DoRA: Enhancing Parameter-Efficient Fine-Tuning with Dynamic Rank Distribution

Yulong Mao*[1,2], Kaiyu Huang*[1,2], Changhao Guan[1,2],
Ganglin Bao[1,2], Fengran Mo[3], and Jinan Xu†[1,2]

[1]Beijing Key Lab of Traffic Data Analysis and Mining, Beijing, China
[2]Beijing Jiaotong University, Beijing, China
[3]Université de Montréal, Montréal, Canada
{maoyulong, kyhuang, jaxu}@bjtu.edu.cn

Abstract

Fine-tuning large-scale pre-trained models is inherently a resource-intensive task. While it can enhance the capabilities of the model, it also incurs substantial computational costs, posing challenges to the practical application of downstream tasks. Existing parameter-efficient fine-tuning (PEFT) methods such as Low-Rank Adaptation (LoRA) rely on a bypass framework that ignores the differential parameter budget requirements across weight matrices, which may lead to suboptimal fine-tuning outcomes. To address this issue, we introduce the Dynamic Low-Rank Adaptation (DoRA) method. DoRA decomposes high-rank LoRA layers into structured single-rank components, allowing for dynamic pruning of parameter budget based on their importance to specific tasks during training, which makes the most of the limited parameter budget. Experimental results demonstrate that DoRA can achieve competitive performance compared with LoRA and full model fine-tuning, and outperform various strong baselines with the same storage parameter budget. Our code is available at https://github.com/MIkumikumi0116/DoRA

1 Introduction

Pre-trained Language Models (PLMs) (Kenton and Toutanova, 2019; Brown et al., 2020; Liu et al., 2019; He et al., 2020, 2021b) play a crucial role in Natural Language Processing (NLP), offering substantial improvements in various downstream tasks (Lee et al., 2020; Mars, 2022; Raffel et al., 2020). Customizing these models for specific tasks typically involves fine-tuning them to adapt pre-trained knowledge to particular requirements (Alabi et al., 2022; Uppaal et al., 2023). However, with the increasing scale of PLMs, the cost of full-model fine-tuning becomes prohibitive (Qiu et al., 2020). This has highlighted the demand for and increased interest in more parameter-efficient fine-tuning (PEFT) methods (Zeng et al., 2023; Ding et al., 2023b).

Common PEFT methods introduce extra parameters to adapt downstream tasks and freeze all original parameters (Li and Liang, 2021a; Liu et al., 2022; Lester et al., 2021a). For instance, Low-Rank Adaptation (LoRA) (Hu et al., 2022a) has gained popularity for its streamlined approach by incorporating low-rank trainable matrices into existing fixed weight matrices in a PLM. However, LoRA assigns trainable parameters uniformly across all matrices, and there are studies (Zhang et al., 2023) indicate that not all weights contribute equally to fine-tuning performance. This could result in inefficient parameter usage. Therefore, for optimal fine-tuning, is it possible to evaluate the parameter budget needs of each matrix and strategically allocate the limited parameters?

Fortunately, there are methods like AdaLoRA (Zhang et al., 2023) that can alleviate the limitations of the prior PEFT methods by introducing a more nuanced parameter distribution strategy. Training with AdaLoRA begins with a higher parameter budget and simulates an SVD (Singular value decomposition) decomposition process, progressively pruning smaller singular values and corresponding singular vectors. It opens the door to implementing adaptive allocation of parameter budgets. However, its dependence on orthogonal regularization for the simulated SVD decomposition might restrict further improvements in fine-tuning efficiency. Additionally, the pruning strategy of AdaLoRA focuses solely on singular values and does not fully exploit all the available information in projection matrices, potentially leading to less-than-optimal decisions.

To address existing challenges, this work introduces the Dynamic Low-Rank Adaptation (DoRA) method, as depicted in Figure 1. Different from LoRA approaches, DoRA decomposes high-rank

*Equal contributions.
†Corresponding author.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 11662–11675

