AutoLoRA: Automatically Tuning Matrix Ranks in Low-Rank Adaptation Based on Meta Learning

Ruiyi Zhang[1*], **Rushi Qiang**[2*], **Sai Ashish Somayajula**[1], **Pengtao Xie**[1]
[1]UC San Diego [2]Tsinghua University
{ruz048, p1xie}@ucsd.edu

Abstract

Large-scale pretraining followed by task-specific finetuning has achieved great success in various NLP tasks. Since finetuning all parameters of large pretrained models poses substantial computational and memory challenges, several efficient finetuning methods have been developed. Among them, low-rank adaptation (LoRA), which finetunes low-rank incremental update matrices on top of frozen pretrained weights, has proven particularly effective. Nonetheless, LoRA's uniform rank assignment across all layers, along with its reliance on an exhaustive search to find the best rank, leads to high computation costs and suboptimal finetuning performance. To address these limitations, we introduce AutoLoRA, a meta learning based framework for automatically identifying the optimal rank of each LoRA layer. AutoLoRA associates each rank-1 matrix in a low-rank update matrix with a selection variable, which determines whether the rank-1 matrix should be discarded. A meta learning based method is developed to learn these selection variables. The optimal rank is determined by thresholding the values of these variables. Our comprehensive experiments on natural language understanding, generation, and sequence labeling demonstrate the effectiveness of AutoLoRA. The code is publicly available at https://github.com/ruz048/AutoLoRA

1 Introduction

Large Language Models (LLMs) (Radford et al., 2019; Brown et al., 2020) have demonstrated state-of-the-art performance across a variety of NLP tasks, spanning from Natural Language Understanding (NLU) (Wang et al., 2018) to Natural Language Generation (NLG) (Radev et al., 2020), a trajectory highlighted by the success of models like ChatGPT (OpenAI, 2023). Their success largely stems from a two-stage process: initial pretraining on vast amounts of unlabeled texts, followed by finetuning on specific downstream tasks. However, as models scale up, for instance transitioning from RoBERTa-large's 355 million parameters (Liu et al., 2019) to GPT-3's staggering 175 billion parameters (Brown et al., 2020), finetuning becomes highly expensive in computation.

To address this challenge, many efficient finetuning methods (Houlsby et al., 2019) have been developed. For instance, the Adapters method (Houlsby et al., 2019) inserts lightweight layers (called adapters) into pretrained networks. During finetuning, only these adapters are updated while the pretrained layers are kept frozen. One limitation of this method is that the adapters incur additional computation overhead during inference. Another approach, prefix tuning (Lester et al., 2021), introduces trainable prefix parameters which are prepended to the input sequence while making the pretrained model parameters frozen. Nevertheless, determining the optimal length of the prefix can be tricky. A prefix that is too short cannot capture enough information, while an overlong prefix may largely reduce the maximum length of the input sequence. To address these limitations, LoRA (Hu et al., 2022) proposes to add low-rank incremental update matrices to pretrained weight matrices. During finetuning, only the incremental matrices are trained while the pretrained ones are frozen. The low-rank parameterization significantly reduces the number of finetuning parameters.

While achieving parameter-efficient finetuning without increasing inference costs, LoRA has two limitations. First, the update matrices at different layers share the same rank, without considering the varying properties across layers. Different layers in a pretrained model have varying importance to a downstream task and should be adapted differently, which requires the number of trainable parameters to be layer-specific. Em-

*Equal contribution.

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 5048–5060

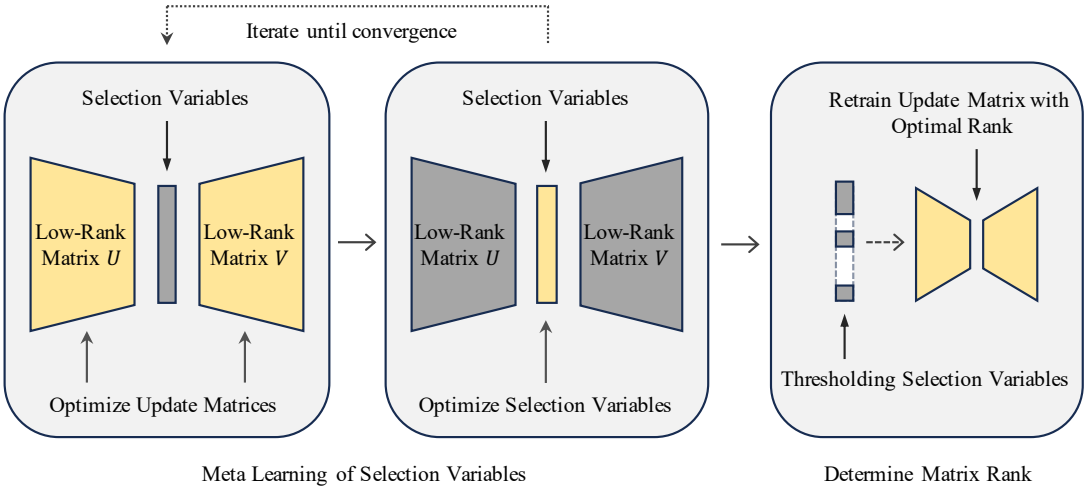

Figure 1: An overview of AutoLoRA. In the meta learning process, AutoLoRA learns selection variables with two iterative steps. Firstly, the weights in the update matrices are optimized on the training dataset. Secondly, the selection variables are updated on the validation dataset. These two steps are iterated until convergence is achieved. Upon acquiring the optimal values of the selection variables, AutoLoRA determines the optimal matrix ranks by thresholding these values. Subsequently, the ranks of update matrices in the LoRA layers are set to the learned optimal ranks and retrained on the combination of training and validation data.

