of the matrix transformation by removing the activation function, allowing us to assess whether activation functions contribute significantly to representation fine-tuning.

Table 4 presents the experimental result of DenseLoRA compared to its variants. The results underscore the crucial of the $Encoder$ and $Decoder$ components in the effective fine-tuning of LLMs. Furthermore, the Only Matrix variant, which removes the activation function, shows a performance decrease. This finding indicates that activation functions are not merely auxiliary components but rather play an essential role in enhancing representation fine-tuning.

Dense Matrix $\boldsymbol{M}$: To verify the advantages of DenseLoRA's dense matrix representation over LoRA, we compare ΔM with ΔA and ΔB from the same adaptation module and layer during training. Since A and B are significantly larger than $\boldsymbol{M}$, we randomly select the slices of A and B that match the size of $\boldsymbol{M}$. Figure 5 presents the incremental updates of $\boldsymbol{M}$, while a detailed comparison with A and B is provided in Appendix B.3. From the figure, we observe that $\boldsymbol{M}$ exhibits a dense update pattern, where the majority of its parameters actively contribute to the adaptation process. In contrast, A and B show sparse updates, with most of their parameters either remaining unchanged or undergoing minimal modifications. These findings demonstrate that DenseLoRA is highly parameter-efficient, effectively utilizing a compact adaptation matrix while maintaining strong performance.

5 Related Works

Parameter-Efficient Fine-Tuning (PEFT): It is a widely adopted strategy aimed at fine-tuning a limited number of parameters in LLMs while keeping the remainder unchanged. These approaches involve fine-tuning only a subset of the existing model parameters or adding new parameters to the model. These methods are designed to reduce the high computational cost of full fine-tuning LLMs (Lialin et al., 2023; Han et al., 2024). Existing PEFT methods are primarily classified into four types. The first type is known as adapter-based methods. These methods involved inserting adapter layers between the existing layers in LLMs (Houlsby et al., 2019; He et al., 2021; Karimi Mahabadi et al., 2021). The second type involves prompt-based methods (Lester et al., 2021; Razdaibiedina et al., 2023; Li and Liang, 2021). These approaches introduced trainable soft tokens (prompts) into the model's input rather than modifying its internal weights. While these methods involved altering the model's input or architecture, they resulted in increased inference latency compared to other fine-tuning methods. The third type of method is the low-rank method like LoRA.

Low-Rank Adaptation (LoRA) (Hu et al., 2022): It is a technique designed for the low-rank properties of model updates to improve parameter efficiency. Specifically, LoRA used two small matrices to approximate the weight increments during fine-tuning. Recently, there have been numerous methods exploring more efficient LoRA variants. For example, AdaLoRA (Zhang et al., 2023) builds upon LoRA by applying Singular Value Decomposition (SVD) to prune less significant singular values, enhancing update efficiency. Additionally, DoRA (Liu et al., 2024) decomposes the pre-trained weight into two components: magnitude and direction, allowing LoRA to focus solely on directional updates. Another recent contribution, VeRA (Kopiczko et al., 2024) introduces the use of "scaling vectors" to adapt frozen random matrices that are shared between layers, significantly reducing the number of trainable parameters compared to traditional LoRA.

Representation Fine-tuning: Prior interpretability research has demonstrated that hidden representations in LLMs encode rich semantic information. As a result, representation fine-tuning has emerged as a new type of PEFT method (Wu et al., 2024c,a; Yin et al., 2024). For example, RED (Wu et al., 2024a) modifies the representations at intermediate layers through scaling and biasing operations, significantly reducing the number of trainable parameters.

Our approach integrates low-rank adaptation with representation fine-tuning, which is different from the existing PEFT method, and achieves superior performance.

6 Conclusion

To address weight redundancy in LoRA, we propose Dense Low-Rank Adaptation (DenseLoRA), which significantly reduces the number of trainable parameters while outperforming LoRA. DenseLoRA achieves this by employing a shared $Encoder$ and $Decoder$ across all layers to refine and compress hidden representations before adaptation. Instead of relying on two redundant low-rank matrices, DenseLoRA fine-tunes LLMs using a dense low-rank matrix, leading to more efficient parameter utilization. We evaluate the effectiveness of DenseLoRA on various benchmarks, demonstrating that it achieves performance comparable to LoRA while requiring only 1/70 of the trainable parameters. DenseLoRA presents a more efficient approach to low-rank adaptation while maintaining and even improving model performance compared to existing methods like LoRA.

7 Limitations

In this paper, we conduct experiments on commonsense reasoning tasks, and arithmetic reasoning tasks by fine-tuning LLaMA2-7B and LLaMA3-8B. There is a broader range of tasks unexplored using DenseLoRA, such as image generation tasks, and visual instruction tuning tasks. We will apply DenseLoRA in these tasks for future work.

8 Acknowledgements

This work is supported by the National Natural Science Foundation of China (No.62206004, No.62272001, No.62072419, No.62406095), and the Natural Science Foundation of Anhui Province (No.2308085MF213).

