BiDoRA: Bi-level Optimization-Based Weight-Decomposed Low-Rank Adaptation

Anonymous authors
Paper under double-blind review

Abstract

Parameter-efficient fine-tuning (PEFT) of large language models (LLMs) has gained considerable attention as a flexible and efficient way of adapting LLMs to downstream tasks. Among these methods, weighted decomposed low-rank adaptation (DoRA) has emerged as a promising approach. DoRA bridges the gap between low-rank adaptation (LoRA) and full fine-tuning (FT) by decomposing the weight matrices into magnitude and direction components, thereby maintaining learning behavior similar to FT. Although DoRA shows encouraging performance, it is over-expressive and potentially increases the risk of overfitting. Moreover, optimizing magnitude and direction simultaneously leads to a coupled updating pattern, limiting its learning capacity. In this work, we propose BiDoRA, a bi-level optimization-based PEFT method. In BiDoRA, the two components are optimized at different optimization levels, mitigating the risk of overfitting. Additionally, the asynchronous optimization promotes a decoupled updating pattern, allowing for more flexible updates suitable for various downstream tasks. Evaluation of BiDoRA on various tasks spanning natural language understanding, natural language generation, token classification, and extremely small datasets reveals that it significantly outperforms DoRA and other PEFT methods. The code for BiDoRA is available at `https://anonymous.4open.science/r/BiDoRA-5D31`.

1 Introduction

Large language models (LLMs) (Radford et al., 2019; Brown et al., 2020) have achieved state-of-the-art results across a broad range of NLP tasks, from natural language understanding (NLU) (Wang et al., 2019) to natural language generation (NLG) (Novikova et al., 2017). Parameter-efficient fine-tuning (PEFT) methods (Houlsby et al., 2019; Hu et al., 2021) have been introduced as a promising solution for adapting LLMs for downstream data. PEFT approaches update only a subset of the pre-trained parameters, achieving performance comparable to full-finetuning (FT) while requiring significantly fewer computational resources.

One popular type of PEFT is low-rank adaptation (LoRA, Hu et al. (2021)), which attaches low-rank matrices to the pre-trained weights and updates only these matrices during fine-tuning. Liu et al. (2024) shows that when decomposing the weights into magnitude and direction, their correlation (as defined in a weight decomposition analysis in Liu et al. (2024)) tends to be positive in LoRA, whereas it is negative in FT. To bridge the training pattern distinction, they introduce an explicit reparameterization of the pre-trained weights matrix. The method, named DoRA, decomposes the weights into the column-wise product of magnitude and direction, which determines the direction and magnitude of the weight update, respectively. This approach enables DoRA to share similar learning patterns with FT, thereby outperforming LoRA in multiple tasks. Nonetheless, DoRA introduces *additional parameters* and *over-expressive architecture* compared to LoRA, which can exacerbate overfitting issues when adapting to small downstream datasets (See Fig. 2). Furthermore, in DoRA, the magnitude and direction components are optimized concurrently, leading to a constrained updating pattern due to shared optimization setup (e.g., learning rate, optimizer, batch size.)

To address the challenges above, we propose BiDoRA, a bi-level optimization-based weight-decomposed low-rank adaptation method for parameter-efficient fine-tuning of LLMs. BiDoRA facilitates an even more

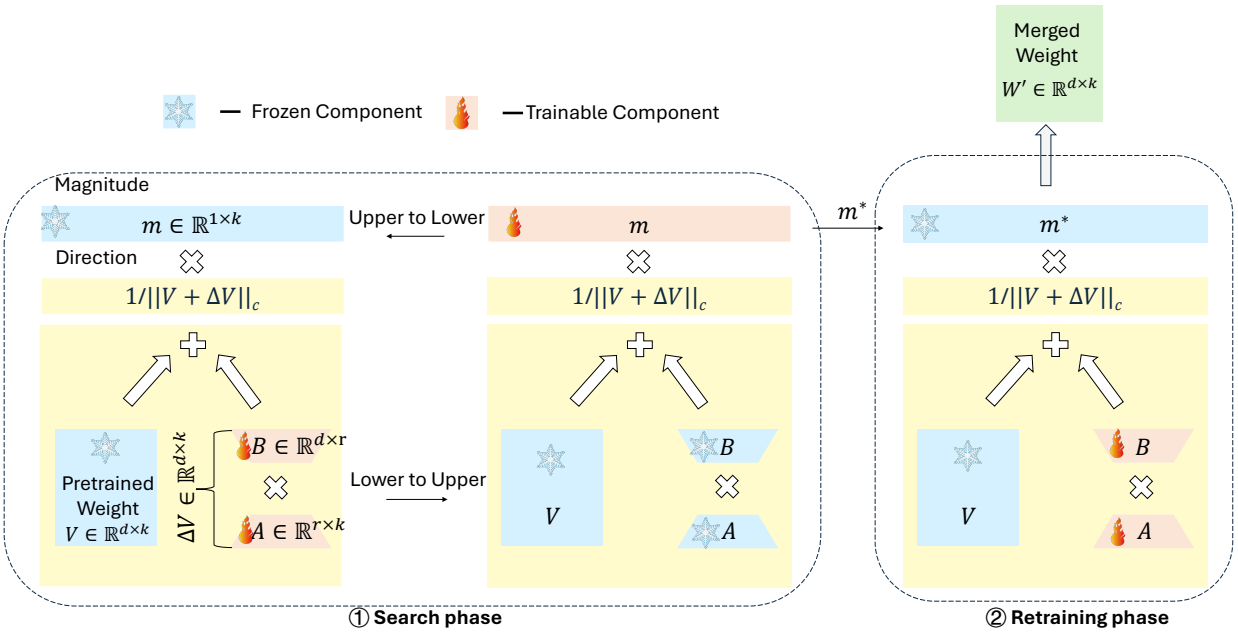


Figure 1: **An overview of BiDoRA.** BiDoRA performs PEFT using a bi-level optimization framework. At the lower level, BiDoRA learns the direction component ΔV of the update matrices using the training split of the downstream dataset. At the upper level, BiDoRA optimizes the magnitude component m with optimized ΔV from the lower level, using the validation split of the dataset. After determining the optimal magnitude, the direction component undergoes further fine-tuning on a combined set of both training and validation splits to maximize overall performance.

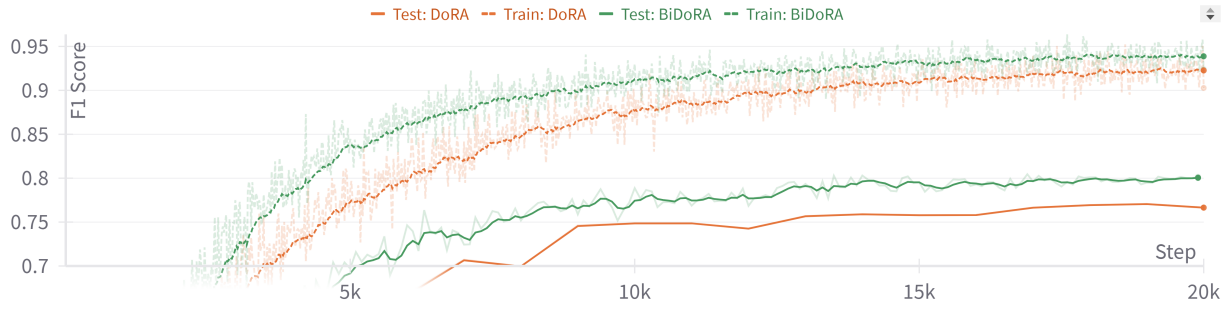


Figure 2: Training and test accuracy versus global training steps on the ModHayes split of the Reuters21578 dataset when fine-tuning a RoBERTa-base model using DoRA and BiDoRA. The training and test curves for DoRA show a larger gap compared to BiDoRA, highlighting the effectiveness of our method in reducing overfitting.

flexible updating pattern and mitigates overfitting by separately optimizing the two components on different data splits with distinct optimization levels. BiDoRA is based on a bi-level optimization framework: At the lower level, the low-rank direction component is updated using the training split, while the magnitude component remains fixed. At the upper level, the magnitude component is updated by minimizing the loss on the validation split via hypergradient descent. Subsequently, the direction component is further fine-tuned with the optimal magnitude frozen to maximize the performance. These two optimization steps are performed iteratively until convergence. Fig. 1 provides an overview of BiDoRA.