ploying a uniform rank across all layers compromises this purpose, which renders some layers to be under-parameterized (leading to suboptimal finetuning performance) while others unnecessarily over-parameterized (leading to computation inefficiency). Second, obtaining the optimal rank in LoRA typically involves an extensive manual hyperparameter search, which is time-consuming and poses scalability issues.

To address the aforementioned limitations of LoRA, we introduce the AutoLoRA framework to automatically determine the optimal rank for each LoRA layer. In AutoLoRA, we first decompose an update matrix into the product of two low-rank matrices (with rank k), in alignment with the LoRA methodology. This product can be expressed as the summation of k rank-1 matrices. For each rank-1 matrix, we assign a continuous trainable selection variable $\alpha \in [0, 1]$ indicating the matrix's relative importance in the summation. After learning, if α is close to zero, the corresponding rank-1 matrix is removed from the summation. These selection variables effectively control the rank of an update matrix. Learning α directly on a training dataset together with the update matrices can result in overfitting, and the network learned in this way lacks generalization ability. To mitigate this problem, we formulate the search process of α as a meta learning (Finn et al., 2017) problem. First, we finetune the weights in the rank-1 matrices on a training dataset. Second, we optimize the α values by minimizing the loss on a validation dataset. These two steps iterate until convergence. Subsequently, we derive the optimal rank of each LoRA layer by thresholding the learned α values. Once the optimal rank is identified for each layer, the weights in the low-rank update matrices are retrained on the combination of training and validation data. An overview of our proposed method is illustrated in Figure 1.

The major contributions of this paper are summarized as follows.

- We propose AutoLoRA, a meta learning based approach that can automatically determine the optimal and layer-specific ranks of update matrices, alleviating the burden of manually tuning them as in LoRA.
- Extensive experiments on natural language understanding and generation tasks demonstrate the effectiveness of AutoLoRA.

2 Related Works

2.1 Parameter Efficient Finetuning Methods

Various methods have been developed for efficiently finetuning pretrained models. These methods update only a small subset of the weights in large pretrained models, leaving the majority of the parameters frozen. According to Aghajanyan et al. (2021), weight matrices in large pretrained models tend to have a small intrinsic dimension, offering

theoretical intuitions for finetuning pretrained models with low-dimensional reparameterization. Impressively, these methods can sometimes surpass the performance of full finetuning, particularly in downstream tasks with limited training data.

Some efficient finetuning methods finetune the pretrained model by updating trainable prompts while leaving its pretrained parameters frozen. For example, Prompt-tuning (Lester et al., 2021) learns "soft prompts" for language models to perform specific downstream tasks. Prefix-tuning (Li and Liang, 2021) optimizes a sequence of continuous task-specific vectors for natural language generation tasks. P-tuning (Liu et al., 2023) optimizes a small neural network which generates continuous prompt embeddings to finetune GPT models for natural language understanding tasks. LLaMA-Adapter (Zhang et al., 2023b) learns trainable prompts for the LLaMA (Touvron et al., 2023a) model. However, selecting appropriate prompt length can be challenging, as short prompts cannot capture sufficient information while overlong prompts significantly reduce the input sequences' length.

Another line of research involves finetuning the pretrained model by inserting trainable modules into the model while keeping pretrained parameters frozen. For example, Adapter (Houlsby et al., 2019) proposes to inject additional trainable adapter layers into pretrained Transformer (Vaswani et al., 2017) models. IA3 (Liu et al., 2022) multiplies the output of activation functions in the pretrained model with trainable vectors. Compacter (mahabadi et al., 2021) inserts hypercomplex multiplication layers (Zhang et al., 2021) to the pretrained model, offering more efficiency than those in Adapters. These methods incur additional inference overhead due to computing the inserted modules.

AdaLoRA (Zhang et al., 2023a) aims to overcome the problem that LoRA evenly distributes the budget of updates across all LoRA layers by adaptively allocating the budget according to their importance scores. However, since both the importance score and update matrices are learned on the same training dataset, there is an increased risk of overfitting.

2.2 Meta Learning

Various meta learning methods have been proposed for better adaptation of models to new tasks with minimal training data. For instance, Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017) is a gradient based meta learning method, aiming to train model weights for fast adaptation to new tasks with small amounts of data in a few gradient descent steps. Meta-SGD is an extension of MAML (Li et al., 2018). It not only learns model weights, but also optimizes learning rates for fast adaptation to new tasks. Reptile (Nichol et al., 2018) is a first-order meta learning algorithm, which serves as a simpler alternative to MAML. Reptile repeatedly moves the initialization of meta parameters towards the model weights trained on a specific task, sidestepping second-order gradient computation. Orthogonal to these previous methods, our meta learning based method is used for tuning matrix ranks in LoRA.

