

# Revisiting Pretraining with Adapters

Seungwon Kim, Alex Shum, Nathan Susanj, Jonathan Hilgart

Georgia Institute of Technology

{skim3222, ashum7, nsusanj3, jhilgart3}@gatech.edu

## Abstract

Pretrained language models have served as the backbone for many state-of-the-art NLP results. These models are large and expensive to train. Recent work suggests that continued pretraining on task-specific data is worth the effort as pretraining leads to improved performance on downstream tasks. We explore alternatives to full-scale task-specific pretraining of language models through the use of adapter modules, a parameter-efficient approach to transfer learning. We find that adapter-based pretraining is able to achieve comparable results to task-specific pretraining while using a fraction of the overall trainable parameters. We further explore direct use of adapters without pretraining and find that the direct fine-tuning performs mostly on par with pretrained adapter models, contradicting previously proposed benefits of continual pretraining in full pretraining fine-tuning strategies. Lastly, we perform an ablation study on task-adaptive pretraining to investigate how different hyperparameter settings can change the effectiveness of the pretraining.

## 1 Introduction

Pretrained Language Models (PLM) are predominant in tackling current Natural Language Processing (NLP) tasks. Most PLMs based on the Transformer architecture (Vaswani et al., 2017) are first trained on massive text corpora with the self-supervised objective to learn word representations (Devlin et al., 2019; Liu et al., 2019), and then are fine-tuned for a specific target task. The pretraining and fine-tuning of PLMs achieves state-of-the-art (SOTA) performance in many NLP tasks. Inspired by the benefits of pretraining, there have been studies demonstrate the effects of continued pretraining on the domain of a target task or the target task dataset (Mitra et al., 2020; Han and Eisenstein, 2019; Gururangan et al., 2020). Gururangan et al., 2020 adapt PLMs on the target task

by further pretraining RoBERTa (Liu et al., 2019) on the target text corpus before it is fine-tuned for the corresponding task and showed that this task adaptation consistently improves the performance for text classification tasks.

However, this full process of pretraining and then fine-tuning can be parameter inefficient for recent PLMs that have millions or billions of parameters (Devlin et al., 2019; Radford et al., 2018). This parameter inefficiency becomes even worse when one continues pre-training all the parameters of PLMs on the task-specific corpus. Furthermore, recent PLMs need more than 100s of MB to store all the weights (Liu et al., 2019; Radford et al., 2018), making it difficult to download and share the pre-trained models on the fly.

Recently, adapters have been proposed as an alternative approach to decrease the substantial number of parameters of PLMs in the fine-tuning stage (Houlsby et al., 2019). Finetuning with adapters mostly matches the performance of those with the full fine-tuning strategy on many NLP tasks including GLUE benchmark (Wang et al., 2018) and reduces the size of the model from 100s of MB to the order of MB (Pfeiffer et al., 2020b). As such, a natural question arises from the successes of the adapter approach: can the adapter alone adapt PLMs to the target task when it is used in the second phase of the pretraining stage and thus lead to the improvement of the performance on the corresponding task?

In this paper, we explore task-adaptive pretraining, termed TAPT (Gururangan et al., 2020), with adapters to address this question and overcome the limitations of the conventional full pretraining and fine-tuning. We only train the adapter modules in the second phase of pretraining as well as the fine-tuning stage to achieve both parameter efficiency and the benefits of continual pretraining and compare those with the adapter-based model without pretraining. Surprisingly, we find that directly

fine-tuning adapters performs mostly on par with the pre-trained adapter model and outperforms the full TAPT, contradicting the previously proposed benefits of continual pretraining in the full pretraining fine-tuning scheme. As directly fine-tuning adapters skips the second phase of pretraining and the training steps of adapters are faster than those of the full model, it substantially reduces the training time. We further investigate different hyperparameter settings that affect the effectiveness of pretraining.