A similar strategy of combating overfitting based on bi-level optimization has been utilized in the well-established practice of differentiable neural architecture search (DARTS, Liu et al. (2018)), where architecture and sub-network are learned using different dataset splits. Optimizing the selection variables and sub-networks in a single loop can result in an over-expressive network since the selection variables tend to select all sub-networks to achieve the best expressiveness, which, however, incurs severe overfitting. In contrast, training the sub-networks with the selection module fixed on the training split while validating the effectiveness of the selection module on the unseen validation split eliminates the risk of overfitting effectively. Similarly, we treat the magnitude component as the architecture and the direction component as the sub-networks and train these components on separate datasets. As shown in Fig. 2, BiDoRA demonstrates better resistance to overfitting compared to DoRA, given the smaller performance gap between the training set and test set. Furthermore, the asynchronous gradient update steps at the two optimization levels in BiDoRA facilitate better decoupling of two components, leading to a more flexible update pattern that closely resembles FT. As illustrated in Fig. 3, the updates across different layers using BiDoRA have a correlation value that is closest to that of FT, highlighting its superior learning capability compared to both DoRA and LoRA.

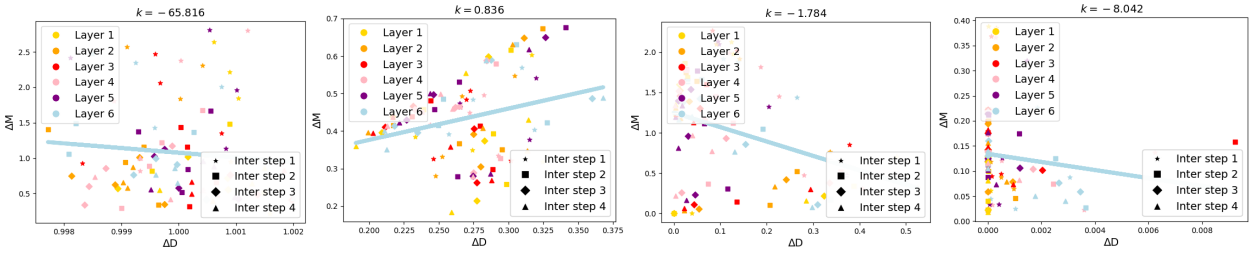
Our work makes the following key contributions:

- We propose BiDoRA, a novel PEFT method based on bi-level optimization. In contrast to DoRA, which trains the magnitude and direction components on a single dataset, BiDoRA optimizes these components at different optimization levels.
- Our strategy effectively mitigates the risk of overfitting and results in a parameter update pattern that more closely resembles full fine-tuning.
- Extensive experiments on various downstream tasks highlight the superior performance of BiDoRA. BiDoRA consistently surpasses several baseline methods, including LoRA and DoRA.

2 Related Work

2.1 Parameter Efficient Fine-Tuning Methods

Parameter-efficient fine-tuning (PEFT) methods aim to reduce the high costs associated with full fine-tuning large-scale models by updating only a relatively small subset of pre-trained parameters, rather than the entire model, to adapt to downstream tasks. Existing PEFT methods can be mainly categorized into three types. The first category, known as adapter-based methods, injects additional trainable modules into the original frozen backbone. For instance, Houlsby et al. (2019) suggests adding linear modules in sequence to existing layers, while He et al. (2021) proposes integrating these modules in parallel with the original layers to enhance performance. The second category is prompt tuning methods, which add extra soft tokens (prompts) to the initial input. During the fine-tuning stage, only these trainable soft tokens are updated, as demonstrated in works such as Lester et al. (2021) and Razdaibiedina et al. (2023). Unfortunately, the first two categories lead to increased inference latency compared to fully fine-tuned models. The third category is low-rank adaptation methods, pioneered by the foundational work LoRA (Hu et al., 2021). These methods attach low-rank matrices to pre-trained weights and use only these matrices for weight updates during fine-tuning. Since low-rank updates can be merged with pre-trained weights before inference, low-rank adaptation-based PEFT methods do not increase inference time. Following LoRA, Zhang et al. (2023) applies SVD decomposition to low-rank matrices and prunes less significant singular values for more efficient updates. Zhang et al. (2024b) uses meta-learning to search for the optimal rank of LoRA matrices, further improving its performance on downstream tasks. Most recently,Liu et al. (2024) uses weight decomposition analysis to reveal that LoRA exhibits a distinct weight updating pattern compared to FT, which may constrain its learning capacity. Therefore, DoRA (Liu et al., 2024) was then proposed to bridge the gap between LoRA and FT. DoRA decomposes the pre-trained weights into two components—magnitude and direction—and fine-tunes both, which results in a more closely aligned updating pattern compared to FT.



(a) Full FT($k = -65.816$) (b) LoRA($k = 0.836$) (c) DoRA($k = -1.784$) (d) BiDoRA($k = -8.042$)

Figure 3: Magnitude and direction updates for (a) FT, (b) LoRA, (c) DoRA, and (d) BiDoRA of the query matrices across different layers and intermediate steps after fine-tuning the GPT2 model on the E2E dataset, where k denotes the correlation value. Different markers represent matrices from different training steps, with each color corresponding to a specific layer.

2.2 Bi-level Optimization

Bi-level optimization (BLO) has been widely applied in various machine learning tasks, including meta-learning (Finn et al., 2017; Rajeswaran et al., 2019), neural architecture search (Liu et al., 2018; Zhang et al., 2021), and hyperparameter optimization (Lorraine et al., 2020; Franceschi et al., 2017). Despite its wide usage, solving BLO problems can be challenging due to the inherent nature of nested optimization problems. Several algorithms have been proposed to address this challenge, including zeroth-order methods such as Bayesian optimization (Cui & Bai, 2019) and first-order algorithms based on hypergradients (Pearlmutter & Siskind, 2008; Lorraine et al., 2020). Among these approaches, gradient-based BLO has received significant attention because it can scale to high-dimensional problems with a large number of trainable parameters. In this work, we extend the application scenarios of gradient-based BLO to develop a robust and effective parameter-efficient fine-tuning method for pre-trained models.

3 Preliminary

LoRA (Hu et al., 2021) involves attaching the product of two low-rank matrices to the pre-trained weights and fine-tuning these low-rank matrices on downstream datasets with the pre-trained weights frozen. It is based on the assumption that parameter updates made during fine-tuning exhibit a low intrinsic rank. Formally, given a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, LoRA attaches a low-rank update matrix $\Delta W \in \mathbb{R}^{d\times k}$ to the pre-trained weight. This update matrix can be decomposed as $\Delta W = BA$, where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ are two low-rank matrices, with $r \ll \min(d, k)$. Consequently, the weight matrix W' is represented as follows:

$$W' = W_0 + \Delta W = W_0 + BA \tag{1}$$

In this setup, only the LoRA matrix ΔW is updated. (Liu et al., 2024) found that LoRA and full fine-tuning exhibit different learning patterns by performing weight decomposition on fine-tuned weight matrices (See Appendix D). To bridge this discrepancy, weight-decomposed low-rank adaptation (DoRA, Liu et al. (2024)) further reparameterizes the weight matrices by explicitly decomposing them into learnable magnitude and direction components. Formally, DoRA performs adaption as follows:

$$W' = m\frac{V + \Delta V}{\|V + \Delta V\|_c} = m\frac{W_0 + BA}{\|W_0 + BA\|_c} \tag{2}$$

where ΔV is a product of two learnable low-rank matrices, B and A, while the magnitude component $m \in \mathbb{R}^{1\times k}$ is a learnable vector. Here, $\|\cdot\|_c$ represents the vector-wise norm of a matrix computed across each column. In DoRA, both components are optimized concurrently on a single downstream dataset. In this work, we aim to improve DoRA by further decoupling the training of two components.

