Dual LoRA: Enhancing LoRA with Magnitude and Direction Updates

Anonymous ACL submission

Abstract

Low-rank adaptation (LoRA) is one of the most popular methods among parameter-efficient fine-tuning (PEFT) methods to adapt pre-trained large language models (LLMs) to specific downstream tasks. However, the model trained based on LoRA often has an unsatisfactory performance due to its low-rank assumption. In this paper, we propose a novel method called Dual LoRA to improve the performance by incorporating an inductive bias into the original LoRA. Specifically, we separate low-rank matrices into two groups: the magnitude group to control whether or not and how far we should update a parameter and the direction group to decide whether this parameter should move forward or backward, to better simulate the parameter updating process of the full fine-tuning based on gradient-based optimization algorithms. We show that this can be simply achieved by adding a ReLU function to the magnitude group and a sign function to the direction group. We conduct several experiments over a wide range of NLP tasks, including natural language understanding (NLU), generation (NLG), and commonsense reasoning datasets on LLaMA-1/2/3, RoBERTa, DeBERTa, and GPT2 as baseline models. Results show that we consistently outperform LoRA and its SOTA variants with the same number of trainable parameters.

1 Introduction

Large language models (LLMs) have shown promising results on almost all natural language processing (NLP) tasks (Touvron et al., 2023a; Achiam et al., 2023) and other multi-modal tasks (Liu et al., 2024a), by adapting a well trained LLM to different downstream applications. Full fine-tuning (FFT) is a straightforward way to achieve this goal, but it requires tremendous computational resources and time to complete the fine-tuning process. Thus, parameter-efficient fine-tuning (PEFT) which updates a small fraction (less than 2%) of parameters has attracted more and more attention due to its low memory and time requirements.

Traditional PEFT methods include adapter tuning (Hu et al., 2023) which adds trainable tiny modules to adapt to downstream tasks, prompt tuning (Peng et al., 2024) that inserts learnable prompt vectors to the existing input, and low-rank adaptation (LoRA) (Hu et al., 2021a) which updates the original parameters by adding low-rank matrices. Among them, LoRA surpasses other methods by achieving better performance without generating additional inference costs.

Many follow-ups manage to improve the fine-tuning performance of LoRA. LoRA+ (Hayou et al., 2024) uses different learning rates to update low-rank matrices A and B and enhance the performance with a well-chosen learning rate ratio. DoRA (Liu et al., 2024b) decomposes the original weight matrix into a normalized matrix and its corresponding norm and applies the original LoRA to the normalized matrix. FLoRA (Si et al., 2024) generates LoRA to high dimensional space and inserts a low-rank core matrix into the original LoRA matrices to improve its performance. MoRA (Jiang et al., 2024) replaces the low-rank matrices with a square matrix to achieve high-rank updating and applies a compress layer and a decompress layer to maintain a roughly similar number of trainable parameters. However, they share a common drawback: as the trainable parameters are much fewer than those of FFT, updating them without incorporating prior knowledge will inevitably result in unsatisfactory model accuracy.

Thus, in this paper we introduce an inductive bias into the original LoRA method, *i.e.,* to simulate the parameter updating process of FFT, which utilizes gradient-based optimization algorithms. Specifically, we divide the low-rank matrices into two groups: the magnitude group, which controls whether and to what extent a parameter should

be updated; and the direction group, which determines the direction of the update——whether it should be positive or negative. The whole fine-tuning process can be treated as adjusting the sign and magnitude of each element in the update matrix and adding them back to the original parameters to gradually achieve the optimal solution. We conduct experiments to validate the effectiveness of our method over a wide range of NLP tasks including natural language understanding (NLU), generation (NLG) and commonsense reasoning to make a fair comparison with SOTA methods. Evaluation results on different LLM models such as LLaMA-7B/13B, LLaMA2-7B, LLaMA3-8B, LLaMA3-70B-Instruct, RoBERTa, DeBERTa, and GPT2 show that we can achieve consistent improvements over these SOTA methods by using the same number of training parameters.

2 Related Works

In this section, we introduce different parameter-efficient fine-tuning (PEFT) methods, followed by a deeper dive into the LoRA series methods.

2.1 PEFT Methods in LLMs

Prefix tuning is the first kind of methods (Li and Liang, 2021; Liu et al., 2022; Zhang et al., 2024) in PEFT. It was first proposed by Li *et.al.* (Li and Liang, 2021), which was a lightweight alternative to FFT that kept LLM parameters frozen and only optimized a sequence of continuous task-specific vectors called prefix. Dynamic prefix-tuning (Liu et al., 2022) proposed a generative template-based event extraction method with dynamic prefixes by integrating context information with type-specific prefixes to learn a context-specific prefix for each context. Selective prefix-tuning (Zhang et al., 2024) showed that prefix tokens carried context-specific information and enhanced their specialization can improve model performance. Thus, they integrated a selective mechanism inspired by selective self-attention and introduced selective loss to encourage diversity in prefix tokens.

Prompt tuning is the second kind of PEFT method that added trainable embeddings to original word embeddings and learned these soft prompts through back-propagation and tuned them to incorporate signals from any number of labeled examples (Lester et al., 2021). P-Tuning v2 (Liu et al., 2021) empirically found that properly optimized prompt tuning can be universally effective across a wide range of model scales and NLU tasks, which increased the capacity of continuous prompts and closed the gap to FFT. Knowledgeable Prompt-tuning (Hu et al., 2021b) improved and stabilized the original prompt-tuning method by expanding the label word space of the verbalizer with external knowledge bases and refining it with PLM before predicting.

Representation fine-tuning (REFT) aims to train interventions that manipulate model representations to steer model behaviors on downstream tasks at inference time. ReFT (Wu et al., 2024) introduced a family of ReFT methods that operated on a frozen base model and learned task-specific interventions on hidden representations.

Although the aforementioned methods improved the performance of LLMs in downstream tasks, they suffered the problem that the original architecture of the baseline model needed to be changed and the inference speed was slowed down. Compared to them, LoRA-based methods had exactly the same inference latency to the baseline LLMs.