References

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*, volume 34, pages 7432–7439.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

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. In *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. Association for Computational Linguistics.

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. *CoRR*, abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.

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 *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical*

Methods in Natural Language Processing (EMNLP), pages 523–533, Doha, Qatar. Association for Computational Linguistics.

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 *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. 2023. LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5254–5276, Singapore. Association for Computational Linguistics.

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

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M Asano. 2024. VeRA: Vector-based random matrix adaptation. In *The Twelfth International Conference on Learning Representations*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. 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)*, pages 4582–4597, Online. Association for Computational Linguistics.

Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. 2023. Scaling down to scale up: A guide to parameter-efficient fine-tuning. *Preprint*, arXiv:2303.15647.

Cheng Lin, Lujun Li, Dezhi Li, Jie Zou, Wei Xue, and Yike Guo. 2024. Nora: Nested low-rank adaptation for efficient fine-tuning large models. *Preprint*, arXiv:2408.10280.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

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

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. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094. Association for Computational Linguistics.

Anastasiia Razdaibiedina, Yuning Mao, Madian Khabsa, Mike Lewis, Rui Hou, Jimmy Ba, and Amjad Almahairi. 2023. Residual prompt tuning: improving prompt tuning with residual reparameterization. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6740–6757, Toronto, Canada. Association for Computational Linguistics.

Pengjie Ren, Chengshun Shi, Shiguang Wu, Mengqi Zhang, Zhaochun Ren, Maarten de Rijke, Zhumin Chen, and Jiahuan Pei. 2024. MELoRA: Mini-ensemble low-rank adapters for parameter-efficient fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3052–3064, Bangkok, Thailand. Association for Computational Linguistics.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. 2023. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22500–22510.

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

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China. Association for Computational Linguistics.

Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. 2023. Large language models in medicine. *Nature medicine*, 29(8):1930–1940.

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

Haoyu Wang, Tianci Liu, Ruirui Li, Monica Xiao Cheng, Tuo Zhao, and Jing Gao. 2024. RoseLoRA: Row and column-wise sparse low-rank adaptation of pre-trained language model for knowledge editing and fine-tuning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 996–1008, Miami, Florida, USA. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Muling Wu, Wenhao Liu, Xiaohua Wang, Tianlong Li, Changze Lv, Zixuan Ling, Zhu JianHao, Cenyuan Zhang, Xiaoqing Zheng, and Xuanjing Huang. 2024a. Advancing parameter efficiency in fine-tuning via representation editing. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13445–13464, Bangkok, Thailand. Association for Computational Linguistics.

Taiqiang Wu, Jiahao Wang, Zhe Zhao, and Ngai Wong. 2024b. Mixture-of-subspaces in low-rank adaptation. *Preprint*, arXiv:2406.11909.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. 2024c. ReFT: Representation finetuning for language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

SHIH-YING YEH, Yu-Guan Hsieh, Zhidong Gao, Bernard B W Yang, Giyeong Oh, and Yanmin Gong. 2024. Navigating text-to-image customization: From lyCORIS fine-tuning to model evaluation. In *The Twelfth International Conference on Learning Representations*.

Fangcong Yin, Xi Ye, and Greg Durrett. 2024. Lofit: Localized fine-tuning on LLM representations. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*.

A Experimental Setting

A.1 Dataset

Commonsense Reasoning: tasks consist of 8 benchmarks and the details are described as follows:

- BoolQ (Clark et al., 2019): This dataset comprises a collection of yes/no question examples, totaling 15942 examples. These questions are naturally occurring and generated in unprompted and unconstrained settings;
- PIQA (Bisk et al., 2020): This dataset consists of questions with two solutions that require physical commonsense to answer;
- SIQA (Sap et al., 2019): This dataset focuses on analyzing people's actions and their social implications;
- HellaSwag: This dataset consists of commonsense Natural Language Inference (NLI) questions, each featuring a context and multiple endings that complete the context;
- WinoGrande (Sakaguchi et al., 2021): This dataset presents a fill-in-a-blank task with binary options. The goal is to select the appropriate option for a given sentence that requires commonsense reasoning;
- ARC-c and ARC-e (Clark et al., 2018): These two datasets are the Challenge Set and Easy Set of ARC (Clark et al., 2018) dataset, which contains genuine grade-school level, multiple-choice science questions;
- OBQA: This dataset comprises questions that require multi-step reasoning, the use of additional common sense knowledge, and thorough text comprehension.

Arithmetic Reasoning: tasks consist of 4 benchmarks and the details are described as follows:

- GSM8K (Cobbe et al., 2021): This dataset consists of 8,500 high-quality, linguistically diverse elementary math problems created by humans;
- AQUA (Ling et al., 2017): This dataset comprises 100,000 multiple-choice questions focused on mathematics, encompassing a wide range of topics and varying levels of difficulty;

HyperParmaters	**LLaMA2-7B**		**LLaMA3-8B**	
Rank r	16	32	16	32
α	32	64	32	64
Dropout	0.05			
Optimizer	AdamW			
LR	3e-4			
LR Scheduler	Linear			
Batch Size	16			
Warmup Stemps	100			
Epochs	2			
Where	Q, K, V, Up, Down			

Table 5: The hyperparameters for DenseLoRA on the commonsense reasoning tasks.

