Derivative-Free Optimization for Low-Rank Adaptation in Large Language Models

Anonymous ACL submission

Abstract

Parameter-efficient tuning methods such as LoRA could achieve comparable performance to model tuning by tuning a small portion of the parameters. However, substantial computational resources are still required, as this process involves calculating gradients and performing back-propagation throughout the model. Much effort has recently been devoted to utilizing the derivative-free optimization method to eschew the computation of gradients and showcase an augmented level of robustness in few-shot settings. In this paper, we prepend the low-rank modules into each self-attention layer of the model and employ two derivative-free optimization methods to optimize these low-rank modules at each layer alternately. Extensive results on various tasks and language models demonstrate that our proposed method achieves substantial improvement and exhibits clear advantages in memory usage and convergence speed compared to existing gradient-based parameter-efficient tuning and derivative-free optimization methods in few-shot settings.

1 Introduction

In recent years, there has been a rapid development in large language models (LLMs) (Radford et al., 2019; Brown et al., 2020; OpenAI, 2023; Touvron et al., 2023), which have showcased impressive capabilities across various natural language processing tasks. However, the sheer number of parameters of large models leads to a linear increase in tuning cost and poses challenges for fine-tuning on common hardware. To this end, parameter-efficient tuning methods (He et al., 2022; Houlsby et al., 2019; Chen et al., 2022b,a) have emerged as a solution that could achieve comparable performance to full fine-tuning while only tuning a small portion of the parameters in large language models. Although these methods can reduce GPU memory requirements by approximately 30% (Sung et al., 2022), tuning a small subset of parameters still involves the computation of gradients and back-propagation, which presents challenges for utilizing and deploying large language models. Currently, the prevailing approach to harness the power of large language models is through in-context learning, treating the model as a service (Brown et al., 2020). The approach only involves forward computation and requires designing appropriate prompts or demonstrations without updating model parameters. However, in-context learning demands a meticulous selection of prompts and demonstrations, and the model's performance relies entirely on the chosen prompts and demonstrations (Gao et al., 2021).

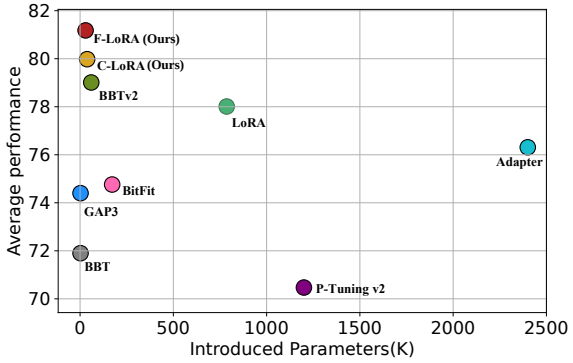

Figure 1: The results of our proposed methods F-LoRA and C-LoRA compared to gradient-based and gradient-free methods on average performance over seven language understanding tasks. We evaluate all the methods on RoBERTa-large.

Recently, much effort has been devoted to black-box tuning methods (Sun et al., 2022b,a; Zhao et al., 2023; Xu et al., 2023; Oh et al., 2023), which utilizes the derivative-free optimization method to optimize the introduced continuous prompts. Black-box tuning methods achieve comparable performance to parameter-efficient tuning methods and full fine-tuning in a few-shot setting without needing gradient computation and back-propagation through the entire LLM. However, it is acknowledged that training the prompt vectors in few-shot

settings is prone to instability and exhibits slow convergence (Lester et al., 2021; Li and Liang, 2021; Liu et al., 2021b), making it challenging to generalize to large language models.

In this paper, we propose a derivative-free optimization approach to address the challenges associated with the introduced low-rank modules in large language models that eschews the computation of gradients and showcases an augmented level of robustness in few-shot settings. Our method eliminates the need for gradient computation and back-propagation, resulting in improved stability and faster convergence compared to previous baselines. We prepend low-rank modules into each self-attention layer of the language model and initialize these modules by computing the mean and variance of hidden states for each layer. To optimize the parameters of the low-rank modules, we employ two derivative-free optimization methods. Recognizing that directly optimizing all low-rank modules of each layer using derivative-free methods in a high-dimensional space may slow down convergence, we adopt a divide-and-conquer strategy. The strategy entails optimizing the low-rank modules of each layer separately. To enable the optimization of the low-rank modules through derivative-free methods, we introduce a linear mapping matrix. The matrix maps the parameters obtained after derivative-free optimization to the desired low-rank modules at each layer. We initialize the linear mapping matrix based on normal distributions, with standard deviations related to the hidden states of each layer.

We conduct comprehensive experiments on RoBERTa-large (Delobelle et al., 2020), GPT2-large, and GPT2-XL (Radford et al., 2019) to assess the effectiveness of our method. The results demonstrate that our proposed method has a significant improvement on average across seven natural language understanding tasks in a few-shot setting. As shown in the Figure 1, our proposed approach achieves substantial improvement compared to existing gradient-based parameter-efficient methods (e.g., Adapter tuning, LoRA, P-Tuning v2, and BitFit) and derivative-free optimization methods (e.g., BBT, GAP3, and BBTv2) on RoBERTa-large. Additionally, our proposed method demonstrates superior performance, clear advantages regarding GPU memory usage, and faster model convergence speed compared to existing derivative-free optimization methods in larger models.

2 Preliminaries

2.1 Derivative-free Optimization

Derivative-free optimization (DFO) algorithms (Wierstra et al., 2014; Rios and Sahinidis, 2013; Qian et al., 2016) are capable of tackling complex problems without relying on the back-propagation. Typically, these DFO algorithms employ a sampling-and-updating framework to enhance the solution iteratively. These algorithms have broad applications spanning various fields, from automatic machine learning (Snoek et al., 2012) to reinforcement learning (Salimans et al., 2017; Bai et al., 2023) and objective detection (Zhang et al., 2015b). Representative DFO algorithms include CMA-ES (Covariance Matrix Adaptation Evolution Strategy) (Hansen and Ostermeier, 2001), Fireworks algorithm (Li and Tan, 2018; Chen and Tan, 2021; Li and Tan, 2020), Genetic algorithms (Mitchell, 1998), among others. CMA-ES is a widely adopted evolutionary algorithm for nonlinear and non-convex continuous optimization. It generates new potential solutions by sampling from a multivariate normal distribution model at each iteration. Besides, we have a Fireworks algorithm (FWA) based on simulating the explosion process of fireworks, introducing randomness and diversity to aid in escaping local minima and conducting a more comprehensive search of the problem space. FWA presents a new search manner that searches the potential space by a stochastic explosion process within a local space. Based on CMA-ES and FWA, we propose two derivative-free optimization methods for low-rank adaptation: C-LoRA and F-LoRA. We detail the optimization processes of the two gradient-free optimization methods in Appendix A.1 and A.2.