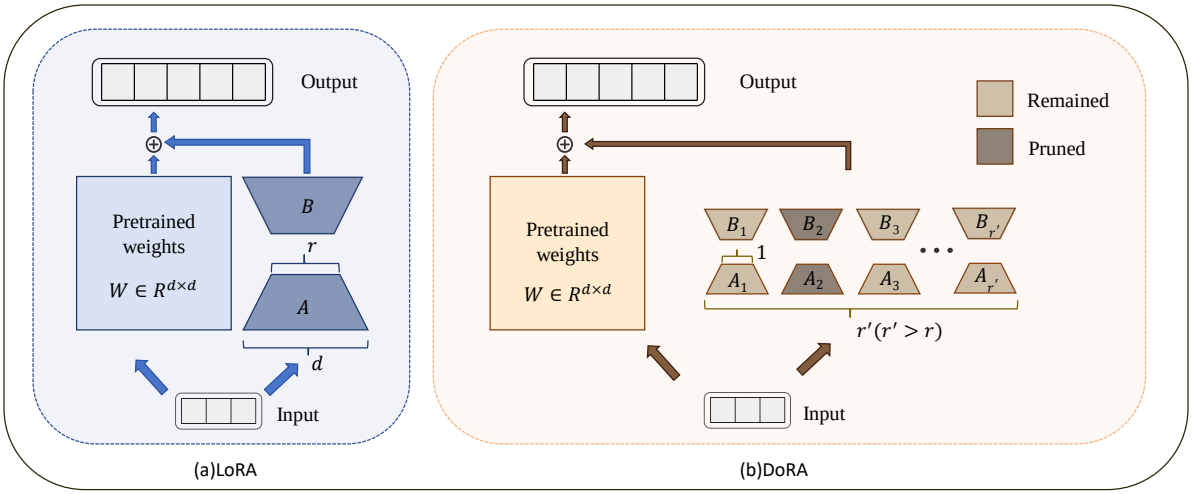

Figure 1: Figure (a) and Figure (b) illustrate the reparameterization of LoRA and DoRA. LoRA introduces a pair of low-rank matrices, A and B, each with a rank of r, into the weight matrix. In contrast, DoRA introduces r' pairs of single-rank matrices, each acting as a LoRA component. During training, DoRA evaluates the contribution of each component to the overall performance and prunes components with smaller contributions, achieving adaptive allocation of parameters.

LoRA layers into sums of single-rank components, evaluates the contribution of each component to overall performance, and prunes components with fewer contributions. This allows for the on-demand allocation of parameter budgets to modules of the PLM, maximizing the use of limited parameter budgets. Compared to existing methods of dynamic parameter allocation (e.g., AdaLoRA), DoRA can allocate parameter budgets more appropriately based on a richer set of information from projection matrices.

To sum up, our contributions are as follows:

- We introduce a novel PEFT method, DoRA, which surpasses the performance of full model fine-tuning with less than 0.3% of the trainable parameters.
- DoRA can efficiently identify the modules in PLMs that play a crucial role in the fine-tuning task, thereby allocating a larger parameter budget to these key modules.
- DoRA maximizes the use of a limited parameter budget. Experiments demonstrate that DoRA outperforms baseline approaches across multiple downstream tasks under the same parameter budget constraints.

2 Background

The emergence of PLMs such as BERT (Kenton and Toutanova, 2019), GPT (Radford et al., 2019), and Llama (Touvron et al., 2023) has meaningfully advanced the field of NLP. Trained on extensive text datasets, PLMs capture intricate linguistic patterns, enabling superior performance in various NLP tasks such as text categorization, named entity recognition, and machine translation (Zhao et al., 2023). Their flexibility in adapting to specific datasets via fine-tuning renders them exceedingly versatile for addressing various linguistic challenges.

PLMs predominantly leverage the Transformer architecture (Vaswani et al., 2017) which features stacked Transformer blocks. Each block comprises two key components: a Multi-Head Attention (MHA) mechanism and a Feed-Forward Neural (FFN) network. In particular, MHA effectively captures contextual relationships in text and is given by:

$$\begin{aligned}\text{MHA}(x) = \text{Concatenate}(&head_1(x),\\ &head_2(x), \ldots, head_h(x))W_o\end{aligned} \tag{1}$$

$$head_i(x) = \text{Softmax}\left(\frac{(xW_{qi})(xW_{ki})^T}{\sqrt{d_h}}\right)xW_{vi} \tag{2}$$

Method	Parameter allocation strategy	Parametric method	Regularization penalty term
LoRA	Equal allocation	Low-rank adaptation	N/A
AdaLoRA	Adaptive allocation	LoRA with SVD decomposition	SVD proximity
DoRA	Adaptive allocation	LoRA with component-wise decomposition	Component variance

Table 1: Comparison of DoRA, LoRA, and AdaLoRA

where $x \in \mathbb{R}^{n \times d}$ is the input feature, n is the sequence length and d is the hidden dimension. The mechanism consists h self-attention heads, each aiming to capture different aspects of information. For each head $head_i$, there are three projection matrices: query W_{qi}, key W_{ki}, and value W_{vi}, each with dimensions $\mathbb{R}^{d \times d_h}$, where d_h is the dimension of each head, typically set to d/h. The output projection matrix $W_o \in \mathbb{R}^{d \times d}$ is used to produce the final output.

Attention scores are calculated by normalizing the dot product of queries and keys through the softmax function and are given by:

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{3}$$

These scores determine the attention each sequence position pays to other positions. Subsequently, these scores are multiplied by the value projection result to yield the output of each head. Finally, the outputs of all heads are concatenated and multiplied by the output projection matrix W_o, forming the final MHA output.

Following MHA, the FFN further processes the information:

$$\text{FFN}(x) = \text{ReLU}(xW_{f1} + b_1)W_{f2} + b_2 \tag{4}$$

This allows for more complex interactions between the features extracted by the self-attention mechanism. Each Transformer block incorporates a residual connection that adds the input of the block directly to its output. This approach helps to alleviate the vanishing gradient problem and ensures a consistent information flow across the layers of the models.

3 Dynamic Low-Rank Adaptation

In this paper, we aim to optimize the use of a limited parameter budget in fine-tuning PLMs with LoRA. We make improvements based on LoRA (Hu et al., 2022a) and AdaLoRA (Zhang et al., 2023), as shown in Table 1. We propose DoRA that stands out due to its innovative approach, comprising three main strategies: a decomposition strategy that views a high-rank LoRA layer as a combination of multiple single-rank LoRA components, a dynamic rank allocation mechanism that adjusts these components based on their contribution to the overall performance of the model and a regularization penalty to ensure stable pruning throughout the process. The overall algorithm is shown in Algorithm 1.