4 Methods

4.1 Overview of BiDoRA

Our method, BiDoRA, optimizes the trainable parameters in DoRA layers by solving a bi-level optimization problem. Let $\mathcal{M} = \{m_1, m_2, \ldots, m_n\}$ denote the set of magnitude components for all n DoRA modules, and $\mathcal{V} = \{\Delta V_1, \Delta V_2, \ldots, \Delta V_n\}$ denote the set of corresponding direction components. Specifically, we first learn the direction components $\mathcal{V}^*(\mathcal{M})$ on the training split of the downstream dataset $\mathcal{D}_{tr}$ at the lower level. The magnitude component $\mathcal{M}$ is tentatively fixed at this level, thus the resulting optimal direction component $\mathcal{V}^*(\mathcal{M})$ is a function of $\mathcal{M}$. At the upper level, we determine the optimal magnitude component $\mathcal{M}^*$ by optimizing the loss on a validation split $\mathcal{D}_{val}$. In practice, $\mathcal{D}_{tr}$ and $\mathcal{D}_{val}$ are typically created by splitting the original training set without using additional data. This bi-level optimization problem is solved using an efficient gradient-based algorithm, where parameters in two levels are optimized iteratively until convergence. Related convergence analyses of this type of gradient-based bi-level optimization algorithms can be found in Pedregosa (2016), Rajeswaran et al. (2019), and references therein. The generalization analysis has also been studied in Bao et al. (2021).

4.2 Orthogonal Regularization

The orthogonality of neural network weights has been identified as a beneficial property (Bansal et al., 2018) and can effectively mitigate the overfitting issue (Balestriero & richard baraniuk, 2018). Therefore, we define a Gram regularization loss (Xie et al., 2017) for the direction component:

$$\mathcal{R}(\mathcal{V}) = \sum_{k=1}^{n} \left\| (V_k + \Delta V_k)^\top (V_k + \Delta V_k) - I \right\|_F^2 \tag{3}$$

where I is the identity matrix and $\| \cdot \|_F$ denotes the Frobenius norm. Intuitively, $\mathcal{R}(\mathcal{V})$ encourages each column of the direction matrix, representing a specific direction, to be orthogonal to one another. Since each column has already been normalized (equivalent to projected to the unit sphere), this also prompts each column to be far away from the other, thereby reducing the redundancy of parameters.

4.3 A Bi-level Optimization Framework

Lower Level At the lower level, we train the low-rank direction component $\mathcal{V}$ by minimizing a loss $\mathcal{L}_{tr}$ defined on the training set $\mathcal{D}_{tr}$. The overall training objective at this level is $\mathcal{L}_{tr}(\mathcal{V}, \mathcal{M}) = C(\mathcal{V}, \mathcal{M}; \mathcal{D}_{tr}) + \gamma \mathcal{R}(\mathcal{V})$. Here, C represents the fine-tuning loss, given the low-rank direction component $\mathcal{V}$, the magnitude component $\mathcal{M}$, and the training split $\mathcal{D}_{tr}$ of the downstream dataset. $\mathcal{R}(\mathcal{V})$ is the orthogonal regularizer defined in Eq. (3), with γ as a trade-off hyperparameter. In this level, we only update $\mathcal{V}$ while keeping $\mathcal{M}$ fixed, resulting in the following optimization problem:

$$\mathcal{V}^*(\mathcal{M}) = \arg\min_{\mathcal{V}} \ \mathcal{L}_{tr}(\mathcal{V}, \mathcal{M}) \tag{4}$$

where $\mathcal{V}^*(\mathcal{M})$ denotes the optimal solution for $\mathcal{V}$ in this problem, which is a function of $\mathcal{M}$.

Upper Level At the upper level, we validate the previously fixed magnitudes $\mathcal{M}$ on the validation set $\mathcal{D}_{val}$, using the optimal direction component $\mathcal{V}^*(\mathcal{M})$ that was learned at the lower level. This results in a validation loss $\mathcal{L}_{val}(\mathcal{V}^*(\mathcal{M}), \mathcal{M}) = C(\mathcal{V}^*(\mathcal{M}), \mathcal{M}; \mathcal{D}_{val})$. We determine the optimal magnitude component $\mathcal{M}$ by minimizing this validation loss:

$$\min_{\mathcal{M}} \mathcal{L}_{val}(\mathcal{V}^*(\mathcal{M}), \mathcal{M}) \tag{5}$$

A Bi-level Optimization Framework Integrating the two levels of optimization problems, we have the following bi-level optimization framework:

Algorithm 1: BiDoRA

Input: Training dataset $\mathcal{D}_{tr}$ and validation dataset $\mathcal{D}_{val}$

Initialize trainable magnitude components $\mathcal{M} = \{m_k\}_{k=1}^{n}$ and low-rank direction components $\mathcal{V} = \{\Delta V_k\}_{k=1}^{n} = \{\{A_k\}_{k=1}^{n}, \{B_k\}_{k=1}^{n}\}$

// Search Phase

while *not converged* **do**

  Update magnitude $\mathcal{M}$ by descending $\nabla_{\mathcal{M}} \mathcal{L}_{val}(\mathcal{V} - \xi \nabla_{\mathcal{V}} \mathcal{L}_{tr}(\mathcal{V}, \mathcal{M}), \mathcal{M})$

  Update direction $\mathcal{V}$ by descending $\nabla_{\mathcal{V}} \mathcal{L}_{tr}(\mathcal{V}, \mathcal{M})$

Derive the optimal magnitude $\mathcal{M}^* = \{m_k^*\}_{k=1}^{n}$

// Retraining Phase

Train $\mathcal{V}$ until converge using $\mathcal{D}_{tr} \bigcup \mathcal{D}_{val}$ and derive the optimal direction $\mathcal{V}^*$

Output: $\mathcal{V}^*$ and $\mathcal{M}^*$

$$\begin{aligned} &\min_{\mathcal{M}} \ \mathcal{L}_{val}(\mathcal{V}^*(\mathcal{M}), \mathcal{M}) \\ s.t. \quad &\mathcal{V}^*(\mathcal{M}) = \arg\min_{\mathcal{V}} \ \mathcal{L}_{tr}(\mathcal{V}, \mathcal{M}) \end{aligned} \tag{6}$$

Note that these two levels of optimization problems are mutually dependent on each other. The solution of the optimization problem at the lower level, $\mathcal{V}^*(\mathcal{M})$, serves as a parameter for the upper-level problem, while the optimization variable $\mathcal{M}$ at the upper level acts as a parameter for the lower-level problem. By solving these two interconnected problems jointly, we can learn the optimal magnitude component $\mathcal{M}^*$ and incremental direction matrices $\mathcal{V}^*$ in an end-to-end manner.

Two reasons exist behind the choice of setting the magnitude component as the upper level instead of the converse one: 1) In literature, the upper level usually has much fewer parameters than the lower level. In our case, the design of setting the magnitude of complexity $\mathcal{O}(k)$ as the upper level and the direction of complexity $\mathcal{O}(dr + kr)$ as the lower level is consistent with the common practice. 2) BiDoRA resembles the DARTS method (Liu et al., 2018) in neural architecture search where the subnets are selected by a selection variable. Specifically, the magnitude vector resembles a selection variable on the direction matrix by softly selecting each direction (subnets) via scaling.

