Systematic Analysis for Pretrained Language Model Priming for Parameter-Efficient Fine-tuning

Shih-Cheng Huang[1*], **Shih-Heng Wang**[1*], **Min-Han Shih**[2*], **Saurav Sahay**[2+], **and Hung-yi Lee**[1*]
[*]National Taiwan University, Taipei, Taiwan
[+]Intel Labs, Santa Clara, CA, USA
{r09942093,r11942079,b08502141,hungyilee}@ntu.edu.tw[*]
saurav.sahay@intel.com[+]

Abstract

Parameter-efficient (PE) methods (like Prompts or Adapters) for adapting pre-trained language models (PLM) to downstream tasks have been popular recently. However, hindrances still prevent these methods from reaching their full potential. For example, two significant challenges are few-shot adaptation and cross-task generalization. To tackle these issues, we propose a general PE **priming** framework to enhance and explore the few-shot adaptation and generalization ability of PE methods. In this framework, PLMs are primed with PE methods for rapidly adapting to various target tasks. To evaluate the generalization ability of these PE methods, we conduct experiments on a few-shot cross-domain benchmark containing 160 diverse NLP tasks. Our experiment not only reveals the best priming strategy but also verifies that priming facilitates the adaptation to target tasks.

1 Introduction

In recent years, pre-trained language models (PLMs) in natural language processing (NLP) are blooming everywhere (Devlin et al., 2018; Lewis et al., 2019; Liu et al., 2019; Joshi et al., 2020; Raffel et al., 2019; Radford et al., 2019; Brown et al., 2020). However, not only the number of PLMs but also their size is rapidly growing, making it harder to perform full fine-tuning. To address the issue, tons of parameter-efficient fine-tuning (PEFT) methods have bubbled up, such as adapters (Houlsby et al., 2019; Pfeiffer et al., 2020; Zaken et al., 2021; Fu et al., 2022), or prompts (Lester et al., 2021; Li and Liang, 2021). These methods have made it equitable for researchers with insufficient computational resources. However, there is still a long way to go for these PE methods to reach their full potential. Because the pre-training objectives are not directly related to PE, it is foreseeable that there is a mismatch between the PLM and PE methods, which may prevent PE methods from unleashing their full power. To address this problem, we introduce an additional "priming" stage between pre-training and downstream fine-tuning. As shown in Fig.1, we prime the PLM on extra few-shot source tasks with multitask learning (MTL) or meta-learning, and then fine-tune PE elements to target tasks. Compared with traditional PEFT methods, the PLM and PE elements can fit each other better after priming. In other words, instead of the conventional PEFT paradigm (pre-train → PEFT), we adopt the "pre-train → priming → PEFT" pipeline.

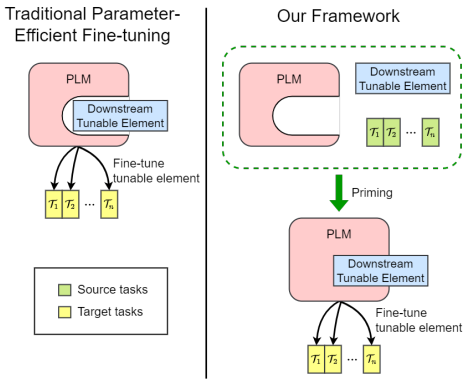

Figure 1: We propose a general framework to improve the performance of parameter-efficient fine-tuning. We prime the PLM with source tasks for parameter-efficient methods.

Some recent studies explore how to bridge the gap between pre-training tasks and target tasks. Gu et al. (2021) pre-trains the soft prompt tokens with self-supervised tasks to give a better initialization. Huang et al. (2022); Hou et al. (2022) exploits optimization-based meta-learning to find an initialization for soft prompts to facilitate faster adaptation to new tasks. Gheini et al. (2022) tweaks the meta-learning algorithm MAML (Finn et al., 2017) for priming and simulates the PEFT procedure in the inner loop. However, they only focus on priming for a single PE method with a specific algorithm. In contrast, our work views priming as a general method to boost PEFT from a higher per-

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 4: Student Research Workshop), pages 1–7
June 18, 2024

spective. Moreover, previous works explore only single-domain tasks, which lack the exploration of generalization ability. On the contrary, our work evaluate PE methods with diverse NLP tasks in various domains.

