BIPEFT: Budget-Guided Iterative Search for Parameter Efficient Fine-Tuning of Large Pretrained Language Models

Aofei Chang[1], **Jiaqi Wang**[1], **Han Liu**[2]
Parminder Bhatia[3], **Cao Xiao**[3], **Ting Wang**[4], **Fenglong Ma**[1*]
[1]Pennsylvania State University, [2]Dalian University of Technology
[3]GE Healthcare, [4]Stony Brook University
[1]{aofei, jqwang, fenglong}@psu.edu, [2]hanliu@dlut.edu.cn
[3]{parminder.bhatia, cao.xiao}@gehealthcare.com, [4]twang@cs.stonybrook.edu

Abstract

Parameter Efficient Fine-Tuning (PEFT) offers an efficient solution for fine-tuning large pretrained language models for downstream tasks. However, most PEFT strategies are manually designed, often resulting in suboptimal performance. Recent automatic PEFT approaches aim to address this issue but face challenges such as search space entanglement, inefficiency, and lack of integration between parameter budgets and search processes. To overcome these issues, we introduce a novel **B**udget-guided **I**terative search strategy for automatic **PEFT** (BIPEFT), which significantly enhances search efficiency. BIPEFT employs a new iterative search strategy to disentangle the binary module and rank dimension search spaces. Additionally, we design early selection strategies based on parameter budgets, accelerating the learning process by gradually removing unimportant modules and fixing rank dimensions. Extensive experiments on public benchmarks demonstrate the superior performance of BIPEFT in achieving efficient and effective PEFT for downstream tasks with a low parameter budget.[1]

1 Introduction

Large pre-trained models (PTMs) (Devlin et al., 2019; Radford et al., 2019) based on Transformer architectures (Vaswani et al., 2017) have achieved significant success across a variety of downstream tasks through fine-tuning, including applications of healthcare (Wang et al., 2024; Luo et al., 2024). However, the computational and storage demands of PTMs limit the feasibility of full fine-tuning. To address this, parameter-efficient fine-tuning (PEFT) methods have garnered considerable attention. Existing PEFT approaches have demonstrated superior performance on downstream tasks (Xu et al., 2023). However, most methods, such as adapters (Houlsby et al., 2019; Lin et al., 2020; Rücklé et al., 2021), BitFit (Ben Zaken et al., 2022), and LoRA (Hu et al., 2022a; Zhang et al., 2023b; Zi et al., 2023; Zhang et al., 2023a; Dettmers et al., 2023) require *manual design* of fine-tuning strategies. Configuring PEFT for different layers of Transformers can result in varying performance outcomes.

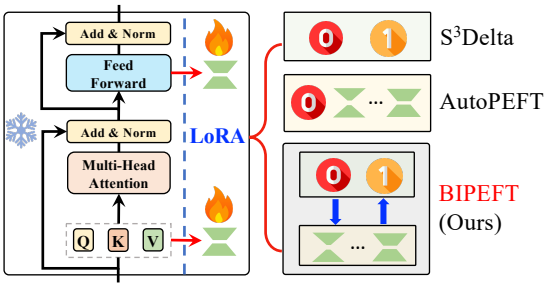


Figure 1: Search space comparison among different automatic PEFT approaches when searching on low-rank adaption (LoRA) (Hu et al., 2022a). S^3Delta (Hu et al., 2022b) uses a $\{0, 1\}$ binary space to determine whether the module is kept. AutoPEFT (Zhou et al., 2024) uses a multi-dimensional space to search module existence and dimension ranks simultaneously, where 0 means the module will be removed. The proposed BIPEFT uses a novel iterative search strategy to disentangle the binary module search and the rank dimension search.

To mitigate this issue, **automatic PEFT** approaches (Hu et al., 2022b; Zhou et al., 2024) have been proposed to automatically search for the optimal PEFT configuration. S^3Delta (Hu et al., 2022b) is the first differential neural architecture search (NAS)-based PEFT approach, which automatically searches for the optimal modules to include in the configuration. AutoPEFT (Zhou et al., 2024) employs Bayesian optimization to conduct the PEFT search across a large space. Although these two automatic approaches are effective, they still suffer from the following issues:

(1) Search Space Entanglement. As shown in Figure 1, the search space of S^3Delta (Hu et al.,

*Corresponding author.
[1]Source code is available at https://github.com/Aofei-Chang/BIPEFT

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 7429–7440
November 12-16, 2024 ©2024 Association for Computational Linguistics

2022b) is limited, focusing only on binary PEFT module searches. In contrast, AutoPEFT (Zhou et al., 2024) combines binary module selection with multiple rank dimension spaces by representing binary module selection with 0 and denoting non-zero values for ranks. Such a mixed search ignores the fact that these two spaces are interdependent and entangled. During the search, a module with a rank of 0 means that the search on this module is not necessary yet. In contrast, a non-zero rank indicates that the module will be kept, and 0 will not be selected further. Consequently, jointly searching within such entangled spaces introduces new optimization challenges, requiring an automatic balance between dimensional choices and binary decisions, especially when the search space is large.

(2) Better Search Efficiency. Using a non-differential optimization strategy makes AutoPEFT (Zhou et al., 2024) less efficient compared to S^3Delta (Hu et al., 2022b). However, even with differential NAS, the efficiency of S^3Delta is still unsatisfactory due to overlooking the unique characteristics of the PEFT task. Unlike traditional automated machine learning tasks that typically learn model parameters and architectures from scratch, automatic PEFT only learns a small set of parameters, with most being fixed. The weight distributions for some modules or dimension ranks may quickly stabilize after a few training steps. Thus, keeping them involved in the PEFT learning until the final training step is unnecessary and lowers search efficiency.

(3) Isolation between Parameter Budgets and Search Efficiency. In practice, a smaller yet effective model is more useful for downstream tasks. S^3Delta (Hu et al., 2022b) uses a parameter budget to control the number of trainable model parameters. However, this budget does not accelerate PEFT optimization. An ideal solution would be to use the parameter budget as a factor to guide the efficient PEFT configuration, potentially yielding a more effective downstream model.