2.2 LoRA-Based Methods

LoRA (Hu et al., 2021a) assumed that only a small number of task-specific parameters needed to be tuned to fit the downstream tasks and updated the weights with two low-rank matrices. These matrices can be merged back into the original weights during inference to avoid additional computational costs. LoRA+ (Hayou et al., 2024) argued that LoRA led to sub-optimal results, and the problem can be corrected by setting different learning rates for the low-rank matrices A and B with a fixed learning rate ratio. MoRA (Jiang et al., 2024) believed that the low-rank updating mechanism limited the ability of LLMs and used a square matrix to achieve high-rank updating with the same number of trainable parameters. Two non-parameter operators were used to reduce the input dimension and increase the output dimension of this square matrix. DoRA (Liu et al., 2024b) decomposed the pre-trained weight into magnitude and direction for fine-tuning, and employed original LoRA for direction component update to accelerate the training process.

The methods mentioned above can improve the performance of downstream tasks. However, their performance is still unsatisfactory because of the low-rank assumption (Hu et al., 2021a; Hayou et al., 2024; Liu et al., 2024b). Although MoRA (Jiang et al., 2024) attempted to address this issue by us-

ing a high-rank matrix, its rank and the number of trainable parameters remained significantly lower than those in FFT. Thus, it is difficult to achieve satisfactory model performance without incorporating prior knowledge into the training process.

Note that both DoRA and our method have magnitude and direction groups, but the meaning behind them is totally different. The direction and magnitude in DoRA can be treated as a normalized weight matrix and its corresponding norm. In our method, we are trying to simulate the parameter updating process of FFT which utilizes gradient-based optimization algorithms. Thus, the direction and magnitude control the sign and to what extent a parameter should be updated.

Another family of methods aim to modify the gradient calculation and backward propagation process of training, such as GaLore (Zhao et al., 2024), FLoRA (Hao et al., 2024) and GaRare (Liu et al.). These methods are orthogonal to the proposed Dual LoRA which only focuses on the architecture and forward pass modification, and a detailed discussion falls outside the scope of this paper.

3 Method

In this section, we first introduce the preliminaries of LoRA and optimization methods. Then, we give a thorough analysis of our proposed Dual LoRA and explain its advantage over previous LoRA-based methods.

3.1 Low-Rank Adaptation (LoRA)

Given a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, LoRA (Hu et al., 2021a) assumes that a low "intrinsic rank" is enough during adaptation on downstream tasks and constrains the updated matrix with a low-rank decomposition:

$$W' = W_0 + \Delta W = W_0 + \frac{\alpha}{r} \cdot BA, \quad (1)$$

where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ are two low-rank matrices with rank $r \ll min(d, k)$, α is a fixed hyper-parameter to control the influence of the low-rank matrices, and W' is the final weight matrix after fine-tuning.

Given an original forward pass $h = W_0x$ with an input x, the modified forward pass can be expressed as:

$$h = W_0x + \Delta Wx = (W_0 + \frac{\alpha}{r} \cdot BA)x. \quad (2)$$

Note that the usage of LoRA does not affect the inference speed since the low-rank matrices A and B can be merged back into the original weight W_0, and the dimension of the final weight matrix W' is the same as the pre-trained weight matrix W_0. Since the trainable low-rank matrices have fewer parameters (less than 2%) compared to the original matrices, LoRA usually has insufficient performance.

3.2 Optimization Methods

Given a loss function $\ell(\hat{y}, y)$ which measures the cost between the prediction $\hat{y}$ and the ground-truth label y, we can choose a family $\mathcal{F}$ of functions $f_w(x)$ with learnable weight w and input x, and seek the function $f \in \mathcal{F}$ to minimize the loss $\ell(f_w(x), y)$ averaged on the input examples:

$$E_n(f_w) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_w(x_i), y_i). \quad (3)$$

In order to minimize the empirical risk $E_n(f_w)$, a global optimum weight w^* needs to be found step by step using a series of optimization methods. Specifically, we have:

$$w_{t+1} = w_t + \gamma\Delta w, \quad (4)$$

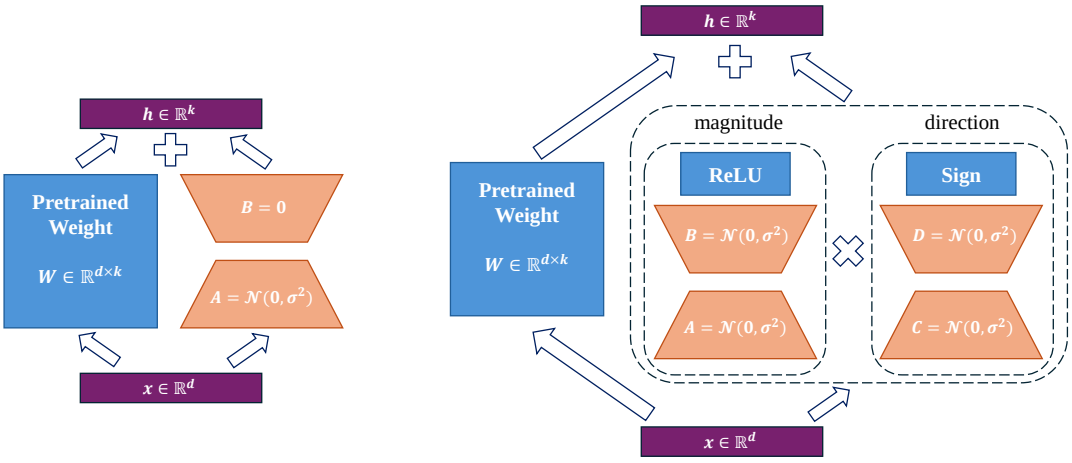
where γ is the learning rate and w_{t+1} is expected to converge to the global optimum w^* as the training proceed.

To achieve this, different optimization methods leverage different ways to compute Δw. For example, gradient descent (Bottou, 2010) uses $\Delta w = \frac{1}{n}\sum_{i=1}^{n} \nabla_w \ell(f(x_i), y_i)$ to compute the update, and Adam (Kingma, 2014) uses $\Delta w = \hat{m}_t/(\sqrt{\hat{v}_t}+\epsilon)$ where $\hat{m}_t$ and $\hat{v}_t$ are the first-moment estimate and second-moment estimate, and $\epsilon = 10^{-8}$.

