Robust Transfer Learning with Pretrained Language Models through Adapters

Wenjuan Han[1*†] **Bo Pang**[2*] **Yingnian Wu**[2]
[1] Beijing Institute for General Artificial Intelligence, Beijing, China
[2] Department of Statistics, University of California, Los Angeles
hanwenjuan@bigai.ai
{bopang, ywu}@ucla.edu

Abstract

Transfer learning with large pretrained transformer-based language models like BERT has become a dominating approach for most NLP tasks. Simply fine-tuning those large language models on downstream tasks or combining it with task-specific pretraining is often not robust. In particular, the performance considerably varies as the random seed changes or the number of pretraining and/or fine-tuning iterations varies, and the fine-tuned model is vulnerable to adversarial attack. We propose a simple yet effective adapter-based approach to mitigate these issues. Specifically, we insert small bottleneck layers (i.e., adapter) within each layer of a pretrained model, then fix the pretrained layers and train the adapter layers on the downstream task data, with (1) task-specific unsupervised pretraining and then (2) task-specific supervised training (e.g., classification, sequence labeling). Our experiments demonstrate that such a training scheme leads to improved stability and adversarial robustness in transfer learning to various downstream tasks. [1]

1 Introduction

Pretrained transformer-based language models like BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) have demonstrated impressive performance on various NLP tasks such as sentiment analysis, question answering, text generation, just to name a few. Their successes are achieved through sequential transfer learning (Ruder, 2019): pretrain a language model on large-scale unlabeled data and then fine-tune it on downstream tasks with labeled data. The most commonly used fine-tuning approach is to optimize all parameters of the pretrained model with regard to the downstream-task-specific loss. This training scheme is widely adopted due to its simplicity and flexibility (Phang et al., 2018; Peters et al., 2019; Lan et al., 2019; Raffel et al., 2020; Clark et al., 2020; Nijkamp et al., 2021; Lewis et al., 2020).

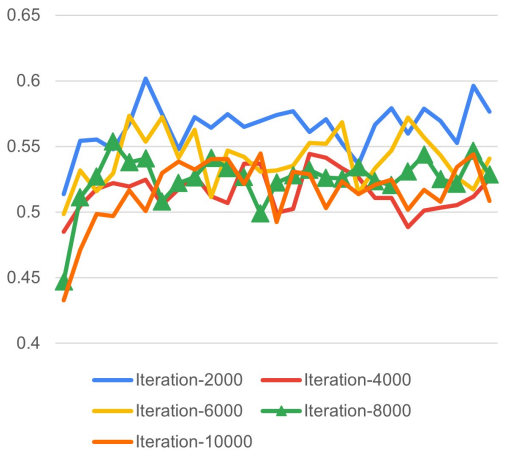


Figure 1: Learning curves of fine-tuning with the task-specific pretraining iterations varied. The curve with triangles represents the model that has converged in the 8000-th pretraining iteration.

Despite the success of the standard sequential transfer learning approach, recent works (Gururangan et al., 2020; Lee et al., 2020; Nguyen et al., 2020) have explored domain-specific or task-specific unsupervised pretraining, that is, masked language model training on the downstream task data before the final supervised fine-tuning on it. And they demonstrated benefits of task-specific pretraining on transfer learning performance. However, both standard sequential transfer learning and that with task-specific pretraining are unstable in the sense that downstream task performance is subject to considerable fluctuation while the random seed is changed or the number of pretraining and/or fine-tuning iterations is varied even after the training has converged (see Section 2 and Section 3

[*] Equal contributions.
[†] Corresponding author.
[1] https://github.com/WinnieHAN/Adapter-Robustness.git

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Short Papers), pages 854–861
August 1–6, 2021.