2.2 Black-Box-Tuning

Common language understanding tasks can be formulated as classification tasks by incorporating task-specific prompts and a few labeled samples or by carefully engineering prompts and verbalizers (Brown et al., 2020; Schick and Schütze, 2021a). For example, an input sentence combined with template P that includes a <MASK> token can be represented as $X = \{x_1, x_2, \cdots, x_L, P, \text{<MASK>}.\}$, and L corresponds to the length of the input sentence. When X is fed into model f, the model can determine whether the corresponding label token of class Y (e.g., "Yes" or "No") is more appropriate to replace the <MASK> token (Gao et al., 2021).

Prompt tuning (Lester et al., 2021) and P-tuning (Liu et al., 2021b) insert continuous prompt vectors $\boldsymbol{p} \in \mathbb{R}^D$ into the input X at the embedding layer, where the objective can be formulated as follows:

$$\boldsymbol{p}^\star = \arg\min_{\boldsymbol{p}\in\Theta} \mathcal{L}(f(\boldsymbol{p}; X), Y) \tag{1}$$

where $\mathcal{L}$ is the loss function, Θ is the search space, and $\boldsymbol{p}^\star$ is the optimal prompt vector after a gradient-based optimization through the model of f.

Recently, a gradient-free prompt tuning method, BBT (Sun et al., 2022b), was proposed to learn the continuous prompts without back-propagation through the model. BBT utilizes derivative-free optimization algorithms to optimize the continuous prompt as follows:

$$\boldsymbol{z}^\star = \arg\min_{\boldsymbol{z}\in\mathcal{Z}} \mathcal{L}(f(\boldsymbol{A}\boldsymbol{z}; X), Y) \tag{2}$$

where $\boldsymbol{A} \in \mathbb{R}^{D\times d}$ is the random projection matrix, $\mathcal{Z}$ is the search space, and $\boldsymbol{z} \in \mathbb{R}^d$ is a low-dimensional subspace. BBT adopts the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen and Ostermeier, 2001) to obtain the optimal prompt $\boldsymbol{z}^\star$. Inspired by the success of deep prompt tuning (Li and Liang, 2021; Liu et al., 2021a), BBTv2 (Sun et al., 2022a) extends BBT by optimizing the deep prompt with derivative-free methods injected at every intermediate layer of the language model.

3 Approach

The intrinsic dimensionality refers to the minimum dimension required to address high-dimensional optimization problems. It plays a crucial role in explaining the effectiveness of fine-tuning in language models (Li et al., 2018; Aghajanyan et al., 2021). Previous research on the intrinsic dimensionality of pre-trained language models has led to the development of approaches such as LoRA (Hu et al., 2022). LoRA introduces down-sampling and up-sampling matrices at each layer of the Transformer and employs a low-rank decomposition optimization method to modify the original weight matrices within the self-attention modules. However, LoRA requires gradient back-propagation throughout the entire model, which may be computationally intensive when dealing with large language models. To further explore the potential benefits of combining low-rank adaptation optimization with derivative-free optimization, as depicted in Figure 2, we formally introduce our method, a derivative-free optimization for low-rank adaptation in large language models.

Similar to the manual prompt learning family of models (Schick and Schütze, 2021a,b; Gao et al., 2021), we first design our model to be close to the pre-training stage (e.g., keeping consistent objective function) by converting each input X to masked language model (MLM) input, which contains a <MASK> token. Then, the model determines the corresponding verbalizers of class Y to substitute for the <MASK> token.

Building upon the success of Low-rank adaptation (LoRA) (Hu et al., 2022) and Black-Box prompt tuning (BBT) (Sun et al., 2022b,a), we incorporate low-rank matrices into the self-attention module of each layer in the pre-trained language model. We optimize the introduced low-rank matrix parameters using derivative-free optimization methods. As shown in Figure 2, the LoRA module consists of two low-rank matrices: $A \in \mathbb{R}^{r\times k}$ and $B \in \mathbb{R}^{D\times r}$. Here, r represents the rank size, typically chosen as 2, 4, 8, 16, etc. For the weight matrices in the self-attention module $W \in \mathbb{R}^{D\times k}$ of the self-attention layer, the parameter updates are performed through matrix decomposition $W + \Delta W = W + BA$. During training, the parameters of W are frozen and do not receive gradient updates, while the parameters of A and B are updated using gradient-free methods.

Considering that large language models have a low intrinsic dimensionality, we further optimize the parameters within a low-rank space using two gradient-free optimization methods. Figure 2 illustrates this process. At the self-attention module of each layer in the language model, we optimize the vectors $\boldsymbol{m_L^1} \in \mathbb{R}^d$, $\boldsymbol{m_L^2} \in \mathbb{R}^d$, $\boldsymbol{m_L^3} \in \mathbb{R}^d$, and $\boldsymbol{m_L^4} \in \mathbb{R}^d$ using gradient-free optimizers such as FWA (Li and Tan, 2018; Chen and Tan, 2021; Li and Tan, 2020) and CMA-ES (Hansen and Ostermeier, 2001), where d is the intrinsic dimension. These optimized vectors are then projected into the low-rank space using specific random projection modules $\boldsymbol{G_L^1} \in \mathbb{R}^{r\times k\times d}$, $\boldsymbol{G_L^2} \in \mathbb{R}^{D\times r\times d}$, $\boldsymbol{G_L^3} \in \mathbb{R}^{r\times k\times d}$, and $\boldsymbol{G_L^4} \in \mathbb{R}^{D\times r\times d}$. The computation can be expressed as follows:

$$\begin{aligned} A_Q &= \boldsymbol{G_L^1 m_L^1} \\ B_Q &= \boldsymbol{G_L^2 m_L^2} \\ A_K &= \boldsymbol{G_L^3 m_L^3} \\ B_K &= \boldsymbol{G_L^4 m_L^4} \end{aligned} \tag{3}$$