3 Preliminaries

In LoRA (Hu et al., 2022), a weight matrix $W_l \in \mathbb{R}^{m_l \times n_l}$ at layer l in a downstream model is parameterized as $W_l = \widetilde{W}_l + \Delta_l$, where $\widetilde{W}_l$ is the weight matrix at layer l in a pretrained model and Δ_l is an incremental update matrix. Δ_l is parameterized as the product of two low-rank matrices: $\Delta_l = U_l V_l$, where $U_l \in \mathbb{R}^{m_l \times k_l}$ and $V_l \in \mathbb{R}^{k_l \times n_l}$. k_l, which is much smaller than m_l and n_l, is the rank of Δ_l. Equivalently, Δ_l can be written as the summation of k_l rank-1 matrices:

$$\Delta_l = \sum_{j=1}^{k_l} \Delta_l^j, \tag{1}$$

where Δ_l^j is the outer-product between the j-th column of U_l and the j-th row of V_l.

4 Method

4.1 Overview

In AutoLoRA, we aim to automatically determine the rank k_l in Eq.(1), instead of manually specifying it as in LoRA. To achieve this goal, we associate each rank-1 matrix in an update matrix with a selection variable and reparameterize the update matrix as a weighted sum of rank-1 matrices. A meta learning based approach is developed to learn these selection variables. After learning, if the value of a selection variable is close to zero, its corresponding rank-1 matrix is removed. In this way, we can determine the optimal rank for each update matrix based on the selection variables. An overview of the AutoLoRA algorithm is shown in Algorithm 1.

Algorithm 1 AutoLoRA: Automatically Search for Optimal Rank

Initialize selection variables $A = \{\alpha_l\}_{1 \le l \le M}$ and update matrices $\{\Delta_l = \sum_{j=1}^{k_l} \alpha_l^j \Delta_l^j\}_{1 \le l \le M}$
while *not converged* **do**
1. Update weight parameters W by descending $\nabla_W \mathcal{L}_{tr}(W, D_{tr})$.
2. Update selection variables A by descending $\nabla_A \mathcal{L}_{val}(W - \eta \nabla_W \mathcal{L}_{tr}(W, D_{tr}), D_{val})$.

end while
Derive the best rank $k_l^* = |\{\alpha_{l,j}^* | 1 \le j \le k_l, \alpha_{l,j}^* \ge \lambda\}|$ for LoRA layer l from optimal values α_l^*.

4.2 Reparameterize Update Matrices

We associate each rank-1 matrix Δ_l^j in Eq.(1) with a selection variable $\alpha_l^j \in [0, 1]$ and reparameterize Δ_l as a weighted sum of rank-1 matrices:

$$\Delta_l = \sum_{j=1}^{k_l} \alpha_l^j \Delta_l^j. \tag{2}$$

α_l^j can be interpreted as the importance of Δ_l^j. If α_l^j is close to 0, Δ_l^j will be removed from Δ_l, which effectively reduces the rank of Δ_l by one. In other words, the rank of Δ_l is equivalent to the number of non-zero values in $\{\alpha_l^j\}_{j=1}^{k_l}$. By learning these selection variables based on their fitness to data, we can automatically determine the rank of Δ_l. We add a constraint that the sum of $\{\alpha_l^j\}_{j=1}^{k_l}$ is equal to one: $\sum_{j=1}^{k_l} \alpha_l^j = 1$. This constraint renders the optimization of $\{\alpha_l^j\}_{j=1}^{k_l}$ difficult. To address this problem, instead of optimizing $\{\alpha_l^j\}_{j=1}^{k_l}$ directly, we parameterize them using softmax:

$$\alpha_l^j = \frac{\exp(\beta_l^j)}{\sum_{i=1}^{k_l} \exp(\beta_l^i)}, \tag{3}$$

and learn the unconstrained variables $\{\beta_l^j\}_{j=1}^{k_l}$.

4.3 Learn Selection Variables

Let $A = \{\alpha_l^j | 1 \le j \le k_l, 1 \le l \le M\}$ denote all selection variables, where M is the number of layers in the pretrained model. We propose a meta learning based approach to learn A. Let $\mathcal{L}_{tr}$ denote the downstream task's training loss defined on a training dataset D_{tr}. Given the weight parameters $W_l = \widetilde{W}_l + \Delta_l$ at layer l in the downstream model, we first perform a one-step gradient descent update of W_l:

$$\widehat{W}_l = W_l - \eta \nabla_{W_l} \mathcal{L}_{tr}(\{W_l\}_{l=1}^M, D_{tr}), \tag{4}$$

where η is a learning rate. Then we evaluate $\{\widehat{W}_l\}_{l=1}^M$ on a validation dataset D_{val}. The validation loss $\mathcal{L}_{val}(\{\widehat{W}_l\}_{l=1}^M, D_{val})$ is a function of A since $\mathcal{L}_{val}$ depends on $\{\widehat{W}_l\}_{l=1}^M$ which depends on A. We optimize A by minimizing the validation loss:

$$\min_A \mathcal{L}_{val}(\{\widehat{W}_l\}_{l=1}^M, D_{val}). \tag{5}$$

We use an approximate gradient-based algorithm (Choe et al., 2023) to solve this problem. The updates of W and A in Eq.(4) and Eq.(5) are iteratively performed until convergence.