Both FFT and LoRA fine-tune the model based on the optimization methods mentioned above. However, FFT assumes Δw is a full-rank matrix while LoRA decomposes Δw into two low-rank matrices and trains them without any other prior knowledge, which is the main reason that causes the performance drop.

3.3 Dual LoRA

Note that the update matrix Δw can always be decomposed into magnitude and direction regardless of the optimization method used. Learning these components separately can be treated as adding an inductive bias into the original LoRA, aiding in facilitating the search for the optimal solution within the solution space.



(a) LoRA (b) Dual LoRA

Figure 1: The architecture of the original LoRA and our proposed Dual LoRA. The low-rank update matrices are separated into the magnitude group and the direction group.

Instead of using two low-rank matrices, we use four low-rank matrices and separate them into a magnitude group and a direction group in Dual LoRA, as shown in Fig. 1.

Magnitude group. Given two low-rank matrices $A \in \mathbb{R}^{r_1 \times k}$ and $B \in \mathbb{R}^{d \times r_1}$, the magnitude group can be computed as:

$$W_m = \mathrm{ReLU}(BA), \tag{5}$$

which has two effects. Firstly, non-negative outputs can be treated as learning the magnitude of the update during the training process. Secondly, we can easily freeze some of the elements that are already well-trained for the downstream tasks in the original weight matrix by learning the output elements of BA to be negative and filter them out with ReLU function, which is hard for previous LoRA-based methods to achieve such a goal.

Direction group. Given two low-rank matrices $C \in \mathbb{R}^{r_2 \times k}$ and $D \in \mathbb{R}^{d \times r_2}$, the direction group can be computed as:

$$W_d = \mathrm{Sign}(DC), \tag{6}$$

where $\mathrm{Sign}(\cdot)$ is an element-wise operation that outputs $+1$ for positive input and -1 otherwise. Note that the gradient of the sign function is zero almost everywhere, and backward propagation cannot be applied during training. Thus, given $x_b = Sign(x)$, the straight-through estimator (STE) method (Bengio et al., 2013) is introduced to compute its gradient as:

$$\frac{\partial \mathcal{L}}{\partial x} = \mathrm{Clip}(\frac{\partial \mathcal{L}}{\partial x_b}, -1, 1), \tag{7}$$

in which $\mathcal{L}$ is the corresponding loss function for a downstream task and:

$$\mathrm{Clip}(x, -1, 1) = \begin{cases} -1, & \text{if } x < -1, \\ x, & \text{if } -1 \le x < 1, \\ 1, & \text{otherwise.} \end{cases} \tag{8}$$

The direction group can control the sign of each element in the update matrix, which is a two-way direction to decide whether the element in the original weight matrix should move forward or backward.

Overall update. Given a pre-trained weight matrix W_0, the overall update of our Dual LoRA can be expressed as:

$$W' = W_0 + \Delta W = W_0 + \frac{\alpha}{\sqrt{r_1 r_2}} W_m \odot W_d, \tag{9}$$

where $\odot$ represents an element-wise product (Hadamard product) between two matrices. Similarly, given the original forward pass $h = W_0 x$, the modified forward pass is:

$$h = W_0 x + \Delta W x = (W_0 + \frac{\alpha}{\sqrt{r_1 r_2}} W_m \odot W_d) x, \tag{10}$$

which does not affect the inference process as long as we merge ΔW into W_0.

Initialization. LoRA uses random Gaussian initialization for A and zero for B to make sure the update matrix is zero at the beginning of training, as shown in Fig. 1(a). In Dual LoRA, however, none of the low-rank matrices in the magnitude group should be initialized with zero. Otherwise, either all trainable parameters are dead, or we cannot achieve the goal that the update matrix is zero due to the $\mathrm{ReLU}(\cdot)$ function and $\mathrm{Sign}(\cdot)$ function.

Table 1: The results of the proposed Dual LoRA and other competitors with LLaMA-7B/13B, LLaMA2-7B, LLaMA3-8B and LLaMA3-70B-Instruct on commonsense reasoning datasets. For all matrices, higher is better.

Model	Methods	Trainable Param. (%)	Commonsense Reasoning Datasets BoolQ	PIQA	SIQA	HellaS	WinoG	ARC-e	ARC-c	OBQA	Avg.
L-7B	Adapter-P	3.54	67.9	76.4	78.8	69.8	78.9	73.7	57.3	75.2	72.2
	LoRA ($r = 32$)	0.83	67.5	80.8	78.2	83.4	80.4	78.0	62.6	79.1	76.3
	DoRA ($r = 32$)	0.84	69.7	83.4	78.6	87.2	81.0	81.9	66.2	79.2	78.4
	DoRA ($r = 64$)	1.65	70.1	82.0	75.6	85.9	79.7	79.1	63.7	78.4	76.8
	Dual LoRA ($r = 16$)	0.83	**71.3**	**83.4**	**81.0**	**88.1**	**83.7**	**85.0**	**71.0**	**82.4**	**80.7**
L-13B	LoRA ($r = 32$)	0.67	71.6	83.4	80.0	89.9	84.2	81.2	67.7	80.8	79.9
	DoRA ($r = 16$)	0.35	71.7	84.2	80.6	90.5	85.2	83.1	68.4	80.4	80.5
	DoRA ($r = 32$)	0.68	**72.4**	84.9	81.2	**91.5**	83.7	84.6	68.9	81.6	81.1
	Dual LoRA ($r = 16$)	0.67	**72.4**	**87.4**	**81.9**	90.9	**86.1**	**89.1**	**77.4**	**88.2**	**84.2**
L2-7B	LoRA ($r = 16$)	0.41	70.4	82.9	79.0	81.3	81.5	82.4	69.2	80.4	78.4
	LoRA ($r = 32$)	0.83	68.9	82.2	78.1	86.9	81.2	79.3	65.4	78.4	77.6
	DoRA ($r = 16$)	0.43	72.0	83.1	79.9	89.1	83.0	84.5	71.0	81.2	80.5
	DoRA ($r = 32$)	0.84	71.8	83.7	76.0	89.1	82.6	83.7	68.2	82.4	79.7
	HiRA ($r = 32$)	0.83	71.2	83.4	79.5	88.1	84.0	86.7	73.8	84.6	81.4
	Dual LoRA ($r = 16$)	0.83	**73.8**	**83.8**	**80.3**	88.1	**85.2**	**87.7**	**74.1**	**85.0**	**82.3**
L3-8B	RandLoRA	0.70	**76.3**	88.1	80.3	95.7	86.1	90.4	80.9	87.0	85.6
	LoRA ($r = 16$)	0.35	71.7	86.8	79.5	93.9	84.4	87.4	76.3	84.2	83.0
	LoRA ($r = 32$)	0.70	71.2	85.1	79.3	92.1	82.6	85.2	70.1	81.4	80.9
	DoRA ($r = 16$)	0.35	74.5	88.8	80.3	95.5	84.7	90.1	79.1	87.2	85.0
	DoRA ($r = 32$)	0.71	74.6	89.3	79.9	95.5	85.6	90.5	80.4	85.8	85.2
	HiRA ($r = 32$)	0.70	75.4	**89.7**	81.2	95.4	87.7	93.3	82.9	**88.3**	86.7
	Dual LoRA ($r = 16$)	0.70	76.2	89.1	**83.2**	**96.0**	**89.1**	**93.7**	**85.4**	87.6	**87.5**
L3-70B	LoRA ($r = 16$)	0.197	78.6	92.8	83.4	92.7	92.6	97.5	91.7	94.4	90.5
	DoRA ($r = 16$)	0.202	78.4	93.0	83.8	96.5	92.3	97.6	92.3	94.6	91.1
	Dual LoRA ($r = 8$)	0.197	**82.5**	**95.2**	**85.8**	**97.7**	**94.6**	**98.2**	**93.6**	**96.6**	**93.0**