Algorithm 1 DoRA

1: **Input:** Dataset D; total steps T; initial budget $b(0)$; final budget $b(T)$; learning rate γ; regularization coefficient η; smoothing factor β, DoRA parameters A, B, and c.
2: **for** $t = 1$ to T **do**
3: Sample a mini-batch d from D and compute the true label loss $\mathcal{L}_{\text{true}} = \mathcal{L}(A, B, c, d)$;
4: Compute the regularization loss $\mathcal{L}_{\text{reg}}$ as Equation 10;
5: Combine losses by adding true label loss and regularization loss $\mathcal{L}_{\text{combined}} = \mathcal{L}_{\text{true}} + \eta\mathcal{L}_{\text{reg}}$;
6: Perform backpropagation to compute the gradients of $\mathcal{L}_{\text{combined}}$, and update the parameters with the learning rate γ;
7: Compute the importance score s as Equation 7, update smoothed importance score $\widetilde{s}(t)$ as Equation 8;
8: Compute the current parameter budget $b(t)$ as Equation 9;
9: Prune the components with smaller $\widetilde{s}(t)$ based on $b(t)$, set their c to 0;
10: **end for**
11: **Output:** The fine-tuned parameters $\{A, B, c\}$.

3.1 Parameterization

DoRA introduces a novel perspective on PEFT for PLMs, building upon and enhancing the foundational LoRA technique. A standard LoRA layer is defined as:

$$W = W_0 + \Delta W = W_0 + AB \tag{5}$$

where W is the weight matrix after fine-tuning, W_0 denotes the original weight matrix, and A, B are the low-rank matrices introduced by LoRA. By contrast, DoRA reinterprets this configuration and is given by:

$$W = W_0 + \sum_{i=1}^{r'} \Delta W_i = W_0 + \sum_{i=1}^{r'} A_i B_i c_i \quad (6)$$

here, r' represents the number of LoRA components, which will be explained in detail in Section 3.3. A LoRA component is a triplet of A_i, B_i, and c_i, where A_i and B_i are single-rank matrices, shaped as $d \times 1$ and $1 \times d$ respectively. c_i is a scalar used for pruning the component, it is set to 0 if the component is to be pruned.

3.2 Importance Scoring

To evaluate the importance of each LoRA component, we employ an importance scoring mechanism that quantifies the contribution of each ΔW_i and is given by:

$$\begin{aligned} s_i &= \|\Delta W_i\|_F / \|\sum_{j=1}^{r'} \Delta W_j\|_F \\ &= \|A_i B_i c_i\|_F / \|\sum_{j=1}^{r'} A_j B_j c_j\|_F \end{aligned} \quad (7)$$

here, $\|x\|_F$ denotes the Frobenius norm, a measure that calculates the square root of the sum of the squares of all elements in a matrix.

Employing the Frobenius norm allows us to measure the proportion of each LoRA component contribution to the total update magnitude of its corresponding LoRA layer. This metric facilitates an estimation of the potential impact on the total update of the LoRA layer if a particular component were to be pruned. Components with smaller impacts on the overall update magnitude are prioritized for pruning. This ensures that the pruning process minimally affects the performance, focusing on removing components that contribute the least to the effectiveness of the LoRA layer.

Compared to previous methods (Zhang et al., 2023), we use $\|\Delta W_i\|_F$ instead of c_i to assess the importance of components, thereby incorporating information from A_i and B_i for a more comprehensive evaluation of component importance.

Moreover, to enhance the precision of the importance score, we employ a smoothing method by applying an exponential moving average to the importance scores. The smoothed importance score for the i-th LoRA component at time t, denoted as $\widetilde{s}_i(t)$, blends the current importance score s_i with the previous one, adjusted by a factor β:

$$\widetilde{s}_i(t) = \beta \cdot \widetilde{s}_i(t-1) + (1-\beta) \cdot s_i \quad (8)$$

3.3 Parameter Scheduling and Pruning Strategy

Parameter budget refers to the average number of LoRA components in each LoRA layer. It starts with an initial parameter budget, $b^{(0)} = r'$, which is deliberately set higher than the eventual target budget, $b^{(T)} = r$, where r' and r are hyperparameters. Setting r' greater than r allows DoRA to explore a wider range of potential parameter allocations, facilitating the search for the optimal distribution.

DoRA adopts a gentle pruning strategy. For the pruned triplets A_i, B_i, and c_i, pruning is performed merely by setting c_i to 0 while keeping A_i and B_i unchanged. During subsequent training, the pruned triplets can be restored as long as c_i is updated to a non-zero value by backpropagation and is not pruned again.

DoRA warms up the training without pruning for the first t_i step, i denotes initial steps and then follows a cubic decrement pattern to prune components with lower importance scores until the remaining components reach the budget $b^{(T)}$. Subsequently, it fixes the component distribution in the last t_f steps, f denotes the final steps. The overall budget scheduler is given by:

$$b^{(t)} = \begin{cases} b^{(0)} & \text{if } 0 \le t < t_i, \\ b^{(0)} - \dfrac{(b^{(0)} - b^{(T)})}{b^{(0)}} \cdot \left(\dfrac{t - t_i}{t_f - t_i} \right)^3 & \text{if } t_i \le t \le T - t_f, \\ b^{(T)} & \text{if } t > T - t_f. \end{cases} \quad (9)$$

3.4 Dimensional Equilibrium Modulator

DoRA utilizes the Frobenius norm of components for pruning, with a preference for clipping those with smaller norms. However, a potential issue arises when a component has most elements near zero and a few with considerably high values, leading to a relatively low Frobenius norm and thus being selected for pruning. This scenario can result

in remarkable alterations in a limited number of dimensions of the total update, ΔW, resembling the effects of gradient explosion and adversely impacting model stability and fine-tuning performance.