		WNLI	RTE	MRPC	STS-B	CoLA	SST-2	QNLI	QQP	MNLI
Metrics		*Acc.*	*Acc.*	*F1/Acc.*	*P/S corr.*	*M corr.*	*Acc.*	*Acc.*	*Acc./F1*	*M acc.*
WO.		56.34	65.7	88.85/84.07	88.64/88.48	56.53	92.32	90.66	90.71/87.49	84.10
W.	*F.*	45.07	61.73	89.47/85.29	83.95/83.70	49.23	91.97	87.46	88.40/84.31	81.08
	TSP.+F.	56.34	68.59	89.76/86.37	89.24/88.87	64.87	92.78	91.12	90.92/87.88	84.14

Table 1: Performance on the development dataset of GLUE. Results of W.(F.) are reported in Adapter-Hub. We report results of WO. using the implementation from Wolf et al. (2020). Acc.: Accuracy. M acc.: Mismatched Acc. P/S acc.: Person/Spearman corr. M corr.: Matthew's corr. TSP.: Task-Specific Pretrain. F.: Finetune. WO.: Without adapter. W.: With adapter.

for details). For instance, as observed in Fig. 1, as the number of task-specific pretraining iteration varies, CoLA's performance is severely unstable in fine-tuning. Besides instability, we also observe that task-specific pretraining is vulnerable to adversarial attack. Last but not least, task-specific pretraining and/or fine-tuning on the entire model is highly parameter-inefficient given the large size of these models (e.g., the smallest BERT has 110 million parameters).

In this work, we propose a simple yet effective adapter-based approach to mitigate these issues. Adapters are some small bottleneck layers inserted within each layer of a pretrained model (Houlsby et al., 2019; Pfeiffer et al., 2020a,b). The adapter layers are much smaller than the pretrained model in terms of the number of parameters. For instance, the adapter used in (Houlsby et al., 2019) only adds 3.6% parameters per task. In our approach, we adapt the pretrained model to a downstream task through 1) task-specific pretraining and 2) task-specific supervised training (namely, fine-tuning) on the downstream task (e.g., classification, sequence labeling) by only optimizing the adapters and keeping all other layers fixed. Our approach is parameter-efficient given that only a small number of parameters are learned in the adaptation.

The adapted model learned through our approach can be viewed as a residual form of the original pretrained model. Suppose x is an input sequence and h_{original} is the features of x computed by the original model. Then the feature computed by the adapted model is,

$$h_{\text{adapted}} = h_{\text{original}} + f_{\text{adapter}}(x), \qquad (1)$$

where $f_{\text{adapter}}(x)$ is the residual feature in addition to h_{original} and f_{adapter} is the adapter learned in the adaptation process. h_{original} extracts general features that are shared across tasks, while f_{adapter} is learned to extract task-specific features. In prior work (Houlsby et al., 2019; Pfeiffer et al., 2020b), f_{adapter} is learned with task-specific supervised learning objective, distinctive from the unsupervised pretraining objective, and might not be compatible with h_{original}, as evidenced in our experiments. In our approach, f_{adapter} is first trained with the same pretraining objective[2] on the task-specific data before being adapted with the supervised training objective, encouraging the compatibility between h_{original} and f_{adapter}, which is shown to improve the downstream task performance in our experiments (see Table 1).

Some prior works have examined the potential causes of the instability of pretrained language models in transfer learning. Lee et al. (2019) proposed that catastrophic forgetting in sequential transfer learning underlined the instability, while Mosbach et al. (2020) proposed that gradient vanishing in fine-tuning caused it. Pinpointing the cause of transfer learning instability is not the focus of the current work, but our proposed method seems to able to enhance transfer learning on both aspects.

The standard sequential transfer learning or that with task-specific pretraining updates all model parameters in fine-tuning. In contrast, our approach keeps the pretrained parameters unchanged and only updates the parameters in the adapter layers, which are a small amount compared to the pretrained parameters. Therefore, our approach naturally alleviates catastrophic forgetting considering the close distance between the original pretrained model and the adapted model. On the other hand, we do not observe gradient vanishing with our transfer learning scheme (see Section 2 for more details). This might be because optimizing over a much smaller parameter space in our approach, compared to the standard sequential transfer learning scheme where all parameters are trained, renders the op-

[2]In this work, we conduct experiments with the most widely used pretraining objective, masked language modeling. The same training scheme can be extended to other pretraining objectives.