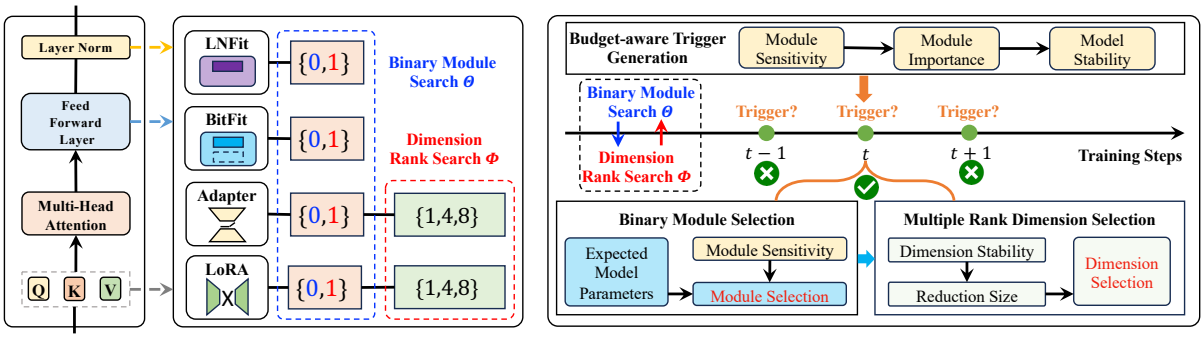
To solve all the aforementioned issues simultaneously, in this paper, we propose a parameter **B**udget-guided **I**terative search strategy `BIPEFT` for boosting the search efficiency of automatic **PEFT**, as shown in Figure 2. `BIPEFT` works as follows: At each training step t, `BIPEFT` uses the designed iterative search strategy to search optimal architecture weights for the binary PEFT module search space and rank dimension search space alternatively, which can handle the first issue. After a few training steps, `BIPEFT` will trigger the selection modules, where the trigger is generated based on the parameter budget $\mathcal{B}$ and the current module state, as detailed in §Sec. 3.2. `BIPEFT` contains a novel parameter budget-guided module selection strategy in §Sec. 3.3 to gradually remove the unimportant modules and a new parameter history-based dimension selection strategy in §Sec. 3.4 to adaptively fix rank dimensions during the search. These two new selection strategies can perfectly address the second and third issues. After selecting modules and fixing rank dimensions, `BIPEFT` will repeat again until the number of triggers achieves the maximum number Z.

To sum up, this work has the following contributions:

- We recognize the importance of disentangling the binary module and rank dimension search spaces for automatic PEFT optimization.
- We introduce a novel automatic PEFT model `BIPEFT`, which is an iterative differential NAS-based approach to effectively search for downstream task models with parameter budget constraints.
- We design early selection strategies for different space searches, which significantly accelerates the optimal PEFT model learning.
- We conduct extensive experiments on two public benchmarks and compare `BIPEFT` against state-of-the-art baselines. Experimental results demonstrate the efficacy and efficiency of `BIPEFT` for automatic PEFT.

2 Related Work

Parameter Efficient Fine-Tuning (PEFT). Generally, PEFT is designed based on a Transformer architecture and only optimizes a small portion of parameters and leaves the vast majority of parameters frozen for efficient adaptation to downstream tasks. The PEFT methods can be broadly classified into four categories (Han et al., 2024): (1) *Additive PEFT* such as Adapter (Houlsby et al., 2019) and Prefix-Tuning (Li and Liang, 2021) inserts new trainable modules or parameters in the model. (2) *Selective PEFT* aims to optimize model performance by selectively fine-tuning a subset of



(a) Fine-tuning a PTM layer using disentanglement search spaces

(b) The proposed BIPEFT model with iterative search and selection strategies

Figure 2: Overview of the proposed BIPEFT, which conducts an iterative search on disentanglement search spaces with novel module and rank dimension selection strategies to accelerate search efficiency.

the model's parameters by masking (Guo et al., 2021; Fu et al., 2023; Liao et al., 2023; Sung et al., 2021) or manual design (Ben Zaken et al., 2022). (3) *Reparameterized PEFT* like LoRA (Hu et al., 2022a), constructs a reparameterization of the original model parameters for training, then equivalently transforms it back at the inference stage (Zhang et al., 2023b; Luo et al., 2023). (4) *Hybrid PEFT* focuses on combining the benefits of diverse PEFT modules from the last three categories (Mao et al., 2021; Chen et al., 2023). A series of automated hybrid PEFT methods are developed to automatically search for an effective hybrid PEFT structure, and our work falls into this category.

Automated Configuration Search for PEFT. AutoPEFT (Zhou et al., 2024) using Bayesian optimization and S^{3}Delta (Hu et al., 2022b) utilizing differential neural architecture search (NAS) techniques to search for optimal architectures for natural language processing (NLP) tasks. In contrast, PrunePEFT (Lawton et al., 2023) adopts a straightforward pruning approach to identify essential PEFT parameters. In the vision domain, NOAH (Zhang et al., 2022b) applies an evolutionary NAS strategy to tailor PEFT configurations specifically for Vision Transformer (Dosovitskiy et al., 2021). Despite these advancements, significant challenges remain in optimizing search space design and improving search efficiency.

Early Stopping in Neural Architecture Search. Existing early stopping strategies in neural architecture search address the overfitting issues on selecting many skip connections in convolutional neural networks (CNNs) when using methods like DARTS (Liu et al., 2019). OLES (Jiang et al., 2023) introduces an operation-level early stopping method to mitigate this issue. DARTS+ (Liang et al., 2019) implements a manual threshold for stopping the search. SGAS-es utilizes an indicator for stabilizing search results on CNNs. Notably, early stopping techniques have not yet been adapted for the PEFT search, which is characterized by faster convergence compared to CNNs.

3 Methodology

3.1 Overview

Our approach aims to automatically search for the optimal PEFT structure from N modules of LLMs through a novel budget-aware search strategy named BIPEFT. This strategy iteratively searches from two distinct spaces: a binary position search space and a dimension search space, with module and dimension selection mechanisms. As shown in Figure 2, at each training step t, BIPEFT will first optimize the architecture weights $\mathbf{\Theta}_t \in \mathbb{R}^{N\times 2}$ by fixing the weights of $\mathbf{\Phi}_{t-1} \in \mathbb{R}^{N\times K}$ for the dimension search space, where K is the number of dimension candidates. Subsequently, BIPEFT updates $\mathbf{\Phi}_t$ using the optimized $\mathbf{\Theta}_t$.

After each training step, BIPEFT checks whether the budget-aware selection mechanisms are triggered according to module sensitivity scores and the targeted budget $\mathcal{B}$ detailed in §Sec. 3.2. If triggered, BIPEFT will first estimate the number of reduced parameters R_z at this trigger stage and then discard unimportant modules according to the estimated R_z with a designed module selection strategy in §Sec. 3.3. BIPEFT also selects an appropriate dimension size for certain modules in §Sec. 3.4. These selection operations lead to further updates $\mathbf{\Theta}_t$ and $\mathbf{\Phi}_t$. BIPEFT stops when the maximum trigger count Z is reached, ensuring that the model size approximates the targeted budget $\mathcal{B}$. The search process of BIPEFT is outlined in Algorithm 1, with the details of the designed strategy discussed in the following subsections.