To avoid this, we introduce the Dimensional Equilibrium Modulator (DEM) loss, which penalizes the variance of components as:

$$R = \frac{1}{n}\sum_{i=1}^{n}(\text{Var}(A_i) + \text{Var}(B_i)) \qquad (10)$$

where $\text{Var}(A_i)$ and $\text{Var}(B_i)$ represent the variances of components A_i and B_i, with n indicating the number of components. DEM encourages a uniform distribution of elements within components, avoiding disproportionate impacts from isolated or few dimensions, effectively reducing perturbations from model pruning, and enhancing model stability.

4 Experiments

4.1 Experimental Setup

We compared DoRA with existing baseline methods to evaluate its performance in natural language understanding (NLU), question answering (QA), and text generation (summarization) tasks. We chose RoBERTa (Liu et al., 2019) and Bart (Lewis et al., 2019) as the foundational models, used respectively for NLU and QA tasks, and for summarization tasks.

RoBERTa is an optimized version of the BERT (Kenton and Toutanova, 2019) architecture, which significantly improves performance on a variety of language understanding tasks through extended training, larger datasets, and finer tuning of parameters. Bart is a Transformer-based (Vaswani et al., 2017) sequence-to-sequence pre-trained model specifically designed for text generation tasks, such as summarization. It effectively handles various generation tasks by combining bidirectional and autoregressive Transformer architectures.

We tested the performance on several standard datasets: using the GLUE (General Language Understanding Evaluation) (Wang et al., 2018) dataset to evaluate NLU tasks, SQuAD (Rajpurkar et al., 2016) (Stanford Question Answering Dataset) for QA, and Xsum (Narayan et al., 2018) for text summarization. GLUE is a set of dataset for training and testing NLU systems, including various tasks such as sentiment analysis and textual entailment. SQuAD is a question answering dataset that consists of questions generated from Wikipedia articles and their corresponding answers. Xsum provides a testing environment for extreme summarization tasks aimed at generating single-sentence summaries, challenging the models under extreme information compression conditions.

We selected several mainstream fine-tuning methods as baselines, including LoRA, AdaLoRA, Adapter Tuning, BitFit, and full model fine-tuning. LoRA fine-tunes the model weights by adding low-rank matrices to the pre-trained matrices; AdaLoRA is an improvement of LoRA, adding an adaptive adjustment mechanism. Adapter Tuning fine-tunes by inserting lightweight network modules into the PLM. BitFit adjusts only the bias parameters in the PLM. Full model fine-tuning is a traditional method that involves comprehensive adjustment of all model weights.

We report the average results based on 5 random seeds, as shown in Table 2, Table 3, and Table 4, The hyperparameter settings for the experiments can be found in Appendix E.

4.2 Results

We investigate the performance of DoRA and baseline methods across subtasks of the GLUE benchmark, conducting experiments under two different parameter budget scenarios.

As shown in Table 2, DoRA and AdaLoRA, employing adaptive parameter allocation strategies, outperform all baseline methods using uniform parameter distribution, demonstrating the remarkable effectiveness of adaptive parameter allocation. Across the GLUE benchmark, DoRA surpasses LoRA by 0.84% and 0.88%, and AdaLoRA by 0.59% and 0.45% under two parameter budgets, further proving the broad applicability and effectiveness of DoRA's adaptive parameter allocation strategy in multiple tasks.

Especially noteworthy is DoRA's performance on the CoLA dataset, where it shows the highest improvement, surpassing the highest performing baseline method by 1.48% and 1.73% under two parameter budgets. This highlights DoRA's advantage in handling the linguistic acceptability task, showcasing its efficiency in dealing with challenging NLP tasks. However, DoRA's performance on the MNLI task slightly lags behind AdaLoRA, likely due to MNLI being the largest dataset in GLUE with high task complexity, indicating a need for further optimization of the adaptive parameter allocation strategy when dealing with large-scale

Method	Trainable Parameters	RTE	MRPC	STS-B	CoLA	SST-2	QNLI	QQP	MNLI	Avg
Full FT	124.65M	78.63	88.33	90.31	60.26	94.73	92.58	90.75	87.68	85.41
BitFit	0.10M	79.57	89.07	90.55	61.16	94.38	90.99	88.08	85.50	84.91
H-Adapter	1.20M	80.43	89.90	90.16	62.62	93.73	92.82	90.83	86.53	85.88
P-Adapter	1.19M	80.51	89.51	90.65	63.87	93.83	92.61	90.53	86.75	86.03
LoRA	1.33M	80.65	89.90	90.91	63.54	93.71	92.76	90.44	87.11	86.13
AdaLoRA	1.27M	81.23	89.02	91.22	63.23	95.11	92.84	90.48	**87.89**	86.38
DoRA	1.31M	**81.73**	**90.05**	**91.34**	**65.35**	**95.21**	**92.97**	**91.32**	87.81	**86.97**
H-Adapter	0.31M	78.56	88.64	90.88	61.76	93.54	92.52	90.16	86.31	85.30
P-Adapter	0.30M	79.07	88.74	90.44	62.92	93.24	92.59	89.94	86.23	85.40
LoRA	0.33M	78.63	88.68	91.24	62.40	93.25	92.75	90.12	87.01	85.50
AdaLoRA	0.32M	79.04	88.81	91.06	63.17	94.79	92.87	90.07	**87.64**	85.93
DoRA	0.34M	**79.15**	**89.72**	**91.28**	**64.90**	**94.98**	**92.93**	**90.64**	87.45	**86.38**