4.4 Determine Matrix Rank

Given the optimally learned selection variables A^*, we determine the rank of each update matrix based on A^*. For each layer l, we count the number of entries in $\{\alpha_l^j\}_{j=1}^{k_l}$ that satisfy $\alpha_l^j \ge \lambda$, where λ denotes a threshold. This number would be the optimal rank for Δ_l. We set λ to be $\lambda = 1/k_l$. This threshold guarantees the automatically determined rank is at least one.

4.5 Retrain Update Matrices

The thresholding operations in Section 4.4 incurs a discrepancy: when training the update matrices in Section 4.3, all rank-1 matrices are used to make predictions; however, after thresholding, some rank-1 matrices are dropped, which may hurt performance. To bridge this discrepancy, we retrain the update matrices. Specifically, for each update matrix, we set its rank to be the optimal value determined in Section 4.4, then train them by minimizing the finetuning loss on the combination of training and validation datasets.

5 Experiments

5.1 Experimental Setup

The baseline methods used in this work include Adapter (Houlsby et al., 2019), LoRA (Hu et al., 2022), and AdaLoRA (Zhang et al., 2023a).

We examine the efficacy of AutoLoRA by finetuning a RoBERTa-base model (Liu et al., 2019), a RoBERTa-large model, and a GPT2-medium model (Radford et al., 2019) on natural language

Method	Params	CoLA	SST-2	MRPC	QQP	MNLI	QNLI	RTE	STS-B	Avg.
Full FT	125.0M	61.6	94.8	89.3	90.3	86.7	92.8	76.9	91.2	85.5
Adapter	0.9M	58.8	94.0	88.4	89.1	86.5	92.5	71.2	89.9	83.8
LoRA	0.3M	59.0	94.5	89.1	89.6	86.9	92.9	75.8	**91.1**	84.9
AdaLoRA	0.3M	58.8	94.0	**89.4**	89.9	**87.0**	**93.0**	75.9	90.6	85.0
AutoLoRA	0.3M	**61.3**	**94.9**	**89.4**	**90.3**	**87.0**	92.9	**77.0**	90.8	**85.5**

Table 1: Performance and the number of parameters (Params) of AutoLoRA and baseline methods for finetuning RoBERTa-base on the GLUE benchmark. Higher value is better for all metrics. The best results are shown in **bold**. We also provide the performance of full finetuning (Full FT) for reference.

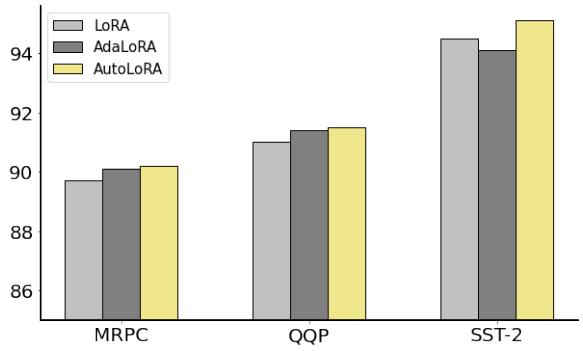

Figure 2: Results of finetuning the RoBERTa-large model on the MRPC, QQP, and SST-2 datasets. Y-axis represents accuracy on GLUE development sets.

understanding (NLU), natural language generation (NLG), and sequence labeling datasets. We include detailed comparison of these two pretrained models in Appendix B.

A Transformer (Vaswani et al., 2017) model consists of several stacked Transformer blocks (layers), and each block contains a multi-head attention (MHA) module and a fully-connected neural network. Each head in an MHA module includes a query projection layer, a key projection layer, and a value projection layer. In adherence to the standard setting in LoRA, we select only the query and value projection layers as trainable LoRA layers, leaving other layers frozen. Both RoBERTa-base and GPT2-medium possess 12 Transformer layers, which results in 24 trainable LoRA layers. The RoBERTa-large model, with 24 Transformer layers, has 48 trainable LoRA layers.

We set the initial dimension of selection variables α_l to be 8 at each layer, i.e., $k_l = 8$. The rank for each layer in LoRA baselines is set as 4, resulting in a similar number of trainable parameters as that in AutoLoRA. We use AdamW (Loshchilov and Hutter, 2019) as the optimizer for both AutoLoRA and baseline methods. We set the batch size as 16 for NLU and NLG tasks, and 32 for the sequence labeling task. We set the learning rate for optimizing weight parameters W in Eq.(4) to be $1e-4$, and the learning rate for optimizing selection variables A in Eq.(5) to be $1e-3$. All experiments were conducted on NVIDIA A100 GPUs. Our implementation is based on Pytorch (Paszke et al., 2019), HuggingFace Transformers (Wolf et al., 2020), and the Betty library (Choe et al., 2023).