Algorithm 1: Algorithm of BIPEFT

Input: N PEFT modules with trainable weights $\Delta\mathbf{W}$, architecture parameters $\mathbf{\Theta}_0$ and $\mathbf{\Phi}_0$, stability trigger threshold τ, parameter budget $\mathcal{B}$, total steps T, maximum trigger count Z, binary selection indicator $\mathbf{b}_z$, dimension selection indicator $\mathbf{d}_z$

Output: Searched PEFT architecture A

for $t = 1, \ldots, T$ **do**
- Update $\Delta\mathbf{W}$ by gradient descent;
- Optimize $\mathbf{\Theta}_{t-1}$ and $\mathbf{\Phi}_{t-1}$ iteratively according to the objective;
- Accumulate the sensitivity score $\bar{s}^t$ of each PEFT module using Eq. (4);
- Calculate module importance indicator $\mathbf{I}_t$;
- Evaluate model stability β_t using Eq. (5);
- **if** $(\beta_t \geq \tau)$ **then**
  - Estimate expected parameters E_t using Eq. (7);
  - Calculate expected parameter reduction R_z using Eq. (6);
  - Rank N modules by sensitivity score $\bar{s}^t$;
  - Perform early module selection until the reduction R_z is achieved;
  - Update binary selection indicator $\mathbf{b}_z$;
  - Evaluate dimension stability λ_z^n using Eq. (9);
  - Estimate the number of modules Y_z for dimension selection using Eq. (10);
  - Fix dimensions for Y_z modules with the lowest stability scores λ_z^n;
  - Update dimension selection indicator $\mathbf{d}_z$;
  - Freeze part of the $\mathbf{\Theta}_{t-1}$ and $\mathbf{\Phi}_{t-1}$ according to selection indicators $\mathbf{b}_z$, $\mathbf{d}_z$;
  - $Z \leftarrow Z - 1$;
- **if** $Z \leq 0$ **then**
  - Exit;

return A

3.2 Budget-aware Trigger Generation

A naive solution of automatically searching for the optimal PEFT structure is to gradually reduce the number of parameters during model fine-tuning. However, if the fine-tuned model is unstable, we cannot decide which modules will be pruned. Thus, evaluating model stability is important. To achieve this goal, we design a new budget-aware model stability strategy according to module-level sensitive scores and the targeted budget $\mathcal{B}$.

3.2.1 Module Sensitivity Estimation

In our setting, we apply the differential neural architecture search (NAS) for optimal PEFT configuration. If a module $\mathcal{M}_n$ plays an important role in the PEFT model, it should be more stable and contribute greatly to the loss function. In the NAS-based model training, we have both training and validation data, denoted as $\mathcal{D}_{tra}$ and $\mathcal{D}_{val}$. The module sensitivity can be evaluated on these two kinds of data as follows:

$$s_n^t = f_n^t(\mathcal{D}_{tra}) + \alpha_n^t f_n^t(\mathcal{D}_{val}), \tag{1}$$

$$f_n^t(\mathcal{D}) = \frac{1}{|\mathcal{M}_n|} \sum_{w \in \mathcal{M}_n} \left| w\mathbf{G}_n^t(w, \mathcal{D}) \right|, \tag{2}$$

$$\alpha_n^t = \cos(\mathbf{G}_n^t(\mathcal{D}_{tra}), \mathbf{G}_n^t(\mathcal{D}_{val})). \tag{3}$$

Following (Molchanov et al., 2019), we use f_n^t to calculate the parameter-level average magnitude of the gradient-weight product. $|\mathcal{M}_n|$ is the number of fine-tuned parameters of the n-th module. $\mathbf{G}_n^t(w)$ denotes the gradient of weight w. α_n^t denotes the gradient cosine similarity of the module $\mathcal{M}_n$ on both training and validation data.

The sensitivity score s_n^t of each module can be measured after each training step t. To mitigate the impact of stochastic data batch sampling, we propose to smooth the sensitivity score using an exponential moving average following (Zhang et al., 2022a) as follows:

$$\bar{s}_n^t = \gamma \bar{s}_n^{t-1} + (1 - \gamma) s_n^t, \tag{4}$$

where γ is a predefined hyperparameter.

3.2.2 Trigger Generation

Determining the optimal timing for starting parameter reduction is pivotal. To address this challenge, we propose an adaptive trigger generation approach to automatically estimate the optimal timing according to module sensitivity scores learned by Eq. (4) along with the targeted budget $\mathcal{B}$.

Module Importance Indicator. Ideally, the finally selected modules should be greatly yet continuously contributed to the model fine-tuning. Moreover, the total parameters of the finally selected modules should be close to the targeted budget $\mathcal{B}$ in our setting.

Based on these motivations, we design a module importance indicator $\mathbf{I}_t \in \{0, 1\}^N$ for each training step t to record the estimated importance of modules. Specifically, we initialize $\mathbf{I}_t = \mathbf{0}$ and rank the module sensitivity scores in descending order. We accumulate the parameters from top-ranked modules one by one. If the current sum is smaller than the targeted budget $\mathcal{B}$, we set the indicator value as 1 for that module. Otherwise, all the remaining indicator values are 0.

Trigger. Intuitively, the model performs stably if the important modules change slightly within a time window H. We use an average cosine similarity between two consecutive module importance

indicators within H steps to evaluate the model stability as follows:

$$\beta_t = \frac{1}{H} \sum_{j=t-H}^{t} \cos(\mathbf{I}_j, \mathbf{I}_{j-1}), \tag{5}$$

where $\beta_t \geq \tau$ means that the model is stable now, and the trigger can be generated, where τ is a hyperparameter. After triggering the parameter reduction, BIPEFT then uses two strategies to reduce the number of training parameters, i.e., binary module selection and multiple dimension selection.

3.3 Binary Module Selection

The majority of parameters will be reduced in the binary module selection stage. In the design of BIPEFT, we aim to gradually obtain the optimal PEFT configuration after triggering the reduction Z times. Thus, estimating the number of parameter reductions at the trigger step z is essential. Let E_t denote the expected number of parameters at the t-th training step when the trigger counter is z. We can estimate the expected number of reductions is

$$R_z = \frac{E_t - \mathcal{B}}{Z - z}. \tag{6}$$

Here, the challenge is how can we estimate E_t.

3.3.1 Expected Parameters Estimation

We use the current status of the PEFT model at the t-th training step (i.e., the z-th trigger counter) and the selection status at the $(z-1)$-th trigger stage to estimate the expected parameters E_t. The PEFT model contains both architecture weights $\mathbf{\Theta}_t$ and $\mathbf{\Phi}_t$. The selection status at $z-1$ contains a binary module selection indicator denoted as $\mathbf{b}_{z-1} \in \{0,1\}^N$ and a module dimension selection indicator vector $\mathbf{d}_{z-1} \in \{0,1\}^N$. $\mathbf{b}^n_{z-1} = 1$ means that the module $\mathcal{M}_n$ is kept in the PEFT training. Otherwise, the module has been removed. $\mathbf{d}^n_{z-1} = 1$ indicates that $\mathcal{M}_n$ has a selected or fixed dimension value with index k^*_n, but $\mathbf{d}^n_{z-1} = 0$ means the dimension is undetermined. Based on these notations, we can estimate E_t as follows:

$$E_t = \sum_{n=1}^{N} p_n \sum_{k=1}^{K} C^k_n, \tag{7}$$

where $p_n = \mathbf{b}^n_{z-1} * \text{softmax}(\mathbf{\Theta}^n_t)[1]$ is the probability of the kept module, and C^k_n is the estimated number of parameters of each module, which is defined as follows:

$$C^k_n = \begin{cases} \mathbf{q}_n[k^*_n], & \text{if } \mathbf{d}^n_{z-1} = 1, \\ \mathbf{q}_n \cdot \text{softmax}(\mathbf{\Phi}^n_t)^\top, & \text{if } \mathbf{d}^n_{z-1} = 0, \end{cases} \tag{8}$$

where $\mathbf{q}_n \in \mathbb{R}^K$ is the parameter vector for all candidate dimensions, which can be precalculated. When the dimension is fixed, we use the corresponding parameter values directly. Otherwise, we use a weighted sum over the estimated probabilities.