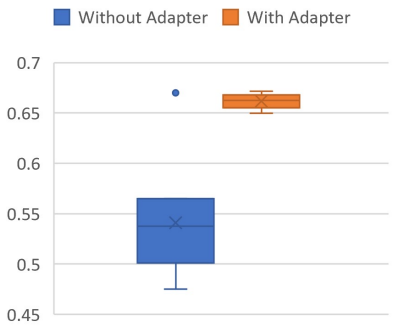


Figure 2: Distribution of dev scores on RTE from 10 random seed restarts when finetuning (1) BERT (Devlin et al., 2019) and (2) BERT with the adapter architecture.

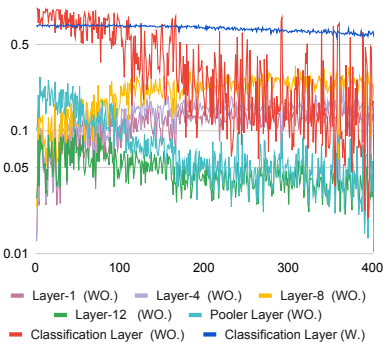


Figure 3: Gradient norms (on log scale) of intermediate layer and classification layer on RTE for with/without-adapter fine-tuning run. WO.: Without adapter. W.: With adapter.

timization easier. We leave it to future work for further theoretical analysis.

In addition to its improved stability, the proposed transfer learning scheme is also likely to be more robust to adversarial attack. Given that it updates the entire model, the standard transfer learning approach might suffer from overfitting to the downstream task, and thus a small perturbation in the input might result in consequential change in the model prediction. In turn, it might be susceptible to adversarial attack. Our approach only updates a much smaller portion of parameters, and hence might be more robust to these attacks, which is confirmed in our empirical analysis (see Section 4).

Contributions. In summary our work has the following contributions. (1) We propose a simple and parameter-efficient approach for transfer learning. (2) We demonstrate that our approach improves the stability of the adaptation training and adversarial robustness in downstream tasks. (3) We show the improved performance of our approach over strong baselines. Our source code is publicly available at https://github.com/WinnieHAN/Adapter-Robustness.git.

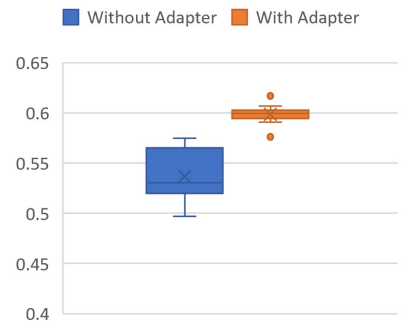


Figure 4: Box plots showing the TSP. stability of BERT with/without adapter on CoLA.

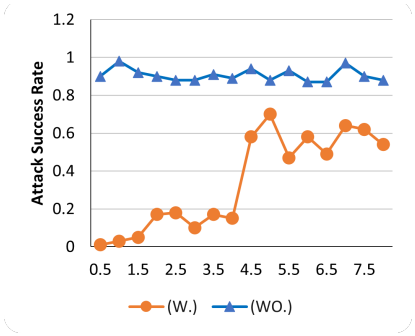


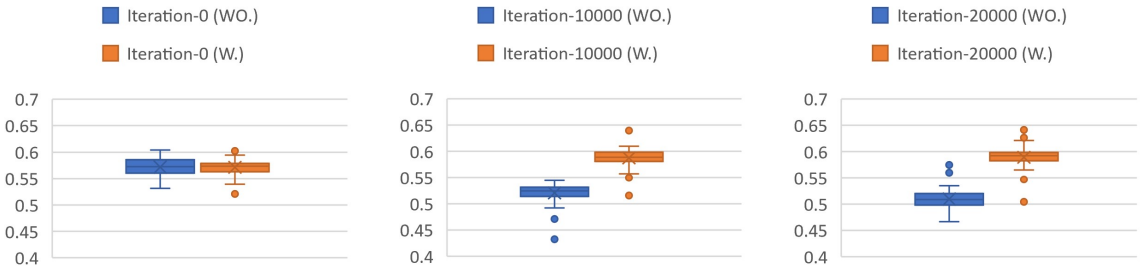
Figure 5: Attack success rate of BERT with/without adapter during task-specific pretraining. WO.: Without adapter. W.: With adapter.