5.2 Experiments on Natural Language Understanding Tasks

We conduct extensive experiments on eight datasets from the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018) to evaluate the performance of AutoLoRA on NLU tasks. The GLUE benchmark contains single sentence classification, sentence pair classification, and regression tasks for language acceptability evaluation, sentiment analysis, sentence similarity measurement, and natural language inference. We use accuracy as the evaluation metrics for the SST-2, MRPC, QQP, MNLI, QNLI, and RTE tasks. We use Matthew's correlation for the CoLA task and Spearman's correlation for the STS-B task. Since the test sets of the GLUE benchmark are not publicly available, following previous studies (Zhang et al., 2022), we use the AutoLoRA framework to finetune a RoBERTa-base model on the GLUE training set and evaluate it on the GLUE development set. We split the original training set into a new training set and a validation set with a ratio of 1:1, which are used as D_{tr} and D_{val} in Eq.(4) and Eq.(5) respectively. Please note that baselines methods are trained on the original training set and our method does not unfairly use more data than baselines.

Table 1 shows the performance of AutoLoRA

Method	Param	E2E					WebNLG		
		BLEU	NIST	MET	ROUGE-L	CIDEr	BLEU	MET	TER
Full FT	354.9M	68.0	8.61	46.1	69.0	2.38	46.5	38.0	0.53
Adapter	11.1M	67.0	8.50	45.2	66.9	2.31	50.2	38.0	0.46
LoRA	0.3M	67.1	8.54	45.7	68.0	2.33	50.7	39.5	0.46
AdaLoRA	0.3M	67.0	8.55	45.5	68.1	2.32	50.6	39.4	**0.44**
AutoLoRA	0.3M	**67.9**	**8.68**	**46.0**	**68.9**	**2.37**	**50.8**	**39.6**	**0.44**

Table 2: Performance and the number of parameters of AutoLoRA and baseline methods for finetuning GPT-medium on the E2E and WebNLG datasets. Higher value is better for all metrics except TER. Lower TER value indicates better performance.

on the GLUE development sets, compared with baseline methods. AutoLoRA achieves the best performance on 6 out of 8 datasets, and obtains an average performance of 85.5, outperforming all baseline methods. As AutoLoRA outperforms LoRA on average, we can conclude that the optimal ranks learned by AutoLoRA are better than the manually tuned ranks in LoRA. The reasons are two-fold. First, AutoLoRA allows different layers to have distinct ranks, sufficiently accounting for the fact that different layers have varying properties and need to have layer-specific amounts of tunable parameters. In contrast, LoRA uniformly uses the same rank for all layers, without considering the difference across layers. Second, AutoLoRA learns the continuous selection variables (which determine the ranks) by maximizing the finetuning performance on validation data via gradient descent. The search space is continuous, which allows more comprehensive exploration of rank configurations. In contrast, LoRA performs manual tuning of ranks in a discrete space, where the number of rank configurations is relatively limited.

Furthermore, AutoLoRA outperforms the AdaLoRA baseline on average. The reason is that AdaLoRA uses a single dataset to simultaneously learn rank-1 matrices and their importance scores, which can easily lead to overfitting. In contrast, our method splits the training dataset into two disjoint sets, learns rank-1 matrices on one set, and optimizes selection variables on the other set, which is more resilient to overfitting.

In addition, we present the results of fully finetuning a RoBERTa-base model. Results indicate that AutoLoRA attains performance on par with the full finetuning method, while utilizing significantly fewer parameters.

We further examine the efficacy of AutoLoRA with larger pretrained models. Specifically, we applied AutoLoRA to finetune a RoBERTa-large model (Liu et al., 2019) on the MRPC, QQP, and SST-2 datasets. The RoBERTa-large model comprises 355 million parameters, in contrast to the RoBERTa-base, which only contains 125 million. As shown in Figure 2, the performance of AutoLoRA surpasses both baseline methods across all three datasets, demonstrating AutoLoRA's robust effectiveness in finetuning pretrained models with various sizes.

5.3 Experiments on Natural Language Generation Tasks

In addition to NLU tasks, we also evaluate the effectiveness of AutoLoRA in NLG tasks. The experiments were conducted on two datasets: E2E (Novikova et al., 2017) and WebNLG (Gardent et al., 2017). The E2E dataset contains around 50,000 data-sentence pairs in the restaurant domain. Given the data record of a restaurant, the task is to generate a text description for the restaurant. The WebNLG dataset contains more than 10,000 data-sentence pairs extracted from DBpedia. The data contains triples with a format of (subject, property, object), and the task is to generate a text as a verbalisation of these triples. We use BLEU (Papineni et al., 2002), NIST (Lin and Och, 2004), METEOR (Banerjee and Lavie, 2005), ROUGE-L (Lin and Hovy, 2004), and CIDEr (Vedantam et al., 2015) as evaluation metrics for the E2E dataset. For the WebNLG dataset, we use BLEU, METEOR, and TER (Snover et al., 2006) as evaluation metrics. AutoLoRA was applied to finetune a GPT-medium model.

Table 2 shows the performance of AutoLoRA on the E2E test set and WebNLG test set. AutoLoRA achieves the best performance in terms of all five metrics on the E2E dataset. It outperforms or is on par with baseline methods on the WebNLG dataset

Method	Param	Precision	F1
Full FT	125.0M	70.3	74.9
Adapter	0.9M	66.9	71.3
LoRA	0.3M	68.5	72.2
AdaLoRA	0.3M	69.4	73.0
AutoLoRA	0.3M	**70.1**	**74.2**

Table 3: Performance and the number of parameters of AutoLoRA and baseline methods for finetuning a RoBERTa-base model on the BioNLP dataset.

in terms of all three metrics. This demonstrates AutoLoRA's effectiveness in finetuning pretrained models for NLG tasks. The analysis of reasons that AutoLoRA outperforms LoRA and AdaLoRA is similar to that in Section 5.2. Moreover, the performance of AutoLoRA is on par with that of the full finetuning method, while the number of parameters in AutoLoRA is substantially less.