Optimization Algorithm We use a gradient-based optimization algorithm (Choe et al., 2023) to solve the bi-level optimization problem presented in Eq. (6). A significant challenge in this process is that precisely computing the gradient of the upper-level loss $\mathcal{L}_{val}$ with respect to the magnitude component $\mathcal{M}$ can be computationally prohibitive due to the lack of an analytical solution for $\mathcal{V}^*(\mathcal{M})$ at the lower-level optimization problem. To address this issue, we use the following one-step-unrolled approximation of $\mathcal{V}^*(\mathcal{M})$ inspired by previous work (Liu et al., 2018):

$$\nabla_{\mathcal{M}} \mathcal{L}_{val}(\mathcal{V}^*(\mathcal{M}), \mathcal{M}) \approx \nabla_{\mathcal{M}} \mathcal{L}_{val}(\mathcal{V} - \xi \nabla_{\mathcal{V}} \mathcal{L}_{tr}(\mathcal{V}, \mathcal{M}), \mathcal{M})$$

where ξ is the learning rate at the lower level, and the one-step-unrolled model $\bar{\mathcal{V}} = \mathcal{V} - \xi \nabla_{\mathcal{V}} \mathcal{L}_{tr}(\mathcal{V}, \mathcal{M})$ is used as a surrogate for the optimal solution $\mathcal{V}^*(\mathcal{M})$. We then compute the approximated gradient as follows:

$$\begin{aligned} &\nabla_{\mathcal{M}} \mathcal{L}_{val}(\mathcal{V} - \xi \nabla_{\mathcal{V}} \mathcal{L}_{tr}(\mathcal{V}, \mathcal{M}), \mathcal{M}) \\ =&\nabla_{\mathcal{M}} \mathcal{L}_{val}(\bar{\mathcal{V}}, \mathcal{M}) - \xi \nabla^2_{\mathcal{M},\mathcal{V}} \mathcal{L}_{tr}(\mathcal{V}, \mathcal{M}) \nabla_{\bar{\mathcal{V}}} \mathcal{L}_{val}(\bar{\mathcal{V}}, \mathcal{M}) \qquad (7) \\ \approx&\nabla_{\mathcal{M}} \mathcal{L}_{val}(\bar{\mathcal{V}}, \mathcal{M}) - \xi \frac{\nabla_{\mathcal{M}} \mathcal{L}_{tr}(\mathcal{V}^+, \mathcal{M}) - \nabla_{\mathcal{M}} \mathcal{L}_{tr}(\mathcal{V}^-, \mathcal{M})}{2\epsilon} \qquad (8) \end{aligned}$$

where ϵ is a small scalar and $\mathcal{V}^{\pm} = \mathcal{V} \pm \epsilon \nabla_{\bar{\mathcal{V}}} \mathcal{L}_{val}(\bar{\mathcal{V}}, \mathcal{M})$. Since directly computing the matrix-vector multiplication term in Eq. (7) is computationally expensive, we use finite difference to approximate this

Table 1: RoBERTa$_{\text{base/large}}$ ($R_{b/l}$) and DeBERTa$_{XXL}$ (D_{XXL}) with different fine-tuning methods on the GLUE benchmark. A higher value is better for all datasets. The best results are shown in **bold**.

Method	Param	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
R_b(FT)	125.0M	90.3	94.8	89.3	61.6	86.7	92.8	76.9	91.2	85.5
R_b(Adapter)	0.9 M	86.5	94.0	88.4	58.8	92.5	89.1	71.2	89.9	83.8
R_b(LoRA)	0.15 M	86.8	94.3	88.0	60.3	**93.0**	89.6	72.9	90.1	84.4
R_b(DoRA)	0.17 M	86.8	94.2	89.2	60.5	92.9	89.6	73.2	**90.2**	84.6
R_b(BiDoRA)	0.17 M	**87.1**	**94.4**	**89.4**	**61.3**	92.7	**90.6**	**76.0**	90.1	**85.2**
R_l(FT)	355.0M	90.2	96.4	90.9	68.0	94.7	92.2	86.6	92.4	88.9
R_l(Adapter)	0.8M	90.3	96.3	87.7	66.3	**94.7**	91.5	72.9	91.5	86.4
R_l(LoRA)	0.39 M	**90.6**	96.3	90.0	66.9	94.5	91.2	86.3	91.7	88.4
R_l(DoRA)	0.39 M	**90.6**	**96.4**	89.8	65.8	**94.7**	91.2	86.6	**92.0**	88.4
R_l(BiDoRA)	0.39 M	**90.6**	96.1	**90.1**	**67.0**	94.6	**91.7**	**86.9**	**92.0**	**88.6**
D_{XXL}(DoRA)	4.9M	91.2	**96.3**	92.3	71.1	**95.3**	91.6	91.8	**90.8**	90.0
D_{XXL}(BiDoRA)	4.9M	**91.7**	**96.3**	**92.6**	**72.3**	95.2	**92.0**	**92.3**	**90.8**	**90.4**

product as in Eq. (8), following Liu et al. (2018). As detailed in Algorithm 1, the direction component $\mathcal{V}$ and the magnitude component $\mathcal{M}$ are updated using gradient descent iteratively until convergence. After acquiring the optimal magnitudes $\mathcal{M}^*$ through the process above, the direction component $\mathcal{V}$ is retrained on the union of training and validation splits to achieve the best performance on downstream tasks, resulting in the final learned $\mathcal{V}^*$.

5 Experiments

5.1 Experimental Setup

We compare BiDoRA with several PEFT methods, including Adapter tuning (Houlsby et al., 2019), LoRA (Hu et al., 2021), and DoRA (Liu et al., 2024). BiDoRA does not use any additional data compared to other baselines, as we create the validation set for upper-level optimization by splitting the original training set with an 8:2 ratio for all tasks. All methods in the experiment, including ablation studies, are trained until convergence for a fair comparison. Detailed descriptions of these baseline methods are provided in Appendix C.

Table 2: RoBERTa$_{\text{base/large}}$ ($R_{b/l}$) with different fine-tuning methods on the Reuters21578 benchmark. A higher value is better for all datasets. The best results are shown in **bold**.

Method	Param	ModApte	ModHayes	ModLewis
R_b(FT)	125.0M	85.4	77.6	77.1
R_b(Adapter)	0.9 M	**85.3**	77.5	76.8
R_b(LoRA)	0.15 M	84.7	74.3	74.7
R_b(DoRA)	0.17 M	84.8	78.2	76.6
R_b(BiDoRA)	0.17 M	**85.3**	**79.9**	**77.6**
R_l(FT)	355.0M	84.8	77.5	76.6
R_l(Adapter)	0.44 M	84.8	77.9	76.7
R_l(LoRA)	0.39 M	84.7	77.7	76.7
R_l(DoRA)	0.39 M	84.8	77.4	76.7
R_l(BiDoRA)	0.39 M	**84.9**	**78.9**	**77.3**

Our experiments cover a wide range of tasks, including natural language understanding (NLU), natural language generation (NLG), and token classification. For NLU tasks, we fine-tune the RoBERTa-base (125M), RoBERTa-large (355M), and DeBERTa XXL (1.5B) models on the GLUE benchmark (Wang et al., 2019) and the Reuters21578 dataset (Padmanabhan et al., 2016) using all baseline PEFT methods and BiDoRA. Detailed descriptions of these datasets and pre-trained models are provided in Appendix A. Following existing practices, the development set is used in GLUE as the test data since the actual test set is not publicly available. We report the overall (matched and mismatched) accuracy for MNLI, Matthew's correlation for CoLA, Pearson correlation for STS-B, and accuracy for the other tasks. On the Reuters21578 dataset, the F1 score is used as the evaluation metric across all three splits. For NLG tasks, we fine-tune GPT-2 medium on the E2E (Novikova et al., 2017) dataset. We use BLEU (Papineni et al., 2002), NIST (Lin & Och, 2004), METEOR (Banerjee & Lavie, 2005), ROUGE-L (Lin, 2004), and CIDEr (Vedantam et al., 2015) as evaluation metrics. For token classification, we fine-tune the RoBERTa-base and RoBERTa-large models on the BioNLP (Collier et al., 2004) dataset and the CoNLL2003 (Sang & De Meulder, 2003) dataset. Accuracy, precision, recall, and F1 score are used as evaluation metrics.

For all experiments, our implementation is based on the Huggingface Transformers library (Wolf et al., 2019) and the Betty library (Choe et al., 2023). We use a single NVIDIA A100 GPU for all experiments. More detailed experimental settings are provided in Appendix B.

Table 3: $\text{RoBERTa}_{\text{base/large}}$ ($\text{R}_{\text{b/l}}$) with different fine-tuning methods on BioNLP data and CoNLL2003 dataset. A higher value is better for all metrics. The best results are shown in **bold**.

