On the Effectiveness of Parameter-Efficient Fine-Tuning

Zihao Fu,[1] **Haoran Yang,**[2] **Anthony Man-Cho So,**[2]
Wai Lam,[2] **Lidong Bing,**[3] **Nigel Collier**[1]

[1]Language Technology Lab, University of Cambridge,
[2]The Chinese University of Hong Kong, [3]DAMO Academy, Alibaba Group
{zf268,nhc30}@cam.ac.uk,
{hryang,manchoso,wlam}@se.cuhk.edu.hk, l.bing@alibaba-inc.com

Abstract

Fine-tuning pre-trained models has been ubiquitously proven to be effective in a wide range of NLP tasks. However, fine-tuning the whole model is parameter inefficient as it always yields an entirely new model for each task. Currently, many research works propose to only fine-tune a small portion of the parameters while keeping most of the parameters shared across different tasks. These methods achieve surprisingly good performance and are shown to be more stable than their corresponding fully fine-tuned counterparts. However, such kind of methods is still not well understood. Some natural questions arise: How does the parameter sparsity lead to promising performance? Why is the model more stable than the fully fine-tuned models? How to choose the tunable parameters? In this paper, we first categorize the existing methods into random approaches, rule-based approaches, and projection-based approaches based on how they choose which parameters to tune. Then, we show that all of the methods are actually sparse fine-tuned models and conduct a novel theoretical analysis of them. We indicate that the sparsity is actually imposing a regularization on the original model by controlling the upper bound of the stability. Such stability leads to better generalization capability which has been empirically observed in a lot of recent research works. Despite the effectiveness of sparsity grounded by our theory, it still remains an open problem of how to choose the tunable parameters. Currently, the random and rule-based methods do not utilize task-specific data information while the projection-based approaches suffer from the projection discontinuity problem. To better choose the tunable parameters, we propose a novel Second-order Approximation Method (SAM) which approximates the original problem with an analytically solvable optimization function. The tunable parameters are determined by directly optimizing the approximation function. We conduct extensive experiments on several tasks. The experimental results show that our proposed SAM model outperforms many strong baseline models and it also verifies our theoretical analysis. The source code of this paper can be obtained from https://github.com/fuzihaofzh/AnalyzeParameterEfficientFinetune

1 Introduction

Fine-tuning the model parameters for a specific task on a pre-trained model (Peters et al. 2018; Kenton and Toutanova 2019; Lan et al. 2020; Radford et al. 2018, 2019; Liu et al. 2019; Brown et al. 2020; Lewis et al. 2020; Raffel et al. 2020) has become one of the most promising techniques for NLP in recent years. It achieves state-of-the-art performance on most of the NLP tasks. However, as the parameter number grows exponentially to billions (Brown et al. 2020) or even trillions (Fedus, Zoph, and Shazeer 2021), it becomes very inefficient to save the fully fine-tuned parameters (He et al. 2021a) for each downstream task. Many recent research works propose a parameter-efficient (Houlsby et al. 2019; Zaken, Ravfogel, and Goldberg 2021; He et al. 2021a) way to solve this problem by tuning only a small part of the original parameters and storing the tuned parameters for each task.

Apart from the efficiency of the parameter-efficient models, it has also been observed in many recent research works that the parameter-efficient methods achieve surprisingly good performance. These models are more stable (He et al. 2021b; Lee, Cho, and Kang 2019; Houlsby et al. 2019; Zaken, Ravfogel, and Goldberg 2021; Sung, Nair, and Raffel 2021; Liu et al. 2021; Ding et al. 2022) and even achieve better overall scores than the fully fine-tuned models (Lee, Cho, and Kang 2019; Houlsby et al. 2019; Zaken, Ravfogel, and Goldberg 2021; Sung, Nair, and Raffel 2021; Liu et al. 2021; Xu et al. 2021; Guo, Rush, and Kim 2021; He et al. 2021a; Ding et al. 2022) on some tasks. Currently, it remains unclear why the parameter-efficient models can improve the stability and performance in many prevalent works. In this paper, we first categorize the existing methods into three categories (i.e. random approaches, rule-based approaches, and projection-based approaches) depending on how they choose the tunable parameters. Then, we define the generalized sparse fine-tuned model and illustrate that most of the existing parameter-efficient models are actually a sparse fine-tuned model. Afterwards, we introduce the widely used pointwise hypothesis stability of the sparse fine-tuned model and show theoretically that the sparsity actually controls the upper bound of the stability. Based on the stability analysis, we further give a theoretical analysis of the generalization bound for the sparse fine-tuned model.

Though promising results have been achieved by existing parameter-efficient models, it still remains a challenging problem to select suitable parameters as it is an NP-hard problem. Currently, the random (Lee, Cho, and Kang

2019) and rule-based (Zaken, Ravfogel, and Goldberg 2021; Han, Mao, and Dally 2015; Houlsby et al. 2019; Pfeiffer et al. 2020) approaches propose to optimize fixed parameters. These methods are straightforward and easy to implement but they do not utilize task-specific data information. To solve this problem, the projection-based approaches (Mallya, Davis, and Lazebnik 2018; Guo, Rush, and Kim 2021; Xu et al. 2021) propose to calculate a score for each parameter based on the data and project the scores onto the parameter selection mask's feasible region (an L_0 ball). However, as the feasible region is non-convex, we will show that such projection suffers from the projection discontinuity problem which makes the parameter selection quite unstable. To solve these problems, we propose a novel Second-order Approximation Method (SAM) to approximate the NP-hard optimization target function with an analytically solvable function. Then, we directly choose the parameters based on the optimal value and optimize the parameters accordingly. We conduct extensive experiments to validate our theoretical analysis and our proposed SAM model.

Our contributions can be summarized as follows: 1) We propose a new categorization scheme for existing parameter-efficient methods and generalize most of these methods with a unified view called the sparse fine-tuned model. 2) We conduct a theoretical analysis of the parameter-efficient models' stability and generalization. 3) We propose a novel SAM model to choose the suitable parameters to optimize. 4) We conduct extensive experiments to verify our theoretical analysis and the SAM model.

2 Unified View of Parameter Efficient Fine-tuning

In this section, we first define the unified sparse fine-tuned model which is simpler and easier for theoretical analysis. Then, we give a unified form of the optimization target. Afterwards, similar to previous works (Ding et al. 2022; He et al. 2021a; Mao et al. 2021), we categorize these models into three categories based on how the parameters are chosen. Finally, we show that all the models are sparse fine-tuned model.

2.1 Sparse Fine-tuned Model

We first give the definition of sparse fine-tuned model as well as a unified optimization target. The equivalent model is also defined to help understand the models with modified structures.