Table 2: Results of fine-tuning RoBERTa base on GLUE. We report results on development set, Pearson correlation for STS-B, Matthew's correlation for CoLA, average accuracy for MNLI (matched and mismatched), and accuracy for other tasks. "Full FT", "H-Adapter", and "P-Adapter" represent full fine-tuning, Houlsby adapter, and Pfeiffer adapter. The best results are in **bold**.

	Trainable Parameters	SQuAD v1	SQuAD v2
Full FT	124.65M	85.32/91.49	79.95/83.09
BitFit	0.10M	82.34/89.45	74.28/77.46
H-Adapter	1.20M	84.95/91.07	79.14/82.08
P-Adapter	1.19M	84.86/90.86	78.86/81.84
LoRA	1.33M	85.13/91.39	79.25/82.34
AdaLoRA	1.28M	85.34/91.72	79.87/82.84
DoRA	1.30M	**85.97/92.24**	**80.43/83.53**
H-Adapter	0.31M	84.60/90.44	78.48/81.55
P-Adapter	0.30M	84.44/90.34	78.22/81.34
LoRA	0.33M	84.91/90.91	78.83/81.78
AdaLoRA	0.32M	85.13/91.32	79.47/82.40
DoRA	0.34M	**85.73/91.88**	**79.90/82.92**

Table 3: Results of fine-tuning RoBERTa based on SQuAD. We report the exact match and F1 scores on the development set. The best results are in **bold**.

	Trainable Parameters	Xsum
Full FT	124.65M	40.61/17.76/32.91
LoRA	1.33M	38.77/15.63/30.66
AdaLoRA	1.27M	39.14/16.23/31.34
DoRA	1.31M	**39.67/16.73/31.78**
LoRA	0.33M	37.17/14.57/29.72
AdaLoRA	0.32M	38.32/15.69/30.74
DoRA	0.34M	**38.94/16.22/31.36**

Table 4: Results of fine-tuning Bart base on Xsum. We report the Rouge-1, Rouge-2, and Rouge-L scores on the development set. The best results are in **bold**.

complex tasks.

It is worth mentioning that DoRA demonstrates exceptional parameter efficiency, surpassing the performance of full model fine-tuning with only 0.34M parameters, less than 0.3% of full model fine-tuning, highlighting DoRA's capability in effectively utilizing a limited parameter budget.

Similar results are also observed in the experiments on SQuAD and Xsum, where DoRA outperformes all baseline PEFT methods under both parameter settings.

5 Analysis and Discussion

5.1 Effectiveness of DEM

To verify the effectiveness of DEM, we tested fine-tuning on datasets including STS-B, CoLA, and SST-2, with and without DEM, without DEM means setting hyper-parameter regularization coefficient η to 0, as shown in Table 5.

Model	STS-B	CoLA	SST-2
with DEM	**91.34**	**65.35**	**95.21**
without DEM	91.23	64.17	95.12

Table 5: Performance comparison of DEM

Enabling DEM imposes penalties on the variance of LoRA components, encouraging a uniform weight distribution, and avoiding extreme variations in overall update ΔW across a few dimensions due to pruning. Fine-tuning with DEM enabled achieved higher results, demonstrating the effectiveness of DEM.