Specifically, given

$$\Delta W = \frac{\alpha}{\sqrt{r_1 r_2}} \mathrm{ReLU}(BA) \odot \mathrm{Sign}(DC), \quad (11)$$

we can compute the gradient of the loss function $\mathcal{L}$ with respect to four low-rank matrices as:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial A} &= \frac{\partial \mathcal{L}}{\partial \Delta W} \cdot \frac{\alpha}{\sqrt{r_1 r_2}} B^\top \cdot \mathrm{Sign}(DC) \cdot \mathbb{1}_{BA>0}, \\
\frac{\partial \mathcal{L}}{\partial B} &= \frac{\partial \mathcal{L}}{\partial \Delta W} \cdot \frac{\alpha}{\sqrt{r_1 r_2}} \mathrm{Sign}(DC) \cdot \mathbb{1}_{BA>0} \cdot A^\top, \\
\frac{\partial \mathcal{L}}{\partial C} &= \mathrm{Clip}(\frac{\partial \mathcal{L}}{\partial \Delta W}, -1, 1) \cdot \frac{\alpha}{\sqrt{r_1 r_2}} D^\top \cdot \mathrm{ReLU}(BA) \\
\frac{\partial \mathcal{L}}{\partial D} &= \mathrm{Clip}(\frac{\partial \mathcal{L}}{\partial \Delta W}, -1, 1) \cdot \frac{\alpha}{\sqrt{r_1 r_2}} \mathrm{ReLU}(BA) \cdot C^\top,
\end{aligned} \quad (12)$$

where $\mathbb{1}$ is the indicator function.

It is easy to know that when setting $A = 0$ or $B = 0$, we will have $\mathbb{1}_{BA>0} = 0$ and $\mathrm{ReLU}(BA) = 0$ and all four gradients in Eq. 12 are zeros which will cause the training process to be dead. Setting $C = 0$ or $D = 0$ will not result in such a problem, but it cannot achieve the goal that the update matrix Eq. 11 is zero at the beginning of training since $\mathrm{Sign}(x)$ always outputs $+1$ or -1 depending on the input. Thus, during the experiments, we use random Gaussian initialization for all four low-rank matrices and apply a warm-up strategy for the first few training steps to make sure that $\Delta W = 0$ at the start.

4 Experiments

In this section, we evaluate the effectiveness of the proposed Dual LoRA on various NLP tasks. We compare our methods with other PEFT competitors by fine-tuning LlaMA-7B/13B, LLaMA2-7B, LLaMA3-8B, and LLaMA3-70B-Instruct models on a series of commonsense reasoning datasets. Then, we explore the ability of our method on the neural language understanding (NLU) dataset GLUE by fine-tuning RoBERTa base/large and DeBERTa XXL. Furthermore, we conduct experiments on neural language generation (NLG) with E2E NLG Challenge dataset using GPT2_M and GPT2_L as backbones (see Appendix A). All experiments above show that Dual LoRA can surpass other LoRA-based methods with the same or fewer trainable parameters and achieve state-of-the-art results. Finally, we further analyze our method

by performing a series of ablation studies. In the following experiments, we set the rank of the magnitude group and the direction group as the same, *i.e.*, $r_1 = r_2 = r$ unless specified.

Competitors. We compare Dual LoRA with a series of baseline methods including LoRA-based methods (LoRA (Hu et al., 2021a), LoRA+ (Hayou et al., 2024), GaLore (Zhao et al., 2024), GaRare (Liu et al.), Delta-LoRA (Zi et al., 2023), CorDA (Yang et al., 2024), VeRA (Kopiczko et al., 2024), RandLoRA (Albert et al., 2025), DoRA (Liu et al., 2024b), and HiRA (Huang et al., 2025)) and other PEFT methods (efficient adapter design with LayerNorm (Adapter-L) (Lin et al., 2020), parallel adapter tuning (Adapter-P) (He et al., 2021) and prefix-layer tuning (Prefix) (Li and Liang, 2021)).

4.1 Commonsense Reasoning

Datasets and baseline models. We evaluate Dual LoRA and different PEFT methods on the commonsense reasoning task which is composed of eight different sub-tasks including BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), Social IQa (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-easy/challange (Clark et al., 2018) and OpenBookQA (Mihaylov et al., 2018). Similarly to DoRA, we merge the training sets from all subtasks to get the final training set and perform evaluations on their own testing datasets for each task.