Definition 1 (p-Sparse Fine-tuned Model)**.** *Given a pre-trained model $\mathcal{M}^0$ with parameters θ^0, if a fine-tuned model $\mathcal{M}$ with parameters θ has the same structure as $\mathcal{M}^0$ such that $\|\theta - \theta^0\|_0 \leq p \dim(\theta), p \in (0, 1)$, we say the model $\mathcal{M}$ is a p-sparse fine-tuned model with the sparsity p.*

Many previous works propose different methods of selecting proper parameters to fine-tune. We unify these methods by denoting M as a mask matrix on the parameters and the parameter θ can be denoted as $\theta = \theta^0 + M\Delta\theta$, where $\Delta\theta$ is the difference vector. For a fixed sparsity coefficient p, the sparse fine-tuned model is trying to solve the following problem:

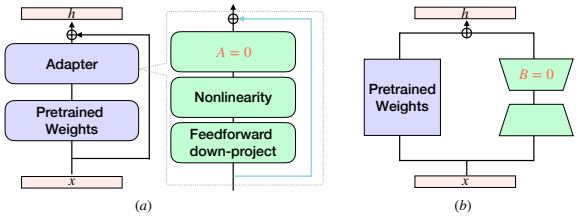

Figure 1: Equivalent model for Adapter (a) and LoRA (b).

$$\begin{aligned} &\min_{\Delta\theta, M} \mathcal{L}(\theta^0 + M\Delta\theta) \\ s.t. \quad &\|M\|_0 = \lfloor mp \rfloor; \quad M_{ij} = 0, \forall i \neq j; \quad M_{ii} \in \{0, 1\}, \end{aligned} \tag{1}$$

where $\lfloor \cdot \rfloor$ is the floor function, $m = \dim(\theta)$ is the parameter number, $M \in \{0, 1\}^{m \times m}$ is the parameter mask matrix with the diagonal equal to 0 or 1 while other elements are equal to 0 and $\mathcal{L}$ is the loss function. We will show that most of the existing methods are sparse fine-tuned models. However, in Definition 1, we assume that the fine-tuned model $\mathcal{M}$ has the same structure as $\mathcal{M}^0$. This assumption hinders us from analyzing many models that alter the structure including Adapter (Houlsby et al. 2019; Pfeiffer et al. 2020; Rücklé et al. 2021; He et al. 2021b), LoRA (Hu et al. 2022), and etc. We define the notion of equivalent model to solve this problem.

Definition 2 (Equivalent Model)**.** *Given a pre-trained model $\mathcal{M}^0$ with parameters θ^0, we say that a model $\tilde{\mathcal{M}}^0$ with parameters $\tilde{\theta}^0$ is an equivalent model for model $\mathcal{M}^0$ if $\forall x, \mathcal{M}^0(x) = \tilde{\mathcal{M}}^0(x)$.*

Here, we do not require that the equivalent model shares the same structure as the original model. As a result, for models fine-tuned with additional structures (e.g. Adapter and LoRA), we can still get a sparse fine-tuned model with respect to an equivalent model $\tilde{\mathcal{M}}^0$ instead of the original pre-trained model $\mathcal{M}^0$. Therefore, our analysis for the sparse fine-tuned model is also applicable to them.

2.2 Parameter Efficient Fine-tuning as Sparse Fine-tuned Model

Unfortunately, Problem (1) is NP-Hard due to the nonconvexity of the feasible region of the matrix M. Many existing methods propose to solve this problem by first estimating M and then optimizing other parameters. Based on different strategies for choosing M, the methods can be divided into three categories, namely, random approaches, rule-based approaches, and projection-based approaches. We first give a general introduction of the prevalent parameter efficient fine-tuning methods in each category and then show that all of these methods are actually a sparse fine-tuned model. Then, in next section, we can prove our theory only based on properties in Definition 1 without refering to any specific model's property.

Random Approaches Random approaches include Random and Mixout models. These models randomly choose

the parameters to be tuned. Such selection does not depend on the task-specific data information. Specifically, **Random** model is very straightforward by randomly selecting the parameters with respect to a given sparsity ratio and then training the selected parameters. Therefore, according to Definition 1, it is a sparse fine-tuned model. **Mixout** (Lee, Cho, and Kang 2019) proposes to directly reset a portion of the fine-tuned model's parameters to the pre-trained parameters with respect to a ratio. Therefore, according to Definition 1, it is a sparse fine-tuned model.

Rule-Based Approaches The rule-based approaches include BitFit, MagPruning, Adapter, and LoRA. This kind of methods directly uses a pre-defined rule to fix the parameters to be tuned. It can be viewed as incorporating prior knowledge to recognize important features and can thus alleviate the problem of random approaches. However, the selection rules are still irrelevant to the specific data. Specifically, **BitFit** (Zaken, Ravfogel, and Goldberg 2021) only fine-tunes the bias-terms and achieves considerably good performance. Therefore, according to Definition 1, it is a sparse fine-tuned model with pre-defined tuning weights. **MagPruning** (Han, Mao, and Dally 2015; Han et al. 2015; Lee et al. 2021; Lagunas et al. 2021) follows the idea that large weights are more important in the model. It ranks the weights by the absolute value and tunes the parameters with high absolute values. Therefore, according to Definition 1, it is a sparse fine-tuned model. **Adapter** (Houlsby et al. 2019; Pfeiffer et al. 2020; Rücklé et al. 2021; He et al. 2021b; Karimi Mahabadi, Henderson, and Ruder 2021; Mahabadi et al. 2021) proposes to add an adapter layer inside the transformer layer. Therefore, the model structure is different from the original model. To make it easier to analyze, Adapter can be viewed as fine-tuning an equivalent model shown in Fig. 1 (a) which initializes the matrix A as an all-zero matrix. The equivalent model has the same output as the original pre-trained model for arbitrary input while the structure is the same as the Adapter model. Therefore, fine-tuning the adapter model can be viewed as fine-tuning partial parameters of the equivalent model with the same structure. According to Definition 1, it is a sparse fine-tuned model with respect to the equivalent model. **LoRA** (Hu et al. 2022; Karimi Mahabadi, Henderson, and Ruder 2021; Panahi, Saeedi, and Arodz 2021) proposes to add a new vector calculated by recovering an hidden vector from a lower dimension space. The model is illustrated in Fig. 1 (b). It is interesting to notice that the original initialization makes the LoRA model already an equivalent model for the original pre-trained model as the matrix B is set to 0. Therefore, according to Definition 1, fine-tuning a LoRA model can also be viewed as fine-tuning partial parameters of the equivalent model with the same structure.