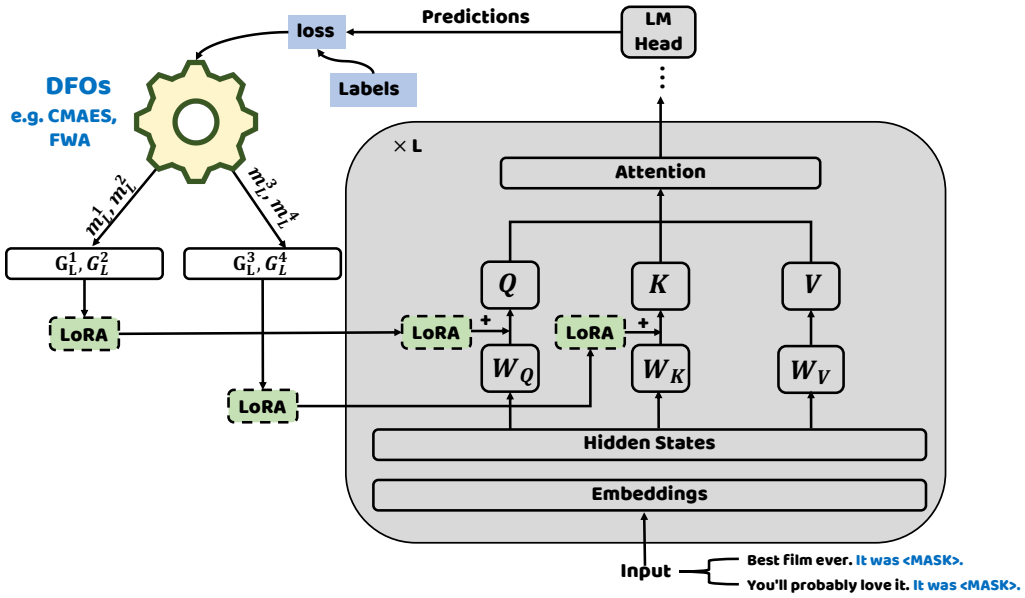

Figure 2: An illustration of derivative-free optimization for low-rank adaptation. We apply the low-rank matrices (green boxes) at the self-attention module of each layer and initialize them with model-specific normal distributions. We use two derivative-free methods (e.g., CMA-ES and Firework algorithm) to alternately optimize low-rank modules at the self-attention module of each layer.

After obtaining $A_Q \in \mathbb{R}^{r\times k}$, $B_Q \in \mathbb{R}^{D\times r}$, $A_K \in \mathbb{R}^{r\times k}$, and $B_K \in \mathbb{R}^{D\times r}$ as low-rank matrices, we explore the weight matrices in self-attention modules with the application of CMA-ES for LoRA in Appendix D and update the weight matrices W_Q and W_K as follows:

$$\begin{aligned} W_Q &= W_Q + B_Q A_Q \\ W_K &= W_K + B_K A_K \end{aligned} \tag{4}$$

This process is performed separately for each model layer, treating the optimization of the low-rank matrices concatenated to the self-attention layer as a subproblem optimization process. Inspired by the divide-and-conquer approach, we employ a gradient-free optimization strategy for the introduced parameters across the entire model.

$$\boldsymbol{m}_{\boldsymbol{L}}^{\boldsymbol{i}\star} = \underset{\boldsymbol{m}_{\boldsymbol{L}}^{\boldsymbol{i}} \in \mathcal{M}}{\arg\min}\, \mathcal{L}(f(\boldsymbol{G}_{\boldsymbol{L}}^{\boldsymbol{i}} \boldsymbol{m}_{\boldsymbol{L}}^{\boldsymbol{i}}; X), Y) \tag{5}$$

where $\mathcal{M}$ is the search space, $\mathcal{L}$ is the loss function, and $\boldsymbol{m}_{\boldsymbol{L}}^{\boldsymbol{i}\star}$ is the optimal vector after a derivative-free optimization through the model f. A detailed description is shown in Algorithm 1.

The initialization of the modules $\boldsymbol{G}_{\boldsymbol{L}}^{\boldsymbol{1}}$, $\boldsymbol{G}_{\boldsymbol{L}}^{\boldsymbol{2}}$, $\boldsymbol{G}_{\boldsymbol{L}}^{\boldsymbol{3}}$, and $\boldsymbol{G}_{\boldsymbol{L}}^{\boldsymbol{4}}$ plays a crucial role in the performance of the model. We analyze two different initialization methods: random initialization with the normal distribution (e.g., initially set to $\mathcal{N}(0, 0.5)$) and initialization with the distribution of the hidden states at each layer of the language model similar to BBTv2 (Sun et al., 2022a). We show the details in Appendix E. In section 5.4, our findings reveal that random initialization with the normal distribution leads to a slight decline in the performance of the language model and slows down the convergence. Therefore, we initialize the modules $\boldsymbol{G}_{\boldsymbol{L}}^{\boldsymbol{1}}$, $\boldsymbol{G}_{\boldsymbol{L}}^{\boldsymbol{2}}$, $\boldsymbol{G}_{\boldsymbol{L}}^{\boldsymbol{3}}$, and $\boldsymbol{G}_{\boldsymbol{L}}^{\boldsymbol{4}}$ using the distribution of the hidden states at each layer of the language model. This initialization strategy helps maintain the performance and convergence speed of the model, leading to better results.

4 Experiments

This section details the experimental results of several natural language understanding (NLU) tasks. The results demonstrate that our proposed method outperforms the current gradient-based and gradient-free methods on several tasks in a few-shot setting.

4.1 Dataset Statistics

We conduct extensive experiments on seven standard NLU datasets, which cover a range of tasks, including natural language inference, paraphrase identification, sentiment analysis, and topic classification. The detailed statistics of these datasets are shown in Appendix C.

