Stochastic Fine-Tuning of Language Models Using Masked Gradients

Mohammad Akbar-Tajari [1,2] [†] **Mohammad Taher Pilehvar** [3]
[1]Sharif University of Technology [3]Cardiff University, United Kingdom
[2]Tehran Institute for Advanced Studies, Iran
m.akbarTajari@gmail.com mp792@cam.ac.uk

Abstract

Large Language Models (LLMs) have emerged as the dominant paradigm in Natural Language Processing owing to their remarkable performance across various target tasks. However, naively fine-tuning them for specific downstream tasks often requires updating a vast number of parameters, resulting in high computational costs and overfitting when training data is limited. In this paper, we propose a novel approach, called *Stochastic Tuning*, that addresses these challenges by selectively updating a small subset of parameters in each step of the tuning process. Our approach is characterized by its customization of updates based on task-specific partial gradients with respect to stochastic sub-networks. The advantage of Stochastic Tuning over existing solutions lies in its ability to consider both parameter weights as well as forward values which guarantees a context-sensitive fine-tuning. Our experiments demonstrate that Stochastic Tuning outperforms existing lightweight fine-tuning methods, improving average performance by over two points on RoBERTa across several tasks in the GLUE benchmark while updating merely 0.08% of the model's parameters. The code for our implementation can be found at https://github.com/m-Tajari/StocTuning_LLMs.

1 Introduction and Related Work

Full fine-tuning of a pre-trained language model (PLM), a widely adopted approach in modern NLP, can be computationally expensive due to the need for updating all parameters of the model. By limiting the number of updatable parameters during the fine-tuning process, *lightweight mechanisms* reduce the computational cost by a large factor while retaining the performance. A class of lightweight methods augments models with small trainable modules. Adapters (Houlsby et al., 2019) are a prominent technique in this category. They inject a few modules into the transformer blocks, enabling comparable performance to the full fine-tuning scenario. Instead of introducing new modules, Prefix-Tuning (Li and Liang, 2021) optimizes virtual tokens, called the Prefix, which are prepended to the normal activation vectors of transformers. This method excels in low-data settings. In contrast to both these techniques, Low-Rank Adaptation (Hu et al., 2022, LoRA) avoids adding new parameters. Instead, it freezes the pre-trained ones and optimizes low-rank weights inserted into existing layers, achieving better results across various tasks. However, a drawback of all these methods is that they still require extra parameters to be injected into already large models.

According to the *lottery ticket hypothesis* (Frankle and Carbin, 2019), large transformer-based models consist of sparse sub-networks, fine-tuning of which results in competitive performance to the full fine-tuning. Various studies have tried to find optimal sub-networks which effectively transfer knowledge from a pre-trained model to different downstream tasks. Notably, BIas-Term FIne-Tuning (Ben Zaken et al., 2022, BitFit) freezes all the transformer-encoder parameters but the biases. Akbar-Tajari et al. (2022) take a step further to generalize this approach by fine-tuning similar transformer modules across layers, including LayerNorms, the expressive power of whose variants has been theoretically studied (Giannou et al., 2023). The results suggest that each transformer module can act as a winning ticket due to its ability in effectively transferring knowledge. However, while these localized fine-tuning techniques achieve acceptable performance across a variety of tasks, they rely too much on tuning predefined modules and neglect the importance of adapting parameters based on the specific requirements of the task at hand. This limitation prevents more fine-grained, task-specific parameter updates, which are

[†]Research conducted during my time as a Research Assistant at Tehran Institute for Advanced Studies, Iran.

crucial for optimal performance on diverse downstream tasks.