3.3.2 Adaptive Module Selection

Using Eq. (7), we can obtain the expected reductions R_z. To prune unimportant modules, we still use the module-level sensitivity scores calculated by Eq. (4) at step t. We rank all sensitivity scores of the currently kept modules and remove the modules with the lowest scores if their total parameter size is smaller than R_z. Correspondingly, we update $\mathbf{b}_{z-1}$ to obtain $\mathbf{b}_z$ as the new module indicator.

3.4 Multiple Rank Dimension Selection

We can also determine the dimension size for each module $\mathcal{M}_n$ if there is a clearly stable pattern on learned architecture weights $\mathbf{\Phi}^n_t$. Since evaluating the sensitivity score for each dimension as each module is hard, we propose a new strategy to measure the weight distributions between two trigger counters, $z-1$ and z. Note that there are several training steps between the trigger gap. Assume that at the j-th training step, BIPEFT triggers the reduction $z-1$, and at t ($t > j$), the z-th trigger happens. We use the historical architecture weights $[\mathbf{\Phi}^n_j, \cdots, \mathbf{\Phi}^n_t]$ to evaluate the stability of each dimension.

3.4.1 Dimension Stability Estimation

Intuitively, a stable dimension needs to satisfy two conditions. On the one hand, the intra-dimension weight at each training step may not change much, i.e., the standard deviation of dimension-level weights $[\sigma^n_{z,1}, \cdots, \sigma^n_{z,K}]$ should be as small as possible. On the other hand, the cross-dimension weights, i.e., the weight distributions, should also be as similar as possible. Specifically, we can use the Kullback-Leibler (KL) divergence to evaluate the similarity of two distributions. Based on these motivations, we design a dimension stability indicator as follows:

$$\lambda^n_z = \frac{1}{K} \sum_{k=1}^{K} \sigma^n_{z,k} \text{KL}(\mathbf{\Phi}^n_j, \mathbf{\Phi}^n_t). \tag{9}$$

3.4.2 Adaptive Dimension Selection

Similar to the module selection, we need to determine which modules' dimensions should be selected automatically according to the calculated dimension stability scores $[\lambda_z^1, \cdots, \lambda_z^N]$ using Eq. (9). However, the challenge here is estimating the expected number of modules for dimension reduction.

Expected Reduction Module Size Estimation. Intuitively, we can fix more modules' dimensions if the potential dimension selections at $z-1$ and z are similar, which motivates us to use the similarity score to potential dimension vectors, i.e., $\mathbf{v}_{z-1}$ and $\mathbf{v}_z$. The potential dimension of the n-th module that is not fixed can be obtained through its architecture weights, i.e., $\mathbf{v}_z^n = \mathsf{Dim}[\text{argmax}_k(\mathbf{\Phi}_t^n)]$, where Dim is all the possible dimension vector. Note that if the module is removed, then $\mathbf{d}_{z-1}^n = 1, \mathbf{v}_z^n = 0$, and $\mathbf{v}_z^n = \mathsf{Dim}[k_n^*]$ for the fixed module. Finally, we can estimate the number of modules for dimension reduction as follows:

$$Y_z = \left\lfloor \frac{\text{sum}(\mathbf{1} - \mathbf{d}_{z-1}) * \cos(\mathbf{v}_{z-1}, \mathbf{v}_z)}{Z - z} \right\rfloor. \quad (10)$$

Adaptive Dimension Selection. After obtaining the expected number of reduction modules, we then use the dimension stability scores $[\lambda_z^1, \cdots, \lambda_z^N]$ to select dimension-unfixed modules among the top Y_z lowest scores, whose dimensions are then fixed using the corresponding values in $\mathbf{v}_z^n$. Finally, we can update $\mathbf{d}_z$ based on $\mathbf{d}_{z-1}$ and the newly dimension-fixed modules.

3.5 Iterative Optimization

The proposed `BIPEFT` is a differential NAS-based PEFT model, which can be optimized as DARTS (Liu et al., 2019). The optimization objective is defined as follows:

$$\min_{\mathbf{\Theta}^*, \mathbf{\Phi}^*} \mathcal{L}_{\text{val}}\left(\mathcal{D}_{val}; \mathbf{\Theta}, \mathbf{\Phi}, \mathbf{W}_0 + \Delta\mathbf{W}^*\right),$$
$$s.t.\ \Delta\mathbf{W}^* = \arg\min_{\Delta\mathbf{W}} \mathcal{L}_{\text{tra}}\left(\mathcal{D}_{tra}; \mathbf{\Theta}^*, \mathbf{\Phi}^*, \mathbf{W}_0 + \Delta\mathbf{W}\right),$$

where the network parameters contain two parts – fixed pre-trained LLM weights $\mathbf{W}_0$ and trainable PEFT parameters $\Delta\mathbf{W}$. However, we have two distinct search spaces with architecture weights $\mathbf{\Theta}$ and $\mathbf{\Phi}$. These two kinds of architecture weights depend on each other, making optimizing them simultaneously hard.

To address this issue, we propose an iterative optimization approach. We first use the first-order approximation in DARTS to optimize $\Delta\mathbf{W}$, which improves the search efficiency by considering that $\Delta\mathbf{W}$ converges fast based on the pre-trained $\mathbf{W}_0$. After obtaining $\Delta\mathbf{W}^*$, we then fix the dimension parameters $\mathbf{\Phi}^*$ and optimize the module parameters $\mathbf{\Theta}$ first. Subsequently, we use the optimized $\mathbf{\Theta}^*$ to learn the optimal dimension parameters $\mathbf{\Phi}^*$. We repeat the previous steps until `BIPEFT` converges.

Besides, to bridge the gap between the search and validation stages, we follow (Chang et al., 2019) by employing the Gumbel-Softmax (Jang et al., 2017) function to normalize $\mathbf{\Theta}^*$ and $\mathbf{\Phi}^*$, where the nodes with maximal weight will have the highest probability to be selected. In addition, during the search, we implement weight entanglement, as utilized in AutoFormer (Chen et al., 2021), allowing shared weights across different dimensions to enhance stability, promoting faster convergence and reducing memory costs.