## 2 Pretraining and Adapters

**Pre-trained language model** We use RoBERTa (Liu et al., 2019), a Transformer-based language model that is pre-trained on a massive text corpus, following Gururangan et al., 2020. RoBERTa is an extension of BERT (Devlin et al., 2019) with optimized hyperparameters and a modification of the pretraining objective, which excludes next sentence prediction and only uses the randomly masked tokens in the input sentence. To evaluate the performance of RoBERTa on a certain task, a classification layer is appended on top of the language model after the pretraining and all the parameters in RoBERTa are trained in a supervised way using the label of the dataset. In this paper, training word representations using RoBERTa on a masked language modeling task will be referred to as pretraining. Further, taking this pretrained model and adding a classification layer with additional updates to the language model parameters will be referred to as fine-tuning.

**Task-adaptive pretraining (TAPT)** Although RoBERTa achieves strong performance by simply fine-tuning the PLMs on a target task, there can be a distributional mismatch between the pretraining and target corpora. To address this issue, pretraining on the target task or the domain of the target task can be usefully employed to adapt the language models to the target task and it further improves the performance of the PLMs. Such methods can be referred to as Domain-Adaptive Pretraining (DAPT) or Task Adaptive-Pretraining (TAPT) (Gururangan et al., 2020). In this paper, we limit the scope of our works to TAPT as domain text corpus is not always available for each task, whereas TAPT can be easily applied by directly using the dataset of the target task while its performance often matches with DAPT (Gururangan et al., 2020). In TAPT, the second phase of pretraining is per-

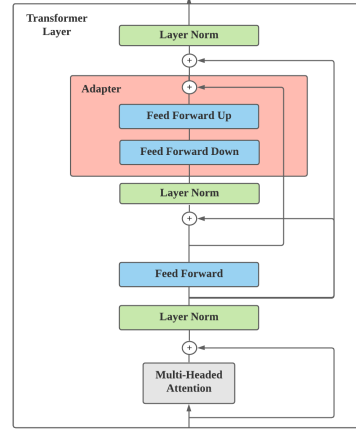


Figure 1: The adapter architecture in the Transformer layer (Pfeiffer et al., 2020a)

formed with RoBERTa using the unlabeled text corpus of the target task, and then it is fine-tuned on the target task.

**Adapter** Adapter modules have been employed as a feature extractor in computer vision (Rebuffi et al., 2017) and have been recently adopted in the NLP literature as an alternative approach to fully fine-tuning PLMs. Adapters are sets of new weights that are typically embedded in each transformer layer of PLMs and consist of feed-forward layers with normalizations, residual connections, and projection layers. The architectures of adapters vary with respect to the different configuration settings. We use the configuration proposed by Pfeiffer et al., 2020a in Figure 1, which turned out to be effective on diverse NLP tasks, and add the adapter layer to each transformer layer.

Pfeiffer et al., 2020c use two types of adapter: language-specific adapters and task-specific adapters for cross-lingual transfer. These two types of adapter modules have similar architecture as in Figure 1. However, the language adapters involve invertible adapters after the embedding layer to capture token-level language representation when those are trained via masked language modeling in the pretraining stage, whereas the task adapters are simply embedded in each transformer layer and trained in the fine-tuning stage to learn the task representation. Following Pfeiffer et al., 2020c, we employ language adapter modules with invertible layers to perform pretraining adapters on the unlabeled target dataset. However, we perform fine-tuning pre-trained parameters of the language adapter modules for evaluation to align with

Domain	Task	Label type	Number of inst (Train/Dev/Test)	Classes
Biomedical	CHEMPROT	Relationship classification	4169 / 2427 / 3469	13
Biomedical	RCT	Abstract sentence roles	18040 / 30212 / 30135	5
Computer Science	ACL-ARC	Citation intent	1688 / 114 / 139	6
Computer Science	SCIERC	Relation classification	3219 / 455 / 974	7
News	HYPERPARTISAN	Partisanship	515 / 65 / 65	2
News	AGNEWS	Topic	115000 / 5000 / 7600	4
Reviews	HELPPFULNESS	Review helpfulness	115251 / 5000 / 25000	2
Reviews	IMDB	Review sentiment	20000 / 5000 / 25000	2