Projection-Based Approaches To utilize the task-specific data to help select the model's tunable parameters, many researchers propose projection-based approaches including the DiffPruning, ChildPruning, and etc. These methods propose to choose the optimal parameter mask M and optimize the parameters θ alternately to solve Problem (1). Specifically, they first relax M as a continuous variable to get an optimized value and then project the optimized value onto the feasible region which can be denoted as $\hat{M} = \Pi_\Omega(M) = \arg\min_{\hat{M}\in\Omega} \|\hat{M} - M\|$, where $\Omega = \{M | \|M\|_0 = \lfloor mp \rfloor; M_{ij} = 0, \forall i \neq j; M_{ii} \in \{0, 1\}\}$ and Π_Ω denotes the projection operator onto the feasible region Ω which is an L_0 ball. Specifically, **DiffPruning** (Mallya, Davis, and Lazebnik 2018; Sanh, Wolf, and Rush 2020; Guo, Rush, and Kim 2021; Lagunas et al. 2021) proposes to model the parameter selection mask as a Bernoulli random variable and optimize the variable with a reparametrization method. It then projects the mask onto M's feasible region Ω and do the optimization alternately. Therefore, according to Definition 1, it is also a sparse fine-tuned model. **ChildPruning** (Xu et al. 2021; Mostafa and Wang 2019) proposes to iteratively train the full model parameters and then calculates the projected mask to find the child network. Therefore, it also agrees with the sparse fine-tuned model's definition.

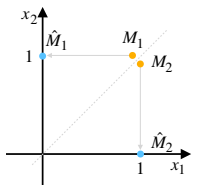

Figure 2: Projection discontinuity problem.

Projection Discontinuity Problem. Though projection-based methods can utilize task-specific data information, such kind of methods suffers from the projection discontinuity problem. Specifically, the feasible region Ω (the L_0 ball) of M is non-convex. Therefore, it does not have the non-expansion property which is generally guaranteed for projection onto a closed convex set. As a result, a small perturbation on M can lead to a totally different projection. For example, as illustrated in Fig. 2, suppose that $p = 0.5$ and $M_1 = \text{diag}\{0.99, 1\}, M_2 = \text{diag}\{1, 0.99\}$. Though $M_1 \approx M_2$, we have $\Pi_\Omega(M_1) = \text{diag}\{0, 1\}$ while $\Pi_\Omega(M_2) = \text{diag}\{1, 0\}$, which is quite different. Consequently, the projection is very sensitive to the parameters updating noise. As a result, it is hard to keep consistent with the previous parameters selection which leads to a big change for the parameters selection. Such inconsistency will impair the overall performance.

3 Theoretical Analysis of the Sparse Fine-tuned Model

Suppose that we have a pre-trained model $\mathcal{M}^0$ with parameters θ^0, we fine-tune the sparse fine-tuned model $\mathcal{M}$ by updating only $\lfloor pm \rfloor$ parameters. We will first show that sparsity implies a regularization of the original model. Then, we prove that if a model is a sparse fine-tuned model, the model stability can benefit from the sparsity. Next, we give a theoretical analysis of the model generalization error bound and show that sparsity contributes to reducing the generalization error. It should be noted that in the proofs, we only use properties from Definition 1. Therefore, our theory is applicable to all model categories (random approaches, rule-based approaches, and projection-based approaches) that agrees with

Definition 1.

3.1 Sparse Fine-tuned Model as a Regularizer

As analyzed in section 2.2, most of the models choose the parameter mask M with different approaches and optimize the parameters θ accordingly. Here, we treat the matrix M as a given parameter and denote $\theta = \theta^0 + M\Delta\theta$. The sparse fine-tuned optimization in Problem (1) can be reformulated as:

$$\begin{aligned} &\min_{\theta} \mathcal{L}(\theta) \\ &s.t.\ \|(I-M)(\theta-\theta^0)\|^2 = 0, \end{aligned} \tag{2}$$

where $M = \text{diag}\{M_{11}, \cdots, M_{mm}\}$ is a diagonal matrix with $M_{ii} \in \{0, 1\}$. By Lagrangian duality, solving Problem (2) is equivalent to solving the following problem:

$$\bar{L} = \min_{\theta} \max_{\lambda} \mathcal{L}(\theta) + \lambda \|(I-M)(\theta-\theta^0)\|^2. \tag{3}$$

Then, we derive a new regularized problem with the following proposition.

Proposition 1. *Optimizing Problem (2) implies to optimizing the upper bound $\bar{L}$ of the following regularized problem:*

$$L_R = \min_{\theta} \mathcal{L}(\theta) + \|(I-M)(\theta-\theta^0)\|^2 \leq \bar{L}. \tag{4}$$

The proof can be found in Appendix A.1 (Fu et al. 2022). It can be concluded that optimizing Problem (2) is the same as optimizing the upper bound of the original loss function $\mathcal{L}(\theta)$ with a regularization term $\|(I-M)(\theta-\theta^0)\|^2$. We will show later that such regularization contributes to the stability of the sparse fine-tuned model.

3.2 Stability Analysis

Stability has been studied in a lot of previous research works (Bousquet and Elisseeff 2002; Shalev-Shwartz et al. 2010; Shalev-Shwartz and Ben-David 2014; Hardt, Recht, and Singer 2016; Kuzborskij and Lampert 2018; Charles and Papailiopoulos 2018; Fu et al. 2021) in many different forms. We focus on one of the commonly used notions, namely, the Pointwise Hypothesis Stability (PHS) which focuses on analyzing the change of model output after a training sample is removed. Following (Charles and Papailiopoulos 2018), we denote the original training data as $S = \{z_1, \cdots, z_n\}$ and the dataset without one sample as $S^i = S \backslash z_i = \{z_1, \cdots, z_{i-1}, z_{i+1}, \cdots, z_n\}$, where z_i is the ith training sample. We also define $i \sim U(n)$ as a sampling procedure from a uniform distribution with n samples. $\mathcal{A}(S)$ is defined as model parameters obtained by running algorithm $\mathcal{A}$ on data S.

Definition 3 (Pointwise Hypothesis Stability, (Bousquet and Elisseeff 2002))**.** *We say that a learning algorithm $\mathcal{A}$ has **pointwise hypothesis stability** ϵ with respect to a loss function ℓ, if*

$$\mathbb{E}_{S,i\sim U(n)}[|\ell(\mathcal{A}(S^i), z_i) - \ell(\mathcal{A}(S), z_i)|] \leq \epsilon. \tag{5}$$

Here, $\ell(\theta, z_i)$ is the single sample loss for z_i when the model parameter is θ. We assume that $\mathcal{A}(S^i)$ is close to $\mathcal{A}(S)$. As $\mathcal{A}(S)$ is the optimal solution, the Hessian matrix at $\mathcal{A}(S)$ is a positive-semidefinite matrix. We can derive our bound for PHS in the following theorem.