		BioNLP				CoNLL2003			
Method	Param	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
R_b(FT)	125.0M	93.9	69.0	78.9	73.6	99.3	95.7	96.3	96.0
R_b(Adapter)	0.9 M	93.9	69.1	78.8	73.7	**99.3**	95.7	96.4	96.0
R_b(LoRA)	0.15 M	93.9	69.0	78.8	73.6	**99.3**	95.4	96.3	95.8
R_b(DoRA)	0.17 M	**94.0**	69.2	**79.1**	73.8	**99.3**	95.3	96.2	95.8
R_b(BiDoRA)	0.17 M	93.9	**71.2**	78.6	**74.7**	**99.3**	**95.9**	**96.5**	**96.2**
R_l(FT)	355.0M	94.0	69.4	79.6	74.1	99.4	96.2	97.0	96.6
R_l(Adapter)	0.44 M	**94.0**	69.4	79.7	74.2	**99.4**	96.1	97.0	96.6
R_l(LoRA)	0.39 M	93.9	69.2	79.3	73.9	**99.4**	96.2	97.0	96.6
R_l(DoRA)	0.39 M	**94.0**	69.4	**79.7**	74.2	**99.4**	96.2	**97.1**	96.6
R_l(BiDoRA)	0.39 M	**94.0**	**71.3**	79.3	**75.1**	**99.4**	**96.4**	**97.1**	**96.7**

5.2 Experiments on Natural Language Understanding Tasks

In this section, we evaluate the performance of BiDoRA on NLU tasks, with a particular focus on text classification. Table 1 presents the results of fine-tuning the RoBERTa-base, RoBERTa-large, and DeBERTa XXL models on the GLUE benchmark with baseline PEFT methods and BiDoRA. The results show that BiDoRA achieves superior or comparable performance compared to baseline methods across all datasets with the same number of trainable parameters. Table 2 presents the results of fine-tuning RoBERTa models on the Reuters21578 datasets, where BiDoRA outperforms all baseline methods by an even larger margin. Notably, BiDoRA achieves performance comparable to or even better than full fine-tuning. The superior performance of BiDoRA on both benchmarks verifies the effectiveness of its bi-level optimization mechanism. By training the magnitude and direction components on two distinct sub-datasets, BiDoRA enhances the flexibility of the learning process and improves learning capacity compared to DoRA, resulting in a performance boost.

5.3 Experiments on Natural Language Generation Tasks

In this section, we evaluate BiDoRA's performance on the NLG task. Table 5 presents the results of fine-tuning a GPT-2 model on the E2E dataset with baseline PEFT methods and BiDoRA. The results show

Table 4: Ablation studies. We evaluate the performance of BiDoRA without retraining (w/o retraining), without bi-level optimization ($\xi = 0$) and without orthogonal regularization (w/o cst.).

Method	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg
BiDoRA (w/o retraining)	87.0	94.2	89.0	57.3	92.4	90.6	71.6	90.0	84.0
BiDoRA ($\xi = 0$)	86.9	94.2	89.0	59.4	90.8	**91.2**	75.9	90.0	84.7
BiDoRA (w/o cst.)	87.0	**94.4**	88.6	**61.3**	**92.7**	90.2	76.0	**90.1**	85.0
BiDoRA	**87.1**	**94.4**	**89.4**	**61.3**	**92.7**	90.6	**76.1**	**90.1**	**85.2**

Table 5: Performance of BiDoRA and baseline methods for fine-tuning GPT2-medium on the E2E dataset. A higher value is better for all metrics. The best results are shown in **bold**.

Method	Param	BLEU	NIST	MET	ROUGE-L	CIDEr
FT	354.9M	68.0	8.61	46.1	69.0	2.38
Adapter	11.1M	67.0	8.50	45.2	66.9	2.31
LoRA	0.39M	67.1	8.54	45.7	68.0	2.33
DoRA	0.39M	67.0	8.48	45.4	70.1	2.33
BiDoRA	0.39M	**69.0**	**8.72**	**46.2**	**70.9**	**2.44**

that BiDoRA achieves the best performance across all five evaluation metrics, demonstrating the superiority of BiDoRA in fine-tuning pre-trained models for NLG tasks.

5.4 Experiments on Token Classification

Further evidence of the effectiveness of BiDoRA can be observed in Table 3, which reports the results of token classification tasks. Unlike the NLU tasks discussed in the previous section, which involve classifying entire sentences and focusing on capturing global semantics, token classification requires classifying each token within a sentence, highlighting the importance of capturing local context. On the BioNLP dataset, BiDoRA consistently outperforms baseline methods by a large margin in terms of F1 score. On the CoNLL2003 dataset, BiDoRA either outperforms or matches all baseline methods across all metrics. Consistent with our previous findings, BiDoRA effectively fine-tunes pre-trained models for token classification tasks.

5.5 Experiments on Extremely Small Datasets

Table 6: Fine-tuning ESM on the thermostability prediction task. A higher value is better for all metrics, with the best results highlighted in **bold**.

Methods	#Params	Accuracy	Precision	Recall	F1
FT	652.7M	79.8	81.2	79.8	78.4
LoRA	1.5M	75.9	78.2	75.9	75.5
DoRA	1.6M	76.9	78.7	76.9	76.2
BiDoRA	1.6M	**78.8**	**79.1**	**78.8**	**78.2**

The ESM (Evolutionary Scale Modeling, Rives et al. (2021)) model is a transformer-based protein language model designed for protein sequence analysis, leveraging the transformer architecture to capture evolutionary patterns. We fine-tune the ESM model using the Protein Aligner checkpoint (Zhang et al., 2024a) on two classification tasks—thermostability prediction (Chen et al., 3,695 training samples) and blood-brain barrier peptide prediction (BBP, Dai et al. (2021), 936 training samples). Notably, protein analysis datasets are typically much smaller than those in NLP, in which case the large pre-trained models are prone to overfitting, even when using PEFT methods. The trainable parameters (on the order of millions) are significantly overparameterized compared to the available samples (thousands or even hundreds), highlighting the need for our overfitting-resilient counterpart.

Table 7: Fine-tuning ESM on the BBP task. A higher value is better for all metrics, with the best results highlighted in **bold**.

Methods	#Params	Accuracy	Precision	Recall	F1
FT	652.9M	89.4	89.9	89.4	89.4
LoRA	1.9M	86.8	87.7	86.8	86.7
DoRA	2.0M	89.4	91.3	89.4	89.3
BiDoRA	2.0M	**92.1**	**93.1**	**92.1**	**92.0**

The results are presented in Tables 6 and 7, respectively. For the classification tasks, we use accuracy, precision, recall, and F1 score to evaluate performance. Consistent with our previous findings, BiDoRA effectively fine-tunes pre-trained models on extremely small datasets. Our method outperforms the baselines by a larger margin as the dataset size decreases, confirming our previous conclusion that our method effectively combats the overfitting issue on various network architectures and diverse tasks.

5.6 Ablation Studies

In this section, we perform ablation studies to investigate the effectiveness of individual modules or strategies in BiDoRA. We fine-tune a RoBERTa-base model on the GLUE benchmark under different ablation settings, and the results are shown in Table 4.

Retraining We test the model directly obtained from the search phase to evaluate the effectiveness of further retraining the direction component. The results show that BiDoRA outperforms BiDoRA (w/o retraining) on average, highlighting the necessity of retraining.

Bi-level Optimization We set ξ to zero in Algorithm 1 to assess the effectiveness of the bi-level optimization framework. This ablation setting can be interpreted as an alternative learning method where two optimization steps are carried out alternately on two different splits of the training dataset. Notably, in the alternative learning method, the updating of each component is unaware of each other, making the training less stable. In contrast, the hyper-gradient used in bi-level optimization avoids this issue by connecting the two levels in a certain way. The results show that BiDoRA outperforms BiDoRA ($\xi = 0$) on average, demonstrating the efficacy of the bi-level optimization strategy.

Orthogonal Regularization We examine the effectiveness of the orthogonality constraint in Eq. (3) by setting γ to zero. Results show that BiDoRA outperforms BiDoRA (w/o cst.) on average, indicating the effectiveness of applying the orthogonality regularizer to alleviate overfitting.