Table 1: Datasets used for experimentation. Datasets include both high-resource (RCT (Dernoncourt and Lee, 2017), AGNEWS (Zhang et al., 2015), HELPPFULNESS (McAuley et al., 2015), IMDB (Maas et al., 2011)) and low-resource (CHEMPROT (Kringelum et al., 2016), ACL-ARC (Jurgens et al., 2018), SCIERC (Luan et al., 2018), HYPERPARTISAN (Kiesel et al., 2019) settings.

TAPT, whereas Pfeiffer et al., 2020c employ both the language and the task adapters by stacking task adapters on top of the language adapters.

### 3 Experiments

We now propose an adapter-based approach that is a parameter efficient variant of Task-Adaptive Pretraining (TAPT) and measure the margin of the performance between the pre-trained adapter model and the adapter model without pretraining. For pre-training adapters, we added the adapter module in each transformer layer of RoBERTa using adapter-transformer (Pfeiffer et al., 2020b)<sup>1</sup> and continued pretraining all the weights in adapter layers on target text corpus while keeping the original parameters in RoBERTa fixed. After finishing the second phase of pretraining, we performed fine-tuning of RoBERTa by training the weights in the adapters and the final classification layers while keeping all of the parameters in RoBERTa frozen.

#### 3.1 Dataset

Following Gururangan et al., 2020<sup>2</sup>, we consider 8 classification tasks from 4 different domains. The specification of each task is shown in Table 1. We covered news and review texts that are similar to the pretraining corpus of RoBERTa as well as scientific domains in which text corpora can have largely different distributions from those of RoBERTa. Furthermore, the pretraining corpora of the target tasks include both large and small cases to determine whether the adapter-based approach can be applicable in both low and high-resource settings.

<sup>1</sup><https://github.com/Adapter-Hub/adapter-transformers>

<sup>2</sup>Downloadable link for task dataset: <https://github.com/allenai/dont-stop-pretraining>

#### 3.2 Implementation Details

Our implementation is based on HuggingFace since we found AllenNLP (Gardner et al., 2018) used in Gururangan et al., 2020 is incompatible with adapter-transformer (Pfeiffer et al., 2020b). We follow the hyperparameters setting in Gururangan et al., 2020, and each model in the pretraining and fine-tuning stage is trained on a single GPU (NVIDIA RTX 3090). Details of hyperparameters are described in Appendix A. Note that for the pretraining step, we use a batch size of 8 and accumulate the gradient for every 32 steps to be consistent with the hyperparameter setting in Gururangan et al., 2020.

We perform pretraining with the self-supervised objectives, which are randomly masked tokens, with a probability of 15% for each epoch and we do not apply validation to pretraining and save the model at the end of the training from a single seed. For TAPT, we train the entire parameters of the RoBERTa via masked language modeling (MLM) on the target dataset, whereas for the adapter-based model, we embed the language adapters in each transformer layer and add invertible adapters after the embedding layers to perform MLM while freezing the original parameters of RoBERTa, following Pfeiffer et al., 2020c. Fine-tuning step is straightforward. We perform fine-tuning parameters that are pretrained via MLM for both TAPT and the adapter model. Validation is performed after each epoch and the best checkpoint is loaded at the end of the training to evaluate the performance on the test set.

#### 3.3 Experimental setup

Experiments cover four different models. First, we reproduce the performance of RoBERTa and TAPT in Gururangan et al., 2020 as presented in Appendix C. Then we proceed to the adapter-based approach.