Theorem 1 (Stability)**.** *If the loss function ℓ is $\rho-$Lipschitz, $\mathcal{A}(S^i)$ is close to $\mathcal{A}(S)$, the Hessian matrix $\nabla^2\mathcal{L}(\mathcal{A}(S))$ at $\mathcal{A}(S)$ is positive-semidefinite with a singular value decomposition $U\,\text{diag}(\Lambda)U^{-1}$, $\Lambda = \{\Lambda_1, \cdots, \Lambda_m\}$ and $\Lambda_{min} = \min\{\Lambda_1, \cdots, \Lambda_m\}$, then the expectation of the loss $\mathbb{E}_M L_R$ has a pointwise hypothesis stability as:*

$$\mathbb{E}_{S,i\sim U(n)}[|\ell(\mathcal{A}(S^i), z_i) - \ell(\mathcal{A}(S), z_i)|] \leq \frac{2\rho^2}{(\Lambda_{min} + 2(1-p))n}. \tag{6}$$

The proof can be found in Appendix A.2 (Fu et al. 2022). It can be observed from Theorem 1 that as the sparsity parameter p decreases, the upper bound also decreases. Therefore, sparse models imply better stability which explains most of the empirical results observed in many recent works (He et al. 2021b; Lee, Cho, and Kang 2019; Houlsby et al. 2019; Zaken, Ravfogel, and Goldberg 2021; Sung, Nair, and Raffel 2021; Liu et al. 2021; Ding et al. 2022). It should also be noted that if p is small enough, the upper bound will not change significantly as p continues to decrease. This is because in this case, the denominator is dominated by Λ_{min} which is related to the landscape of the function. Empirically, if the sparsity is too small, the landscape will heavily depend on how the parameters are chosen and thus the stability is impaired.

3.3 Generalization Analysis

With the bound for the stability, we can then get the generalization error bound for the sparse fine-tuned model.

Theorem 2 (Generalization)**.** *We denote the generalization error as $R(\mathcal{A}, S) = \mathbb{E}_z\, \ell(\mathcal{A}(S), z)$ and the empirical error as $\hat{R}(\mathcal{A}, S) = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathcal{A}(S), z)$. Then, for some constant C, we have with probability $1-\delta$,*

$$R(\mathcal{A}, S) \leq \hat{R}(\mathcal{A}, S) + \sqrt{\frac{C^2 + \frac{24C\rho^2}{\Lambda_{min}+2(1-p)}}{2n\delta}}. \tag{7}$$

The proof can be found in Appendix A.3 (Fu et al. 2022). This result shows that the generalization error upper bound becomes smaller as the fine-tuned parameters become sparser. Intuitively, if a model is stable, a perturbation makes less effect on the model and the model is less likely to overfit. It should be noted that the generalization error bound is determined by both the empirical error $\hat{R}(\mathcal{A}, S)$ and sparsity. Therefore, as the mask becomes sparser, even though the second term decreases, the training error $\hat{R}(\mathcal{A}, S)$ will possibly increase when the tunable parameters are not enough to fit the data. Consequently, as the sparsity decreases, the generalization error will first decrease and then increase. We will further examine this conjecture in experiments.

4 Second-order Approximation Method

In Section 3, we theoretically prove the effectiveness of sparsity in fine-tuning. However, it still remains a problem of

how to choose the tunable parameters. As discussed in Section 2.2, the random and the rule-based approaches are robust to noise perturbation as the tunable parameters are fixed during training. However, these methods tune the same parameters on all kinds of tasks without utilizing the information from the task-specific data. On the other hand, the projection-based approaches solve this problem by getting full utilization of the data information but they suffer from the projection discontinuity problem. The noise in the parameter may change the selection of the parameters frequently, thus making the optimization procedure unstable.

To solve the problems, we propose a novel Second-order Approximation Method (SAM), namely, utilizing the data information to help decide the parameter mask while avoiding the projection discontinuity problem. Instead of choosing the parameters randomly or simply by some rules, we propose a novel second-order approximation of Problem (1) to make the optimization target analytically solvable. Then, we directly get the optimal solution for the parameter mask M and fix the mask to train the other parameters θ. Specifically, as indicated by Radiya-Dixit and Wang (2020), the fine-tuned parameters are close to the pre-trained parameters. We can approximate the loss function with its second-order Taylor expansion as $\mathcal{L}(\theta^0+M\Delta\theta) \approx \mathcal{L}(\theta^0)+\nabla\mathcal{L}(\theta^0)^{\mathrm{T}}M\Delta\theta+\frac{1}{2}(M\Delta\theta)^{\mathrm{T}}HM\Delta\theta$. Unfortunately, the Hessian matrix H is expensive to compute especially for a large neural model. To solve this problem, we adopt the widely used technique (Bishop and Nasrabadi 2006; Xu, Roosta, and Mahoney 2020; Yao et al. 2021) of approximating the Hessian matrix with a diagonal matrix denoted as $H = \mathrm{diag}\{h_1, h_2, \cdots, h_n\}$. We also assume that H is positive semidefinite as the pre-trained weights is close to the global minimizer (Radiya-Dixit and Wang 2020) in each downstream task. Then, Problem (1) can be reformulated as:

$$\begin{aligned} &\min_{\Delta\theta} \mathcal{L}(\theta^0)+\nabla\mathcal{L}(\theta^0)^{\mathrm{T}}M\Delta\theta+\frac{1}{2}(M\Delta\theta)^{\mathrm{T}}HM\Delta\theta \\ &s.t. \quad \|M\|_0=\lfloor mp\rfloor; \quad M_{ij}=0,\forall i\neq j; \quad M_{ii}\in\{0,1\}. \end{aligned} \tag{8}$$

With the above setup, we can get the optimal parameter mask M for Problem (8) based on the following theorem:

Theorem 3. *If* $\hat{M}_{ii} = \mathbb{1}(\sum_{j=1}^{m}\mathbb{1}(|\frac{\nabla\mathcal{L}(\theta^0)_i^2}{h_i}| > |\frac{\nabla\mathcal{L}(\theta^0)_j^2}{h_j}|) \geq m-\lfloor mp\rfloor)$*, where* $\nabla\mathcal{L}(\theta^0)_i$ *is the* i*th element of the gradient vector* $\nabla\mathcal{L}(\theta^0)$*, then*

$$\inf_{\Delta\theta}\mathcal{L}(\theta^0+\hat{M}\Delta\theta) \leq \inf_{\substack{\Delta\theta,\|M\|_0=\lfloor mp\rfloor;\\ M_{ij}=0,\forall i\neq j;M_{ii}\in\{0,1\}}} \mathcal{L}(\theta^0+M\Delta\theta). \tag{9}$$