For the baseline models, we use LLaMA-7B/13B (Touvron et al., 2023a), LLaMA2-7B (Touvron et al., 2023b), LLaMA3-8B (Dubey et al., 2024), and LLaMA3-70B-Instruct (Dubey et al., 2024). We halve the rank of our low-rank matrices to ensure that the same number of trainable parameters are used compared to other LoRA-based methods. We tune the learning rate for our method, and all other training hyper-parameters are kept unchanged as in HiRA in order to make a fair comparison. We train one epoch for LLaMA3-70B-instruct, and three epochs for other baseline models.

Results. The results in Tab. 1 show that we can consistently outperform DoRA and HiRA with less trainable parameters on all of the baseline models. For example, Dual LoRA enhances the average accuracy by 2.3%/3.1% compared to the previous best result on LLaMA-7B/13B. The performance gains are still notable on LLaMA2-7B, LLaMA3-8B and LLaMA3-70B-Instruct, which are 0.9%, 0.8% and 1.9%.

4.2 Neural Language Understanding (NLU)

Datasets and baseline models. We evaluate our method on a widely used natural language understanding dataset GLUE. It consists of eight different datasets includes MNLI (Williams et al., 2017), SST-2 (Socher et al., 2013), MRPC (Dolan and Brockett, 2005), CoLA (Warstadt, 2019), QNLI (Rajpurkar et al., 2018), QQP, RTE, and STS-B (Cer et al., 2017). The diversity makes the GELU benchmark a robust dataset for evaluating LLMs on NLU tasks.

For the baseline models, we use RoBERTa base/large (Liu, 2019) and DeBERTa XXL (He et al., 2020) as pretrained baseline models from the HuggingFace Transformers library (Wolf et al., 2020). Similarly to LoRA, we initialize the model to the LoRA-adapted MNLI checkpoint for MRPC, RTE, and STSB rather than the pre-trained baseline model. All other training parameters are the same as LoRA except for the learning rate.

Results. As shown in Tab. 2, the proposed Dual LoRA shows state-of-the-art results on all three baseline models. For example, on the small model RoBERTa base we can defeat previous methods LoRA, LoRA+, and DoRA by 1.6%, 1.2%, and 1.8% average accuracy. Similarly, on medium-sized model RoBERTa large, our Dual LoRA surpasses LoRA, LoRA+, and DoRA by 1.1%, 0.4%, and 0.9%. On DeBERTa XXL model with over 1500M total parameters, Dual LoRA can still exceed LoRA, LoRA+, and DoRA by 1.3%, 1.1%, and 1.1%. Note that we can even surpass the FFT methods on these baseline models by 1.9%, 0.8%, and 0.5%, which shows the priority of the proposed method.

4.3 Ablation Study

We conduct several ablation studies to further verify the effectiveness of the proposed method.

Dealing with the sign function. In the previous section and experiments, we use the straight-through estimator (STE) method (Bengio et al., 2013) to compute the gradient of the sign function. Note that the sign function is a standard operator in the area of binary neural networks (BNNs), and there are many studies on dealing with the forward and backward passes of the sign function. For example, XNOR-Net (Rastegari et al., 2016) scales

Table 2: The results of the proposed Dual LoRA and other competitors with RoBERTa base/large and DeBERTa XXL on GLUE datasets. For all matrices, higher is better.

Model	Methods	Trainable Param. (M)	GLUE								
			MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
RoB_{base}	FFT	125.0	87.6	94.8	90.2	63.6	92.8	**91.9**	78.7	91.2	86.4
	GaLore ($r = 8$)	0.3	87.2	94.4	92.0	61.8	92.3	91.2	79.1	90.8	85.9
	GaRare ($r = 8$)	0.3	87.2	94.4	91.5	61.1	92.3	90.9	79.3	90.3	85.9
	Delta-LoRA ($r = 8$)	0.3	87.5	95.1	90.2	63.8	93.1	90.9	87.0	91.6	87.4
	CorDA ($r = 128$)	21	-	93.1	89.7	59.6	91.5	-	**88.1**	90.2	-
	VeRA	0.3	-	91.9	88.4	59.9	90.5	-	74.9	90.4	-
	RandLoRA	0.7	-	92.2	88.0	59.4	91.3	-	74.7	90.3	-
	LoRA ($r = 8$)	0.3	87.0	94.6	89.2	60.9	92.9	90.7	92.0	91.1	86.1
	LoRA ($r = 16$)	0.6	87.0	95.1	89.0	63.9	93.0	91.2	83.4	91.1	86.7
	LoRA+ ($r = 16$)	0.6	**87.8**	95.2	90.4	65.9	92.6	91.2	82.3	91.4	87.1
	DoRA ($r = 16$)	0.6	87.7	95.3	87.8	64.8	92.6	90.8	82.2	90.8	86.5
	Dual LoRA ($r = 8$)	0.6	**87.8**	**95.8**	**91.7**	**67.8**	**93.3**	90.7	**88.1**	**91.7**	**88.3**
$\text{RoB}_{\text{large}}$	FFT	355.0	90.2	**96.4**	90.9	68.0	94.7	**92.2**	86.6	92.4	88.9
	GaLore ($r = 16$)	1.6	90.8	96.1	91.7	68.3	95.7	91.9	87.0	92.5	89.3
	GaRare ($r = 16$)	1.6	**91.3**	96.2	91.7	67.9	94.6	91.8	87.4	92.3	89.2
	VeRA	0.3	-	95.8	89.3	65.3	94.1	-	81.6	91.8	-
	RandLoRA	1.8	-	95.5	90.1	67.4	94.1	-	84.5	91.4	-
	LoRA ($r = 8$)	0.8	90.2	95.6	89.5	63.8	94.5	91.5	88.8	92.5	88.3
	LoRA ($r = 16$)	1.6	90.2	95.9	90.9	66.0	94.4	91.6	87.4	92.3	88.6
	LoRA+ ($r = 16$)	1.6	90.3	96.3	91.4	68.7	94.7	91.6	88.8	92.5	89.3
	DoRA ($r = 16$)	1.6	90.5	96.2	89.7	68.5	92.6	91.5	89.2	92.3	88.8
	Dual LoRA ($r = 8$)	1.6	90.5	**96.4**	**91.9**	**70.2**	**95.1**	91.2	**89.5**	**92.6**	**89.7**
DeB_{XXL}	FFT	1500.0	91.8	97.2	**92.0**	72.0	96.0	**92.7**	93.9	92.9	91.1
	LoRA ($r = 16$)	4.7	91.7	96.6	89.7	70.8	95.7	92.6	95.0	92.4	90.6
	LoRA ($r = 32$)	9.4	**92.0**	**97.5**	91.2	68.7	96.0	91.9	92.8	92.4	90.3
	LoRA+ ($r = 32$)	9.4	91.7	**97.5**	91.2	68.7	96.0	92.3	94.6	92.4	90.5
	DoRA ($r = 32$)	9.4	91.9	96.9	90.9	71.2	95.8	92.3	92.6	92.3	90.5
	Dual LoRA ($r = 16$)	9.4	91.9	97.1	91.9	**74.0**	**96.2**	92.6	**95.3**	**93.4**	**91.6**