Method	#Params	SST-2 Acc.	Yelpp Acc.	AG's News Acc.	DBPedia Acc.	MRPC F1.	SNLI Acc.	RTE Acc.	Avg.
Gradient-Based Methods									
Model tuning	**355M**	85.49(2.84)	91.82(0.79)	86.36(1.85)	97.98(0.14)	77.35(5.70)	54.64(5.29)	58.60(6.21)	78.88
Adapter Tuning	**2.4M**	83.91(2.90)	90.99(2.86)	86.01(2.18)	97.99(0.07)	69.20(3.58)	57.46(6.63)	48.62(4.74)	76.31
BitFit	**172K**	81.19(6.08)	88.63(6.69)	86.83(0.62)	94.42(0.94)	66.26(6.81)	53.42(10.63)	52.59(5.31)	74.76
LoRA	**786K**	88.49(2.90)	90.21(4.00)	87.09(0.85)	97.86(0.17)	72.14(2.23)	61.03(8.55)	49.22(5.12)	78.01
Prompt Tuning	**50K**	68.23(3.78)	61.02(6.65)	84.81(0.66)	87.75(1.48)	51.61(8.67)	36.13(1.51)	54.69(3.79)	63.46
P-Tuning v2	**1.2M**	64.33(3.05)	92.63(1.39)	83.46(1.01)	97.05(0.41)	68.14(3.89)	36.89(0.79)	50.78(2.28)	70.47
Gradient-Free Methods									
Manual Prompt	**0**	79.82	89.65	76.96	41.33	67.40	31.11	51.62	62.56
In-Context Learning	**0**	79.79(3.06)	85.38(3.92)	62.21(13.46)	34.83(7.59)	45.81(6.67)	47.11(0.63)	60.36(1.56)	59.36
Feature-MLP	**1M**	64.80(1.78)	79.20(2.26)	70.77(0.67)	87.78(0.61)	68.40(0.86)	42.01(0.33)	53.43(1.57)	66.63
Feature-BiLSTM	**17M**	65.95(0.99)	74.68(0.10)	77.28(2.83)	90.37(3.10)	71.55(7.10)	46.02(0.38)	52.17(0.25)	68.29
GAP3	**2.5K**	89.70(2.80)	93.00(2.30)	83.20(3.20)	83.70(2.90)	70.20(4.50)	51.10(4.60)	49.70(1.50)	74.40
BBT	**2.5K**	89.56(0.25)	91.50(0.16)	81.51(0.79)	79.99(2.95)	61.56(4.34)	46.58(1.33)	52.59(2.21)	71.90
BBTv2	**60K**	90.33(1.73)	92.86(0.62)	85.28(0.49)	93.64(0.68)	77.01(4.73)	57.27(2.27)	56.68(3.32)	79.01
C-LoRA	**38K**	90.70(1.30)	93.37(0.44)	85.55(0.57)	93.70(0.88)	**80.12(1.88)**	59.11(1.89)	57.34(2.34)	79.98
F-LoRA	**38K**	**91.56(0.87)**	**94.84(0.57)**	**86.64(0.55)**	**94.95(0.54)**	79.99(1.67)	**61.42(1.47)**	**60.93(1.42)**	**81.48**

Table 1: Performance of gradient-based and gradient-free methods on RoBERTa-large. We report average and standard deviation performance over five different seeds. Bold fonts indicate the best results on derivative-free methods.

Algorithm 1 DFOs for Low-Rank Adaptation

Require: L-layer language model f,
Budget of API calls: $\mathcal{B}$,
Optimizers: $\{\mathcal{O}_i\}_{i=1}^{L}$,
Loss function $\mathcal{L}$

Hidden variable: $\boldsymbol{m_L^i}, i = 1, 2, 3, 4$
Random projections: $\boldsymbol{G_L^i}, i = 1, 2, 3, 4$
Low-rank modules of each layer: A and B
repeat
  for each hidden layer **do**
    Evaluate: $loss = \mathcal{L}(f(\boldsymbol{G_L^i m_L^i}))$
    Update $\boldsymbol{m_L^i}$ by DFOs: $\mathcal{O}_L^i(\boldsymbol{m_L^i}, loss)$
    Update the Low-rank modules A and B through $\boldsymbol{G_L^i m_L^i}$ while keep $\boldsymbol{G_L^i}$ frozen
  end for
until $\mathcal{B}/L$ times f call

4.2 Experimental Settings

Datasets The implementation of our method is based on HuggingFace (Wolf et al., 2020) and Pytorch (Paszke et al., 2019). Considering the incredible power of large language models in a few-shot setting (Brown et al., 2020), we conduct our experiments with the same procedure as Zhang et al. (2021), Gu et al. (2022), and Sun et al. (2022a) to construct the true few-shot learning settings (Perez et al., 2021). We sample n instances for each class $\mathcal{Y}$ from the original training set to form the true few-shot training set $\mathcal{D}_{\text{train}}$ and validation sets $\mathcal{D}_{\text{dev}}$, and ensure that $|\mathcal{D}_{\text{train}}| = |\mathcal{D}_{\text{dev}}|$. In particular, in our experiments, the size of the test sets is significantly larger than that of the training and validation sets.

Hyperparameters We use a default setting training with a population size of 20 and a budget of 6,000 API calls to all the tasks for CMA-ES and a default setting training with a population size of 5 and a budget of 6,000 API calls to all the tasks for FWA. We set the rank r of the low-rank matrix to be 2 or 4. We train our proposed method on one NVIDIA 3090 with 24G of memory and report the accuracy or F1 score for several NLU tasks. For generating random projections, we use normal distributions with standard deviations initialized with the distribution of the hidden states at each layer of the language model.

4.3 Baselines

To ensure a fair comparison, we use RoBERTa-large (Delobelle et al., 2020) as the pre-trained model for both gradient-based and gradient-free methods in our experiments. Specifically, to verify the effectiveness of our method on large language models, we chose GPT2-XL (Radford et al., 2019)