Dataset	Baseline RoBERTa	TAPT	Adapter w/o PT	Adapter w/ PT
CHEMPROT	81.9 <sub>1.0</sub>	82.6 <sub>0.4</sub>	82.69 <sub>0.4</sub>	<b>82.71</b> <sub>0.4</sub>
RCT	87.2 <sub>0.1</sub>	<b>87.7</b> <sub>0.1</sub>	87.35 <sub>0.04</sub>	87.4 <sub>0.1</sub>
ACL-ARC	63.0 <sub>5.8</sub>	67.4 <sub>1.8</sub>	<b>69.47</b> <sub>2.4</sub>	69.25 <sub>2.5</sub>
SCIERC	77.3 <sub>1.9</sub>	79.3 <sub>1.5</sub>	81.5 <sub>0.9</sub>	<b>82.37</b> <sub>1.0</sub>
HYPERPARTISAN	86.6 <sub>0.9</sub>	90.4 <sub>5.2</sub>	<b>93.01</b> <sub>4.7</sub>	84.97 <sub>6.4</sub>
AGNEWS	93.9 <sub>0.2</sub>	<b>94.5</b> <sub>0.1</sub>	94.00 <sub>0.1</sub>	93.94 <sub>0.1</sub>
HELPFULNESS	65.1 <sub>3.4</sub>	68.5 <sub>1.9</sub>	<b>70.96</b> <sub>0.6</sub>	70.83 <sub>0.8</sub>
IMDB	95.0 <sub>0.2</sub>	95.5 <sub>0.1</sub>	95.51 <sub>0.1</sub>	<b>95.57</b> <sub>0.1</sub>
Average $F_1$	81.3	83.24	<b>84.31</b>	83.38
Trainable params per task (PT/FT)	-/124.64M	163.35M/124.64M	-/1.78M	2.18M/2.08M
Ratio to total params (PT/FT)	-/100%	100% /100%	-/1.42%	1.32%/1.65%
Relative training speed (PT/FT)	-/1.0	1.0/1.0	-/1.29	1.14/1.24
Relative inference speed (PT/FT)	-/1.0	1.0/1.0	-/0.98	0.88/0.98

Table 2: Average  $F_1$  score with standard deviation on test set. Each score is averaged over 5 random seeds. Evaluation metric is macro- $F_1$  scores on test set for each task except for CHEMPROT and RCT which use micro- $F_1$ . We report the results of baseline RoBERTa and TAPT from Gururangan et al., 2020. Following Rücklé et al., 2020, we measure the average relative speed for the training and the inference time across all tasks except for the inference speed in fine-tuning stage, which excludes low-resource tasks. PT and FT indicate pretraining and fine-tuning respectively.

To investigate the benefits of task-adaptive pretraining with adapters, we compare the performance of the pre-trained adapter model with the model without pretraining, i.e., directly fine-tuning adapters in RoBERTa on the target task.

For the adapter-based approach, we compare the adapter-based model with the second phase of pretraining and the model without the pretraining. Since the weights of the adapters are randomly initialized, we empirically found that a larger learning rate worked well compared to the full fine-tuning experiments. We sweep the learning rates in  $\{2e-5, 1e-4, 3e-4, 6e-4\}$  and the number of epochs in  $\{10, 20\}$  on the validation set and report the test score that performs the best on the validation set.

### 3.4 Results

The results are summarized in Table 2. Surprisingly, for the average  $F_1$  score, the adapter-based model without task-adaptive pretraining performs best, followed by the other adapter with the pretraining model, TAPT, and the baseline RoBERTa. Except for Hyperpartisan news, the adapter model without pretraining performs mostly on par with the counterpart adapter model that involves pretraining on target text corpus, suggesting that the benefits of additional task-adaptive pretraining diminish when we use the adapter-based approach. Furthermore, directly fine-tuned adapter model only trains 1.42% of the entire parameters which leads to the 30% faster-training step than the full model and skips the pretraining stage that typically expensive to train than the fine-tuning, substantially reducing