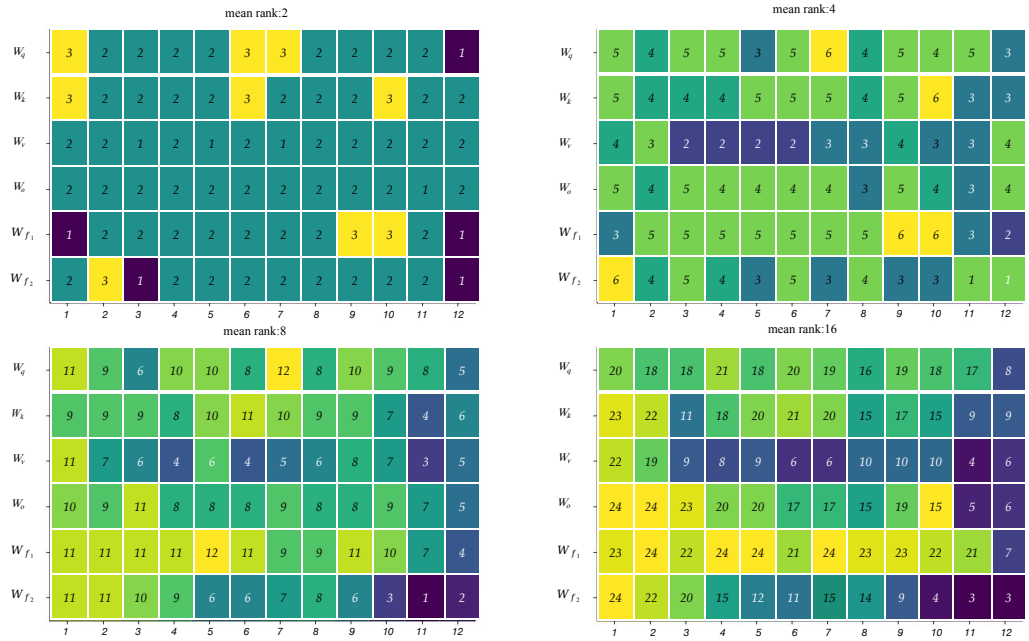

Figure 2: Rank distribution under four parameter budgets

5.2 Parameter Allocation Preference

To validate whether DoRA can identify key modules in PLMs, we set the final budgets $b^{(T)}$ to 2, 4, 8, and 16, with 1.5 times the final budget as the initial budget $b^{(0)}$, and conducted fine-tuning experiments on SST-2 dataset respectively.

The results are visually presented in Figure 2, which shows that, in the intermediate layers, the query and key matrices are allocated with more parameter budget, while the value matrices are allocated with fewer budget. The initial output matrices receive more budget. In the feed-forward neural network, the lower projection matrices, represented as W_{f_2} in the figure, at the backend, especially in the last few layers, are allocated with very few budgets.

DoRA exhibited the same parameter allocation tendencies across all four configurations, demonstrating its ability to consistently identify key modules in PLMs and allocate more parameter budgets to them accordingly.

5.3 Impact of Initial Budget

We investigated the impact of the initial budget $b^{(0)}$ across the MRPC, STS-B, and SST-2 datasets. We fine-tuned models starting from various initial budgets and pruned them to a consistent final budget $b^{(T)}$ of 2. The results are presented in Table 6. The first row indicates that when the initial budget is 2, it matches the final budget, which means no model pruning was performed.

Intriguingly, our findings suggest that maintaining a constant final parameter budget while starting with a higher initial parameter budget improves model performance. We attribute this improvement to a more generous initial parameter budget offering a wider exploration space for DoRA, thereby increasing the chance of preserving essential parameters during pruning and optimizing the model's final performance.

Initial Budget	MRPC	STS-B	SST-2
2	76.32	90.97	94.88
3	89.74	91.28	94.98
4	89.92	91.40	95.01
6	90.14	91.62	95.05
8	90.17	91.65	95.11

Table 6: The Impact of Initial Rank Size

6 Related Work

PEFT is crucial for the fine-tuning of PLMs in practical applications. These techniques primarily focus on updating a select subset of the model's parameters or introducing new parameters on a small scale, enabling more efficient use of resources. These approaches are particularly valuable in scenarios constrained by computational resources. Existing PEFT methods can generally

be divided into three categories: addition-based methods, specification-based methods, and reparameterization-based methods (Ding et al., 2022).

Addition-based methods achieve adjustment by adding extra modules or trainable parameters to PLMs, such as trainable adapters or soft prompts. These methods are scalable and applicable to models of various sizes, with the performance gap between them and full model fine-tuning narrows as model size increases. Examples include adapters (Houlsby et al., 2019; Pfeiffer et al., 2021; He et al., 2021a; Zhu et al., 2021), which insert small neural modules into transformer layers for adjustment, and prompt-based tuning (Li and Liang, 2021b; Gao et al., 2021; Hu et al., 2022b; Tan et al., 2022; Lester et al., 2021b; Vu et al., 2021), which stimulates PLMs by adding additional context around the original input.

Specification-based methods focus on fine-tuning a few inherent parameters within the model without altering its internal structure (Vucetic et al., 2022; Holmes et al., 2021). By directly specifying which parts of the parameters to optimize, these approaches achieve efficient model adaptation, maintaining performance close to full parameter fine-tuning while reducing the number of parameters adjusted. Examples include BitFit (Ben Zaken et al., 2022), which optimizes only the bias terms in the model, and Diff Pruning (Guo et al., 2021), which introduces sparsity by optimizing a difference vector.

Reparameterization-based methods optimize models by transforming adaptive parameters into more efficient forms, often based on the low-rank hypothesis. These methods (Holmes et al., 2021; Karimi Mahabadi et al., 2021; Edalati et al., 2022; Zhang et al., 2023; Lialin et al., 2023; Ding et al., 2023a; Valipour et al., 2023; Su et al., 2024) aim to reduce computational and memory costs by optimizing low-dimensional proxy parameters while maintaining or surpassing the performance of full parameter fine-tuning. They are grounded in the theory that PLM adaptations to downstream tasks are inherently low-rank. Examples include LoRA (Hu et al., 2022a), which optimizes based on the hypothesis of a low intrinsic rank of weight changes.

7 Conclusion

In this paper, we introduce Dynamic Low-Rank Adaptation (DoRA), a novel method aiming at enhancing the efficiency of fine-tuning PLMs by dynamically adjusting parameter distribution. DoRA innovatively allocates parameter budgets based on their importance to specific tasks, demonstrating considerable improvements in NLP applications. Experimental results indicate that DoRA surpasses baseline methods, highlighting its potential for broader adoption in model optimization efforts.

The innovation of DoRA lies in its adoption of an adaptive parameter allocation strategy, which, unlike traditional uniform distribution, dynamically adjusts the distribution of parameter budgets based on their contribution. Additionally, DoRA employs a component-wise decomposition approach for handling LoRA layers, treating high-rank LoRA layers as a combination of single-rank LoRA components. These components are adjusted through a dynamic rank allocation mechanism, pruned according to their contribution to the overall model performance. To ensure stable pruning throughout the process, DoRA incorporates a regularization penalty term focused on reducing component variance.