4 Experiments

4.1 Experimental Setups

Datasets. We use two widely used natural language processing (NLP) benchmarks: GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019) in our experiments. All datasets are downloaded from the HuggingFace Datasets (Lhoest et al., 2021).

We follow S^3Delta (Hu et al., 2022b) to generate the training, validation, and test splits. For larger datasets, including QQP, QNLI, ReCoRD, SST-2, and MNLI, we allocate 2,000 random samples from the training set to form a new validation set, use the remaining samples as the training set, and repurpose the original validation set as the test set. For smaller datasets, we equally split the original validation set into new validation and test sets, while the original training set remains unchanged. Each dataset is split using different random seeds to introduce variability in the dataset configurations. Note that we remove the COPA dataset in SuperGLUE since its performance varies dramatically following S^3Delta (Hu et al., 2022b).

Baselines. We follow existing studies and use the following models as baselines: (1) Automatic PEFT methods: **AutoPEFT** (Zhou et al., 2024) uses multi-objective Bayesian optimization to discover a Pareto-optimal set of PEFT configurations. **S^3Delta** (Hu et al., 2022b) automatically searches sparse PEFT structure by determining the usage of each module using differential NAS. **PrunePEFT** (Lawton et al., 2023) implements a

Table 1: Results on the GLUE and SuperGLUE benchmarks. "Ratio" specifies the ratio of trainable parameters compared to T5. Average metrics (AVG) and parameter ratios (Ratio) are averaged over all the values. $*$ denotes the results that are directly copied from **S^{3}Delta** (Hu et al., 2022b).

GLUE									
Ratio	**Method**	**CoLA**	**SST2**	**MRPC**	**QQP**	**STSB**	**MNLI**	**QNLI**	**AVG**
10000%%	Fine-tune*	62.25 ± 3.96	95.87 ± 0.42	91.86 ± 1.19	89.50 ± 0.22	91.86 ± 0.46	89.61 ± 0.30	94.22 ± 0.35	87.88
Manual PEFT Methods									
65.33%%	Adapter*	59.03 ± 3.06	95.90 ± 0.29	93.02 ± 0.28	88.39 ± 0.06	91.77 ± 0.25	89.53 ± 0.07	94.17 ± 0.19	87.40
21.32%%	LoRA(r=8)*	58.43 ± 4.16	95.79 ± 0.27	92.21 ± 0.88	88.35 ± 0.25	91.78 ± 0.31	89.38 ± 0.32	94.14 ± 0.12	87.15
8.13%%	BitFit*	56.98 ± 3.89	96.24 ± 0.33	92.16 ± 0.68	88.12 ± 0.07	91.59 ± 0.08	89.10 ± 0.09	94.07 ± 0.21	86.90
1.70%%	LNFit*	56.15 ± 4.06	95.81 ± 0.20	91.71 ± 0.39	88.17 ± 0.10	91.37 ± 0.24	89.11 ± 0.09	93.99 ± 0.20	86.62
Automated PEFT Methods									
18.90%%	AutoPEFT	59.59 ± 1.24	95.87 ± 0.23	91.06 ± 0.52	88.21 ± 0.04	91.42 ± 0.08	89.16 ± 0.10	93.90 ± 0.15	87.03
1.39%%	BIPEFT	59.04 ± 8.20	95.70 ± 0.08	92.10 ± 0.61	88.28 ± 0.09	91.83 ± 0.40	89.14 ± 0.01	94.06 ± 0.01	**87.16**
1.39%%	S^{3}Delta-M*	59.34 ± 4.75	95.84 ± 0.14	92.13 ± 2.09	88.04 ± 0.23	91.58 ± 0.25	89.14 ± 0.13	94.12 ± 0.12	87.17
1.39%%	PrunePEFT	60.39 ± 5.90	95.87 ± 0.09	92.83 ± 1.41	88.12 ± 0.11	91.62 ± 0.61	89.19 ± 0.15	93.95 ± 0.08	87.42
1.39%%	BIPEFT	62.97 ± 3.72	96.27 ± 0.24	93.22 ± 0.62	88.28 ± 0.02	92.23 ± 0.28	89.28 ± 0.02	94.25 ± 0.06	**88.07**
SuperGLUE									
Ratio	**Method**	**BoolQ**	**CB**	**MultiRC**	**ReCORD**	**RTE**	**WIC**	**AVG**	
10000%%	Fine-tune*	86.67 ± 0.21	96.43 ± 2.92	76.65 ± 1.01	85.03 ± 0.67	88.49 ± 2.12	73.12 ± 1.71	84.40	
Manual PEFT Methods									
65.33%%	Adapter*	85.98 ± 0.68	94.64 ± 6.19	77.60 ± 0.84	85.96 ± 0.37	89.21 ± 2.94	71.63 ± 0.90	84.17	
21.32%%	LoRA(r=8)*	85.06 ± 0.70	91.96 ± 3.42	76.94 ± 1.16	85.84 ± 0.21	87.05 ± 0.59	72.10 ± 1.31	83.16	
21.32%%	BitFit*	85.02 ± 0.48	89.29 ± 2.92	75.79 ± 1.15	85.85 ± 0.32	86.15 ± 1.48	72.34 ± 1.61	82.41	
1.70%%	LNFit*	84.07 ± 0.50	82.14 ± 2.92	75.52 ± 1.16	86.14 ± 0.11	86.69 ± 1.81	69.28 ± 1.49	80.64	
Automated PEFT Methods									
17.69%%	AutoPEFT	83.39 ± 0.13	93.75 ± 4.49	76.49 ± 0.41	85.59 ± 0.13	85.99 ± 0.42	72.17 ± 1.19	82.89	
1.39%%	BIPEFT	84.55 ± 0.13	92.86 ± 3.58	75.90 ± 0.12	85.94 ± 0.05	88.85 ± 1.53	70.38 ± 0.66	**83.08**	
1.39%%	S^{3}Delta-M*	84.92 ± 0.68	92.86 ± 2.92	76.38 ± 0.92	86.10 ± 0.11	86.69 ± 1.90	71.63 ± 1.07	83.10	
1.39%%	PrunePEFT	85.16 ± 0.47	91.66 ± 4.12	76.66 ± 0.60	85.46 ± 0.81	87.29 ± 0.42	70.37 ± 0.23	82.77	
1.39%%	BIPEFT	85.44 ± 1.12	94.65 ± 2.52	75.84 ± 0.04	85.70 ± 0.07	88.49 ± 2.04	72.25 ± 0.22	**83.73**	

simple unstructured pruning strategy to search for optimal PEFT structures. (2) Manually designed PEFT methods: We use **LoRA** (Hu et al., 2022a), **Adapter** (Houlsby et al., 2019), **BitFit**, and **LNFit** as baselines following S^{3}Delta (Hu et al., 2022b).