HyperParmaters	**LLaMA3-8B**	
Rank r	16	32
α	32	64
Dropout	0.05	
Optimizer	AdamW	
LR	3e-4	
LR Scheduler	Linear	
Batch Size	16	
Warmup Stemps	100	
Epochs	2	
Where	Q, K, V, Up, Down	

Table 6: The hyperparameters for DenseLoRA on the arithmetic reasoning tasks.

- SVAMP (Simple Variations on Arithmetic Math Word Problems) (Patel et al., 2021): This dataset comprises arithmetic word problems appropriate for fourth-grade students and below;
- AddSub (Hosseini et al., 2014) This dataset consists of 395 problems specifically focused on addition and subtraction.

A.2 Hyperparameters

Table 5 shows the detailed hyperparameters for commonsense reasoning tasking when fine-tuning the LLaMA3-8B and LLaMA2-7B. Table 6 shows the detailed hyperparameters for arithmetic reasoning tasking when fine-tuning the LLaMA3-8B. During the inference, we set the hyperparameters $max\_new\_tokens = 128$.

B Additional Experimental

B.1 Low Resources Experimental

Table 7 shows the details for low resources setting of commonsense reasoning tasks when fine-tuning the LLaMA3-8B and using the rank equal to 32.

B.2 Tuning Granularity Analysis

This section is a detail experiment result of adapting different weight modules using DenseLoRA. Each module is represented by its first letter as follows: (Q)uery, (K)ey, (V)alue, (O)utput, (G)ate, (U)p, (D)own. We conduct experiments using LLaMA3-8B with a rank of 32 on commonsense reasoning training samples. The result, shown Table 8

B.3 Compared $\boldsymbol{M}$ and A, B

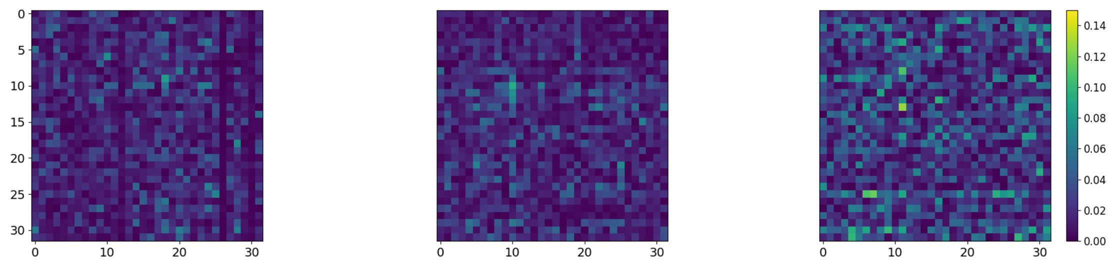
we compare ΔM with ΔA and ΔB from the same adaptation module and layer during training. Since A and B are significantly larger than $\boldsymbol{M}$, we randomly select the slices of A and B that match the size of $\boldsymbol{M}$. The detailed result is shown in Figure 6.

Ratio	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	Avg.
10%	71.6	85.0	76.4	90.3	79.2	89.6	77.7	78.6	81.1
20%	63.5	86.6	78.6	92.2	81.8	77.7	89.8	81.8	81.5
40%	73.4	86.8	78.7	93.2	85.1	77.5	90.0	83.0	83.6
60%	68.6	87.4	79.5	93.9	84.4	78.2	90.2	84.4	83.3
80%	72.9	88.4	79.3	94.1	85.5	78.2	90.6	84.0	84.1

Table 7: Accuracy(%) comparison of fewer training samples methods fine-tuning LLaMA3-8B on the commonsense reasoning 170k dataset in (Hu et al., 2023). **Ratio** denotes the ratio of trained parameters. We set the rank = 32.

# Params(%)	Method		BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	Avg.
	LoRA	DenseLoRA									
0.25	QKV	UD	74.3	86.7	79.9	94.2	85.6	77.0	88.5	83.8	83.8
0.49	UD	QKV	72.5	88.0	81.4	93.8	84.1	74.7	86.8	84.0	83.2
0.01	-	QKV	72.2	86.5	78.6	90.8	82.4	77.0	90.2	80.4	82.3
0.02	-	UD	71.0	87.3	78.9	94.0	84.8	79.3	90.3	84.4	83.8

Table 8: Accuracy(%) comparison of several different tuning granularity of DenseLoRA fine-tuning LLaMA3-8B. Each module is represented by its first letter as follows: (Q)uery, (K)ey, (V)alue, (O)utput, (U)p, (D)own.



(a) (a) ΔA (b) (b) ΔB (c) (c) $\Delta \boldsymbol{M}$

Figure 6: Increamts of matrices A and B of LoRA compared to matrix $\boldsymbol{M}$ of DenseLoRA. We randomly select the slices of a small set of A and B for demonstration.