2 Instability to Different Random Seeds

We first evaluate the training instability with respect to multiple random seeds: fine-tuning the model multiple times in the same setting, varying only the random seed. We conduct the experiments on RTE (Wang et al., 2018) when fine-tuning 1) BERT-base-uncased (Devlin et al., 2019) and 2) BERT-base-uncased with the adapter (Houlsby et al., 2019) [3]. As shown in Figure 2, the model without adapter leads to a large standard deviation on the fine-tuning accuracy, while the one with adapter results in a much smaller variance on the task performance.

Gradient Vanishing Mosbach et al. (2020) argues that the fine-tuning instability can be explained by optimization difficulty and gradient vanishing. In order to inspect if the adapter-based approach suffers from this optimization problem, we plot the L_2 gradient norm with respect to different layers of BERT, pooler layer and classification layer, for fine-tuning with or without adapter in

[3]For all the experiments, we use the implementation of Pfeiffer et al. (2020b): https://github.com/Adapter-Hub/adapter-transformers.git.



(a) Pretraining Iteration 0 (b) Pretraining Iteration 10000 (c) Pretraining Iteration 20000

Figure 9: Box plots showing the fine-tuning stability of BERT with/without adapter for different TSP. iterations on CoLA. WO.: Without adapter. W.: With adapter.

Figure 3.

In traditional fine-tuning (without adapter), we see vanishing gradients for not only the top layers but also the pooler layer and classification layer. This is in large contrast to the with-adapter fine-tuning. The gradient norm in the with-adapter fine-tuning does not decrease significantly in the training process. These results imply that the adaptation with adapter does not exhibit gradient vanishing and presents a less difficult optimization problem, which in turn might explain the improved stability of our approach.

3 Instability to Pretraining and Fine-tuning Iterations

Fine-tuning with all parameters also exhibits another instability issue. In particular, fine-tuning a model multiple times on the pretrained language model, varying the task-specific pretraining iterations and fine-tuning iterations, leads to a large standard deviation in downstream task performance. As observed in Figure 1, CoLA's performance when varying the task-specific pretraining iterations is severely unstable during pretraining iterations and fine-tuning iterations. The model has converged at the pretraining iteration of 8000. However, fine-tuning based on this model does not obtain the best performance.

Pretraining Iterations. Figure 4 displays the performance on CoLA of 10 fine-tuning runs with and without the adapter. For each run, we vary only the number of pretraining iterations from 2000 to 20000 with an interval of 2000 and fix the fine-tuning epochs to 10. We clearly observe that most runs for BERT with adapter outperforms the one without adapter. Moreover, the adapter makes pretraining BERT significantly more stable than the standard approach (without adapter).

Fine-tuning Iterations. We then study the stability with regard to the number of fine-tuning iterations. We show box plots for BERT using various pretraining iterations and fine-tuning iterations, with and without adapter in Figure 9. The three sub-figures represent the early, mid, and late stages of pretraining, corresponding to the 0-th, 10000-th, and 20000-th iteration respectively. The 0-th iteration represents the original model without task-specific pretraining. The model suffers from underfitting in the 0-th iteration and overfitting in the 20000-th iteration.

In Figure 9 (a), we plot the distributions of the development scores from 100 runs when fine-tuning BERT with various fine-tuning epochs ranging from 1 to 100. In the early stage, the average development score of the model with the adapter is a little lower than the baseline model while the stability is better. After several epochs of pretraining, the adapter gradually shows improved performance in terms of the mean, minimum and maximum as demonstrated in Figure 9 (b). In the end of the pretrainig, there exists an over-fitting problem for the traditional BERT models. Pretraining transfers the model to a specific domain and fails to maintain the original knowledge. In contrast, the performance with the adapter still grows as training continues and consistently benefit from pretraining. Besides, we observe that the adapter leads to a small variance in the fine-tuning performance, especially in the late stage. Additional plots and learning curves can be found in the Appendix.