Table 3: Different methods are used to deal with sign function. The experiments are conducted on LLaMA-7B and the commonsense reasoning dataset.

Method	Trainable Params (%)	Avg.
STE (ours)	1.64	80.7
XNOR-Net	1.64	79.4
Dorefa-Net	1.64	79.8

the weights after binarized:

$$\textbf{Forward:}\ x_b = \text{Sign}(x) \times \mathbf{E}_F(|x|),$$
$$\textbf{Backward:}\ \frac{\partial \mathcal{L}}{\partial x} = \frac{\partial \mathcal{L}}{\partial x_b}, \tag{13}$$

where $\mathbf{E}_F(|x|)$ is the mean of the absolute value of each output channel of weights. Dorefa-Net (Zhou et al., 2016) uses a constant scalar to scale all of the weights instead of doing channel-wise scaling:

$$\textbf{Forward:}\ x_b = \text{Sign}(x) \times \mathbf{E}(|x|),$$
$$\textbf{Backward:}\ \frac{\partial \mathcal{L}}{\partial x} = \frac{\partial \mathcal{L}}{\partial x_b}. \tag{14}$$

The experimental results of using different methods to deal with the sign function are shown in Tab. 3. The original STE method performs best among different methods. This conclusion is different from that in BNNs. We analyze that this is because both XNOR-Net (Rastegari et al., 2016) and Dorefa-Net (Zhou et al., 2016) modify the forward pass of sign function by adding per-channel scales or a constant scale, which contaminate the direction group and make it unable to focus on giving the correct binary outputs.

The influence of r_1 and r_2. As shown in Sec. 3.3, r_1 and r_2 are the ranks of the low-rank matrices in the magnitude and direction groups, respectively. In previous experiments, we set $r_1 = r_2 = r$ in default to avoid introducing new hyper-parameters compared to other LoRA-based methods. Thus, in this section we dive deeper into the influence of r_1 and r_2.

Specifically, we keep the total trainable parameters unchanged by setting $r_1 + r_2 = 2r$, and adjust the ratio of parameters in the magnitude group and direction group, which are controlled by r_1 and r_2, respectively. We conduct experiments on the commonsense reasoning dataset with LLaMA3-8B as our baseline model and set 15 different ratios. Other training hyper-parameters are kept the same

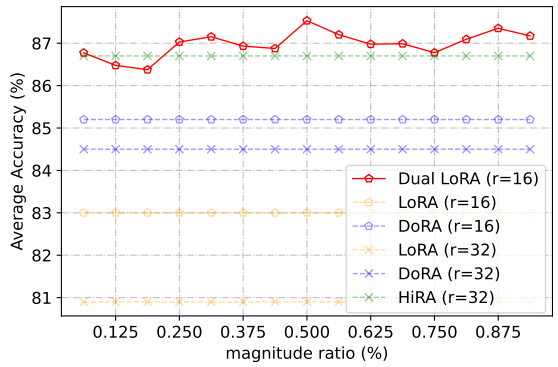


Figure 2: Average accuracy on commonsense reasoning datasets using LLaMA3-8B as the baseline model with $r_1 = \{2, 4, \cdots, 30\}$ and $r_2 = 32 - r_1$ in the experiments. The red line is the proposed Dual LoRA, the blue/orange lines represent DoRA/LoRA with different ranks.

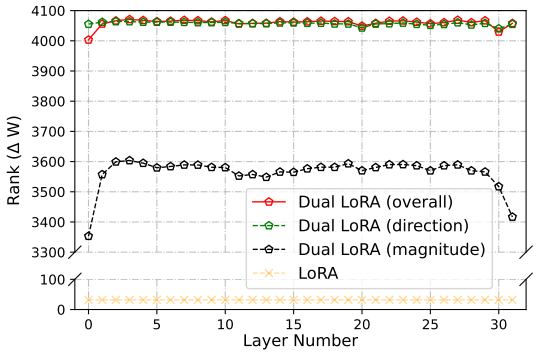


Figure 3: The average rank of ΔW for LoRA, magnitude group of Dual LoRA, direction group of Dual LoRA, and the overall Dual LoRA. The experiments are conducted on LLaMA2-7B.

as in the previous experiments. The results are shown in Fig. 2. We can see that the proposed Dual LoRA can consistently outperform LoRA, DoRA, and HiRA on a wide range of parameter ratio between the magnitude group and direction group (from 25% to 94%), which shows the robustness of our method.

More ablation studies are shown in Appendix B.