Method	#Params	SST-2 Acc.	Yelpp Acc.	AG's News Acc.	DBPedia Acc.	MRPC F1.	SNLI Acc.	RTE Acc.	Avg.
				GPT2-large					
BBT	**2.5K**	75.53(1.98)	80.75(0.53)	77.63(1.89)	77.46(0.69)	65.56(2.34)	32.28(2.43)	52.44(3.32)	65.95
BBTv2	**60K**	83.72(3.05)	85.46(2.33)	**79.96(0.75)**	91.36(0.73)	75.92(3.42)	35.79(1.47)	55.78(1.42)	72.57
C-LoRA	**38K**	84.86(2.02)	87.75(0.91)	79.46(0.87)	92.23(0.57)	76.20(1.22)	37.08(0.99)	57.40(1.67)	73.56
F-LoRA	**38K**	**85.88(1.51)**	**88.52(0.55)**	79.21(3.01)	**92.44(0.76)**	**76.44(0.67)**	**37.56(0.33)**	**57.88(1.38)**	**74.00**
				GPT2-XL					
BBT	**5K**	78.56(1.46)	82.34(2.46)	78.21(2.46)	79.76(3.65)	68.44(2.74)	33.42(2.63)	54.08(2.49)	67.83
BBTv2	**120K**	85.86(2.45)	86.43(0.53)	79.10(3.20)	92.14(1.35)	76.03(2.22)	35.98(1.12)	55.23(2.47)	72.97
C-LoRA	**76K**	86.96(1.50)	**88.70(0.72)**	79.55(2.20)	92.83(0.55)	77.45(1.67)	38.13(1.22)	**58.65(1.44)**	74.61
F-LoRA	**76K**	**87.33(1.67)**	88.47(1.24)	**79.84(1.88)**	**93.87(0.94)**	**78.09(1.26)**	**38.89(1.34)**	58.45(0.97)	**74.99**

Table 2: Performance of gradient-free methods on GPT2-large. We report average and standard deviation performance over five different seeds. Bold fonts indicate the best results.

as a large language model for gradient-free methods. We compare our proposed method (**C-LoRA**) and FW-LoRA (**F-LoRA**) with several baselines as follows:

Gradient-based methods For Gradient-based methods, we compare with (1) **Model tuning:** The vanilla transformer fine-tuning (Delobelle et al., 2020). (2) **Adapter tuning:** Inserting a small task-specific module between the self-attention module (and the MLP module) and the subsequent residual connection at each Transformer layer (Houlsby et al., 2019). (3) **BitFit:** Tuning the biases of the pre-trained language model in our few-shot settings (Ben Zaken et al., 2022). (4) **LoRA:** Merging the low-rank and trainable matrices with the frozen weights at each layer of the Transformer (Hu et al., 2022). (5) **P-Tuning v2:** Appending trainable continuous prompt vectors at each layer of the Transformer (Liu et al., 2021a). (6) **Prompt tuning:** Appending trainable continuous prompt vectors at embedding layer of the Transformer (Lester et al., 2021).

Gradient-free methods For Gradient-free methods, we compare with (1) **Manual Prompt:** Using the templates and label words to conduct zero-shot evaluation (Gao et al., 2021). (2) **In-context learning:** Selecting up several training samples and concatenating them with the input texts (Brown et al., 2020). (3) **Feature-MLP** and (4) **Feature-BiLSTM:** Training a MLP/BiLSTM classifier on the features extracted by the language model (Peters et al., 2019). (5) **GAP3:** Black-box prompt tuning with genetic algorithm (Zhao et al., 2023). (6) **BBT:** Black-box prompt tuning with CMA-ES algorithm. (7) **BBTv2:** Black-box deep prompt tuning with CMA-ES algorithm.

4.4 Main Results

Overall Comparison on RoBERTa Model. As illustrated in Table 1, we demonstrate the experimental results of our proposed methods and the baselines across seven datasets. We observe that the proposed methods outperform gradient-based and gradient-free methods, exhibiting varying levels of improvement across different NLP tasks in a few-shot settings, on average. Specifically, for gradient-based methods, the proposed method improves the performance compared to Model tuning, Adapter tuning, BitFit, LoRA, and P-Tuning v2 by an average of 2.60, 5.17, 6.72, 3.47, and 11.01 points, respectively, across the seven NLU datasets. Furthermore, compared to these parameter-efficient fine-tuning methods, we introduce fewer parameters, yielding superior model performance. For gradient-free methods, the proposed method improves the performance compared to Manual Prompt, In-Context-Learning, GAP3, BBT, and BBTv2 by an average of 18.92, 22.12, 7.05, 9.58, and 2.47 points, respectively, across the seven NLU datasets. Our proposed methods perform better than gradient-free methods across all datasets (e.g., especially on RTE, MRPC, and SST-2), demonstrating the effectiveness of our methods. It is important to emphasize that the primary distinction between LoRA and our

proposed method lies in the optimization algorithm employed. LoRA utilizes gradient descent (Adam optimizer), whereas our method employs the DFOs algorithm (C-LoRA and F-LoRA). Our experimental results indicate that gradient-based optimization may lead to overfitting on limited training data, while DFOs, with their exploration mechanism, tend to discover more effective solutions.

Overall Comparison on GPTs Models. To evaluate the efficacy of our proposed approach on larger models, we conduct experiments on the GPT2-large and GPT2-XL models, as illustrated in Table 2. We show the performance of the two gradient-free optimization methods across models of varying sizes compared to other baselines. Specifically, for GPT2-large, the proposed method improves the performance compared to BBT and BBTv2 by 8.05 and 1.43 points, respectively, on average. It even outperforms BBTv2 across 6/7 datasets (e.g., SST2, Yelpp, DBPedia, MRPC, SNLI, and RTE). For the larger model GPT2-XL, the proposed method improves the performance by 7.16 and 2.02 points on average, respectively, and even outperforms BBTv2 across all datasets. Moreover, as the parameters of the model continue to increase, it can be found that the proposed method is still effective and performs better than other gradient-free optimization methods when fewer parameters are introduced, demonstrating that the proposed method can generalize to more large language models.

5 Analysis

5.1 Memory Usage and Training Time

Table 3 compares our proposed methods (C-LoRA and F-LoRA) with BBTv2 regarding memory usage and training time on SST2, AG's News, and MRPC datasets. The experiments are conducted with a batch size of 16 and a sequence length 512 in GPT2-XL. To monitor GPU memory usage during training, we use Nvidia-smi. Our proposed methods demonstrate superior performance with reduced GPU memory consumption compared to BBTv2. To ensure a fair comparison of training time, we utilize a single NVIDIA 3090 GPU with 24GB of memory, implementing early stopping if the development accuracy does not improve after 1,500 steps. Our methods exhibit faster convergence than BBTv2, achieving improvements of 8.7 minutes on SST2, 15.4 minutes on AG's News, and 7.7 minutes on MRPC. This indicates that our approach has the potential to be applied to large language models, offering parameter-efficient, memory-efficient, and faster convergence.