Limitation

Our study confirms the effectiveness of DoRA in several NLP tasks. However, its evaluation has been limited to these tasks, and its efficacy in handling more complex NLP challenges, such as machine translation or multimodal tasks, has yet to be established. Moreover, the models used in our experiments are somewhat limited in scale, as we have not conducted experiments with large language models (LLMs). Addressing this limitation, future work could explore DoRA's potential in these sophisticated areas of NLP.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No.62376019, 61976015, 61976016, 61876198 and 61370130) and the Talent Fund of Beijing Jiaotong University (2024JBRC005). We sincerely thank the reviewers for their insightful comments and suggestions to improve the quality of the paper.

References

Jesujoba O. Alabi, David Ifeoluwa Adelani, Marius Mosbach, and Dietrich Klakow. 2022. Adapting pre-trained language models to African languages via multilingual adaptive fine-tuning. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4336–4349, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland. Association for Computational Linguistics.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. *TAC*, 7:8.

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

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. 2023a. Sparse low-rank adaptation of pre-trained language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4133–4145.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2023b. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Ali Edalati, Marzieh Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. 2022. Krona: Parameter efficient tuning with kronecker adapter. *arXiv preprint arXiv:2212.10650*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Demi Guo, Alexander Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4884–4896, Online. Association for Computational Linguistics.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021b. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

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

Connor Holmes, Minjia Zhang, Yuxiong He, and Bo Wu. 2021. Nxmtransformer: Semi-structured sparsification for natural language understanding via admm. *Advances in neural information processing systems*, 34:1818–1830.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

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

Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Jingang Wang, Juanzi Li, Wei Wu, and Maosong Sun. 2022b. Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2225–2240, Dublin, Ireland. Association for Computational Linguistics.

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

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

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

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

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461.

Xiang Lisa Li and Percy Liang. 2021a. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

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

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. 2023. Relora: High-rank training through low-rank updates. In *Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ NeurIPS 2023)*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Mourad Mars. 2022. From word embeddings to pre-trained language models: A state-of-the-art walk-through. *Applied Sciences*, 12(17):8805.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *ArXiv*, abs/1808.08745.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. *Advances in neural information processing systems*, 30.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew 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, Seattle, Washington, USA. Association for Computational Linguistics.

Zhan Su, Fengran Mo, Prayag Tiwari, Benyou Wang, Jian-Yun Nie, and Jakob Grue Simonsen. 2024. Mixture of experts using tensor products. *arXiv preprint arXiv:2405.16671*.

Zhixing Tan, Xiangwen Zhang, Shuo Wang, and Yang Liu. 2022. MSP: Multi-stage prompting for making pre-trained language models better translators. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6131–6142, Dublin, Ireland. Association for Computational Linguistics.

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

Rheeya Uppaal, Junjie Hu, and Yixuan Li. 2023. Is fine-tuning needed? pre-trained language models are near perfect for out-of-domain detection. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12813–12832, Toronto, Canada. Association for Computational Linguistics.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2023. Dylora: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3274–3287.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2021. Spot: Better frozen model adaptation through soft prompt transfer. *arXiv preprint arXiv:2110.07904*.

Danilo Vucetic, Mohammadreza Tayaranian, Maryam Ziaeefard, James J Clark, Brett H Meyer, and Warren J Gross. 2022. Efficient fine-tuning of bert models on the edge. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1838–1842. IEEE.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

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

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 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, Online. Association for Computational Linguistics.

Guangtao Zeng, Peiyuan Zhang, and Wei Lu. 2023. One network, many masks: Towards more parameter-efficient transfer learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7564–7580, Toronto, Canada. Association for Computational Linguistics.

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

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Yaoming Zhu, Jiangtao Feng, Chengqi Zhao, Mingxuan Wang, and Lei Li. 2021. Counter-interference adapter for multilingual machine translation. *arXiv preprint arXiv:2104.08154*.

A Potential Risks of Our Method

Since our proposed method adapts pre-trained models to specific tasks, it has the potential to extend the range of languages supported by these models. However, there is a risk that some malicious users might exploit this new capability to provide services in politically sensitive languages and tasks. For instance, a malicious user could use our method to generate hateful or offensive sentences in some politically sensitive languages.

B Dataset Detail

GLUE (Wang et al., 2018) benchmark is a collection of diverse natural language understanding tasks designed to evaluate and analyze the performance of models across a wide range of linguistic challenges. The benchmark encompasses a variety of tasks, including linguistic acceptability (CoLA (Warstadt et al., 2019)), sentiment analysis (SST-2 (Socher et al., 2013)), paraphrase detection (MRPC (Dolan and Brockett, 2005), QQP (Wang et al., 2018)), semantic textual similarity (STS-B (Wang et al., 2018)), and natural language inference (MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), RTE (Bentivogli et al., 2009)). The dataset statistics are presented in Table 7.