Another branch of research takes a non-localized approach. These methods form sub-networks where any parameter of the model can be updated, without considering their role in different transformer modules. Influenced by Dropout (Srivastava et al., 2014), Mixout (Lee et al., 2020) regularizes fine-tuning process by randomly freezing sub-networks during sequential iterations, ignoring the significance of parameters. In a more efficient way, Dynamic Parameter Selection (Zhang et al., 2022, DPS) adaptively selects promising sub-networks composed of important parameters. Despite being dynamically selected, these sub-networks usually contain a substantial portion of model's parameters. In addition, DPS only utilizes gradients of back-propagation without directly considering the value of parameters, which have successfully been used in similar pruning strategies (Lee et al., 2019).

To mitigate these issues, we propose *Stochastic Tuning*, a method that estimates the importance of individual parameters by using gradients of back-propagation and the value of parameters together with forwarded values. Based on the importance scores, our approach stochastically forms a task-specific binary mask of a predefined size. The optimization process is then constrained to sub-networks of parameters with highest importance scores. Stochastic Tuning utilizes both task-specific data and the encoded knowledge of pre-trained models to select sub-networks. This approach enables PLMs to seamlessly adapt to diverse tasks while retaining their inherent generalization capabilities. Moreover, the randomness in our method acts as a regularizer; therefore, it can prevent models form overfitting, bringing about better generalization.

Stochastic Tuning provides the following three key advantages compared to existing solutions: (1) It utilizes the encoded knowledge of PLMs by taking into account the value of parameters in the masking process, which ultimately reduces the time required for the fine-tuning process; (2) It tailors its selection of update sub-networks to the specific downstream task at hand; and (3) The stochasticity in the selection of the binary mask ensures that less important parameters also have the opportunity to participate in the fine-tuning process, bringing about regularization effect. Our experiments on the GLUE benchmark (Wang et al., 2019) demonstrate that Stochastic Tuning yields consistent improvements of about two points on average over previous state-of-the-art fine-tuning methods.

2 Stochastic Tuning

2.1 Background: Gradient Descent

Since the forward pass of our method and the related works being compared to in this study are similar to the full fine-tuning scenario, we focus our attention on back-propagation. Let $f_{\boldsymbol{\theta}_0} : \mathcal{X} \to \mathcal{Y}$ denote the PLM with parameters $\boldsymbol{\theta}_0 \in \mathbb{R}^n$. Fine-tuning methods try to solve the following optimization problem:

$$\boldsymbol{\theta}^\star \in \operatorname*{arg\,inf}_{\boldsymbol{\theta} \in \mathbb{R}^n} \mathcal{L}(\theta, \mathcal{D}),$$

where $\mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y}$ denotes the labeled dataset used for fine-tuning and

$$\mathcal{L}(\boldsymbol{\theta}, \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x}, y) \in \mathcal{D}} \ell\left(f_{\boldsymbol{\theta}}\left(\mathbf{x}\right), y\right)$$

represents the training loss for the model with parameters $\boldsymbol{\theta}$ and the loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$[1]. Stochastic Gradient Descent (SGD)[2] tries to find a solution by repeatedly updating parameters using the following rule:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \frac{\partial \mathcal{L}(\boldsymbol{\theta}_t, \mathcal{B}_t)}{\partial \boldsymbol{\theta}_t} \tag{1}$$

for $t \in \{0, 1, \cdots, T-1\}$ and $\mathcal{B}_t \subseteq \mathcal{D}$. Note that Equation (1) makes use of learning rate $\eta \in \mathbb{R}_+$ and in-batch training loss: $\mathcal{L}(\boldsymbol{\theta}_t, \mathcal{B}_t)$.

Fine-tuning sub-networks corresponds to imposing a restriction on the number of parameters that are modified in each step of SGD. Under these circumstances, the optimization problem can be formulated as:

$$\begin{aligned} &\boldsymbol{\theta}^\star \in \operatorname*{arg\,inf}_{\boldsymbol{\theta} \in \mathbb{R}^n} \mathcal{L}(\boldsymbol{\theta}, \mathcal{D}) \\ &\textbf{s.t.} \;\; ||\boldsymbol{\theta}||_0 \leq k, \end{aligned}$$

with $k \in \mathbb{R}_{++}$[3] denoting the number of parameters within the sub-network. Using indicator varibales

[1] $\mathbb{R}_+ = \{x \in \mathbb{R} : x \geq 0\}$.