5.7 Weight Decomposition Analysis

One important motivation of DoRA is to bridge the inherent differences between LoRA and FT. Similar to DoRA, we conduct a weight decomposition analysis on the correlation between the change of magnitudes and that of directions (detailed in Appendix D) for BiDoRA and baseline methods by fine-tuning a GPT2-medium model on the E2E dataset. As shown in Fig. 3, FT, DoRA, and BiDoRA all exhibit negative correlation values, while LoRA shows a positive correlation, consistent with the findings in Liu et al. (2024). Notably, BiDoRA achieves a negative correlation of -8.042, closer to FT than DoRA's -1.784. This improvement is attributed to the decoupled training process of the two layers, which allows for a higher learning capacity compared to DoRA.

5.8 Discussion

The advantage of BiDoRA is supported by both theoretical insights and empirical evidence, as illustrated as follows.

Motivation. Theoretically, Liu et al. (2024) showed that LoRA's training pattern tends to be coupled in terms of magnitude-direction correlation, which degrades learning capacity. Their solution was to introduce a reparameterization that decouples these components in the formulation. We build upon DoRA following

their theory and further decouple magnitude and direction in terms of training dynamics. Specifically, the two components are trained in separate loops within a bilevel optimization framework, which is expected to improve performance in an intuition similar to DoRA.

Empirical evidences. We performed a Wilcoxon signed-rank test to compare the performance of DoRA and BiDoRA. Specifically, we used the results from Table 1. For each PEFT method, we collected 9 values (8 values from each dataset plus the average performance) from one base model and concatenated the results from three base models (RoBERTa-base, RoBERTa-large, and DeBERTa-XXL) to obtain a list of 27 values. A comparison of these 27 values between DoRA and BiDoRA reveals that BiDoRA is significantly better than DoRA, with a p-value of 2.4×10^{-4}. This result demonstrates that BiDoRA offers a non-marginal improvement over DoRA.

Additionally, the weight decomposition analysis, including (Fig. 3 and Fig. 4), indicates that BiDoRA achieves better decoupling of the components compared to DoRA. Evaluation metrics across various tasks demonstrate the superior performance of BiDoRA, confirming that our decoupled optimization loop leads to improved outcomes.

6 Conclusion and Future Works

We propose BiDoRA, a novel bi-level optimization framework for parameter-efficient fine-tuning of large-scale pre-trained models. By conducting weight decomposition following the DoRA approach, our method trains the two components separately in two interconnected optimization levels using different sub-datasets. In this way, BiDoRA not only decouples the learning process of the two components, resulting in a learning pattern closer to FT, but also effectively alleviates overfitting. Empirical studies on various NLP tasks demonstrate that BiDoRA outperforms DoRA and other baselines, highlighting the effectiveness of our method.

References

Randall Balestriero and richard baraniuk. A spline theory of deep learning. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 374–383. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/balestriero18b.html`.

Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pp. 65–72, 2005.

Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. Can we gain more from orthogonality regularizations in training deep cnns? In *Neural Information Processing Systems*, 2018. URL `https://api.semanticscholar.org/CorpusID:55704502`.

Fan Bao, Guoqiang Wu, Chongxuan Li, Jun Zhu, and Bo Zhang. Stability and generalization of bilevel programming in hyperparameter optimization. *Advances in neural information processing systems*, 34: 4529–4541, 2021.

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

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

Tianlong Chen, Chengyue Gong, Daniel Jesus Diaz, Xuxi Chen, Jordan Tyler Wells, Zhangyang Wang, Andrew Ellington, Alex Dimakis, Adam Klivans, et al. Hotprotein: A novel framework for protein thermostability prediction and editing. In *The Eleventh International Conference on Learning Representations*.

Sang Keun Choe, Willie Neiswanger, Pengtao Xie, and Eric Xing. Betty: An automatic differentiation library for multilevel optimization. In *The Eleventh International Conference on Learning Representations*, 2023.

Nigel Collier, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Jin-Dong Kim. Introduction to the bio-entity recognition task at jnlpba. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*, pp. 73–78, 2004.

Hua Cui and Jie Bai. A new hyperparameters optimization method for convolutional neural networks. *Pattern Recognition Letters*, 125:828–834, 2019.

Ruyu Dai, Wei Zhang, Wending Tang, Evelien Wynendaele, Qizhi Zhu, Yannan Bin, Bart De Spiegeleer, and Junfeng Xia. Bbppred: sequence-based prediction of blood-brain barrier peptides with feature representation learning and logistic regression. *Journal of Chemical Information and Modeling*, 61(1):525–534, 2021.

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

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and reverse gradient-based hyperparameter optimization. In *International Conference on Machine Learning*, pp. 1165–1173. PMLR, 2017.

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

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

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2021.

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

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Chin-Yew Lin and Franz Josef Och. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd annual meeting of the association for computational linguistics (ACL-04)*, pp. 605–612, 2004.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2018.

Shih-yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*, 2024.

Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International conference on artificial intelligence and statistics*, pp. 1540–1552. PMLR, 2020.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. The e2e dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pp. 201–206, 2017.

Divya Padmanabhan, Satyanath Bhat, Shirish Shevade, and Y Narahari. Topic model based multi-label classification. In *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 996–1003. IEEE, 2016.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.

Barak A Pearlmutter and Jeffrey Mark Siskind. Reverse-mode ad in a functional framework: Lambda the ultimate backpropagator. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 30(2): 1–36, 2008.

Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In *International conference on machine learning*, pp. 737–746. PMLR, 2016.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. *Advances in neural information processing systems*, 32, 2019.

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

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, Jimmy Ba, and Amjad Almahairi. Residual prompt tuning: Improving prompt tuning with residual reparameterization. *arXiv preprint arXiv:2305.03937*, 2023.

Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15): e2016239118, 2021.

Erik Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pp. 142–147, 2003.

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

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4566–4575, 2015.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Di Xie, Jiang Xiong, and Shiliang Pu. All you need is beyond a good init: Exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5075–5084, 2017. URL https://api.semanticscholar.org/CorpusID:8199182.

Li Zhang, Han Guo, Leah V Schaffer, Young Su Ko, Digvijay Singh, Hamid Rahmani, Danielle Grotjahn, Elizabeth Villa, Michael Gilson, Wei Wang, et al. Proteinaligner: A multi-modal pretraining framework for protein foundation models. *bioRxiv*, pp. 2024–10, 2024a.

Miao Zhang, Steven W Su, Shirui Pan, Xiaojun Chang, Ehsan M Abbasnejad, and Reza Haffari. idarts: Differentiable architecture search with stochastic implicit gradients. In *International Conference on Machine Learning*, pp. 12557–12566. PMLR, 2021.

Q Zhang, M Chen, A Bukharin, P He, Y Cheng, W Chen, and T Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. preprint (2023). *arXiv preprint arXiv:2303.10512*, 2023.

Ruiyi Zhang, Rushi Qiang, Sai Ashish Somayajula, and Pengtao Xie. AutoLoRA: Automatically tuning matrix ranks in low-rank adaptation based on meta learning. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5048–5060, Mexico City, Mexico, June 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.282. URL https://aclanthology.org/2024.naacl-long.282.

A Datasets and Models

A.1 Natural Language Understanding

The GLUE Benchmark comprises a diverse array of tasks that are widely employed for evaluation in natural language understanding. It encompasses two single-sentence classification tasks, three tasks assessing similarity and paraphrasing, and four tasks focusing on natural language inference. Specifically, it includes MNLI (MultiNLI, Williams et al. (2017)), SST-2 (Stanford Sentiment Treebank, Socher et al. (2013)), MRPC (Microsoft Research Paraphrase Corpus, Dolan & Brockett (2005)), CoLA (Corpus of Linguistic Acceptability, Warstadt et al. (2019)), QNLI (Question NLI, Rajpurkar et al. (2018)), QQP (Quora Question Pairs), RTE (Recognizing Textual Entailment), and STS-B (Semantic Textual Similarity Benchmark, Cer et al. (2017)). We summarize the statistical data for all datasets within the GLUE Benchmark in Table 8.