On top of that, we conduct comprehensive experiments over well-known PE methods like adapters and prompt tuning under different settings. The experiment result reveals that priming by tuning only PLM leads to the best adaptation result on target tasks. In addition, we shows priming does help the whole model to converge more easily, which validates the necessity of priming.

2 Related Work

2.1 Adapter

Adapters (Houlsby et al., 2019; Stickland and Murray, 2019; Wang et al., 2020; Bansal et al., 2022; Fu et al., 2022; Pfeiffer et al., 2020; Karimi Mahabadi et al., 2021; Hu et al., 2021; Zaken et al., 2021; He et al., 2021) are lightweight modules introduced for the transformer architecture. Adapters add extra trainable parameters and freeze the original PLM parameters during fine-tuning.

2.2 Prompt

Prompt-based tuning (Li and Liang, 2021; Lester et al., 2021; Liu et al., 2021b; Huang et al., 2022; Gu et al., 2021; Liu et al., 2021a; Han et al., 2021; Hou et al., 2022; Vu et al., 2021; Liu et al., 2022) is an innovative method to use the power of PLMs efficiently. Gu et al. (2021) proposed the concept of prompt initialization pre-training. Huang et al. (2022) proposed Meta-learned Prompt Tuning (MetaPT) to further improve prompt initialization.

3 Methodology

3.1 Framework

Our work aims to comprehensively analyze the priming strategies by comparing the performance of PE methods under few-shot scenarios. We introduce a general framework to prime the whole model (may include PLMs) to better adapt to downstream tasks. Our training approach consists of two distinct stages: the **upstream priming stage** and the **downstream fine-tuning stage**. Initially, the model acquires knowledge from source tasks during the upstream priming stage, followed by few-shot fine-tuning on target tasks in the downstream stage.

Specifically, we name parameters fine-tuned in the upstream stage as **upstream tunable elements**, while those in the downstream stage as **downstream tunable elements**. **Upstream tunable elements** and **downstream tunable elements** may be fully-overlapping, partially-overlapping or non-overlapping.

3.2 Upstream Priming Stage

The upstream priming stage is designed to prime the model's upstream tunable elements, enabling it to quickly adapt to a range of downstream few-shot tasks. We employ a **priming algorithm**, predominantly Meta Learning or Multitask Learning (MTL), to update the upstream tunable elements on source tasks. Upstream tunable elements comprise **Pre-trained Language Models (PLM)**, **adapters**, and **prompts**. In other words, the combination of upstream tu

3.2.1 Multi-task Learning

In Multi-task Learning (MTL), multiple tasks are learned concurrently by minimizing their combined loss. This method enhances the model's capability to learn cross-task features and accelerates adaptation. Our implementation of MTL focuses on training the upstream tunable elements during the upstream priming stage. We define ψ as the whole model's parameters, with subsets ψ_u for upstream and ψ_d for downstream tunable elements. The objective is to minimize loss across training tasks while adjusting only ψ_u:

$$\psi_u' = \arg\min_{\psi_u} \sum_{\mathcal{T}_i \in \mathcal{T}} \mathcal{L}\left(\psi, \mathcal{T}_i\right) \tag{1}$$

Here, $\mathcal{L}$ is the loss function, and $\mathcal{T}_i$ represents the i^{th} task from the set of source tasks $\mathcal{T}$.

It is important to note that if ψ_d is not included in ψ_u, it is initialized but remains unchanged during the upstream priming stage. For example, if PLM is selected as the upstream tunable element and adapters as the downstream element, the adapters are initialized in the upstream stage but only tuned during the downstream fine-tuning phase.