5.4 Experiments on Sequence Labeling

In this section, we evaluate AutoLoRA on a sequence labeling task. Different from the GLUE tasks which perform classification on an entire sentence (focusing on capturing global semantics), sequence labeling performs classification on each token in a sentence (emphasizing capturing local context). The experiments were conducted on the BioNLP (Collier and Kim, 2004) dataset, which is a Named Entity Recognition dataset containing biological entities such as DNA, RNA, and protein. F1 is used as the evaluation metric. AutoLoRA was applied to finetune a RoBERTa-base model for this task.

Table 3 shows the performance of AutoLoRA on the BioNLP test set, compared with baseline methods. AutoLoRA outperforms all baseline methods in terms of F1 score. The analysis of reasons is similar to that in Section 5.2. In line with our previous findings on NLU and NLG tasks, AutoLoRA can effectively finetune pretrained models for sequence labeling.

5.5 Ablation Studies

In this section, we perform ablation studies to investigate the effectiveness of individual modules in our method. The studies were performed on the GLUE benchmark.

No Constraints. We examine the effectiveness of the sum-to-one constraint in Section 4.2 by removing the constraints from AutoLoRA. Specifically, we directly use a threshold $\lambda = \sum_{j=1}^{k_l} \alpha_l^j / k_l$ to obtain the optimal discrete ranks without any constraints in the meta learning process (AutoLoRA w/o cst.). Results in Table 4 show that AutoLoRA outperforms this ablation setting on average, indicating the effectiveness of this sum-to-one constraint. The reason is that adding such a constraint can make the selection variables better represent the relative importance of rank-1 matrices, which facilitate accurate pruning of less important rank-1 matrices.

Element-wise Sigmoid. We further examine the effectiveness of the sum-to-one constraint in Section 4.2 by comparing AutoLoRA with an ablation setting that applies element-wise sigmoid operations on selection variables. Specifically, for each α_l^j, we use sigmoid to constrain its value into $[0, 1]$ in the meta learning process, and a threshold of 0.5 is used to obtain discrete ranks (AutoLoRA sigmoid). Results in Table 4 show that AutoLoRA outperforms this ablation setting on average. In this ablation setting, α_l^j no longer directly indicates the relative importance of rank-1 matrices, making it challenging to select an appropriate threshold.

Meta Learning. We examine the effectiveness of the meta learning framework by setting $\eta = 0$ in Algorithm 1. This ablation setting can be interpreted as an alternative learning method where two optimization steps are carried out alternatively on two different splits of the training dataset. Results in Table 4 show that AutoLoRA outperforms AutoLoRA ($\eta = 0$) on average, demonstrating the efficacy of the meta learning strategy.

5.6 Qualitative Analysis

Figure 3 presents the optimal rank determined by AutoLoRA for the QQP, MNLI, and E2E datasets. For the QQP and MNLI datasets, we utilized a RoBERTa-base backbone, while a GPT2-medium backbone was employed for the E2E dataset. In this figure, column i corresponds to the i-th Transformer block in the pretrained model. Each row corresponds to a dataset and a layer type (query projection and value projection layer). As can be seen, the optimal ranks learned by AutoLoRA for different layers have varying values. This is aligned with the hypothesis discussed in Section 1 that different layers need different matrix ranks. Vanilla LoRA ignores this difference and uniformly uses the same

Method	CoLA	SST-2	MRPC	QQP	MNLI	QNLI	RTE	STS-B	Avg.
AutoLoRA (w/o cst.)	61.0	93.7	88.5	90.0	**87.2**	92.1	**77.5**	90.5	85.1
AutoLoRA (sigmoid)	59.7	94.1	88.3	89.8	86.9	92.6	75.7	90.7	84.7
AutoLoRA ($\eta = 0$)	61.2	94.8	89.3	90.1	87.1	92.8	77.3	90.5	85.2
AutoLoRA	**61.3**	**94.9**	**89.4**	**90.3**	87.0	**92.9**	77.0	**90.8**	**85.5**

Table 4: Ablation studies. Performance comparison of AutoLoRA, AutoLoRA without constraint (AutoLoRA w/o cst.), AutoLoRA with element-wise sigmoid (AutoLoRA sigmoid), and AutoLoRA with $\eta = 0$. We evaluated all methods on the development sets of the GLUE benchmark.