The proof can be found in Appendix A.4 (Fu et al. 2022). It can be observed that selecting features according to Theorem 3 achieves the minimal value of the approximation in Problem (8). The remaining problem is how to calculate the diagonal of the Hessian matrix. Unfortunately, calculating the diagonal Hessian is as complex as calculating the whole Hessian. To solve this problem, instead of minimizing the target function in Problem 8, we propose to optimize its upper bound

$$\begin{aligned} &\min_{\Delta\theta} \mathcal{L}(\theta^0)+\nabla\mathcal{L}(\theta^0)^{\mathrm{T}}M\Delta\theta+\frac{1}{2}(M\Delta\theta)^{\mathrm{T}}DM\Delta\theta \\ &s.t. \quad \|M\|_0=\lfloor mp\rfloor; \quad M_{ij}=0,\forall i\neq j; \quad M_{ii}\in\{0,1\}. \end{aligned} \tag{10}$$

where $D = \mathrm{diag}\{|\lambda_{max}|, |\lambda_{max}|, \cdots, |\lambda_{max}|\}$ and λ_{max} is the maximal eigenvalue of H. This can be directly calculated from the Rayleigh quotient that $\forall x \neq 0, x^THx \leq x^Tx\lambda_{max} \leq x^Tx|\lambda_{max}| = x^TDx$. Therefore, the SAM algorithm is quite straightforward based on Theorem 3. We first get the gradient $\nabla\mathcal{L}(\theta^0)_i$ for the ith parameter θ_i. Then, we calculate $|\nabla\mathcal{L}(\theta^0)_i^2|$ and take the top $\lfloor mp\rfloor$ parameters to optimize. We will not change the selected parameters during the optimization procedure.

5 Experiments

5.1 Experimental Setup

Following most previous works (Lee, Cho, and Kang 2019; Dodge et al. 2020; Xu et al. 2021), we use the original development set as the test set to report the scores as the original test sets are only available via the leaderboard with a limited submission number. Different from many previous works that train models without validation, we split the original training set by randomly sampling 10% as the new development set while using the remaining 90% samples to train the model. Instead of training the model for fixed epoch number, we use the new development set to do an early stop training by setting the tolerance for all models to 40. We build our models with the jiant[1] (Phang et al. 2020) framework and test our models on several GLUE (Wang et al. 2018) and SuperGLUE (Wang et al. 2019) tasks. Following the setting of Lee, Cho, and Kang (2019); Xu et al. (2021), we choose several tasks including Corpus of Linguistic Acceptability (CoLA) (Warstadt, Singh, and Bowman 2019), Semantic Textual Similarity Benchmark (STS-B) (Cer et al. 2017), Microsoft Research Paraphrase Corpus (MRPC) (Dolan and Brockett 2005), Recognizing Textual Entailment (RTE) (Dagan, Glickman, and Magnini 2005; Bentivogli et al. 2009), Commitment Bank (CB) (De Marneffe, Simons, and Tonhauser 2019), Choice of Plausible Alternatives (COPA) (Roemmele, Bejan, and Gordon 2011), and Winograd Schema Challenge (WSC) (Levesque, Davis, and Morgenstern 2012). We compare our model with many strong baseline models including Random, Mixout, BitFit, MagPruning, Adapter, LoRA, DiffPruning, and ChildPruning. The details of these models have been extensively discussed in Section 2.2 and we adopt the same evaluation methods as Wang et al. (2018, 2019) to evaluate the models. We run each experiment 10 times with different random seeds and report the scores with corresponding standard deviations. As many previous experiments are conducted under different settings, we re-implement all the baseline models with the jiant framework to give a fair comparison. For the Adapter and LoRA model, we incorporate AdapterHub [2] (Pfeiffer et al. 2020) and loralib [3] into jiant. Following the setting of Guo, Rush, and Kim (2021), we set

[1] https://jiant.info/

[2] https://adapterhub.ml/

[3] https://github.com/microsoft/LoRA

	CoLA	STS-B	MRPC	RTE	CB	COPA	WSC	AVG
FullTuning	58.36±1.74	89.80±0.52	**89.55**$_{[1]}$±0.81	76.03±2.14	88.93$_{[2]}$±2.37$_{[2]}$	67.70±4.41	53.10±6.18	74.78±2.60
Random	58.35±1.05$_{[2]}$	89.81±**0.11**$_{[1]}$	88.73±0.80	72.71±3.23	**90.54**$_{[1]}$±3.39	68.80±2.64	52.88±5.97	74.55±2.46
MixOut	58.66±1.96	90.15$_{[3]}$±0.17	88.69±0.60$_{[3]}$	**77.55**$_{[1]}$±**1.64**$_{[1]}$	86.51±4.13	71.30±4.84	52.98±6.78	75.12$_{[3]}$±2.88
Bitfit	56.67±1.45	90.12±0.14$_{[3]}$	87.35±0.58$_{[2]}$	72.74±2.47	86.96±3.20	71.20±3.79	55.10±5.39	74.31±2.43
MagPruning	56.57±2.47	90.30$_{[2]}$±0.14$_{[3]}$	88.09±0.79	73.53±1.84$_{[3]}$	81.25±3.50	71.50$_{[3]}$±2.46$_{[2]}$	55.67±**2.73**$_{[1]}$	73.85±1.99$_{[2]}$
Adapter	**62.11**$_{[1]}$±1.22$_{[3]}$	90.05±0.13$_{[2]}$	89.29$_{[3]}$±0.60$_{[3]}$	76.93$_{[3]}$±2.05	87.32±4.62	69.50±2.54$_{[3]}$	57.02$_{[2]}$±5.27	76.03$_{[2]}$±2.35
LoRA	60.88$_{[3]}$±1.48	87.19±0.51	89.53$_{[2]}$±0.62	76.97$_{[2]}$±1.92	84.64±3.76	69.70±2.83	56.84$_{[3]}$±4.52	75.11±2.24$_{[3]}$
DiffPruning	58.53±1.49	89.59±0.34	78.79±6.09	69.93±7.87	86.25±2.65$_{[3]}$	72.10$_{[2]}$±2.91	53.37±3.60$_{[3]}$	72.65±3.57
ChildPruning	60.00±1.29	89.97±1.51	87.19±3.86	75.76±4.38	86.61±3.22	69.40±4.00	55.59±3.81	74.93±3.15
SAM	60.89$_{[2]}$±**0.96**$_{[1]}$	**90.59**$_{[1]}$±0.14$_{[3]}$	88.84±**0.49**$_{[1]}$	76.79±1.72$_{[2]}$	88.93$_{[2]}$±**1.75**$_{[1]}$	**74.30**$_{[1]}$±**2.45**$_{[1]}$	**59.52**$_{[1]}$±3.08$_{[2]}$	**77.12**$_{[1]}$±**1.51**$_{[1]}$

Table 1: Main experiment. We run each experiment 10 times with different random seeds and report means and standard deviations. The number in the bracket is the rank for the scores in the corresponding column. Due to the space limit, we attach the training time analysis and the significance test in Appendix A.5 and A.7 of Fu et al. (2022).