Pretrained Backbone Model. Our experiments utilize the T5 large model for all the baselines and BIPEFT, which contains approximately 770 million parameters. We freeze the pretrained parameters $\mathbf{W}_0$ across all PEFT settings.

Search Spaces. Since the search space of existing work is different, for a fair comparison, we follow existing work and make comparisons within the space that they use. Setting 1 (**S1**): For AutoPEFT, we use a mixture of Serial Adapter (Houlsby et al., 2019), Parallel Adapter (He et al., 2022), and Prefix-Tuning (Li and Liang, 2021). Setting 2 (**S2**): When comparing with S^{3}Delta and PrunePEFT, we use a mixture of LoRA, Adapter-LR, BitFit (Ben Zaken et al., 2022), and LNFiT modules as the search space. For both S1 and S2, each module has a binary selection space $\{0, 1\}$. For baseline S^{3}Delta, its dimension size is fixed, which is 1. For other approaches, the candidate dimensions for all the PEFT modules in both search space settings are $\mathsf{Dim} = \{1, 4, 8\}$. Additional experiment settings and hyperparameters are listed in Appendix B.

4.2 Main Results

We report the results on GLUE and SuperGLUE in Table 1 by averaging 3 runs with different random seeds. We follow previous work (Hu et al., 2022b) and use different metrics to validate the performance of different tasks, as detailed in Appendix A. Notably, BIPEFT typically outperforms both manual PEFT methods and automated PEFT baselines regarding average scores on both benchmarks under different search spaces, with a small budget. On the GLUE benchmark, BIPEFT even surpasses the full fine-tuning with only 1.39%% parameters.

When comparing the automated PEFT methods with manually designed PEFT, we find that BIPEFT and other automated PEFT methods achieve better average scores on GLUE and SuperGLUE with

Table 2: Results of ablation study within the **S1**. "**Entanglement**" means that we directly use differential NAS to search from the entangled search space. "**b** → **d**" is a two-stage search, where we first run a binary search until the model converges, then run the dimension rank search based on the binary search results. "**d** → **b**" means to exchange the search order of "**b** → **d**". "***w.o.* Selection**" is a variant of BIPEFT by removing the early selection and only conducting the iterative search. "***w.* Selection**" is BIPEFT. Red and Green highlight the best and second-best.

Entanglement	Disentanglement				Ratio	COLA	MRPC	RTE	AVG
	w.o. Iteration		*w.* Iteration						
	b → d	d → b	*w.o.* Selection	*w.* Selection					
✓					33.81%%	60.52 ± 2.58	91.54 ± 2.48	87.77 ± 1.01	79.94
	✓				29.01%%	60.03 ± 2.94	91.66 ± 1.89	88.01 ± 0.83	79.90
		✓			21.98%%	60.33 ± 3.70	92.03 ± 1.36	87.77 ± 1.86	80.04
			✓		29.22%%	62.95 ± 3.03	92.04 ± 2.29	88.25 ± 0.83	81.08
				✓	**1.39%%**	**62.97 ± 3.72**	**93.22 ± 0.62**	**88.49 ± 0.83**	**81.56**

fewer parameters. We also find that the differences in search space design result in performance change. Under the **S1** setting designed by AutoPEFT, BIPEFT searched with an equivalent low parameter budget of 1.39%% does not match the performance achieved under the **S2** setting, which contains more types of PEFT modules, highlighting the benefits of including a diverse range of searchable PEFT modules in BIPEFT.

4.3 Ablation Study

The proposed BIPEFT mainly considers the disentanglement search spaces and uses both binary module and dimension rank early selection strategies to enhance efficiency and boost performance. We use four baselines in the ablation study to validate the effectiveness of our model design. The experimental setting follows (Hu et al., 2022b), and the results of these baselines are shown in Table 2. We can observe that the iterative search approaches (the last two rows) outperform the entanglement and non-iterative ones, indicating the effectiveness of our proposed iterative search solution. Besides, using the designed selection strategies, BIPEFT further enhances its performance and reduces the number of parameters. This comparison validates the usefulness of our early selection strategies.

4.4 Efficiency Analysis

In addition to the promising performance, we also evaluate the search efficiency of our method by comparing the time consumed during the search stage against the re-training duration, as detailed in Table 3. All the training time is tested with the same batch size on the same GPU devices. The time is averaged from three datasets, RTE, STSB, and CoLA, under two search space settings, S1 and S2. It clearly shows that BIPEFT achieves a high search efficiency with the design of early selection.

Table 3: Efficiency Comparison.

Setting	Method	Avg. Time (min)		
		Search	Re-train	Ratio
S1	AutoPEFT	348	22	15.80
	BIPEFT	13	22	**0.59**
S2	S^3Delta	125	20	6.30
	PrunePEFT	20	20	1.00
	BIPEFT	13	20	**0.65**

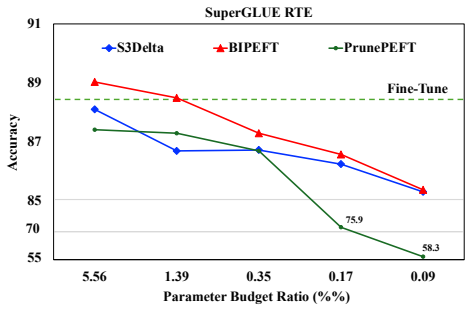


Figure 3: Performance vs. different levels of budget.

4.5 Different Parameter Budgets

We also explore the influence of the parameter budget. Ideally, a larger budget will lead to better performance. Figure 3 shows the results with different ratios of parameter budgets, and the models are searched in space **S1** on the RTE dataset in the SuperGLUE benchmark, compared with S^3Delta and PrunePEFT. We can observe that BIPEFT consistently sustains a higher accuracy and saturates to full fine-tuning performance by preserving essential PEFT modules even at very low budget levels.

4.6 Generalization Ability Analysis

We evaluate the task generalization capability of the structures searched within search space **S1**, as presented in Table 4. The results indicate that the structures searched for source datasets such as MRPC, MNLI, and QQP exhibit robust generalization to various target datasets, which encompass different types of NLP tasks. This demonstrates the abil-

Table 4: PEFT structure generalization from source datasets to target datasets. "No Transfer" means using the original structures searched from the target datasets.

Source Datasets	Target Datasets			
	STSB	SST2	QNLI	COLA
No Transfer	92.23 ± 0.28	96.27 ± 0.24	94.34 ± 0.06	62.95 ± 3.03
MRPC	92.28 ± 0.22	96.04 ± 0.08	94.18 ± 0.16	60.51 ± 6.21
MNLI	92.04 ± 0.12	96.16 ± 0.24	94.04 ± 0.17	60.17 ± 2.96
QQP	92.20 ± 0.05	96.10 ± 0.16	94.16 ± 0.12	62.08 ± 5.72

ity of our searched structures in maintaining high performance across diverse NLP downstream applications.