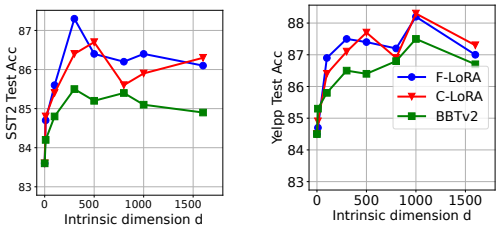

Figure 3: The results of different dimensions on the SST2 and SNLI datasets with GPT2-XL model.

5.2 Effect of Subspace Dimensionality

In exploring the impact of subspace dimensionality on our proposed method, we employ the GPT2-XL model and conduct experiments on the SST2 and SNLI datasets, as illustrated in Figure 3. We explore the dimensionality ranges from 10 to 1600 using BBTv2, C-LoRA, and F-LoRA while maintaining a consistent batch size and population size. As observed in Figure 3, our proposed method consistently outperforms BBTv2 with the increasing subspace dimensionality. Simultaneously, we note that the performance improvement of the model gradually stabilizes when the subspace dimension $d > 500$. Considering that increasing the dimension d may cost much training time for gradient-free algorithms, we keep the range of d between 500 and 1600.

5.3 Effect of Low-Rank r

Considering the impact of the low-rank r on the performance of our proposed method, we conduct experiments on SST2 and Yelpp datasets with the GPT2-XL model to analyze the importance of r. As indicated in Figure 4, we observe that as the low rank r increases, the performance of our two proposed methods gradually decreases on the model. This suggests that when optimizing with gradient-free methods, the model does not require optimization in high dimensions, and achieving good results only requires optimization in a low rank r. In our experiments, we choose $r = 2$ or $r = 4$, allowing the model to achieve good results by introducing very few parameters.

5.4 Effect of Initialization of Module G

In Table 4, we analyze two different initialization methods for module G using DFO (C-LoRA) on Yelpp and RTE datasets: one involves random

Datasets/Methods	SST2			AG's News			MRPC		
	BBTv2	C-LoRA	F-LoRA	BBTv2	C-LoRA	F-LoRA	BBTv2	C-LoRA	F-LoRA
Accuracy (%)	85.86	86.96	87.33	79.10	79.55	79.84	76.03	77.45	78.09
Memory Usage (MB)	12698	12044	12044	17838	17037	17037	13388	12780	12780
Training Time (mins)	20.8	16.4	12.1	35.8	25.7	20.4	22.6	18.5	14.9

Table 3: Comparison of BBTv2, C-LoRA, and F-LoRA on accuracy, memory usage and training time on a single NVIDIA 3090 GPU with 24GB of memory. Batch sizes are 16 and sequence lengths are 512.

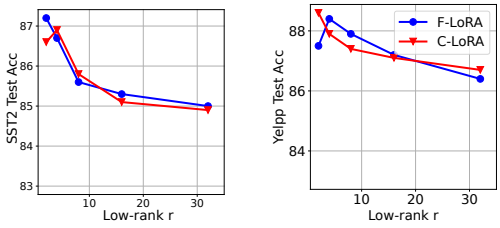

Figure 4: The results of different low-rank r on the SST2 and Yelpp datasets with GPT2-XL model.

Methods	Yelpp (Acc.)	RTE (Acc.)
RI	82.22(2.56)	52.71(3.11)
RI+DFO	87.12(2.41)	57.10(2.69)
RIL	83.42(1.34)	53.01(2.11)
RIL+DFO	88.70(0.72)	58.65(1.44)

Table 4: Test accuracy on Yelpp and RTE with GPT2-XL on different types of initialization of module G. 'RI' denotes random initialization with the normal distribution. 'RIL' denotes random initialization with the distribution of the hidden states at each layer of language model.

initialization with a normal distribution, and the other utilizes the distribution of hidden states at each layer of the language model for initialization. Experimental analysis reveals a significant performance degradation with the random initialization while initializing based on the distribution of the language model's hidden states further mitigates this phenomenon. Additionally, we observe that both the application of DFO (C-LoRA) and these projection matrices yield gains, with DFO making a more pronounced contribution. This suggests the effectiveness of gradient-free optimization methods in optimizing low-rank matrices.

6 Related Work

Gradient-free Optimization of LLMs Gradient-free optimization methods have always been widely used in practice. It shows powerful potential in large language models. Sung et al. (2022) introduced a method that eliminates the need for gradient updates by directly applying a pruned model to downstream tasks. Xiao et al. (2023) and Jin et al. (2023b) propose an efficient transfer learning framework that can adapt large language models to downstream tasks without access to full model parameters. Recently, black-box tuning methods (Sun et al., 2022b,a; Xu et al., 2023; Oh et al., 2023) have employed Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen and Ostermeier, 2001; Hansen et al., 2003) to optimize continuous prompt vectors, bringing substantial benefits to the application of large models with low complexity. However, it is acknowledged that training the introduced prompt vectors is unstable and exhibits slower convergence (Lester et al., 2021; Li and Liang, 2021; Liu et al., 2021b). Therefore, we propose gradient-free optimization for low-rank adaptation to overcome training instability and improve the speed of convergence.

7 Conclusion

In this work, we introduce a novel method for optimizing low-rank modules in large language models in a derivative-free way. The method involves integrating low-rank modules into each self-attention layer of the model and employing two derivative-free optimization methods to optimize these modules at each layer iteratively. Extensive experiments on different tasks and language models show that our proposed method demonstrates superior performance, lower GPU memory usage, and faster model convergence speed compared to existing derivative-free optimization methods in few-shot settings, suggesting that our method presents a promising direction for effectively and economically utilizing LLMs.

Limitations

The proposed method is limited in its applicability to large models where obtaining weights is not feasible, as it requires modifying the specific model structure. Additionally, it is crucial to highlight that our method has only been validated in the context of language understanding tasks. Further exploration and investigation are necessary to assess its effectiveness in generation tasks.

References

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7319–7328, Online. Association for Computational Linguistics.

Fengshuo Bai, Hongming Zhang, Tianyang Tao, Zhiheng Wu, Yanna Wang, and Bo Xu. 2023. Picor: Multi-task deep reinforcement learning with policy correction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(6):6728–6736.

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.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

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

Maiyue Chen and Ying Tan. 2021. Exponentially decaying explosion in fireworks algorithm. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 1406–1413.

Yifan Chen, Devamanyu Hazarika, Mahdi Namazifar, Yang Liu, Di Jin, and Dilek Hakkani-Tur. 2022a. Empowering parameter-efficient transfer learning by recognizing the kernel structure in self-attention. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1375–1388, Seattle, United States. Association for Computational Linguistics.