5 Analysis of the Rank of Update Matrix

In this section, we given an analyze of the rank of the update matrix Eq. 11 to further show the priority of our method. Note that for a given matrix $X \in \mathbb{R}^{m\times n}$, $\text{Rank}(x) \leq \min(m, n)$ always holds true. Thus, in the original LoRA, given $A \in \mathbb{R}^{r\times k}$ and $B \in \mathbb{R}^{d\times r}$ with $r \ll \min(d, k)$, the rank of the update matrix $\Delta W = \frac{\alpha}{r} \cdot BA$ is upper bounded by:

$$\begin{aligned} \text{Rank}(\Delta W) &= \text{Rank}(BA) \\ &\leq min(\text{Rank}(A), \text{Rank}(B)) \leq r. \end{aligned} \quad (15)$$

In the proposed Dual LoRA, however, we found that the rank of the update matrix can achieve a higher upper bound. Specifically, given two low-rank matrices $A \in \mathbb{R}^{r_1\times k}$ and $B \in \mathbb{R}^{d\times r_1}$ in the magnitude group and two low-rank matrices $C \in \mathbb{R}^{r_2\times k}$ and $D \in \mathbb{R}^{d\times r_2}$ in the direction group with $r_1, r_2 \ll \min(d, k)$, the rank of the update matrix $\Delta W' = \frac{\alpha}{\sqrt{r_1 r_2}} \text{ReLU}(BA) \odot \text{Sign}(DC)$ is:

$$\begin{aligned} \text{Rank}(\Delta W') &= \text{Rank}(\text{ReLU}(BA) \odot \text{Sign}(DC)) \\ &\leq \text{Rank}(\text{ReLU}(BA)) \times \text{Rank}(\text{Sign}(DC)) \\ &\leq \min(k, d)^2. \end{aligned} \quad (16)$$

Note that the $\text{ReLU}(\cdot)$ and $\text{Sign}(\cdot)$ operations break the low-rank limitation of the original input matrix and derive output matrices with high rank.

In Fig. 3, we explicitly show the average rank of the update matrix in LoRA and the proposed Dual LoRA (magnitude group, direction group, and overall) over different layers. The experiments are conducted on LLaMA2-7B. We can see that the update matrix and the direction group almost achieve full rank (4096). The magnitude group has a relatively lower rank but is still much larger than that in LoRA. The results show the priority of our method from the perspective of matrix rank.

6 Conclusion

Original LoRA and its followers fine-tune the model without incorporating any prior knowledge and share a common drawback: as the trainable parameters are limited, the model accuracy is unsatisfactory. In this paper, we propose a new LoRA-based method called Dual LoRA, which incorporates an inductive bias into the original LoRA and improve the performance by introducing four low-rank matrices and separating them into the magnitude group and the direction group. The former controls the amplitude and whether or not we should update a parameter, and the latter decides whether or not this parameter should be updated in a positive or negative direction. Parameters in two groups are combined together to simulate the parameter updating process of FFT with gradient-based optimization methods. Experimental results on a wide range of NLP tasks and baseline models show that our Dual LoRA can consistently outperform LoRA and other state-of-the-art methods such as HiRA, DoRA, and LoRA+ with the same or less number of trainable parameters.

7 Limitations

Our Dual LoRA achieves state-of-the-art performance on several benchmarks and baseline models. However, it incurs a slightly higher GPU memory and training time overhead compared to the original LoRA (less than 5%), due to the intermediate feature maps generated by the magnitude group and the direction group.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Paul Albert, Frederic Z Zhang, Hemanth Saratchandran, Cristian Rodriguez-Opazo, Anton van den Hengel, and Ehsan Abbasnejad. 2025. Randlora: Full-rank parameter-efficient fine-tuning of large models. *International Conference on Learning Representations*.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*, pages 177–186. Springer.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

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*.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yongchang Hao, Yanshuai Cao, and Lili Mou. 2024. Flora: Low-rank adapters are secretly gradient compressors. *International conference on machine learning*.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. Lora+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*.

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*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

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

Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Jingang Wang, Juanzi Li, Wei Wu, and Maosong Sun. 2021b. Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification. *arXiv preprint arXiv:2108.02035*.

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*.

Ting Jiang, Shaohan Huang, Shengyue Luo, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, Qi Zhang, Deqing Wang, and 1 others. 2024. Mora: High-rank updating for parameter-efficient fine-tuning. *arXiv preprint arXiv:2405.12130*.

Diederik P Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Dawid J Kopiczko, Tijmen Blankevoort, and Yuki M Asano. 2024. Vera: Vector-based random matrix adaptation. *International Conference on Learning Representations*.

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

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

Zhaojiang Lin, Andrea Madotto, and Pascale Fung. 2020. Exploring versatile generative language model via parameter-efficient transfer learning. *arXiv preprint arXiv:2004.03829*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024a. Visual instruction tuning. *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*.

Xiao Liu, Heyan Huang, Ge Shi, and Bo Wang. 2022. Dynamic prefix-tuning for generative template-based event extraction. *arXiv preprint arXiv:2205.06166*.

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*.

Xu-Hui Liu, Yali Du, Jun Wang, and Yang Yu. On the optimization landscape of low rank adaptation methods for large language models. In *International Conference on Learning Representations*.

Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 364.

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*.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. *arXiv preprint arXiv:1706.09254*.

Cheng Peng, XI Yang, Kaleb E Smith, Zehao Yu, Aokun Chen, Jiang Bian, and Yonghui Wu. 2024. Model tuning or prompt tuning? a study of large language models for clinical concept and relation extraction. *Journal of biomedical informatics*, 153:104630.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer.

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*.

Chongjie Si, Xuehui Wang, Xue Yang, Zhengqin Xu, Qingyun Li, Jifeng Dai, Yu Qiao, Xiaokang Yang, and Wei Shen. 2024. Flora: Low-rank core space for n-dimension. *arXiv preprint arXiv:2405.14739*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 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, and 1 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

A Warstadt. 2019. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and 1 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. 2024. Reft: Representation finetuning for language models. *arXiv preprint arXiv:2404.03592*.

Yibo Yang, Xiaojie Li, Zhongzhu Zhou, Shuaiwen Song, Jianlong Wu, Liqiang Nie, and Bernard Ghanem. 2024. Corda: Context-oriented decomposition adaptation of large language models for task-aware parameter-efficient fine-tuning. *Advances in Neural Information Processing Systems*, 37:71768–71791.

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*.

Hongyi Zhang, Zuchao Li, Ping Wang, and Hai Zhao. 2024. Selective prefix tuning for pre-trained language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 2806–2813.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. 2024. Galore: Memory-efficient llm training by gradient low-rank projection. In *International conference on machine learning*. PMLR.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. 2016. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*.

Bojia Zi, Xianbiao Qi, Lingzhi Wang, Jianan Wang, Kam-Fai Wong, and Lei Zhang. 2023. Delta-lora: Fine-tuning high-rank parameters with the delta of low-rank matrices. *arXiv preprint arXiv:2309.02411*.