5 Conclusion

In this paper, we introduce BIPEFT, a highly efficient search framework for parameter-efficient fine-tuning (PEFT) modules on large pretrained language models. BIPEFT operates within a specifically designed disentangled search space using an iterative search strategy, incorporating novel selection mechanisms that significantly accelerate the search process while delivering promising performance outcomes. Our extensive experiments on two widely used NLP benchmarks demonstrate the superiority of the PEFT structures identified by BIPEFT. Despite requiring limited parameters and incurring minimal search costs, BIPEFT outperforms both manually designed and other automated PEFT methods across a variety of NLP tasks. Additionally, the searched structures exhibit excellent generalization ability, proving effective for other downstream tasks as well. Overall, BIPEFT represents a substantial advancement in the field of automatic PEFT optimization, providing an efficient and effective solution for fine-tuning large pretrained language models.

Limitations

While we conduct our PEFT within the S1 and S2 search space settings, which include many popular PEFT modules, there remains the potential to integrate additional existing or new PEFT modules into our framework to further evaluate search performance and efficiency. Nonetheless, the proposed BIPEFT framework is inherently flexible, allowing for the seamless integration of new PEFT modules as they become available. In addition, although the early stopping design requires manual hyperparameters like the maximum step Z, this mechanism will sustain high efficiency and strong performance because it could effectively prioritize the most critical PEFT modules within the given budget.

Acknowledgements

This work is partially supported by the National Science Foundation under Grant No. 2238275 and 2212323 and the National Institutes of Health under Grant No. R01AG077016.

References

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*, pages 1–9.

Jianlong Chang, Yiwen Guo, Gaofeng Meng, Shiming Xiang, Chunhong Pan, et al. 2019. Data: Differentiable architecture approximation. *NeurIPS*, 32.

Jiaao Chen, Aston Zhang, Xingjian Shi, Mu Li, Alex Smola, and Diyi Yang. 2023. Parameter-efficient fine-tuning design spaces. In *The Eleventh International Conference on Learning Representations*.

Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. 2021. Autoformer: Searching transformers for visual recognition. In *ICCV*, pages 12270–12280.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *NeurIPS*, 36.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Zihao Fu, Haoran Yang, Anthony Man-Cho So, Wai Lam, Lidong Bing, and Nigel Collier. 2023. On the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12799–12807.

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

Zeyu Han, Chao Gao, Jinyang Liu, Sai Qian Zhang, et al. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.

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 *International Conference on Machine Learning*, pages 2790–2799. PMLR.

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

Shengding Hu, Zhen Zhang, Ning Ding, Yadao Wang, Yasheng Wang, Zhiyuan Liu, and Maosong Sun. 2022b. Sparse structure search for delta tuning. In *Thirty-Sixth Conference on Neural Information Processing Systems*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

Shen Jiang, Zipeng Ji, Guanghui Zhu, Chunfeng Yuan, and Yihua Huang. 2023. Operation-level early stopping for robustifying differentiable NAS. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Neal Lawton, Anoop Kumar, Govind Thattai, Aram Galstyan, and Greg Ver Steeg. 2023. Neural architecture search for parameter-efficient fine-tuning of large pre-trained language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8506–8515.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184.

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.

Hanwen Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. 2019. Darts+: Improved differentiable architecture search with early stopping. *arXiv preprint arXiv:1909.06035*.

Baohao Liao, Yan Meng, and Christof Monz. 2023. Parameter-efficient fine-tuning without introducing new latency. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 4242–4260.

Zhaojiang Lin, Andrea Madotto, and Pascale Fung. 2020. Exploring versatile generative language model via parameter-efficient transfer learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 441–459.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*.

Gen Luo, Minglang Huang, Yiyi Zhou, Xiaoshuai Sun, Guannan Jiang, Zhiyu Wang, and Rongrong Ji. 2023. Towards efficient visual adaption via structural re-parameterization. *arXiv preprint arXiv:2302.08106*.

Junyu Luo, Xiaochen Wang, Jiaqi Wang, Aofei Chang, Yaqing Wang, and Fenglong Ma. 2024. CoRelation: Boosting automatic ICD coding through contextualized code relation learning. In *LREC-COLING 2024*, pages 3997–4007, Torino, Italia. ELRA and ICCL.

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Wen tau Yih, and Madian Khabsa. 2021. Unipelt: A unified framework for parameter-efficient language model tuning. In *Annual Meeting of the Association for Computational Linguistics*.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. 2019. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11264–11272.

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

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021. Adapterdrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7930–7946.

Yusheng Su, Chi-Min Chan, Jiali Cheng, Yujia Qin, Yankai Lin, Shengding Hu, Zonghan Yang, Ning Ding, Xingzhi Sun, Guotong Xie, et al. 2023. Exploring the impact of model scaling on parameter-efficient tuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15062–15078.

Yi-Lin Sung, Varun Nair, and Colin A Raffel. 2021. Training neural networks with fixed sparse masks. *NeurIPS*, 34:24193–24205.

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

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *NeurIPS*, 32.

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.

Jiaqi Wang, Junyu Luo, Muchao Ye, Xiaochen Wang, Yuan Zhong, Aofei Chang, Guanjie Huang, Ziyi Yin, Cao Xiao, Jimeng Sun, and Fenglong Ma. 2024. Recent advances in predictive modeling with electronic health records. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 8272–8280. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. 2023. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*.

Feiyu Zhang, Liangzhi Li, Junhao Chen, Zhouqiang Jiang, Bowen Wang, and Yiming Qian. 2023a. Increlora: Incremental parameter allocation method for parameter-efficient fine-tuning. *arXiv preprint arXiv:2308.12043*.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023b. Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*. Openreview.

Qingru Zhang, Simiao Zuo, Chen Liang, Alexander Bukharin, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2022a. Platon: Pruning large transformer models with upper confidence bound of weight importance. In *International conference on machine learning*, pages 26809–26823. PMLR.

Yuanhan Zhang, Kaiyang Zhou, and Ziwei Liu. 2022b. Neural prompt search.

Han Zhou, Xingchen Wan, Ivan Vulić, and Anna Korhonen. 2024. Autopeft: Automatic configuration search for parameter-efficient fine-tuning. *Transactions of the Association for Computational Linguistics*, 12.

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

A Evaluation Metrics

For both GLUE and SuperGLUE benchmarks, we employ various evaluation metrics: Accuracy is reported for the SST-2, MNLI, QNLI, BoolQ, CB, RTE, and WIC tasks. We utilize the F1 score to assess performance on MRPC, QQP, MultiRC, and ReCoRD. Additionally, Matthew's Correlation is used to evaluate CoLA, and the Pearson Correlation coefficient is applied to the STSB task.