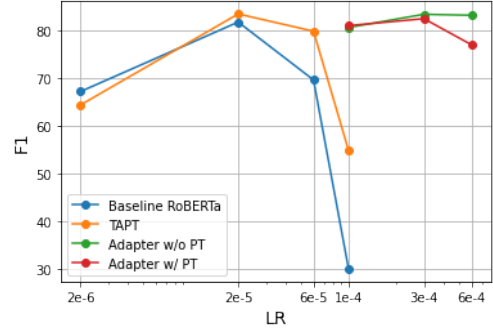


Figure 2:  $F_1$  score as a function of learning rate on test set with log scale on x-axis.  $F_1$  score is averaged over 5 random seeds for low-resource tasks (CHEMPROT, ACL-ARC, SCIERC, HYPER) due to the high variance. For high-resource tasks (RCT, AGNEWS, HELPFULNESS, IMDB), we report the  $F_1$  score from a single random seed for each task. For RoBERTa and TAPT, we follow the hyper-parameter settings in Gururangan et al., 2020 except for the learning rate.

the training time while the relative speed for the inference only decreases by 2% to the full model.

### 3.5 Analysis

We analyze how the adapter alone can surpass or perform on par with both the full model and adapter model with task-adaptive pretraining. Since we sweep the learning rates and the number of epochs in the range that includes larger figures compared to those in the full model when fine-tuning adapters and kept the other hyper-parameters the same as in Gururangan et al., 2020, we hypothesize that



Dataset	Baseline RoBERTa	TAPT
CHEMPROT	<b>82.8</b> <sub>0.9</sub>	82.62 <sub>0.5</sub>
RCT	86.89 <sub>0.1</sub>	<b>87.4</b> <sub>0.2</sub>
ACL-ARC	69.24 <sub>2.6</sub>	<b>70.08</b> <sub>2.3</sub>
SCIERC	80.59 <sub>0.9</sub>	<b>81.28</b> <sub>1.2</sub>
HYPER	<b>94.53</b> <sub>2.0</sub>	86.17 <sub>1.3</sub>
AGNEWS	93.9 <sub>0.2</sub>	<b>94.05</b> <sub>0.1</sub>
HELPFUL	69.63 <sub>0.6</sub>	<b>71.28</b> <sub>0.8</sub>
IMDB	94.93 <sub>0.1</sub>	<b>95.33</b> <sub>0.1</sub>
Average $F_1$	<b>84.06</b>	83.52

Table 3: Best performance of baseline RoBERTa and TAPT (Gururangan et al., 2020) on our implementation. Each score is averaged over 5 random seeds. Best configuration settings for each task is described in Appendix Table 8.

the larger learning rate zeroes out the benefits of pretraining. Figure 2. shows the average  $F_1$  score across all tasks as a function of learning rate.

The adapter model without a second phase of pretraining consistently outperforms or performs on par with the adapter model with pretraining from  $1e-4$  to  $6e-4$ , demonstrating that the additional pretraining turns out to be ineffective. In contrast, TAPT outperforms baseline RoBERTa from  $2e-5$ , where both TAPT and baseline RoBERTa perform best. The results show that different learning rates used in the fine-tuning stage can affect the effectiveness of pretraining and demonstrate that directly fine-tuning a fraction of parameters can provide comparable performance to the full-model as well as the adapter model with pretraining while substantially reducing the training time.

Inspired by the results of the adapter models, we perform the same experiments for the full model (baseline RoBERTa and TAPT) on our implementation by sweeping the learning rates and the number of epochs. We hypothesize that proper hyper-parameter settings such as a larger learning rate or increasing the number of training steps in the fine-tuning stage can improve the performance of baseline RoBERTa, making pretraining on the unlabeled target task less effective. We sweep the learning rate in  $\{1e-5, 2e-5, 3e-5\}$  and the number of epochs in  $\{10, 20\}$  on the validation set and report the test score that performs the best on the validation set. Table 3 shows the best performance of the full models for each task among different hyper-parameter settings. The average  $F_1$  score of baseline RoBERTa greatly increases and surprisingly, it surpasses the performance of TAPT in some tasks. The results ensure that although pretraining PLMs on the target task results in better

performance, one can achieve comparable performance by simply using a larger learning rate or increasing training steps in the fine-tuning stage while skipping the pretraining step that is computationally demanding compared to the fine-tuning.