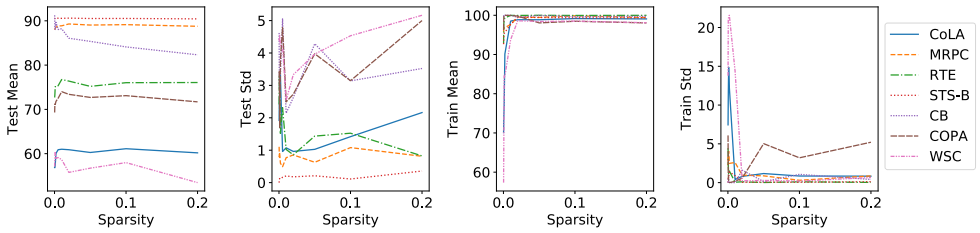

Figure 3: Effectiveness of sparsity.

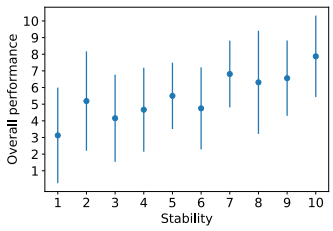

Figure 4: Relation between stability and overall Performance.

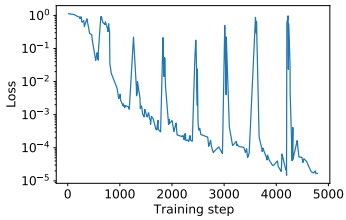

Figure 5: Projection discontinuity problem.

the sparsity to 0.005 for all models for a fair comparison. In SAM, we calculate $\nabla\mathcal{L}(\theta^0)_i$ by accumulating the gradient for a few burn-in steps as we cannot load all the training data into memory, the burn-in steps are chosen from $\{500, 600, 700, 800, 900, 1000, 2000\}$ on the development set as a hyper-parameter (Fu et al. 2022). We fine-tune the models based on RoBERTa-base (Liu et al. 2019) provided by transformers[4] toolkit (Wolf et al. 2020) and we run the models on NVIDIA TITAN RTX GPU with 24GB memory.

5.2 Experimental Results

Main Experiment. The main experimental results are illustrated in Table 1. We can draw the following conclusions based on the results: (1) Most of the parameter-efficient models achieve better performance than the FullTuning model which is also consistent with the observations in many previous works. This observation supports our theoretical analysis in Theorem 2 that the parameter-efficient model has better generalization capability. (2) Most of the parameter-efficient models are more stable than the FullTuning model. This observation is also consistent with many empirical results in previous works and it also supports our theoretical stability analysis in Theorem 1. (3) It is interesting to note that even the Random model outperforms the FullTuning model. It shows that sparsity itself contributes to improving the performance. (4) Our proposed SAM model outperforms several baseline models in several tasks and it ranks in the top 3 of most tasks. This observation validates the effectiveness of our parameter selecting method discussed in Theorem 3. Due to the space limit, we attach the training time analysis and the significance test in Appendix A.5 and A.7 of Fu et al. (2022).

Projection Discontinuity Problem. To give an intuitive illustration of the projection discontinuity problem in projection-based approaches, we plot the training curve of the DiffPruning method on the CB task. As illustrated in Fig. 5, we adjust the mask every 600 training steps. It can be observed from the figure that each time we change the mask, the training error will go back to almost the same value as its initial loss. This result shows that changing the mask severely affects the training procedure due to the projection discontinuity problem.

Relation between Stability and Overall Performance. Theorem 2 shows that stability implies better generalization. To further validate this, we illustrate how the stability ranks and the overall performance ranks are correlated in the main experiment. As shown in Fig. 4, the x-axis is the stability rank in each main experiment while the y-axis is

[4] https://huggingface.co/docs/transformers/model_doc/roberta

	CoLA	STS-B	MRPC	RTE	CB	COPA	WSC	AVG
FullTuning	60.74$_{[2]}$±1.89	90.11$_{[3]}$±0.26	88.74$_{[3]}$±1.08	75.37$_{[3]}$±1.93	84.29±4.21	69.60±2.94	54.81±7.51	74.81±2.83
Random	56.00±1.84	89.79±0.20	88.57±0.72$_{[2]}$	73.00±2.01	89.29$_{[2]}$±4.92	70.30±2.69$_{[3]}$	56.87±4.29	74.83±2.38
MixOut	60.37$_{[3]}$±1.33	90.11$_{[3]}$±0.13$_{[3]}$	88.50±0.78$_{[3]}$	74.51±1.28$_{[2]}$	83.75±3.14$_{[3]}$	69.40±4.80	57.88±6.15	74.93$_{[3]}$±2.52
Bitfit	55.26±**0.78**$_{[1]}$	89.98±0.15	86.87±1.27	71.36±1.71	**91.29**$_{[1]}$±**2.27**$_{[1]}$	71.80$_{[2]}$±3.92	55.29±9.90	74.55±2.86
MagPruning	56.45±1.80	90.26$_{[2]}$±**0.11**$_{[1]}$	87.35±0.85	72.24±2.14	84.46±3.58	69.20±3.54	**59.71**$_{[1]}$±3.88$_{[2]}$	74.24±2.27$_{[2]}$
Adapter	60.05±1.88	89.92±0.19	**88.79**$_{[1]}$±0.80	74.55±1.80	86.61±4.97	68.80±**2.40**$_{[1]}$	55.63±7.53	74.91±2.79
LoRA	**61.46**$_{[1]}$±1.27$_{[3]}$	86.73±0.38	88.28±1.06	**76.46**$_{[1]}$±1.34$_{[3]}$	88.69$_{[3]}$±5.32	67.75±2.49$_{[2]}$	58.85$_{[3]}$±4.27$_{[3]}$	75.46$_{[2]}$±2.30$_{[3]}$
DiffPruning	58.36±1.45	89.52±0.27	77.46±5.31	70.76±9.01	85.18±2.65$_{[2]}$	70.40$_{[3]}$±3.07	55.38±4.30	72.44±3.72
ChildPruning	59.40±2.30	89.33±3.23	88.43±0.80	75.11±2.87	85.71±4.07	70.30±4.54	54.04±7.24	74.62±3.58
SAM	59.52±1.12$_{[2]}$	**90.45**$_{[1]}$±0.12$_{[2]}$	**88.79**$_{[1]}$±**0.69**$_{[1]}$	75.74$_{[2]}$±**1.27**$_{[1]}$	86.79±4.39	**74.00**$_{[1]}$±2.79	59.52$_{[2]}$±**3.32**$_{[1]}$	**76.40**$_{[1]}$±**1.96**$_{[1]}$

Table 2: Data perturbation stability. The setting is the same as the main experiments except that we run the experiments on different sampled datasets. Due to the space limit, we attach the significance test in Appendix A.7 of Fu et al. (2022).

the corresponding overall performance rank. For each vertical line of a specific stability rank, the dot indicates the overall performance mean rank value while the line length indicates the standard deviation. It can be observed from the figure that the two ranks are positively correlated indicating that stabler models usually have better generalization capability. To further show the relationship between the stability and the overall performance, we calculate Spearman's rank correlation coefficient (Spearman 1904) for the two ranks. It can be denoted as $\rho = \frac{\text{cov}(R(S),R(V))}{\sigma_{R(S)}\sigma_{R(V)}}$, where $R(S)$ and $R(V)$ are the rank variables, $\text{cov}(R(S), R(V))$ is the covariance of $R(S)$ and $R(V)$ while $\sigma_{R(V)}$ is the standard deviation of the rank variable V. We have $\rho = 0.4356$ with p-value$= 0.000014 < 0.05$ indicating that the correlation between the two rank variables is significant.