3.2.2 Meta Learning

In our study, we utilize the MAML(Finn et al., 2017) algorithm, for our priming process. MAML is distinctive in its dual-phase training approach: the inner loop and the outer loop. The inner loop is designed for task-specific adaptation, while the outer loop focuses on finding an optimal initialization for quick adaptation in the inner loop. We

modify some parts of MAML algorithm, which are outlined in Alg.1 and illustrates in Fig. 2. It starts by copying the current model parameters ψ as the initial state for the inner loop. In this loop, we specifically tune the downstream tunable element ψ_d. The tuned parameters for the i^{th} task in the inner loop are represented as ψ_i'. The final step involves calculating the loss from the adapted model ψ_i' and the source tasks $\mathcal{T}_i$, which is then used to update ψ_u. The updated ψ_u are initialized for the subsequent inner loop.

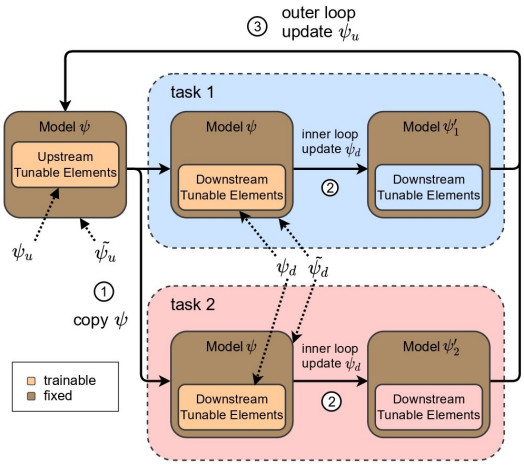

Figure 2: Illustration of Alg. 1

Algorithm 1 Parameter-Efficient MAML

$\mathcal{T} = \{T_1, T_2, ...\}$: A set of source tasks
α, β: Outer lr, Inner lr
θ: PLM parameters
$\{\phi_1, \phi_2, ...\}$: Tunable elements
$\psi = [\theta; \phi_1; \phi_2; ...]$: All parameters of the model

Randomly initialize $\{\phi_1, \phi_2, ...\}$
while not done **do**
  for $T_i \in \mathcal{T}$ **do**
    Split ψ into two parts, ψ_d and $\tilde{\psi}_d$
    Evaluate $\nabla_{\boldsymbol{\psi_d}} \mathcal{L}_{T_i}(f_\psi)$ with respect to K samples
    Compute adapted parameters with gradient descent: $\psi'_{d,i} = \boldsymbol{\psi_d} - \beta \nabla_{\boldsymbol{\psi_d}} \mathcal{L}_{T_i}(f_\psi)$
    $\psi_i' = [\psi'_{d,i}; \tilde{\psi}_d]$
  end for
  Split ψ into two parts, ψ_u and $\tilde{\psi}_u$
  $\psi'_u = \psi_u - \alpha \nabla_{\psi_u} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{T_i}(f_{\psi_i'})$
  $\psi \leftarrow [\psi'_u; \tilde{\psi}_u]$
end while
return ψ

3.3 Downstream Fine-Tuning Stage

In downstream fine-tuning stage, with the primed initialization obtained from the upstream priming stage, we directly fine-tune the downstream tunable elements on the target tasks. Since the backbone of our work is to explore the cross-domain few-shot ability of PEFT methods w & w/o priming, **only prompt or adapter are tunable in downstream stage**.

4 Experiment

4.1 Dataset

We choose **CrossFit Challenge** (Ye et al., 2021) as our benchmark, which provides 160 different NLP few-shot tasks with a unified text-to-text format gathered from existing open-access datasets.

In CrossFit Challenge, they divide all tasks into non-overlapping Train, Dev, and Test tasks. We select **random split** in Ye et al. (2021) to be the task split setting in our work. We select the Train tasks as the source tasks for upstream priming and the Test tasks as the target tasks for downstream fine-tuning. More explicit explanations of tasks can be found in Ye et al. (2021). Briefly speaking, CrossFit Challenge is able to evaluate the authentic few-shot generalization ability of models.