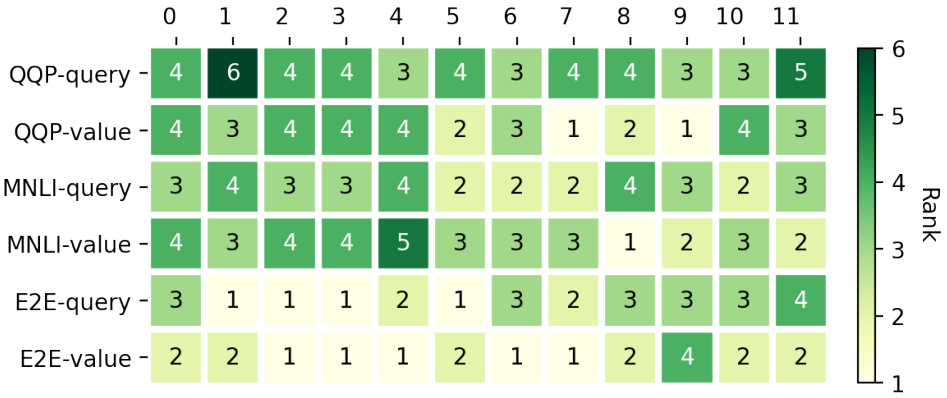

Figure 3: Optimal ranks of the LoRA layers obtained by AutoLoRA on the QQP, MNLI, and E2E datasets. We finetuned RoBERTa-base on QQP and MNLI, and GPT2-medium on E2E. Both RoBERTa-base and GPT2-medium consist of 12 Transformer layers. We only search for the ranks in the query and value projection layers.

Method	AdaLoRA	LoRA+Grid Search	AutoLoRA
Cost	x1	x14.29	x1.91

Table 5: Comparison of average training cost between AutoLoRA and baseline methods on the SST-2, MNLI, and QQP datasets. We normalize the average training time of AdaLoRA as 1.

rank across layers, which leads to inferior performance. Our method provides a computationally efficient mechanism to learn these layer-specific ranks, which takes much less time than grid search (as shown in Section 5.7).

5.7 Computation Costs

Table 5 shows the average training cost of AutoLoRA and two baseline methods on the SST-2, MNLI, and QQP datasets. We normalize the average training time of AdaLoRA as 1 for reference. In LoRA, we use grid search to tune the ranks, with 16 configurations. As can be seen, our method is much more efficient than performing grid search of ranks in LoRA. Grid search is conducted in a discrete space. For each configuration of ranks, LoRA needs to run from scratch, which is very time-consuming. In contrast, our method performs the search in a continuous space via gradient method, which can efficiently explore many configurations without restarting. Compared with AdaLoRA, our method has significantly better performance as shown in Tables 1, 2, and 3, without substantially increasing the computation costs.

6 Discussion

In this section, we discuss the potential issue of overfitting associated with AutoLoRA. In contrast to AdaLoRA (Zhang et al., 2023a), our approach demonstrates enhanced resilience against overfitting by learning rank-1 matrices and selection variables across two separate sub-datasets, thereby reducing the likelihood of overfitting to any single dataset. As shown in Table 1, the robust performance of AutoLoRA on smaller datasets like RTE demonstrates that our method is less prone to overfitting. Nonetheless, if both sub-datasets used are exceedingly small, there exists an increased risk of AutoLoRA overfitting to these datasets.

As shown in Bao et al. (2021), bi-level optimization (BLO) based hyperparameter searching methods, such as AutoLoRA, are more likely to achieve a lower empirical risk on the validation dataset, compared to cross validation based methods. It gives a theoretical proof that the generalization

bound of BLO method is $\mathcal{O}(\frac{C_1}{m})$, while the generalization bound of cross validation is $\mathcal{O}(C_2\sqrt{\frac{1}{m}})$, where m is the size of the validation dataset and C_1, C_2 are other factors in the hyperparameter optimization framework which do not depend on m. As C_1 is much larger than C_2, BLO based methods are more likely to suffer from the overfitting issue when m is relatively small. However, when the size of the validation dataset m is relatively large, BLO based hyperparameter optimization methods like AutoLoRA can better prevent overfitting compared with cross validation.

7 Conclusions and Future Work

In this paper, we introduce AutoLoRA, a meta learning based framework designed to automatically search for the optimal ranks for LoRA layers. Our method associates each rank-1 matrix in LoRA updates with a selection variable and formulates the rank-tuning problem as optimizing the selection variables via meta learning. Thresholding is applied to derive discrete rank values from continuous selection variables and retraining is performed to bridge the gap incurred by thresholding. Comprehensive experiments show the efficacy of AutoLoRA across various NLP tasks.

Similar to the LoRA method, the LoRA layers in AutoLoRA are manually specified, which may be suboptimal. As a future work, we will investigate how to automatically select LoRA layers, by developing a meta learning framework similar to that in Eq.(5).

8 Limitations

In comparison to other rank search techniques like AdaLoRA, our method does introduce some additional computational and memory overhead. However, as shown in Table 5, the increase of training cost is relatively modest. Another limitation is that we did not evaluate our method on more recent large language models (LLMs), such as LLaMA (Touvron et al., 2023a) and LLaMA-2 (Touvron et al., 2023b). It is promising to apply AutoLoRA on these LLMs as they are more powerful compared with previous ones. We did not evaluate our method on LLMs pretrained on non-English texts either. We aim to address these limitations in our future research.

Acknowledgements

We thank the anonymous meta reviewer and reviewers for their valuable suggestions.