Yifan Chen, Devamanyu Hazarika, Mahdi Namazifar, Yang Liu, Di Jin, and Dilek Hakkani-Tur. 2022b. Inducer-tuning: Connecting prefix-tuning and adapter-tuning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 793–808, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Pieter Delobelle, Thomas Winters, and Bettina Berendt. 2020. RobBERT: a Dutch RoBERTa-based Language Model. In *Findings of EMNLP*.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah A. Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *ArXiv*, abs/2002.06305.

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

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *ACL*.

Xiaodong Gu, Kang Min Yoo, and Sang-Woo Lee. 2021. Response generation with context-aware prompt learning. *CoRR*, abs/2111.02643.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2022. PPT: Pre-trained prompt tuning for few-shot learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8410–8423, Dublin, Ireland. Association for Computational Linguistics.

Nikolaus Hansen, Sibylle D. Müller, and Petros Koumoutsakos. 2003. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary Computation*, 11:1–18.

Nikolaus Hansen and Andreas Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*.

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 *ICML*.

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

Feihu Jin, Jinliang Lu, Jiajun Zhang, and Chengqing Zong. 2023a. Instance-aware prompt learning for language understanding and generation. *ACM Trans. Asian Low Resour. Lang. Inf. Process.*, 22(7):199:1–199:18.

Feihu Jin, Jiajun Zhang, and Chengqing Zong. 2023b. Parameter-efficient tuning for large language model without calculating its gradients. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 321–330, Singapore. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *EMNLP*.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. 2018. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*.

Junzhi Li and Ying Tan. 2018. Loser-out tournament-based fireworks algorithm for multimodal function optimization. *IEEE Transactions on Evolutionary Computation*, 22(5):679–691.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*.

Yifeng Li and Ying Tan. 2020. Multi-scale collaborative fireworks algorithm. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021a. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *CoRR*, abs/2110.07602.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. Gpt understands, too. *ArXiv*, abs/2103.10385.

Melanie Mitchell. 1998. *An introduction to genetic algorithms*. MIT Press.

Changdae Oh, Hyeji Hwang, Hee young Lee, Yongtaek Lim, Geunyoung Jung, Jiyoung Jung, Hosik Choi, and Kyungwoo Song. 2023. Blackvip: Black-box visual prompting for robust transfer learning. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24224–24235.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. In *Advances in Neural Information Processing Systems*.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14, Florence, Italy. Association for Computational Linguistics.

Hong Qian, Yi-Qi Hu, and Yang Yu. 2016. Derivative-free optimization of high-dimensional non-convex functions by sequential random embeddings. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1946–1952. IJCAI/AAAI Press.

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).

Luis Miguel Rios and Nikolaos V. Sahinidis. 2013. Derivative-free optimization: a review of algorithms and comparison of software implementations. *J. Glob. Optim.*, 56(3):1247–1293.

Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. 2017. Evolution strategies as a scalable alternative to reinforcement learning. *CoRR*, abs/1703.03864.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *EACL*.

Timo Schick and Hinrich Schütze. 2021b. It's not just size that matters: Small language models are also few-shot learners. In *NAACL-HLT*.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

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.

Tianxiang Sun, Zhengfu He, Hong Qian, Yunhua Zhou, Xuanjing Huang, and Xipeng Qiu. 2022a. BBTv2: Towards a gradient-free future with large language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3916–3930, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. 2022b. Black-box tuning for language-model-as-a-service. In *Proceedings of ICML*.

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. LST: Ladder side-tuning for parameter and memory efficient transfer learning. In *Advances in Neural Information Processing Systems*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

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.

Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. 2014. Natural evolution strategies. *J. Mach. Learn. Res.*, 15(1):949–980.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP*.

Zhuofeng Wu, Sinong Wang, Jiatao Gu, Rui Hou, Yuxiao Dong, V. G. Vinod Vydiswaran, and Hao Ma. 2022. IDPG: an instance-dependent prompt generation method. *CoRR*, abs/2204.04497.

Guangxuan Xiao, Ji Lin, and Song Han. 2023. Offsite-tuning: Transfer learning without full model. *CoRR*, abs/2302.04870.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *ArXiv*, abs/2304.12244.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2021. Revisiting few-sample {bert} fine-tuning. In *International Conference on Learning Representations*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015a. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Yuting Zhang, Kihyuk Sohn, Ruben Villegas, Gang Pan, and Honglak Lee. 2015b. Improving object detection with deep convolutional networks via bayesian optimization and structured prediction. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 249–258. IEEE Computer Society.

Jiangjiang Zhao, Zhuoran Wang, and Fangchun Yang. 2023. Genetic prompt search via exploiting language model probabilities. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 5296–5305. ijcai.org.

A Evolution Strategy

A.1 The CMA Evolution Strategy

CMA-ES, short for Covariance Matrix Adaptation Evolution Strategy, is an evolutionary optimization algorithm designed explicitly for tackling optimization problems that are continuous and non-convex. One of the key distinguishing features of CMA-ES is its ability to dynamically adapt the population's covariance matrix, which facilitates efficient exploration and exploitation of the search space. The algorithm maintains a distribution of candidate solutions generated based on a multivariate normal distribution. Through a series of iterations, CMA-ES continuously adjusts this distribution's mean and covariance matrix. This adaptive process enables the algorithm to explore and exploit promising regions within the search space. During each iteration, CMA-ES refines the distribution by replacing less promising solutions with new candidate solutions generated from the current distribution. By dynamically adjusting the covariance matrix, the algorithm can focus on regions that show potential for improved solutions. This iterative process of adaptation and refinement allows CMA-ES to converge towards optimal or near-optimal solutions for the given optimization problem.

A.2 The Fireworks Algorithm

The Fireworks Algorithm (FWA) utilizes a heuristic search approach based on two critical operations: the explosion and selection operations. During the explosion operation, multiple sparks are generated around existing fireworks within specified explosion amplitudes. These sparks serve as potential solutions for the next generation. Subsequently, the fireworks for the new generation are selected from these sparks. We employ the loser-out tournament-based FWA (LoTFWA) to optimize the low-rank modules. In LoTFWA, fireworks compete, and the losers are forced to restart their search from a new location. This competitive mechanism relies on evaluating the fitness of each firework. If a firework's fitness fails to match the best fitness achieved so far, considering its current progress rate, it is considered a loser. The loser is then eliminated and reinitialized, as continuing its search process would be ineffective. This reinitialization step significantly reduces the likelihood of the algorithm becoming trapped in local minima.