4.2 Setup

4.2.1 Tunable Elements

Our experiment setup mainly follows Ye et al. (2021). In the upstream priming stage, we can tune prompt, adapter and PLM, but we only tune prompt or adapters during the downstream fine-tuning stage to accord with the spirit of PE methods. It's crucial to emphasize that the initialized parameters obtained from the upstream priming stage are carried forward to the downstream fine-tuning stage.

Adapter

In this work, we mainly adopt AdapterBias (Fu et al., 2022) as our adapter module. AdapterBias adds a token-dependent shift to the hidden output of transformer layers, parameterized by only a vector and a linear layer. Compared with the original adapter design (Houlsby et al., 2019), the trainable parameters are further reduced while obtaining comparable performance.

Prompt

Prompt is one of our tunable elements. In our settings, we applied prompt tuning proposed by Lester et al. (2021), which concatenates tunable tokens before the input sentence and ask the PLM to generate corresponding output text. Following the settings in Lester et al. (2021), we set the prompt length to 100 tokens.

4.2.2 Hyperparameters

In our research, we use the BART-base model (Wolf et al., 2019) as our primary language model. For both the MTL and the outer loop of meta-learning, we employ the AdamW (Loshchilov and Hutter, 2017) with a weight decay of 0.01. Specifically in meta-learning, we set different outer loop learning rates for various elements: 8×10^{-5} for PLMs, 8×10^{-3} for prompts, and 1×10^{-5} for adapters. The inner loop has its learning rates set at 0.025 for prompts and 0.001 for adapters. The training is conducted over 80 epochs, with a batch size of 1 for training and an inner batch size of either 4 or 8, contingent on GPU memory limits. For MTL, we maintain a consistent learning rate of 3×10^{-5} for PLMs, prompts, and adapters. The MTL training spans 10 epochs with a train batch size of 32.

4.3 Metrics

For the evaluation metric, we also follow Ye et al. (2021), adopting *Average Relative Gain* (**ARG**) as one of the performance indexes, and the definition for ARG is:

$$\text{ARG} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{P^i - P_0^i}{P_0^i}\right) \quad (2)$$

θ : Adapter and downstream model P_0^i represents the performance of directly fine-tuning PLMs on i^{th} target tasks, and P_i is that of our experiment combination. Since the comparing target is directly fine-tuning PLMs, baselines with ARG greater than 0 are those that surpass fine-tuning PLMs.

4.4 Annotation

We use abbreviations to make the result more concise, including **M** for PLM, **P** for prompts, and **A** for adapters. Additionally, characters before the underline represent the upstream tunable elements, while those after the underline represent the downstream tunable elements of the baseline. For example, P_P represents upstream and downstream tunable elements are both prompts; M+A_A represents upstream tunable elements are PLM and adapters, and downstream tunable element is adapters. Lastly, we use FT_M, FT_A, and FT_P to represent directly fine-tuning the PLM, adapter, and prompt, respectively.

4.5 Main Result

The first block in Table 1 showcases baselines without priming, while subsequent blocks feature combinations of different priming strategies categorized by the priming algorithm. Among the direct fine-tuning results, fine-tuning PLM serves as a competitive baseline with high training costs, while direct fine-tuning prompts/adapters are considered as primary baselines against each priming prompts/adapters baseline.

Table 1 underscores the effectiveness of priming, with most baselines showing noticeable improvements (indicated by asterisks*) across various priming algorithms. Notably, some combinations outperform direct fine-tuning of prompts, and a few even surpass fine-tuning PLM, such as M_P and M+P_P in multi-task learning (ARGs greater than 0). For adapters, certain combinations demonstrate remarkable progress, like M_A in both meta-learning and multi-task learning, while others experience slight drops in performance, such as A_A in both meta-learning and multi-task learning.