A Experiments on Neural Language Generation (NLG)

Datasets. We conduct experiments on E2E NLG Challenge (Novikova et al., 2017) dataset. The E2E dataset consists of approximately 42,000 training data, 4,600 validation data, and 4,600 test data. Each sample is composed of a sequence of slot-value pairs (x, y) and a corresponding natural language reference text.

Results. As in previous experiments, we reduce our rank to half of the other LoRA-based methods to ensure the same number of trainable parameters. All other hyper-parameters are the same as LoRA, except that we tune the learning rate. As the results shown in Tab. 4, Dual LoRA can consistently outperform all other competitors with baseline model GPT2_M and GPT2_L.

B More Ablation Studies

In this section, we conduct more ablation studies on the proposed Dual LoRA.

B.1 Different forms of the update matrix

Specifically, we investigate the following settings:

- Setting 1: Remove $\mathrm{ReLU}(\cdot)$ function in Dual LoRA, which means $\Delta W = \frac{\alpha}{\sqrt{r_1 r_2}}(BA) \odot \mathrm{Sign}(DC)$.
- Setting 2: Remove $\mathrm{Sign}(\cdot)$ function in Dual LoRA, which means $\Delta W = \frac{\alpha}{\sqrt{r_1 r_2}}\mathrm{ReLU}(BA) \odot (DC)$.
- Setting 3: Replace the output of the direction group with a random-initialized binary matrix and fix this matrix during training. Given W_b as the random initialized binary matrix, we have $\Delta W = \frac{\alpha}{r_1}\mathrm{ReLU}(BA) \odot W_b$.

We conduct experiments on LLaMA-7B and perform an evaluation on the commonsense reasoning dataset. The results are shown in Tab. 5. We can see that removing $\mathrm{ReLU}(\cdot)$ and $\mathrm{Sign}(\cdot)$ functions cause marginal performance drop, and using a random initialized binary matrix to replace the direction group severely degrades the performance, which shows the importance of the proposed architecture.

B.2 Different activation functions for the magnitude group

In the main paper, we use $\mathrm{ReLU}(\cdot)$ function for the magnitude group. There are other functions

Table 4: The results of the proposed Dual LoRA and other competitors with GPT2_M and GPT2_L on the E2E NLG Challenge dataset. For all matrices, higher is better.

Model	Methods	Trainable Parameters (M)	E2E NLG Challenge BLEU	NIST	MET	ROUGE-L	CIDEr
GPT2_M	FFT	354.92	68.2	8.62	46.2	71.0	2.47
	Adapter-L	11.09	68.9	8.71	46.1	71.3	2.47
	Prefix	0.35	69.7	8.81	46.1	71.4	2.49
	LoRA ($r = 4$)	0.35	68.9	8.69	46.4	71.4	2.52
	LoRA ($r = 8$)	0.70	69.9	8.77	46.8	71.7	2.50
	LoRA+ ($r = 8$)	0.70	70.2	8.81	46.6	71.6	2.53
	DoRA ($r = 8$)	0.71	69.5	8.75	46.4	71.4	2.52
	Dual LoRA ($r = 4$)	0.70	**70.6**	**8.86**	**46.9**	**72.4**	**2.56**
GPT2_L	FFT	774.03	68.5	8.78	46.0	69.9	2.45
	Adapter-L	23.00	69.1	8.68	46.3	71.4	2.49
	Prefix	0.77	70.3	8.85	46.2	71.7	2.47
	LoRA ($r = 4$)	0.77	70.3	8.85	46.8	71.9	2.52
	LoRA ($r = 8$)	1.54	70.0	8.80	46.8	71.7	**2.54**
	LoRA+ ($r = 8$)	1.54	70.0	8.83	46.8	71.9	2.53
	DoRA ($r = 8$)	1.56	69.8	8.78	46.6	71.6	2.52
	Dual LoRA ($r = 4$)	1.54	**70.6**	**8.86**	**47.1**	**72.5**	**2.54**

Table 5: The results of the previous settings on the commonsense reasoning dataset, with LLaMA-7B as the base model.

Methods	Trainable Param. (%)	Commonsense Reasoning Datasets BoolQ	PIQA	SIQA	HellaS	WinoG	ARC-e	ARC-c	OBQA	Avg.
Dual LoRA ($r = 16$)	0.83	**71.3**	**83.4**	**81.0**	**88.1**	**83.7**	**85.0**	**71.0**	**82.4**	**80.7**
Setting 1 ($r = 32$)	1.64	64.5	81.2	79.9	84.8	82.7	84.8	71.0	79.2	78.5
Setting 2 ($r = 32$)	1.64	63.3	78.8	72.9	78.9	73.3	79.5	58.8	65.6	71.4
Setting 3 ($r = 32$)	0.84	62.0	48.4	33.2	7.9	0	24.1	25.4	28.0	28.6
Setting 3 ($r = 64$)	1.64	62.2	49.9	33.5	25.6	50.5	25.6	26.6	25.8	37.5

Table 6: The results of different activation functions for the magnitude group on the commonsense reasoning dataset, with LLaMA-7B as the base model.

Activation	Commonsense Reasoning Datasets BoolQ	PIQA	SIQA	HellaS	WinoG	ARC-e	ARC-c	OBQA	Avg.
ReLU (ours)	**71.3**	**83.4**	**81.0**	**88.1**	**83.7**	**85.0**	**71.0**	**82.4**	**80.7**
Abs	70.1	82.3	78.6	86.0	81.9	82.3	70.3	81.6	79.1
Sigmoid	60.7	80.6	76.6	81.8	76.6	80.8	64.2	75.0	74.5

that can keep the activation greater than zero, such as $\mathrm{Abs}(x) = |x|$ and $\mathrm{Sigmoid}(x) = 1/(1 + e^{-x})$. We compare them in Tab. 6 on the commonsense reasoning dataset with LLaMA-7B model.

We can see that the original setting with $\mathrm{ReLU}(\cdot)$ function performs the best. This is because neither the $\mathrm{Abs}(\cdot)$ function nor the $\mathrm{Sigmoid}(\cdot)$ function can flexibly give zero output. Note that zero output means that we can easily freeze some of the elements that are already well-trained for the downstream tasks in the original weight matrix, which is the advantage of the $\mathrm{ReLU}(\cdot)$ function.