The Reuters-21578 (Padmanabhan et al., 2016) dataset is one of the most widely used data collections for text categorization research. It was collected from the Reuters financial newswire service in 1987 and is used for text classification and natural language processing tasks. Three splits are available: ModApte, ModHayes, and ModLewis. These documents cover various topics, such as politics, economics, and sports. We summarize the statistical data for all text classification tasks used in our experiments in Table 9.

Table 8: The statistical data for all datasets within the GLUE Benchmark.

Dataset	Metrics	Train	Dev	Test	Label	Task
MNLI	Accuracy	393k	20k	20k	3	NLI
SST-2	Accuracy	67k	872	1.8k	2	Sentiment
MRPC	Accuracy	3.7k	408	1.7k	2	Paraphrase
CoLA	Matthews Corr	8.5k	1k	1k	2	Acceptability
QNLI	Accuracy	108k	5.7k	5.7k	2	QA/NLI
QQP	Accuracy	364k	40k	391k	2	Paraphrase
RTE	Accuracy	2.5k	276	3k	2	NLI
STS-B	Pearson Corr	7.0k	1.5k	1.4k	1	Similarity

Table 9: The statistical data for the Reuters-21578 dataset.

Dataset	Metrics	Train	Test
ModApte	F1	8.8k	3k
ModHayes	F1	18k	0.7k
ModLewis	F1	12k	5.5k

A.2 Natural Language Generation

In our experiments on natural language generation, we use the E2E (Novikova et al., 2017) dataset, which was initially introduced as a dataset for training end-to-end, data-driven natural language generation systems. Multiple references can be associated with each source table used as input. Each sample input (x, y) consists of a series of slot-value pairs accompanied by an associated natural language reference text. The E2E dataset comprises approximately 42,000 training examples, 4,600 validation examples, and 4,600 test examples from the restaurant domain.

We utilize the following five evaluation metrics: BLEU (Papineni et al., 2002), NIST (Lin & Och, 2004), METEOR (Banerjee & Lavie, 2005), ROUGE-L (Lin, 2004), and CIDEr (Vedantam et al., 2015). We summarize its statistical data in Table 10.

Table 10: The statistical data for E2E dataset.

Dataset	Metrics	Train	Validation
E2E	BLEU,NIST,MET,ROUGE-L,CIDEr	42k	4.6k

A.3 Token Classification

BioNLP (Collier et al., 2004) is a Named Entity Recognition dataset that contains biological entities such as DNA, RNA, and protein. It is essentially a token classification task where we want to classify each entity in the sequence. CoNLL-2003 (Sang & De Meulder, 2003) focuses on language-independent named entity recognition. It concentrates on four types of named entities: persons, locations, organizations, and miscellaneous entities that do not belong to the previous three groups. We summarize the statistical data for all used token classification tasks in Table 11.

Table 11: The statistical data for token classification tasks

Dataset	Metrics	Train	Validation	Test
BioNLP	Accuracy,Precision,Recall,F1	17k	1.9k	3.9k
CoNLL2003	Accuracy,Precision,Recall,F1	14k	3.3k	3.5k

B Experimental Settings

In this section, we provide detailed experimental settings. We maintain consistent configurations across experiments, including LoRA rank, LoRA α, batch size, maximum sequence length, and optimizer, to ensure a fair comparison. The hyperparameter tuning for our method is straightforward and convenient.

B.1 RoBERTa

We summarize the experimental settings for the GLUE benchmark in Table 12 and for the Reuters21578 dataset and token classification tasks in Table 13.

Table 12: The hyperparameters we used for RoBERTa on the GLUE benchmark.

Method	Settings	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B
	Optimizer	AdamW							
	Warmup Ratio	0.06							
	Scheduler	Linear							
	LoRA rank	$\text{rank}_\text{a} = \text{rank}_\text{u} = 4$							
	LoRA α	8							
RoBERTa-base	Total batch size	32							
	Global steps	20000	12000	25000	20000	15000	20000	15000	12000
	Lower learning rate	5e-5	1e-5	2e-5	5e-5	2e-5	5e-5	1e-5	1e-5
	Upper learning rate	5e-5	1e-5	2e-5	5e-5	2e-5	5e-5	1e-5	1e-5
	Lower weight decay	0.1							
	Upper weight decay	0.1	0.1	0.1	0.1	0	0.1	0.1	0.01
	Max Seq Length	512							
	Regularization Coefficient	1e-5							
RoBERTa-large	Total batch size	32							
	Global steps	50000	20000	30000	20000	60000	40000	15000	10000
	Lower learning rate	1e-5							
	Upper learning rate	1e-5							
	Lower weight decay	0.5	0.5	0	0.2	0.5	0.5	0.5	0.5
	Upper weight decay	0.5	0.05	0	0.2	0.5	0.5	0.1	0.5
	Max Seq Length	128							
	Regularization Coefficient	0	0	1e-5	1e-5	0	1e-5	0	1e-5

B.2 GPT-2

We summarize the experimental settings for the GPT-2 experiments in Table 14. The experimental configuration, particularly during the inference stage, follows the approach described by Hu et al. (2021).

C Baselines in Experiments

We compare BiDoRA with Full Fine-Tuning (FT), Adapter tuning (Houlsby et al., 2019), LoRA (Hu et al., 2021), and DoRA (Liu et al., 2024) in all our experiments. We provide a brief introduction to these methods here.

Full Fine-Tuning (FT) is a commonly used method for adaptation. The model is initialized with pre-trained weights and biases, and all model parameters are updated through gradient descent.

Adapter tuning (Houlsby et al., 2019) inserts layer adapters between neural modules, such as the MLP module or the self-attention module. It incorporates two fully connected layers within an adapter layer, with a nonlinearity function applied between them.

LoRA (Hu et al., 2021) adds trainable incremental update matrices to pre-trained weight matrices. Following the experimental settings of LoRA, we applied BiDoRA to W_q and W_v matrices (the query and value weight matrices in the self-attention module) for a fair comparison.

DoRA (Liu et al., 2024) proposes weight-decomposed adaptation, which formulates the incremental matrices as a product of magnitude and direction components, thereby accelerating training and aligning the training

Table 13: The hyperparameters we used for RoBERTa on the Reuters21578 dataset, BioNLP dataset, and CoNLL2003 dataset.

Method	Settings	ModApte	ModHayes	ModLewis	BioNLP	CoNLL2003
	Optimizer	AdamW				
	Warmup Ratio	0.06				
	Scheduler	Linear				
	LoRA rank	$\text{rank}_\text{a} = \text{rank}_\text{u} = 4$				
	LoRA α	8				
RoBERTa-base	Total batch size	32				
	Global steps	20000	20000	20000	12000	12000
	Lower learning rate	3e-5	3e-5	3e-5	1e-5	2e-5
	Upper learning rate	3e-5	3e-5	3e-5	1e-5	2e-5
	Lower weight decay	0.1	0.1	0.1	0.1	0.2
	Upper weight decay	0.1				
	Max Seq Length	512				
	Regularization Coefficient	0	1e-5	0	1e-5	0
RoBERTa-large	Total batch size	32				
	Global steps	20000	20000	20000	12000	15000
	Lower learning rate	1e-5	1e-5	1e-5	2e-5	1e-5
	Upper learning rate	1e-5	1e-5	1e-5	2e-5	1e-5
	Lower weight decay	0.2	0.1	0.2	0.02	0.1
	Upper weight decay	0.1	0.1	0.1	0.02	0.1
	Max Seq Length	128				
	Regularization Coefficient	0	1e-5	0	0	1e-5

Table 14: The hyperparameters we used for GPT-2 on the E2E NLG benchmark.