## 4 Conclusion

Our work demonstrates that adapters provide a competitive alternative to large-scale task-adaptive pretraining for NLP classification tasks. We show that it is possible to achieve similar performance to TAPT with pretraining training just 1.32% of the parameters through pretraining with adapters. However, the most computationally efficient option is to skip pretraining and only perform fine-tuning with adapters. We found that skipping pretraining altogether and just fine-tuning with adapters outperforms or performs mostly on par with TAPT and the adapter model with pretraining across our tasks while substantially reducing the training time.

## Acknowledgments

We would like to thank Zsolt Kira, Mandeep Baines, Shruti Bhosale, and Siddharth Goyal for helpful feedback and suggestions. We also would like to thank anonymous reviewers for their insightful comments on the earlier version of the paper.

## References

- Franck Dernoncourt and Ji Young Lee. 2017. [PubMed 200k RCT: a dataset for sequential sentence classification in medical abstracts](#). In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 308–313, Taipei, Taiwan. Asian Federation of Natural Language Processing.
- 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.
- Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. [AllenNLP: A deep semantic natural language processing platform](#). In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. 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](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.
- Xiaochuang Han and Jacob Eisenstein. 2019. [Unsupervised domain adaptation of contextualized embeddings for sequence labeling](#). In *EMNLP*.
- 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.
- David Jurgens, Srikanth Kumar, Raine Hoover, Dan McFarland, and Dan Jurafsky. 2018. [Measuring the evolution of a scientific field through citation frames](#). *Transactions of the Association for Computational Linguistics*, 6:391–406.
- Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. [SemEval-2019 task 4: Hyperpartisan news detection](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 829–839, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Jens Kringelum, Sonny Kim Kjaerulff, Søren Brunak, Ole Lund, Tudor I Oprea, and Olivier Tabouret. 2016. Chemprot-3.0: a global chemical biology diseases mapping. *Database*, 2016.
- 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. *arXiv preprint arXiv:1907.11692*.
- Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. [Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics.
- Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. [Learning word vectors for sentiment analysis](#). In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.
- Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52.
- Arindam Mitra, Pratyay Banerjee, Kuntal Kumar Pal, Swaroop Ranjan Mishra, and Chitta Baral. 2020. [Exploring ways to incorporate additional knowledge to improve natural language commonsense question answering](#). *arXiv:1909.08855v3*.
- 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, Online. Association for Computational Linguistics.
- Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020c. [MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. [Improving language understanding by generative pre-training](#).
- Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. [Learning multiple visual domains with residual adapters](#). In *Advances in Neural Information Processing Systems*, volume 30, pages 506–516. Curran Associates, Inc.
- Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2020. Adapterdrop: On the efficiency of adapters in transformers. *arXiv preprint arXiv:2010.11918*.
- 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*, volume 30, pages 5998–6008. Curran Associates, Inc.
- 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, Brussels, Belgium. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. [Character-level convolutional networks for text classification](#). In *Advances in Neural Information Processing Systems*, volume 28, pages 649–657. Curran Associates, Inc.

## **A Hyperparameter Details**

Details of hyperparameter setting including the learning rates for the best performing results are provided in Table [4](#), [5](#), and [6](#).

## **B Validation Results**

We present validation performance in Table [7](#) and Figure [3](#).

## **C Replication results**

We provide replication results of [Gururangan et al., 2020](#) in Table [9](#).

Hyper-parameter	Value
Optimizer	Adam
Adam epsilon	1e-8, 0.999
Learning rate	1e-4
Batch size	8
Gradient accumulation step	32
Epochs	40 or 100
Adapter reduction factor	12
Maximum sequence length	512

Table 4: Details of hyperparameters used in pretraining experiments. We used 40 number of epochs for HELPFULNESS and 100 for the other tasks.