[2] We employ the term SGD to denote its application in deep learning as mini-batch Stochastic Gradient Descent with momentum.

[3] $\mathbb{R}_{++} = \{x \in \mathbb{R} : x > 0\}$.

$\boldsymbol{\kappa} \in \{0,1\}^n$, the optimization problem may be expressed as:

$$\boldsymbol{\kappa}^\star, \boldsymbol{\theta}^\star \in \underset{\boldsymbol{\kappa}, \boldsymbol{\theta} \in \mathbb{R}^n}{\arg\inf}\ \mathcal{L}(\boldsymbol{\kappa} \odot \boldsymbol{\theta}, \mathcal{D})$$
$$\textbf{s.t.}\ \ \boldsymbol{\kappa} \in \{0,1\}^n, \ \ ||\boldsymbol{\kappa}||_0 \leq k,$$

where $\odot$ is the *Hadamard* product. By formulating the optimization problem as shown, we can easily think of attributing some importance scores to indicator variables, which will subsequently guide our selection of sub-networks in the following subsection. To fine-tune sub-networks by SGD, we have to use partial gradients. To do so, most methods utilize a binary mask $M_t \in \{0,1\}^n$ and change the update rule as follows:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \frac{\partial \mathcal{L}(\boldsymbol{\theta}_t, \mathcal{B}_t)}{\partial \boldsymbol{\theta}_t} \odot M_t. \qquad (2)$$

2.2 Methodology

Algorithm 1 presents the procedure behind Stochastic Tuning. Given a pre-trained language model with parameters $\boldsymbol{\theta}_0$ and a task-specific tuning dataset $\mathcal{D}$, Stochastic Tuning first forwards a batch of data into the model and keeps track of the average value passing through each connection.[4] At iteration t, we denote these values belonging to $\mathbb{R}^n$ with $\boldsymbol{\mu}_t$. After back-propagation, we use the importance function $\mathcal{I} : \mathbb{R}^{3n} \to \mathbb{R}^n_+$ to assign an importance score to individual parameters of the model as follows:

$$\mathcal{I}_t = \mathcal{I}(\boldsymbol{\mu}_t, \boldsymbol{\theta}_t, \mathbf{g}_t) = |\boldsymbol{\mu}_t \odot \boldsymbol{\theta}_t \odot \mathbf{g}_t|,$$

where $\mathbf{g}_t$ denotes in-batch gradients with respect to the parameters of the model: $\mathbf{g}_t = \partial \mathcal{L}(\boldsymbol{\theta}_t, \mathcal{B}_t)/\partial \boldsymbol{\theta}_t$. We use the absolute value of the elements in the Hadamard product vector as the importance scores. Subsequently, we perform a normalization layer on the element-wise squared importance scores vector ($\mathcal{I}_t^2$) to ensure that the sum of its elements equals one, hence a probability distribution. Finally, leveraging this probability distribution over the models' parameters, we stochastically select a sub-network of a pre-defined size, with elements having higher probabilities being more likely to be chosen in the selection process. The chosen sub-network corresponds uniquely to a binary mask, which is employed in Equation (2).

[4]*Passing values through a connection* denotes the value linked to the neuron from which the connection originates. It is worth noting that there exists an injective function connecting the set of any language model parameters to the set of its corresponding neural network connections.