B Experiment Settings and Hyperparameters

For each experiment setting, we report the average performances and standard deviations using results from three different seeds on the final test sets. We configure the maximum sequence lengths as 128 for GLUE tasks and 256 for SuperGLUE tasks, maintaining a consistent batch size of 32 across both benchmarks. For the ReCoRD task, we adjust the settings to a maximum sequence length of 512 and a batch size of 16. We utilize the AdamW optimizer with a linear learning rate decay schedule to optimize our model. All experiments maintain a consistent learning rate of 3×10^{-4} for PEFT training, while the learning rate for architecture parameters in `BIPEFT` is set at 0.01. For `BIPEFT`, following DARTS (Liu et al., 2019), we equally split the original training set into two parts. One part is used for optimizing the model parameters, and the other for optimizing the architecture parameters. The original validation set serves to evaluate and save the searched structures. For the default values of the model hyperparameters used in `BIPEFT`, we set $Z = 100$, $\gamma = 0.85$, $H = 5$, and $\tau = 0.85$.

C Additional Experiment Results

C.1 Hyperparameter Sensitivity Analysis

We perform a detailed hyperparameter sensitivity analysis using the same datasets as in our ablation study. Specifically, we test the following hyperparameters: (1) maximum trigger count Z, (2) γ, used for the smoothed sensitivity in Eq. (4), (3) time window H, employed in the trigger design to measure model stability, and (4) stability threshold τ. All these hyperparameters are configured for the

Table 5: Additional Results on the GLUE benchmark.$*$ denotes the results that are directly copied from **S^{3}Delta** (Hu et al., 2022b). Red denotes the best score.

GLUE									
Ratio	**Method**	**CoLA**	**SST2**	**MRPC**	**QQP**	**STSB**	**MNLI**	**QNLI**	**AVG**
10000%%	Fine-tune*	62.25 ± 3.96	95.87 ± 0.42	91.86 ± 1.19	89.50 ± 0.22	91.86 ± 0.46	89.61 ± 0.30	94.22 ± 0.35	87.88
Automated PEFT Methods									
5.56%%	S^{3}Delta-M*	61.67 ± 5.38	95.88 ± 0.11	91.16 ± 2.09	88.22 ± 0.04	91.81 ± 0.41	89.18 ± 0.08	93.88 ± 0.05	87.40
5.56%%	PrunePEFT	63.83 ± 2.32	95.95 ± 0.19	91.67 ± 1.13	88.45 ± 0.08	92.06 ± 0.47	89.50 ± 0.14	94.35 ± 0.08	87.97
5.56%%	BIPEFT	64.48 ± 2.63	96.04 ± 0.40	93.19 ± 1.24	88.32 ± 0.14	92.09 ± 0.30	89.19 ± 0.02	94.25 ± 0.04	88.22

Table 6: The sensitivity analysis of the maximum trigger count Z. "AVG.Time" denotes the average search time (in minutes).

Value of Z	**COLA**	**MRPC**	**RTE**	**AVG**	**AVG. Time**
10	60.96 ± 1.22	92.98 ± 1.39	86.69 ± 0.36	80.21	6.34
50	61.82 ± 3.85	92.78 ± 0.84	87.77 ± 1.44	80.79	8.41
100	62.97 ± 3.72	93.22 ± 0.62	88.49 ± 2.04	81.56	12.39
200	62.99 ± 5.38	92.98 ± 1.09	89.21 ± 0.72	81.73	16.34

Table 7: The sensitivity analysis of γ.

Value of γ	**COLA**	**MRPC**	**RTE**	**AVG**
0.75	63.01 ± 2.86	93.24 ± 1.16	87.77 ± 0.72	81.34
0.85	62.97 ± 3.72	93.22 ± 0.62	88.49 ± 2.04	81.56
1	56.83 ± 4.70	92.01 ± 1.06	83.45 ± 0.59	77.43

sub-modules of our early selection module, which is designed to enhance search efficiency. The parameter budget ratios are uniformly set to 1.39%%. A comprehensive analysis of each hyperparameter is presented below.

The maximum trigger count Z controls the speed of the early selection process. We report the average search time across three datasets, demonstrating that while larger Z values ensure smoother selection, they may marginally reduce efficiency. As shown in Table 6, increasing Z beyond 100 diminishes marginal returns in performance improvement. The parameter γ is used for smoothed sensitivity in Eq. (4), capturing the ratio of historical sensitivity information retained during the training process. The default value for γ is set to 0.85. Our additional experiments, presented in Table 7, indicate that setting it to 1 prevents the sensitivity from being updated with new data. Stable performance is observed when γ remains 0.7 to 0.9. The time window H is used in the trigger design to measure model stability. As indicated by the results in Table 8, varying H does not produce

Table 8: The sensitivity analysis of time window H used in the trigger design.

Value of H	**COLA**	**MRPC**	**RTE**	**AVG**
3	62.29 ± 4.80	93.26 ± 0.84	88.13 ± 0.36	81.23
5	62.97 ± 3.72	93.22 ± 0.62	88.49 ± 2.04	81.56
10	61.89 ± 4.54	93.45 ± 0.40	88.13 ± 0.36	81.16
20	62.72 ± 3.99	92.64 ± 1.25	88.13 ± 1.08	81.16

Table 9: The sensitivity analysis of stability threshold τ.

Value of τ	**COLA**	**MRPC**	**RTE**	**AVG**
0.1	61.57 ± 4.62	92.74 ± 1.02	87.41 ± 0.36	80.57
0.5	62.18 ± 4.10	92.94 ± 1.04	87.41 ± 0.36	80.84
0.85	62.97 ± 3.72	93.22 ± 0.62	88.49 ± 2.04	81.56

Table 10: Comparison with APET.

Method	**Ratio**	**COLA**	**MRPC**	**RTE**	**AVG**
APET	1.39%%	59.99 ± 4.87	91.92 ± 0.67	85.61 ± 2.16	79.17
APET	5.56%%	60.82 ± 3.95	92.68 ± 1.17	88.49 ± 1.01	80.66
BIPEFT	1.39%%	62.97 ± 3.72	93.22 ± 0.62	88.49 ± 2.04	81.56

significant performance differences across the three datasets. The default value of H is set to 5. The stability threshold τ influences trigger generation, with higher values enforcing stricter stability requirements, leading to smoother trigger generation and improved performance of the searched structures, as demonstrated in Table 9. By default, we set τ to 0.85.

C.2 Higher Sparsity Ratio of Parameters

As demonstrated in the main experiments, BIPEFT outperforms all baselines at a parameter ratio of 1.39%%. To further showcase the effectiveness of our method across diverse budget settings, we provide additional results on the GLUE benchmark at a parameter ratio of 5.56%% under setting **S2**, compared to PrunePEFT and S^{3}Delta. The average score for full fine-tuning is 87.88. As shown in Table 5, BIPEFT continues to outperform other baselines given a higher budget.

C.3 Zero-Cost PEFT Configuration

As introduced in (Su et al., 2023), their zero-cost PEFT configuration method, APET, initially freezes key factors such as the number of trainable parameters. Under this constraint, APET arbitrarily selects tunable parameters using different random seeds, each representing a distinct parameter distribution, and trains them on the tasks. As shown in Table 10, BIPEFT outperforms APET even at a lower parameter ratio, albeit with a small search cost.