B Patterns and Verbalizers

SST-2

- Pattern ["text", "It", "was", "<mask>", "."]
- Verbalizers {"0": "bad", "1": "great"}

Yelpp

- Pattern ["text", "It", "was", "<mask>", "."]
- Verbalizers {"0": "bad", "1": "great"}

AG's News

- Pattern ["<mask>", "News", "text", "."]
- Verbalizers { "0": "World", "1": "Sports", "2": "Business", "3": "Tech"}

DBPedia

- Pattern ["Category: <mask>", "text"]
- Verbalizers { "0": "Company", "1": "Education", "2": "Artist", "3": "Athlete", "4": "Office", "5": "Transportation", "6": "Building", "7": "Natural", "8": "Village", "9": "Animal", "10": "Plant", "11": "Album", "12": "Film", "13": "Written"}

SNLI

- Pattern ["text1", "?", "<mask>", ",", "text2"]
- Verbalizers { "0": "Yes", "1": "Maybe", "2": "No"}

RTE

- Pattern ["text1", "?", "<mask>", ",", "text2"]
- Verbalizers {"0": "Yes", "1": "No"}

MRPC

- Pattern ["text1", "?", "<mask>", ",", "text2"]
- Verbalizers { "0": "No", "1": "Yes"}

C Datasets

Table 5 shows the statistics of datasets used in this work. Specifically, We evaluate our method on AG's News (Zhang et al., 2015a) and DBPedia (Zhang et al., 2015a) for topic classification, RTE (Wang et al., 2018) and SNLI (Bowman et al., 2015) for natural language inference, SST-2 (Socher et al., 2013) and Yelp (Zhang et al., 2015a) for sentiment analysis, and MRPC (Dolan and Brockett, 2005) for semantic paraphrasing.

Datasets	$\|\mathcal{Y}\|$	|Train|	|Test|	Task
			Single-sentence	
SST-2	2	67k	0.9k	Sentiment analysis
Yelpp	2	560k	38k	Sentiment analysis
AG's News	4	120k	7.6k	Topic classification
DBPedia	14	560k	70k	Topic classification
			Sentence-pair	
SNLI	3	549k	9.8k	Natural language inference
RTE	2	2.5k	0.3k	Natural language inference
MRPC	2	3.7k	0.4k	Semantic paraphrasing

Table 5: The datasets evaluated in this work. We sample N $\times|\mathcal{Y}|$ instances from the original training set to form the few-shot training and validation sets.

Weight Type	W_Q	W_K	W_V	W_Q, W_K	W_Q, W_V	W_K, W_V	W_Q, W_K, W_V
Rank r	2	2	2	2	2	2	2
Yelpp (Acc.)	87.64	86.45	85.66	**88.70**	87.34	86.48	86.95
SNLI (Acc.)	37.23	36.67	35.44	**38.13**	37.25	37.22	37.68

Table 6: Test accuracy on Yelpp and SNLI after applying the derivative-free optimized method on different types of weight matrices.

D Applying Gradient-free LoRA to Which Weight Matrices ?

In our investigation of weight matrices in self-attention modules with the application of CMA-ES for LoRA, we have conducted experiments on the GPT2-XL model. Table 6 shows that utilizing the derivative-free optimized LoRA only on the W_Q, W_K, and W_V matrices leads to decreased performance. However, we have observed that simultaneous application of the derivative-free optimized LoRA to the W_Q and W_K matrices yields the best performance. These findings underscore the significance of selecting the weight matrices for derivative-free optimization using the LoRA method. Through exploring various combinations, we can identify the most effective configuration for maximizing performance on the GPT2-XL model.

It is imperative to highlight a noteworthy distinction between our findings and the conventional LoRA approach. Traditionally, LoRA has demonstrated optimal outcomes by concatenating low-rank matrices on the K and V dimensions. However, our novel method, employing a gradient-free optimization approach, exhibits a predilection for achieving superior results by concatenating low-rank matrices on the Q and K dimensions. It is essential to underscore that this empirical observation is derived solely from experimental results and currently lacks a comprehensive theoretical analysis. We defer the in-depth examination of this intriguing phenomenon to future research endeavors.

E How to Initialize the Module G

Inspired by BBTv2, we first set the $\mu = 0$. Then we initialize module G with a normal distribution using the standard deviation as follows:

$$\sigma_m = \frac{\alpha\hat{\sigma}}{\sqrt{d}\sigma_z} \tag{6}$$

where $\hat{\sigma}$ represents the observed standard deviation of hidden states, σ_z denotes the standard deviation of the normal distribution maintained by DFOs, and α is a constant scalar used to stretch the distribution.

F More Related Work

Efficient Few-shot Learners Peters et al. (2019) and Dodge et al. (2020) show that fine-tuning all parameters of language models in few-shot settings can be sub-optimal. As an efficient alternative, parameter-efficient tuning is a promising way of stimulating LLMs. We list mainstream parameter-efficient tuning models as follows: a) Adapter tuning methods learn the task-specific information

(Houlsby et al., 2019) by inserting small-scale task-specific modules within layers of Transformer and only tune the added adapters and layer normalization for model adaptation. b) Prefix tuning methods (Li and Liang, 2021; Liu et al., 2021a) also introduce additional prompt vectors within layers of Transformer to learn task-specific information. Only the parameters of trainable prompt vectors are updated during training while keeping the model parameters frozen. Recently, several prompt tuning methods learn the instance-dependent information (Jin et al., 2023a; Gu et al., 2021; Wu et al., 2022) by inserting the instance-dependent trainable continuous tokens to the input or hidden states of each Transformer layer. c) BitFit (Ben Zaken et al., 2022), a simple but effective method, only optimizes the bias terms inside the model while keeping other parameters frozen. d) LoRA (Hu et al., 2022) merges the low-rank and trainable matrices with the frozen weights at each layer of the Transformer. These parameter-efficient tuning methods still need gradient computation and backpropagation. However, our proposed method utilizes the gradient-free methods and optimizes the model in a memory-efficient and parameter-efficient way.