Corpus	Train	Valid	Test	Metrics
RTE	2.5k	277	3k	Accuracy
MRPC	3.7k	408	1.7k	Accuracy
STS-B	5.7k	1.5k	1.4k	Pearson corr
CoLA	8.5k	1,043	1,063	Matthews corr
SST-2	67k	872	1.8k	Accuracy
QNLI	105k	5.5k	5.5k	Accuracy
QQP	364k	40.4k	391k	Accuracy
MNLI	393k	20k	20k	Accuracy

Table 7: Statistics of the GLUE Benchmark Datasets

SQuAD (Rajpurkar et al., 2016) is an extensive reading comprehension dataset aimed at evaluating the ability of models to understand and answer questions based on Wikipedia article contents. The dataset features two major versions: SQuAD 1.1 and SQuAD 2.0. SQuAD 1.1 consists of over 100,000 question-answer pairs on 500+ articles, where questions are posed by crowd workers on a given passage and the answers are segments of text from the passage. SQuAD 2.0 extends the SQuAD 1.1 dataset with over 50,000 additional unanswerable questions that are written adversarially by crowd workers to look similar to answerable ones but do not have answers in the text. This makes SQuAD 2.0 more challenging and helps models better emulate human reading comprehension abilities by not only retrieving answers but also determining when no valid answer exists within the text.

Xsum (Narayan et al., 2018) is designed specifically for the task of single-sentence news summarization to create a concise abstract of a news article. Xsum consists of approximately 227,000 articles collected from the British Broadcasting Corporation (BBC). Each article comes with a professionally written, single-sentence summary, making it particularly challenging for models due to the extreme brevity and the need for models to abstract rather than simply extract content. Unlike other summarization datasets, which often focus on extracting salient sentences directly from the text, Xsum requires models to generate informative, concise, and grammatically coherent summaries that capture the core essence of the article content, often requiring synthesis and rephrasing skills beyond mere extraction.

C Baseline Detail

LoRA (Hu et al., 2022a): LoRA achieves fine-tuning by integrating low-rank matrices within the weight matrices of the PLM. It maintains efficiency by adjusting a reduced set of parameters, which is particularly advantageous for scaling to large models.

AdaLoRA (Zhang et al., 2023): Building upon LoRA, AdaLoRA optimizes parameter utilization by adaptively adjusting the parameter budget throughout the training process. This adaptive strategy improves fine-tuning efficiency and has demonstrated enhanced performance in a variety of NLP tasks.

Adapter Tuning (Houlsby et al., 2019; Rebuffi et al., 2017; Pfeiffer et al., 2021): Incorporates methods such as the Houlsby Adapter (H-Adapter) and Pfeiffer Adapter (P-Adapter). Adapter Tuning fine-tunes models by inserting small, trainable adapter modules between existing layers of the PLM. This approach does not modify the original weights in the PLM, offering a flexible yet conservative way to adapt the model to new tasks.

BitFit (Ben Zaken et al., 2022): An minimalistic PEFT method that focuses solely on adjusting the bias terms within the model. BitFit has been shown to achieve performance levels comparable to those of full model fine-tuning under specific conditions.

Full Model Fine-tuning: This conventional method updates all the parameters of a PLM to tailor it to specific downstream tasks. Full model fine-tuning demands remarkable computational resources.

D Training Setup

Our experiments were conducted using the PyTorch (Paszke et al., 2019) framework, with models and datasets sourced from the Huggingface (Wolf et al., 2020) platform. The computations were performed on NVIDIA GeForce RTX 3090 GPUs.

We refer to LoRA's initialization method, applying Kaiming initialization (He et al., 2015) to A_i and B_i, and initializing c_i with 0. This ensures that the initial update amount ΔW_i of each component is 0, preserving the behavior of the original model at the initial stage of training.

E Hyper Parameter

We fix the following hyperparameters across all experiments:

- Initial budget $b^{(0)}$: 1.5 times the final budget $b^{(T)}$
- Smoothing factor β: 0.9
- Initial steps t_i: 15% of the total steps
- Final steps t_f: 50% of the total steps

We fixed sentence length and searched remain hyperparameters, including the learning rate, batch size, number of training epochs, regularization coefficient, and pruning step interval, as shown in Table 8

F Computing Efficiency

We evaluate the computational efficiency of DoRA compared to baseline methods based on the training time per epoch on QNLI dataset and GPU memory usage. The results are shown in Table 9. DoRA's computational efficiency is comparable to AdaLoRA, slightly lower than LoRA, and significantly higher than full model fine-tuning.

Corpus	length	learning rate	batch size	epochs	regularization coefficient	pruning step interval
RTE	128	2e-03	16	80	0.5	10
MRPC	128	2.5e-03	16	30	0.3	50
STS-B	128	3e-03	16	45	0.3	10
CoLA	128	2e-03	16	45	0.3	10
SST-2	128	8e-04	64	60	0.5	10
QNLI	128	3.5e-03	16	12	0.5	50
QQP	128	1e-03	16	10	0.3	50
MNLI	128	3e-03	32	10	0.3	50
SQuAD v1	384	5e-03	16	14	0.1	100
SQuAD v2	384	3.4e-03	16	14	0.1	100
Xsum	768	3.4e-03	16	15	0.1	100

Table 8: Hyper-parameter setup

	Time Per Epoch/min	GPU memory consumption/MiB
DoRA	7.8	4112
AdaLoRA	7.9	4130
LoRA	7.2	4094
H-Adapter	6.8	3698
P-Adapter	6.4	3678
BitFit	6.4	3612
Full FT	10.3	6104

Table 9: Computing Efficiency Comparison