Algorithm 1 Stochastic Tuning

```
Require: a PLM with parameters $\boldsymbol{\theta}_0 \in \mathbb{R}^n$, a tuning dataset $\mathcal{D}$, number of tuning steps $T$, learning rate $\eta$, and warm-up ratio w
 1: Initialize:
        model ←CREATEMODEL
        SETPARAMETERS(model, $\boldsymbol{\theta}_0$)
        optim ←CREATEOPTIMIZER($\eta$, w)
 2: for $t = 0, \cdots, T-1$ do
 3:     $\mathcal{B}_t \sim \mathcal{D}$                          ▷ sample a mini-batch
 4:     $\boldsymbol{\mu}_t \leftarrow$ model.forward($\mathcal{B}_t$)          ▷ forward pass
 5:     $\boldsymbol{\theta}_t \leftarrow$ model.get_params($t$)
 6:     $\mathbf{g}_t \leftarrow \mathcal{L}(\boldsymbol{\theta}_t, \mathcal{B}_t)$.backward()        ▷ back-propagate
        ----------------------------------------
 7:     $\mathcal{I}_t \leftarrow |\boldsymbol{\mu}_t \odot \boldsymbol{\theta}_t \odot \mathbf{g}_t|$   ▷ compute importance scores
 8:     $M_t \sim$ norm_layer($\mathcal{I}_t$)          ▷ select a sub-network
 9:     $\mathbf{g}_t \leftarrow M_t \odot \mathbf{g}_t$                ▷ mask the gradients
        ----------------------------------------
10:     optim.step($\mathbf{g}_t$)     ▷ update the selected sub-network
11: end for
```

Our approach enhances task-specific fine-tuning by leveraging a stochastic selection process, inspired by principles from evolutionary algorithms. This randomness in forming the binary mask mirrors natural selection, where diverse parameters are stochastically chosen to participate, fostering robustness and preventing overfitting. The computation of importance scores using a multiplicative function provides a refined strategy for selecting sub-networks, simultaneously accounting for parameters with high magnitudes, gradients, and passing values. While prior research has primarily focused on selection based solely on parameter values and gradients (Lee et al., 2020; Zhang et al., 2022), our inclusion of passing values enables a more targeted selection process, optimizing for task-specific performance.

3 Experiments

We assess the performance of Stochastic Tuning on RoBERTa$_{\text{BASE}}$ and RoBERTa$_{\text{LARGE}}$ (Liu et al., 2019).[5] Following previous work (Zhang et al., 2022; Akbar-Tajari et al., 2022; Ben Zaken et al., 2022), we undertake an extensive set of experiments on six datasets from the GLUE benchmark. To provide a thorough analysis, we report the average and standard deviation of the results obtained

[5]We conducted all experiments using two NVIDIA RTX 6000-24G GPUs for approximately 376 hours.

Model	Method	%Updated	CoLA	SST-2	MRPC	STS-B	QNLI	RTE	*Avg.*
RoBERTa$_{\text{BASE}}$	*Full-FT*	100.0%	62.94±0.7	94.06±0.2	92.52±0.4	90.81±0.1	91.76±0.1	77.38±1.5	84.91
	DPS$_{\text{Mix}}$	50-90%	62.45±1.3	94.08±0.2	91.60±0.7	90.83±0.1	91.72±0.1	75.09±1.9	84.30
	Multi-Head	∼ 23%	62.75±1.5	**94.87±0.3**	92.21±0.3	90.92±0.1	92.68±0.1	76.41±0.9	84.97
	BitFit	0.08%	57.92±1.5	92.82±0.1	91.61±0.4	90.20±0.1	87.49±0.2	73.77±3.0	82.30
	Stoc$_{\text{Tuning}}$	0.08%	62.97±0.9	94.70±0.2	91.98±0.6	**91.10±0.3**	91.93±0.2	**78.65±1.6**	85.22
	Top$_{\text{Tuning}}$	0.08%	**63.40±1.4**	93.78±0.1	**92.56±0.1**	91.02±0.2	**92.85±0.1**	78.34±0.9	**85.33**
	Random	0.08%	55.88±0.9	93.80±0.2	91.17±0.6	89.85±0.1	90.67±0.1	69.92±2.0	81.88
	Frozen	0.00%	18.95±2.2	82.94±0.3	83.22±0.1	57.95±0.7	68.34±0.1	58.48±0.5	61.65
RoBERTa$_{\text{LARGE}}$	*Full-FT*	100.0%	63.23±5.8	96.09±0.2	93.50±0.4	92.09±0.2	94.47±0.1	83.35±3.1	87.12
	DPS$_{\text{Mix}}$	50-90%	68.34±1.3	96.12±0.2	92.47±0.8	92.09±0.3	94.53±0.1	84.48±3.1	88.00
	Multi-Head	∼ 28%	67.56±1.4	**96.26±0.1**	93.09±0.7	92.37±0.2	94.69±0.1	83.51±2.2	87.91
	BitFit	0.08%	67.53±0.9	95.23±0.1	91.75±0.5	91.72±0.1	93.10±0.1	80.27±1.9	86.60
	Stoc$_{\text{Tuning}}$	0.08%	**68.45±0.8**	96.25±0.1	93.15±0.8	**92.41±0.3**	94.63±0.1	**84.71±2.1**	**88.27**
	Top$_{\text{Tuning}}$	0.08%	68.34±1.4	95.60±0.3	**93.72±1.0**	92.19±0.1	**94.77±0.1**	83.37±2.4	88.00
	Random	0.08%	64.45±1.4	95.72±0.2	92.30±0.6	91.60±0.1	93.68±0.0	73.43±2.6	85.20
	Frozen	0.00%	26.47±3.6	86.04±0.3	82.86±0.2	68.33±1.1	72.45±0.3	65.44±2.4	66.93