Effectiveness of Sparsity. To further verify our theoretical analysis in Theorem 1 and Theorem 2, we conduct a new experiment to show how the overall performance and the stability change as we change the sparsity. We change the sparsity of the SAM model in $\{$0.0002, 0.0005, 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2$\}$ and plot the relationship between sparsity and the mean/standard deviation in both the test set and training set. The results are shown in Fig. 3. It can be concluded from the results that (1) as the sparsity ratio decreases, the mean and the standard deviation of most tasks also decrease which means the models become more stable with better generalization. This observation is consistent with our bound in Theorem 1 and Theorem 2. (2) If the sparsity ratio drops below a certain threshold, the models become quite unstable and the performance also sees a sharp drop. This is because the empirical error increases drastically which can be observed in the Train Mean and Train Std scores in Fig. 3. At the same time, under such circumstances, decreasing the sparsity ratio cannot further lower the bound effectively. Therefore, such observation is also consistent with our discussion in Theorem 1 and Theorem 2.

Data Perturbation Stability. In the main experiment, we use different random seeds. However, it is unknown whether the performance is still stable if we have a perturbation on the dataset. We conduct a new experiment to verify the data perturbation stability by training the model on 10 different training sets. Each of them is made by randomly removing 10% training samples from our original training set. The results are shown in Table 2. It can be observed from the results that the data perturbation stability performance is similar to the main experiment and our proposed SAM model still has the best data perturbation stability as well as the overall performance among all the models.

6 Related Works

Fine-tuning on a pre-trained model (Peters et al. 2018; Devlin et al. 2019; Lan et al. 2020; Radford et al. 2018, 2019; Brown et al. 2020; Dong et al. 2019; Liu et al. 2022) has shown to be very promising in recent years. However, fine-tuning the full model yields a large model with the same size for each task and many works indicate that fine-tuning the full model is unstable (Devlin et al. 2019; Lee, Cho, and Kang 2019; Zhu et al. 2020; Dodge et al. 2020; Mosbach, Andriushchenko, and Klakow 2020; Zhao et al. 2021). To solve this problem, many researchers propose the parameter-efficient methods which only fine-tune a small part of the pre-trained parameters. These methods are found to be more stable than fine-tuning the full model (He et al. 2021b; Lee, Cho, and Kang 2019; Houlsby et al. 2019; Zaken, Ravfogel, and Goldberg 2021; Sung, Nair, and Raffel 2021; Liu et al. 2021). Currently, there is still no previous work providing a theoretical analysis for the stability of the parameter-efficient models.

7 Conclusions

In this paper, we propose to understand the effectiveness of the parameter-efficient fine-tuning models. Depending on how the tunable parameters are chosen, we first categorize most of the models into three categories, namely, random approaches, rule-based approaches, and projection-based approaches. Then, we show that all models in the three categories are sparse fine-tuned models and we give a theoretical analysis of the stability and the generalization error. We further show that the random approaches and the rule-based methods do not utilize the task data information while the projection-based approaches suffer from the projection discontinuity problem. We propose a novel SAM model to alleviate both problems and we conduct extensive experiments to show the correctness of our theoretical analysis and the effectiveness of our proposed models.

Acknowledgments

The authors gratefully acknowledge the support of the funding from UKRI under project code ES/T012277/1.

References

Bentivogli, L.; Clark, P.; Dagan, I.; and Giampiccolo, D. 2009. The Fifth PASCAL Recognizing Textual Entailment Challenge. In *TAC*.

Bishop, C. M.; and Nasrabadi, N. M. 2006. *Pattern recognition and machine learning*, volume 4. Springer.

Bousquet, O.; and Elisseeff, A. 2002. Stability and generalization. *JMLR*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *NeurIPS*.

Cer, D.; Diab, M.; Agirre, E.; Lopez-Gazpio, I.; and Specia, L. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Charles, Z.; and Papailiopoulos, D. 2018. Stability and generalization of learning algorithms that converge to global optima. In *ICML*.

Dagan, I.; Glickman, O.; and Magnini, B. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*. Springer.

De Marneffe, M.-C.; Simons, M.; and Tonhauser, J. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*.

Ding, N.; Qin, Y.; Yang, G.; Wei, F.; Yang, Z.; Su, Y.; Hu, S.; Chen, Y.; Chan, C.-M.; Chen, W.; et al. 2022. Delta Tuning: A Comprehensive Study of Parameter Efficient Methods for Pre-trained Language Models. *arXiv preprint arXiv:2203.06904*.

Dodge, J.; Ilharco, G.; Schwartz, R.; Farhadi, A.; Hajishirzi, H.; and Smith, N. A. 2020. Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping.

Dolan, B.; and Brockett, C. 2005. Automatically constructing a corpus of sentential paraphrases. In *IWP*.

Dong, L.; Yang, N.; Wang, W.; Wei, F.; Liu, X.; Wang, Y.; Gao, J.; Zhou, M.; and Hon, H.-W. 2019. Unified language model pre-training for natural language understanding and generation. *NeurIPS*.

Elisseeff, A.; Evgeniou, T.; Pontil, M.; and Kaelbing, L. P. 2005. Stability of Randomized Learning Algorithms. *JMLR*.

Fedus, W.; Zoph, B.; and Shazeer, N. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*.

Fu, Z.; Lam, W.; So, A. M.-C.; and Shi, B. 2021. A theoretical analysis of the repetition problem in text generation. In *AAAI*.

Fu, Z.; Yang, H.; So, A. M.-C.; Lam, W.; Bing, L.; and Collier, N. 2022. On the Effectiveness of Parameter-Efficient Fine-Tuning. *arXiv preprint arXiv:2211.15583. (Full version with Appendix on arXiv)*.

Guo, D.; Rush, A. M.; and Kim, Y. 2021. Parameter-Efficient Transfer Learning with Diff Pruning. In *ACL*.