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.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Fan Bao, Guoqiang Wu, Chongxuan LI, Jun Zhu, and Bo Zhang. 2021. Stability and generalization of bilevel programming in hyperparameter optimization. In *Advances in Neural Information Processing Systems*, volume 34, pages 4529–4541. Curran Associates, Inc.

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

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

Nigel Collier and Jin-Dong Kim. 2004. Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*, pages 73–78, Geneva, Switzerland. COLING.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of*

the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

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

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

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pages 507–523, Berlin, Heidelberg. Springer Berlin Heidelberg.

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

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

Zhenguo Li, Fuxin Zhou, Fei Chen, and Hang Li. 2018. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*.

Chin-Yew Lin and Eduard Hovy. 2004. Rouge: A package for automatic evaluation of summaries. In *Workshop on Text Summarization Branches Out, Association for Computational Linguistics*.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 605–612, Barcelona, Spain.

Marius Lindauer, Katharina Eggensperger, Matthias Feurer, AndrÃ© Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, RenÃ© Sass, and Frank Hutter. 2022. Smac3: A versatile bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(54):1–9.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *arXiv preprint arXiv:2205.05638*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2023. Gpt understands, too. *AI Open*.

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*, abs/1907.11692.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Dougal Maclaurin, David Duvenaud, and Ryan P. Adams. 2015. Gradient-based hyperparameter optimization through reversible learning. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 2113–2122. JMLR.org.

Rabeeh Karimi mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. In *Advances in Neural Information Processing Systems*.

Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA. Association for Computational Linguistics.

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*. Curran Associates Inc., Red Hook, NY, USA.

Dragomir Radev, Ramesh Narasimhan, Ruochen Tang, Abhinand Sivaprasad, Xiangkai Zhang, Amr Saleh, Neha Krishnaswamy, Balazs Gliwa, Yunyao Qiu, Haoran Tang, Yash Vyas, and Rahul Nallapati. 2020. Dart: Open-domain structured data record to text generation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 7492–7503.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. Translation edit rate: A new metric for machine translation evaluation. In *Proceedings of the Association for Machine Translation in the Americas (AMTA)*.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

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.

S. Watanabe, A. Bansal, and F. Hutter. 2023. PED-ANOVA: Efficiently quantifying hyperparameter importance in arbitrary subspaces. *International Joint Conference on Artificial Intelligence*.

S. Watanabe and F. Hutter. 2023. c-TPE: Tree-structured Parzen estimator with inequality constraints for expensive hyperparameter optimization. *International Joint Conference on Artificial Intelligence*.

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.

Aston Zhang, Yi Tay, SHUAI Zhang, Alvin Chan, Anh Tuan Luu, Siu Hui, and Jie Fu. 2021. Beyond fully-connected layers with quaternions: Parameterization of hypercomplex multiplications with $1/n$ parameters. In *International Conference on Learning Representations*.

Haojie Zhang, Ge Li, Jia Li, Zhongjin Zhang, YUQI ZHU, and Zhi Jin. 2022. Fine-tuning pre-trained language models effectively by optimizing subnetworks adaptively. In *Advances in Neural Information Processing Systems*.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023a. Adaptive budget allocation for parameter-efficient fine-tuning. In *ICLR*.

Renrui Zhang, Jiaming Han, Chris Liu, Peng Gao, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, and Yu Qiao. 2023b. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*.

A Datasets

Table 6 shows the statistics of the GLUE datasets.

B Pretrained Models

RoBERTa pretrains a Transformer encoder, which is the same as that in BERT (Devlin et al., 2019). The GPT2 model pretrains a Transformer decoder. The RoBERTa model is pretrained via masked token prediction. The GPT2 model is pretrained via language modeling. RoBERTa is commonly used for natural language understanding (NLU) tasks while GPT2 is often used for natural language generation (NLG) tasks.

C Hyperparameter Optimization

Adequate hyperparameter configuration is crucial for machine learning algorithms to achieve top performance. Compared with grid search and simple random search, Bayesian Optimization (BO) (Lindauer et al., 2022) and gradient-based hyperparameter optimization (Maclaurin et al., 2015) have been widely used because of their sample efficiency. For example, SMAC (Hutter et al., 2011) builds a probabilistic model to estimate the performance of different hyperparameter configurations. It sequentially chooses the next set of hyperparameters to evaluate, with an predefined acquisition function to balance exploration with exploitation in the hyperparameter space. SMAC3 (Lindauer et al., 2022) improves SMAC by evaluating less promising hyperparameters configurations with fewer instances. c-TPE (Watanabe and Hutter, 2023) proposes a constrained tree-structured Parzen estimator to handle constraints such as memory consumption and inference latency of a configuration of hyperparameters. PED-ANOVA (Watanabe et al., 2023) highlights the role of good hyperparameter search space in hyperparameter optimization. It derives a algorithm to compute hyperparameter importance with Pearson divergence. On the other hand, Maclaurin et al. (2015) computes the gradients with respect to hyperparameters, and proposes an efficient method to store related information.

	CoLA	RTE	QNLI	STS-B	MRPC	WNLI	SST-2	MNLI (m/mm)	QQP
Train	8551	2490	104743	5749	3668	635	67349	392702	363871
Dev	1043	277	5463	1500	408	71	872	9815/9832	40432

Table 6: GLUE dataset statistics.