Table 1: The performance comparison of RoBERTa$_{\text{BASE}}$ and RoBERTa$_{\text{LARGE}}$ across six tasks sourced from the GLUE benchmark, utilizing a range of fine-tuning techniques. Evaluation metrics encompass Matthew's correlation for the CoLA task, F1 score for MRPC, Spearman's correlation for STS-B, and Accuracy for the remaining tasks. The table highlights the best and second-best results achieved for each individual task, shedding light on the efficacy of different fine-tuning approaches.

from seven models trained with distinct random seeds. We compare a range of fine-tuning methods for language models, including but not limited to **DPS$_{\text{Mix}}$** (Zhang et al., 2022), which dynamically selects task-relevant sub-networks; **Multi-Head** (Akbar-Tajari et al., 2022), which fine-tunes only attention module parameters; and **BitFit** (Ben Zaken et al., 2022), which updates only the bias terms. Detailed information on datasets and prior methods is provided in Appendices.

3.1 Experimental Setup

We opted for roberta-base and roberta-large, which are readily available in the HuggingFace library for PyTorch (Wolf et al., 2020; Paszke et al., 2019). To handle sequences of varying lengths, we employ a dynamic padding technique to set the maximum input length to 128 and apply a longest-first truncation strategy, facilitating efficient processing of the sequences.

During the fine-tuning process, we employ a batch size of 16 and utilize the AdamW (Loshchilov and Hutter, 2019) optimizer with an epsilon value set to 1e-6, incorporating a linear increase in the learning rate over the initial 10% of steps followed by a linear decay to zero. Our hyperparameter tuning is solely focused on selecting the learning rate from {1e-5, 3e-5, 1e-4, 3e-4, 1e-3, 3e-3}, ensuring a fair comparison with previous work.

3.2 Results

Table 1 reports the results for RoBERTa$_{\text{BASE}}$ and RoBERTa$_{\text{LARGE}}$, where *Full-FT* represents the full fine-tuning scenario, and *Frozen* indicates the method that freezes all the model parameters except for the classification head. To ensure a fair comparison, the size of the selected sub-networks in Stochastic Tuning ($\text{Stoc}_{\text{Tuning}}$) was matched to the number of tunable parameters in BitFit. To check the regularization capabilities of our method, we also report results for Top Tuning ($\text{Top}_{\text{Tuning}}$) which forms sub-networks based on parameters with the highest importance scores (instead of selecting them stochastically as in our method). As a baseline, we also report results for Random Tuning (Random) which forms sub-networks by selecting parameters uniformly at random.