Han, S.; Mao, H.; and Dally, W. J. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. *NeurIPS*.

Hardt, M.; Recht, B.; and Singer, Y. 2016. Train faster, generalize better: Stability of stochastic gradient descent. In *ICML*.

He, J.; Zhou, C.; Ma, X.; Berg-Kirkpatrick, T.; and Neubig, G. 2021a. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*.

He, R.; Liu, L.; Ye, H.; Tan, Q.; Ding, B.; Cheng, L.; Low, J.; Bing, L.; and Si, L. 2021b. On the Effectiveness of Adapter-based Tuning for Pretrained Language Model Adaptation. In *ACL*.

Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; De Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; and Gelly, S. 2019. Parameter-efficient transfer learning for NLP. In *ICML*.

Hu, E. J.; yelong shen; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR*.

Karimi Mahabadi, R.; Henderson, J.; and Ruder, S. 2021. Compacter: Efficient Low-Rank Hypercomplex Adapter Layers. *NeurIPS*.

Kenton, J. D. M.-W. C.; and Toutanova, L. K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*.

Kuzborskij, I.; and Lampert, C. 2018. Data-dependent stability of stochastic gradient descent. In *ICML*.

Lagunas, F.; Charlaix, E.; Sanh, V.; and Rush, A. M. 2021. Block Pruning For Faster Transformers. In *EMNLP*.

Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; and Soricut, R. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *ICLR*.

Lee, C.; Cho, K.; and Kang, W. 2019. Mixout: Effective Regularization to Finetune Large-scale Pretrained Language Models. In *ICLR*.

Lee, J.; Park, S.; Mo, S.; Ahn, S.; and Shin, J. 2021. Layer-adaptive Sparsity for the Magnitude-based Pruning. In *ICLR*.

Levesque, H.; Davis, E.; and Morgenstern, L. 2012. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L.

2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *ACL*.

Liu, H.; Tam, D.; Muqeeth, M.; Mohta, J.; Huang, T.; Bansal, M.; and Raffel, C. 2022. Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning.

Liu, X.; Ji, K.; Fu, Y.; Du, Z.; Yang, Z.; and Tang, J. 2021. P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks. *arXiv preprint arXiv:2110.07602*.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach.

Mahabadi, R. K.; Ruder, S.; Dehghani, M.; and Henderson, J. 2021. Parameter-efficient Multi-task Fine-tuning for Transformers via Shared Hypernetworks. In *ACL*.

Mallya, A.; Davis, D.; and Lazebnik, S. 2018. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *ECCV*.

Mao, Y.; Mathias, L.; Hou, R.; Almahairi, A.; Ma, H.; Han, J.; Yih, W.-t.; and Khabsa, M. 2021. Unipelt: A unified framework for parameter-efficient language model tuning. *arXiv preprint arXiv:2110.07577*.

Mosbach, M.; Andriushchenko, M.; and Klakow, D. 2020. On the Stability of Fine-tuning BERT: Misconceptions, Explanations, and Strong Baselines. In *ICLR*.

Mostafa, H.; and Wang, X. 2019. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *ICML*.

Panahi, A.; Saeedi, S.; and Arodz, T. 2021. Shapeshifter: a Parameter-efficient Transformer using Factorized Reshaped Matrices. *NeurIPS*.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep Contextualized Word Representations. In *NAACL*.

Pfeiffer, J.; Rücklé, A.; Poth, C.; Kamath, A.; Vulić, I.; Ruder, S.; Cho, K.; and Gurevych, I. 2020. AdapterHub: A Framework for Adapting Transformers. In *EMNLP*.

Phang, J.; Yeres, P.; Swanson, J.; Liu, H.; Tenney, I. F.; Htut, P. M.; Vania, C.; Wang, A.; and Bowman, S. R. 2020. `jiant` 2.0: A software toolkit for research on general-purpose text understanding models. http://jiant.info/.

Radford, A.; Narasimhan, K.; Salimans, T.; and Sutskever, I. 2018. Improving language understanding by generative pre-training.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8).

Radiya-Dixit, E.; and Wang, X. 2020. How fine can fine-tuning be? learning efficient language models. In *AISTATS*.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *JMLR*.

Roemmele, M.; Bejan, C. A.; and Gordon, A. S. 2011. Choice of Plausible Alternatives: An Evaluation of Commonsense Causal Reasoning. In *AAAI spring symposium*.

Rücklé, A.; Geigle, G.; Glockner, M.; Beck, T.; Pfeiffer, J.; Reimers, N.; and Gurevych, I. 2021. AdapterDrop: On the Efficiency of Adapters in Transformers. In *EMNLP*.

Sanh, V.; Wolf, T.; and Rush, A. 2020. Movement pruning: Adaptive sparsity by fine-tuning. *NeurIPS*.

Shalev-Shwartz, S.; and Ben-David, S. 2014. *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Shalev-Shwartz, S.; Shamir, O.; Srebro, N.; and Sridharan, K. 2010. Learnability, stability and uniform convergence. *JMLR*.

Spearman, C. 1904. The proof and measurement of association between two things. *The American journal of psychology*, 15(1).

Sung, Y.-L.; Nair, V.; and Raffel, C. A. 2021. Training Neural Networks with Fixed Sparse Masks. *NeurIPS*.

Wang, A.; Pruksachatkun, Y.; Nangia, N.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *NeurIPS*.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *EMNLP Workshop BlackboxNLP*.

Warstadt, A.; Singh, A.; and Bowman, S. R. 2019. Neural network acceptability judgments. *TACL*, 7.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Scao, T. L.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. M. 2020. Transformers: State-of-the-Art Natural Language Processing. In *EMNLP: System Demonstrations*.

Xu, P.; Roosta, F.; and Mahoney, M. W. 2020. Newton-type methods for non-convex optimization under inexact hessian information. *Mathematical Programming*, 184(1).

Xu, R.; Luo, F.; Zhang, Z.; Tan, C.; Chang, B.; Huang, S.; and Huang, F. 2021. Raise a Child in Large Language Model: Towards Effective and Generalizable Fine-tuning. In *EMNLP*.

Yao, Z.; Gholami, A.; Shen, S.; Mustafa, M.; Keutzer, K.; and Mahoney, M. 2021. ADAHESSIAN: An Adaptive Second Order Optimizer for Machine Learning. In *AAAI*.

Zaken, E. B.; Ravfogel, S.; and Goldberg, Y. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.

Zhao, Z.; Wallace, E.; Feng, S.; Klein, D.; and Singh, S. 2021. Calibrate before use: Improving few-shot performance of language models. In *ICML*.

Zhu, C.; Cheng, Y.; Gan, Z.; Sun, S.; Goldstein, T.; and Liu, J. 2020. FreeLB: Enhanced Adversarial Training for Natural Language Understanding. In *ICLR*.