4.6 Parameter Efficiency

Fig 3 visually depicts the relationship between the performance of each method and its tuned parameter scale. The best-performing combinations from Table 1 represent priming prompts/adapters (green ones). Other baselines include fine-tuning prompts/adapters without priming, fine-tuning PLM (BART-base), and existing works like LoRA(Hu et al., 2021) and BitFit(Zaken et al., 2021). Fig 3 demonstrates that priming prompts/adapters baselines locate at the upper-left region, indicating superior results and higher parameter efficiency brought by priming.

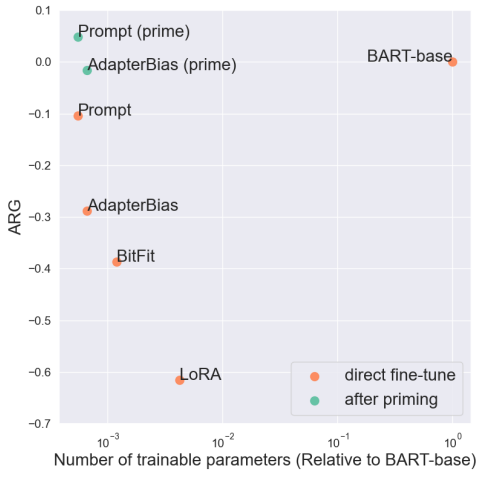

Figure 3: This figure shows the parameter efficiency of different baselines.

			Clf-F1	ACC	EM	Matthew Cor.	QA-F1	Rouge-L	ARG
Without Priming	Direct Fine-tuning	PLM (M) (100%)	**0.6474**	**0.5839**	0.3304	**0.0896**	**0.2919**	0.8033	**0.0000**
		Prompt (P) (0.06%)	0.5570	0.5115	**0.4147**	0.0495	0.2765	0.8030	-0.1040
		Adapter (A) (0.07%)	0.4503	0.4784	0.2824	-0.0276	0.2863	**0.8081**	-0.2886
With Priming	Meta Learning	P_P	**0.6283**	0.5321	**0.4291**	0.0436	0.3261	**0.8010**	-0.0331*
		M_P	0.6267	**0.6441**	0.1961	0.0505	0.3064	0.7962	**-0.0143***
		M+P_P	0.6173	0.5783	0.1777	**0.0626**	0.2509	0.7690	-0.0477*
		A_A	0.4209	0.4257	0.2326	-0.0556	0.2899	0.7958	-0.3534
		M_A	0.5546	0.6457	0.2253	0.0067	0.2656	0.7284	-0.1045*
		M+A_A	0.3528	0.4614	0.2580	-0.0050	**0.3868**	0.7479	-0.3025
	Multi-Task Learning	P_P	0.5524	0.5088	**0.4055**	**0.0765**	0.3390	0.7993	-0.0863*
		M_P	**0.6646**	0.6491	0.2509	0.0612	0.3841	**0.8086**	0.0488*
		M+P_P	0.6610	**0.6519**	0.2622	0.0716	0.3943	0.8032	**0.0571***
		A_A	0.3358	0.4274	0.2743	-0.0469	0.3007	0.7405	-0.3808
		M_A	0.6122	0.6496	0.2821	-0.0128	**0.4432**	0.7383	-0.0158*
		M+A_A	0.3815	0.5709	0.2048	-0.0483	0.4081	0.6713	-0.2531*

Table 1: This table shows the detailed performance of different baselines. We divide the methods by whether it is primed and its priming algorithm. The percentages beside each setting represent the proportion of parameters trained in the downstream fine-tuning stage.