Overall, both $\text{Top}_{\text{Tuning}}$ and $\text{Stoc}_{\text{Tuning}}$ outperform the *Full-FT* baseline by using a very small fraction of the parameters. There is also a consistent improvement over DPS_{Mix} and Multi-Head despite using significantly fewer parameters in the optimization process. The improvement comes from the stable performance of these techniques across different datasets ($\text{Stoc}_{\text{Tuning}}$ is consistently among the top-2 on all the datasets). Compared to BitFit, our approach provides an improvement of over two points on average. Among the two techniques, the $\text{Stoc}_{\text{Tuning}}$ shows to be more effective in most tasks. This proves our assumption that the stochastic selection of parameters can bring about regularization effect. Conclusions are consistent across both $\text{RoBERTa}_{\text{BASE}}$ and $\text{RoBERTa}_{\text{LARGE}}$, with results being generally better for the latter model.

4 Conclusions

We introduce *Stochastic Tuning*, a novel approach for highly efficient fine-tuning of pre-trained language models (PLMs). By utilizing masked gradients to update stochastic sub-networks, our method outperforms previous state-of-the-art fine-tuning techniques in terms of overall performance across multiple GLUE tasks. Notably, we achieve this improvement by updating parameters solely within a small sub-network during each iteration, resulting in substantial computational cost reduction. The inherent randomness in our method acts as a form of regularization, effectively mitigating overfitting and consistently promoting better generalization. Our results demonstrate the significant performance gains achieved with Stochastic Tuning on two distinct PLMs. We anticipate that Stochastic Tuning of PLMs holds promise for a wider range of application scenarios like multimodal model fine-tuning (Liu et al., 2023), leaving room for further exploration for future research endeavors. In line with the methodological choices of previous work, our research did not include any generative models. However, it is worth noting that future studies could broaden the scope of comparative analysis by incorporating fine-tuning approaches using generative models such as GPT (Radford and Narasimhan, 2018), BART (Lewis et al., 2019), and T5 (Raffel et al., 2020). Such an expansion would offer a more comprehensive understanding of the capabilities and trade-offs of different model architectures in natural language processing tasks.

Limitations

Considering the constraints of computational resources, our study focused on RoBERTa models and limited the analysis to the six smallest tasks from the GLUE benchmark. We encountered an additional limitation in PyTorch's lack of native support for random choice from large sets, which necessitated employing NumPy (Harris et al., 2020). This workaround resulted in performance degradation and compromised the overall efficiency of GPU utilization.

Acknowledgment

We would like to thank the anonymous reviewers for their valuable feedback during the review process. Their constructive comments have greatly helped in refining and improving the quality of this work.

References

Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. 2021. Better fine-tuning by reducing representational collapse. In *International Conference on Learning Representations*.

Mohammad Akbar-Tajari, Sara Rajaee, and Mohammad Taher Pilehvar. 2022. An empirical study on the transferability of transformer modules in parameter-efficient fine-tuning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10617–10625, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pages 177–190. Springer.

Dorottya Demszky, Kelvin Guu, and Percy Liang. 2018. Transforming question answering datasets into natural language inference datasets. *CoRR*, abs/1809.02922.

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

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

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. 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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(11):12799–12807.

Angeliki Giannou, Shashank Rajput, and Dimitris Papailiopoulos. 2023. The expressive power of tuning only the norm layers.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature*, 585(7825):357–362.

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

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

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. 2020. Mixout: Effective regularization to finetune large-scale pretrained language models. In *International Conference on Learning Representations*.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. 2019. SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY. In *International Conference on Learning Representations*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

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

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pretraining.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Ming Tu, Visar Berisha, Martin Woolf, Jae-sun Seo, and Yu Cao. 2016. Ranking the parameters of deep neural networks using the fisher information. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2647–2651.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. 2021. Raise a child in large language model: Towards effective and generalizable fine-tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9514–9528, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Haojie Zhang, Ge Li, Jia Li, Zhongjin Zhang, YUQI ZHU, and Zhi Jin. 2022. Fine-tuning pre-trained language models effectively by optimizing subnetworks adaptively. In *Advances in Neural Information Processing Systems*, volume 35, pages 21442–21454. Curran Associates, Inc.