4 Adversarial Robustness

While successfully applied to many domains, the predictions of Transformers (Vaswani et al., 2017)

become unreliable in the presence of small adversarial perturbations to the input (Sun et al., 2020; Li et al., 2020). Therefore, the adversarial attacker has become an important tool (Moosavi-Dezfooli et al., 2016) to verify the robustness of models. The robustness is usually evaluated from attack effectiveness (i.e., attack success rate). We use a SOTA adversarial attack approach to assess the robustness: PWWS attacker (Ren et al., 2019). [4]. Figure 5 shows the attack success rate of BERT with/without adapter during task-specific pretraining on SST-2. The x-axis is the number of epochs for task-specific pretraining. It can be observed that the model with the adapter has better adversarial robustness.

5 Conclusion

We propose a simple yet effective transfer learning scheme for large-scale pretrained language model. We insert small bottleneck layers (i.e., adapter) within each block of the pretrained model and then optimize the adapter layers in task-specific unsupervised pretraining and supervised training (i.e., fine-tuning) while fixing the pretrained layers. Extensive experiments demonstrate that our approach leads to improved stability with respect to different random seeds and different number of iterations in task-specific pretraining and fine-tuning, enhanced adversarial robustness, and better transfer learning task performance. We therefore consider the proposed training scheme as a robust and parameter-efficient transfer learning approach.

Acknowledgments

Y. W. is partially supported by NSF DMS 2015577.

References

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.

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.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. 2019. Mixout: Effective regularization to finetune large-scale pretrained language models. In *International Conference on Learning Representations*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Mike Lewis, Marjan Ghazvininejad, Gargi Ghosh, Armen Aghajanyan, Sida Wang, and Luke Zettlemoyer. 2020. Pre-training via paraphrasing. *Advances in Neural Information Processing Systems*, 33.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. Bert-attack: Adversarial attack against bert using bert. *arXiv preprint arXiv:2004.09984*.

Hairong Liu, Mingbo Ma, Liang Huang, Hao Xiong, and Zhongjun He. 2019. Robust neural machine translation with joint textual and phonetic embedding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3044–3049, Florence, Italy. Association for Computational Linguistics.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2020. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. In *International Conference on Learning Representations*.

[4] We use the implementation in OpenAttack toolkit https://github.com/thunlp/OpenAttack.git. It generates adversarial examples and evaluation the adversarial robustness of the victime model using thee adversarial examples. We use the default settings including all the hyper-parameter values.

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. BERTweet: A pre-trained language model for English Tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14.

Erik Nijkamp, Bo Pang, Ying Nian Wu, and Caiming Xiong. 2021. SCRIPT: Self-critic PreTraining of transformers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5196–5202, Online. Association for Computational Linguistics.

Matthew E Peters, Sebastian Ruder, and Noah A Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14.

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

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020b. Adapterhub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

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.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1085–1097.

Sebastian Ruder. 2019. *Neural transfer learning for natural language processing*. Ph.D. thesis, NUI Galway.

Lichao Sun, Kazuma Hashimoto, Wenpeng Yin, Akari Asai, Jia Li, Philip Yu, and Caiming Xiong. 2020. Adv-bert: Bert is not robust on misspellings! generating nature adversarial samples on bert. *arXiv preprint arXiv:2003.04985*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

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.

A Hyper-Parameters Setting

We conduct the experiments on the task of GLUE tasks (Wang et al., 2018) when fine-tuning 1) BERT-base-uncased (Devlin et al., 2019) and 2) BERT-base-uncased with the adapter architecture (Houlsby et al., 2019). For all the experiments, we use the implementation from https://github.com/Adapter-Hub/adapter-transformers.git. For the model with adapter, we follows the setup from Mosbach et al. (2020). For all experiments, we use the default hyper-parameters except for the number of epochs. Please refer to the provided link.

The main hyper-parameters are listed in Table 2 and Table 3.

Max Sequence Length	256
Batch Size	32
Learning rate	1e-4
Number of Epochs	20

Table 2: Hyper-parameters for BERT with Adapter.

Max Sequence Length	128
Batch Size	32
Learning rate	2e-5
Number of Epochs	10

Table 3: Hyper-parameters for BERT without Adapter.

B Instability to Pretraining and Fine-tuning Iterations

We provide box plots for BERT using various pre-training iterations and fine-tuning iterations, with and without adapter on CoLA in Figure 10. The corresponding learning curves are in Figure 13.