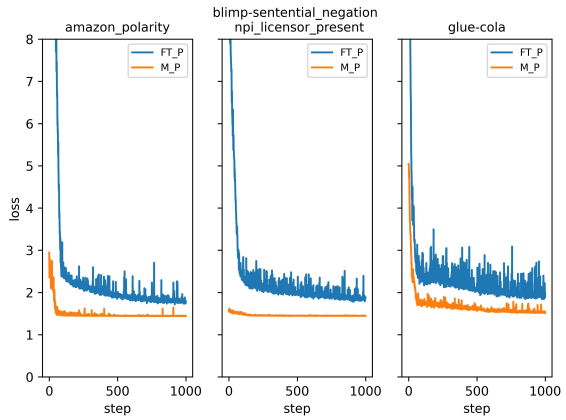

Figure 4: The loss curves of w & w/o priming prompt in the fine-tuning stage.

5 Analysis

In this section, we emphasize the advantages of priming by examining training loss during downstream fine-tuning. We compare loss curves between models w/ or w/o priming (FT_P and MTL M_P, respectively) across three diverse tasks from the target task training set. These tasks, unseen by prompts/adapters, feature distinct evaluation metrics. Figure 4 presents the results, where the blue curves represent FT_P (w/o priming) and the orange curves represent MTL M_P (w/ priming). The primed model not only converges faster but also achieves a superior final level of loss. Furthermore, the orange curves exhibit steadier convergence compared to the fluctuating and glitch-prone blue curves. These findings underscore the benefits of priming for prompts/adapters, facilitating rapid adaptation to various target tasks.

6 Conclusion

In this paper, we systematically analyze priming PEFT within a comprehensive framework. Our framework not only incorporates existing priming approaches but also explores previously uncharted strategies. Our experimental results demonstrate that the majority of priming strategies enhance the performance of PE methods. Notably, "Priming PLM only" emerges as the top-performing strategy when used in conjunction with multi-task learning. Crucially, our study provides concrete evidence that priming significantly facilitates the convergence of fine-tuning prompts/adapters on unseen tasks, underscoring the efficacy of priming.

7 Limitation

We provide a systematic analysis of different priming strategies on PE methods and successfully improve the few-shot performance on diverse downstream tasks. However, there are some limitations to our work. Though we empirically show that MTL outperforms meta-learning, there are no further explanations for it. Besides, a small proportion of the priming strategies lead to a performance drop, but the actual reason remains unexplained. In addition, all the experiments are conducted on the pre-trained BART-base model. Extra experiments on other large language models may strengthen the results.

References

Trapit Bansal, Salaheddin Alzubi, Tong Wang, Jay-Yoon Lee, and Andrew McCallum. 2022. Meta-adapters: Parameter efficient few-shot fine-tuning through meta-learning. In *First Conference on Automated Machine Learning (Main Track)*.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks.

Chin-Lun Fu, Zih-Ching Chen, Yun-Ru Lee, and Hung-yi Lee. 2022. Adapterbias: Parameter-efficient token-dependent representation shift for adapters in nlp tasks. *arXiv preprint arXiv:2205.00305*.

Mozhdeh Gheini, Xuezhe Ma, and Jonathan May. 2022. Know where you're going: Meta-learning for parameter-efficient fine-tuning. *arXiv preprint arXiv:2205.12453*.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2021. Ppt: Pre-trained prompt tuning for few-shot learning.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2021. Ptr: Prompt tuning with rules for text classification.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*.

Yutai Hou, Hongyuan Dong, Xinghao Wang, Bohan Li, and Wanxiang Che. 2022. MetaPrompting: Learning to learn better prompts. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3251–3262, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *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. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Yukun Huang, Kun Qian, and Zhou Yu. 2022. Learning a better initialization for soft prompts via meta-learning.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems*, 34:1022–1035.

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

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461.

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

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

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Annual Meeting of the Association for Computational Linguistics*.

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

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. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2020. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*.

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.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer.

Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Matthew Cer. 2021. Spot: Better frozen model adaptation through soft prompt transfer. In *Annual Meeting of the Association for Computational Linguistics*.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2020. K-adapter: Infusing knowledge into pre-trained models with adapters.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021. CrossFit: A few-shot learning challenge for cross-task generalization in NLP. pages 7163–7189.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.