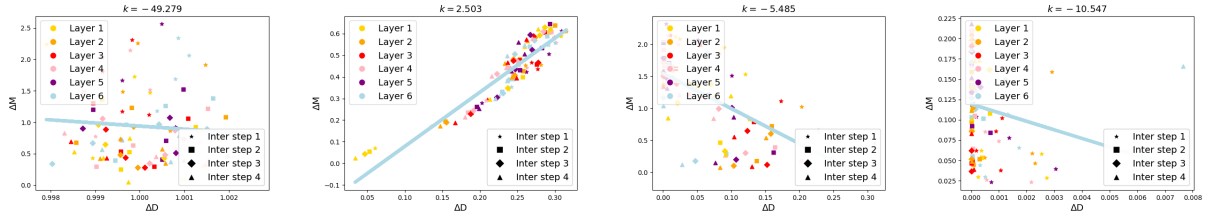
Settings	Training
Optimizer	AdamW
Warmup Ratio	0.06
Scheduler	Linear
LoRA rank	$\text{rank}_\text{a} = \text{rank}_\text{u} = 4$
LoRA α	32
Label Smooth	0.1
Lower learning rate	1e-3
Upper learning rate	1e-4
Lower weight decay	1
Upper weight decay	1
Max Seq Length	512
Regularization Coefficient	1e-5
Settings	**Inference**
Beam Size	10
Length Penalty	0.9
no repeat ngram size	4

behavior with full fine-tuning. In contrast, our BiDoRA trains the two components on distinct sub-datasets to alleviate overfitting.

D Weight Decomposition Analysis

We provide a brief review of the weight decomposition analysis proposed in Liu et al. (2024). Define the weight decomposition of a weight matrix $W \in \mathbb{R}^{d\times k}$ (e.g., query matrix in an attention layer) as $W = m\frac{V}{\|V\|_c} = \|W\|_c\frac{W}{\|W\|_c}$, where $m \in \mathbb{R}^{1\times k}$ is the magnitude vector, and $V \in \mathbb{R}^{d\times k}$ is the directional matrix, with $\|\cdot\|_c$ representing the vector-wise norm of a matrix across each column. This decomposition ensures that each column of $V/\|V\|_c$ remains a unit vector, and the corresponding scalar in m defines the magnitude of each vector. Liu et al. (2024) examine the magnitude and directional variations between W_0 and W_{FT}, defined as $\Delta M^t_{\text{FT}} = \frac{\sum_{n=1}^{k}|m^{n,t}_{\text{FT}} - m^n_0|}{k}$ and $\Delta D^t_{\text{FT}} = \frac{\sum_{n=1}^{k}(1-\cos(V^{n,t}_{\text{FT}}, W^n_0))}{k}$. Here, ΔM^t_{FT} and ΔD^t_{FT} represent the magnitude and direction differences between W_0 and W_{FT} at the t-th training step, respectively, with $\cos(\cdot,\cdot)$ denoting cosine similarity. $m^{n,t}_{\text{FT}}$ and m^n_0 are the n^{th} scalars in their respective magnitude vectors, while $V^{n,t}_{\text{FT}}$ and W^n_0 are the n^{th} columns in V^t_{FT} and W_0. Intuitively, a consistent positive slope trend across all the intermediate steps implies a difficulty in concurrent learning of both magnitude and direction, suggesting that slight directional changes are challenging to execute alongside more significant magnitude alterations. In contrast, a relatively negative slope signifies a more varied learning pattern, with a more pronounced negative correlation indicating a larger learning capacity.

Complementary to Fig. 3 in the main paper on the query matrix, we provide additional results of weight decomposition analysis in Fig. 4 on the value matrix to complement the findings in Section 5.7. We can draw two key observations from Fig. 4: 1) Consistent with the results in Liu et al. (2024), both FT and DoRA exhibit negative correlation values of -49.279 and -5.485, respectively, while LoRA shows a positive correlation with a value of 2.503. 2) BiDoRA achieves a negative correlation value of -10.547, indicating closer alignment with FT compared to DoRA. The analysis of how BiDoRA achieves this improvement is similar to that discussed in Section 5.7.



(a) FT($k = -49.279$) (b) LoRA($k = 2.503$) (c) DoRA($k = -5.485$) (d) BiDoRA($k = -10.5$)

Figure 4: **Magnitude and direction updates** for (a) FT, (b) LoRA, (c) DoRA, and (d) BiDoRA of the value matrices across different layers and intermediate steps after fine-tuning the GPT2 model on the E2E dataset. Different markers represent matrices from different training steps, while different colors indicate matrices from each layer. The values of negative correlation are shown at the top, denoted by k.

E Training Cost

E.1 Computation Costs

Table 15: Average training time cost on the MNLI, QQP, and SST-2 datasets.

Method	LoRA	DoRA	BiDoRA
Cost	×1	×1.30	×3.92

Since BiDoRA has the same architecture as DoRA, our method only requires two extra forward and backward passes of the lower level for the hypergradient calculation of the upper level, as shown in Eq. 8. In principle, this would make our method roughly threefold computationally costly. A similar analysis holds

Partition	ModApte	ModHayes	ModLewis
0.6	85.32	79.76	77.69
0.7	85.32	**80.01**	**77.74**
0.8	**85.34**	79.93	77.63
0.9	85.27	79.85	77.64
1.0	85.23	79.59	77.42

Table 16: Experiment results on different data partitions of BiDoRA

for the memory consumption analysis. Empirically, Table 15 shows the average training cost of BiDoRA and two baseline methods on the MNLI, QQP, and SST-2 datasets from the GLUE benchmark, being consistent with the theoretical analysis. We normalize the cost of LoRA to 1 for reference. Given BiDoRA's superior performance across various tasks, the increase in computational costs is acceptable in most cases, underscoring its practicality.

F The Role of Hyperparameter

The hyperparameter tuning for BiDoRA is simple, convenient, and straightforward. We further conducted experiments regarding the dataset partition of $\mathcal{D}_{tr}$ and $\mathcal{D}_{val}$ to provide insights into its role in BiDoRA. The dataset partition helps maintain the balance of inner/outer optimization by assigning different portions of data. The direction component has more trainable parameters, so it is reasonable to use more data for training the lower level while using the remaining data for training magnitudes. As shown in Table 16, We varied the inner-level dataset $\mathcal{D}_{tr}$ partition from 0.6 to 1.0 with 0.1 intervals and experimented with RoBERTa-base on three splits of the Reuters21578 dataset to examine its influence.

The results indicate that both extreme cases are negative to the overall performance. When the inner partition is too small (≤ 0.6), directions are not well-trained, and when the inner partition is 1.0, magnitudes are not trained at all, leading to a significant performance drop. These findings demonstrate that bi-level optimization is effective in the sense that both levels are necessary for enhancing performance. Although tuning the partition ratio may further improve overall performance, we maintain a consistent data partition of 8:2 in all the experiments for simplicity. A fixed configuration of data partition already consistently yields superior performance of BiDoRA, demonstrating that our method is robust to this hyperparameter within a certain range.

G Comparison with Other General Methods for Addressing Overfitting

There are some common experimental settings that may be used to reduce overfitting. For DoRA, two promising methods are increasing weight decay and adopting a more aggressive dropout rate. We conducted experiments on these two methods separately. We kept hyperparameters that have been well-tuned in DoRA and can achieve optimal results while only tuning the weight decay value. Similarly, we tune the dropout rate of DoRA while keeping the weight decay value to be optimized. We conducted experiments on Roberta-base on three datasets. The results are presented in Table 17 and 18.

We can draw the observation that neither of these approaches effectively addresses overfitting issues or enhances the model's generalization ability. On the other hand, BiDoRA exploits the specific magnitude-direction structure of DoRA and the strategy of training the two distinct components on separate splits of the dataset. An advantage of our methodology is that it can be easily combined with other general-purpose overfitting-alleviating strategies since BiDoRA does not modify the original DoRA architecture.

Table 17: Experiment results on different weight decay values of DoRA

Method	CoLA	MRPC	RTE
DoRA (weight decay = 0)	59.3	88.7	72.9
DoRA (weight decay = 0.05)	60.1	89.2	73.3
DoRA (weight decay = 0.1)	60.5	89.2	73.2
DoRA (weight decay = 0.2)	60.3	89.0	73.2
BiDoRA	**61.3**	**89.4**	**76.0**

Table 18: Experiment results on different dropout rates of DoRA

Method	CoLA	MRPC	RTE
DoRA (dropout rate = 0)	59.2	89.2	72.9
DoRA (dropout rate = 0.1)	60.2	88.9	71.4
DoRA (dropout rate = 0.2)	55.1	87.8	64.2
BiDoRA	**61.3**	**89.4**	**76.0**