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

A The General Language Understanding Evaluation (GLUE) benchmark

Our experiments encompasses a range of tasks: linguistic acceptability (Warstadt et al., 2019, CoLA), sentiment prediction (Socher et al., 2013, SST-2), paraphrase detection (Dolan and Brockett, 2005, MRPC), sentence similarity (Cer et al., 2017, STS-B), and natural language inference (Wang et al., 2019; Demszky et al., 2018; Dagan et al., 2005, QNLI, RTE). Given the restricted online submission quota for the test set, we adhere to the approach followed by several previous studies: fine-tuning on the training data and reporting results based on the development sets (Zhang et al., 2022; Dodge et al., 2020; Aghajanyan et al., 2021; Zhang et al., 2021; Fu et al., 2023).

CoLA. (Warstadt et al., 2019) The **C**orpus **o**f **L**inguistic **A**cceptability dataset comprises English sentences annotated for grammatical acceptability, using the Matthews correlation coefficient as the evaluation metric on an unbalanced binary classification task. The performance is reported on the combined in-domain and out-of-domain sections of the standard dev set.

SST-2. (Socher et al., 2013) The **S**tanford **S**entiment **T**reebank comprises movie review sentences with human-annotated sentiments. The objective is to predict sentence-level sentiment using a binary (positive/negative) classification.

MRPC. (Dolan and Brockett, 2005) The **M**icrosoft **R**esearch **P**araphrase **C**orpus dataset consists of sentence pairs extracted from online news sources, annotated for semantic equivalence. Due to class imbalance, F1 score is reported for evaluation.

STS-B. (Cer et al., 2017) **S**emantic **T**extual **S**imilarity **B**enchmark is a dataset consisting of sentence pairs from various sources, annotated with similarity scores. Evaluation is performed using Pearson and Spearman correlation coefficients.

QNLI. (Wang et al., 2019; Demszky et al., 2018) **Q**uestion-Answering **NLI** transforms the Stanford Question Answering Dataset into a sentence pair classification task, involving question-paragraph pairs. The goal is to determine whether a given context sentence contains the answer to the corresponding question, removing the requirement for the model to select the exact answer while challenging the assumption that the answer is always present and that lexical overlap reliably indicates the answer.

RTE. (Dagan et al., 2005) **R**ecognizing **T**extual **E**ntailment datasets are a compilation of examples from annual challenges, combining data from multiple sources. The datasets are based on news and Wikipedia text, and are converted into a two-class split for consistency.

B Comparison Methods

Here, we provide a brief overview of the baselines to which we have compared our proposed method, highlighting key approaches in efficient fine-tuning techniques.

$\mathbf{DPS_{Mix}}$. Proposed by Zhang et al. (2022), it is an enhanced variant of Child-Tuning (Xu et al., 2021), a dynamic sub-network optimization algorithm aimed at efficient fine-tuning of LLMs. By estimating the importance of parameters using empirical Fisher Information (Tu et al., 2016) from multiple mini-batches of downstream task data, DPS dynamically selects the most task-related subnetwork for updating during fine-tuning, effectively addressing the issue of overfitting.

Multi-Head. Akbar-Tajari et al. (2022) freeze all the modules' parameters to their pre-trained value, except for those in the attention modules of the transformer blocks during the fine-tuning process. Their goal is to illustrate that each module functions as a winning ticket, achieving performance comparable to that of the full fine-tuning scenario.

BitFit. Proposed by Ben Zaken et al. (2022), BitFit is a baseline approach that focuses on updating only the bias components in PLMs while keeping the remaining parameters frozen. By isolating the bias components for updating, BitFit aims to explore the impact of bias adjustments on the overall performance of PLMs.