C Instability for Large Dataset

In contrast to relatively large datasets, smaller data is more suitable and convincing as an example to analyze stability. Small dataset is easier to encounter over-fitting problems and often not stable (Devlin et al., 2019). We use MNLI to evaluate the training instability in terms of 5 random seeds with the same setup in Figure 2. The interquartile range of BERT with adapter on the distribution of dev scores is smaller than BERT without adapter. It shows that the model without adapter consistently leads to the instability issue on the fine-tuning accuracy, while the adapter architecture brings less benefit with larger dataset.

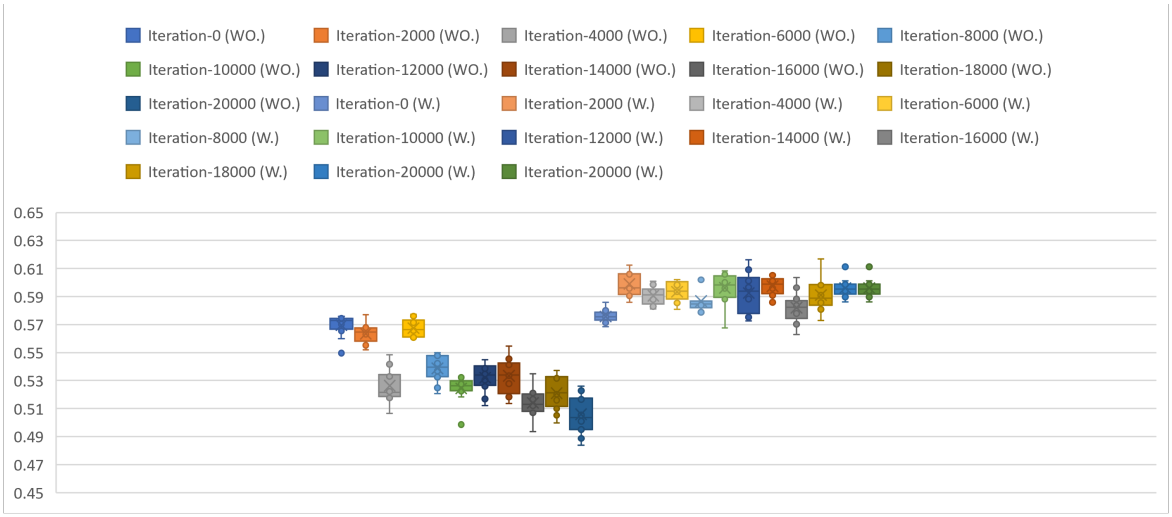
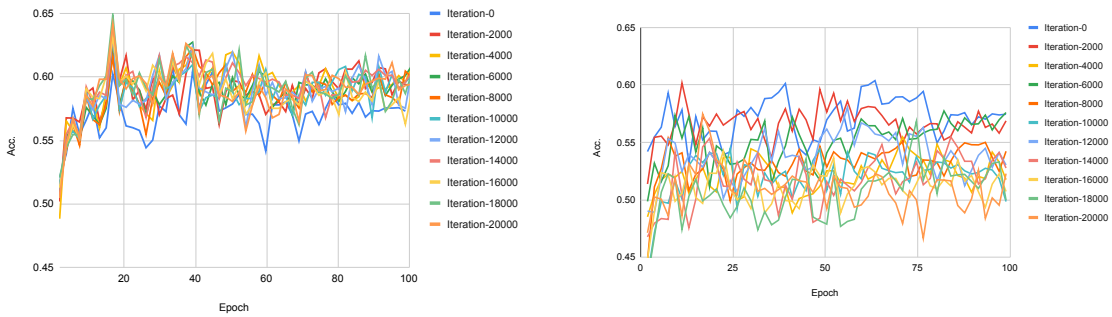


Figure 10: Box plots showing the fine-tuning stability of BERT with/without adapter for different pretraining iteration from 0 to 20000.



(a) BERT with adapter.

(b) BERT without adapter.

Figure 13: Learning curves of fine-tuning when varying the pretraining iterations.