Hyper-parameter	Value
Optimizer	Adam
Adam epsilon	1e-8, 0.999
Batch size	16
Gradient accumulation step	1
Epochs	10 or 20
Patience	3 or 5
Adapter reduction factor	12
Dropout	0.1
Feedforward layer	1
Feedforward nonlinearity	tanh
Classification layer	1
Learning rate	see Table 6
Learning rate decay	linear
Warmup proportion	0.06
Maximum sequence length	512

Table 5: Details of hyperparameters used in fine-tuning experiments. For baseline RoBERTa and TAPT, we used 10 number of epochs with patience of 3 and the learning rate of 2e-5. For adapter experiments, see Table 6.

Dataset	Adapter w/o PT (LR, Epochs, Patience)	Adapter w/ PT (LR, Epochs, Patience)
CHEMPROT	3e-4, 20, 5	6e-4, 20, 5
RCT	1e-4, 10, 3	1e-4, 10, 3
ACL-ARC	6e-4, 10, 3	6e-4, 20, 5
SCIERC	3e-4, 20, 5	6e-4, 20, 5
HYPER	3e-4, 20, 5	1e-4, 20, 5
AGNEWS	1e-4, 10, 3	1e-4, 10, 3
HELPFUL	3e-4, 20, 5	1e-4, 20, 5
IMDB	1e-4, 10, 3	1e-4, 10, 3

Table 6: Learning rate, the nubmer of epochs and patience for best-performing models. For adapter experiments, we sweep the learning rates in  $\{1e-4, 3e-4, 6e-4\}$ , the number of epochs in  $\{10, 20\}$ , and patience factor in  $\{3, 5\}$  on validation set.



Dataset	Adapter w/o pretraining	Adapter w/ pretraining
CHEMPROT	83.77 <sub>0.5</sub>	84.02 <sub>0.7</sub>
RCT	88.16 <sub>0.1</sub>	88.13 <sub>0.1</sub>
ACL-ARC	72.41 <sub>2.2</sub>	77.31 <sub>2.9</sub>
SCIERC	86.86 <sub>0.5</sub>	87.87 <sub>0.3</sub>
HYPER	86.33 <sub>1.4</sub>	86.00 <sub>3.5</sub>
AGNEWS	94.28 <sub>0.1</sub>	94.57 <sub>0.1</sub>
HELPFUL	70.83 <sub>1.2</sub>	70.8 <sub>0.7</sub>
IMDB	95.52 <sub>0.1</sub>	95.6 <sub>0.1</sub>
Average $F_1$	84.77	85.54

Table 7: Validation performance of adapter experiments. Each score is averaged over 5 random seeds. Evaluation metric is macro- $F_1$  scores for each task except for CHEMPROT and RCT which use micro- $F_1$ .

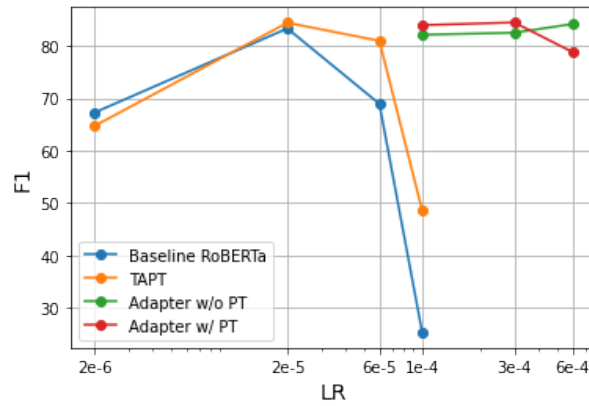


Figure 3:  $F_1$  score as a function of learning rate on development set with log scale on x-axis.  $F_1$  score is averaged over 5 random seeds for low-resource tasks (CHEMPROT, ACL-ARC, SCIERC, HYPER) due to the high variance. For high-resource tasks (RCT, AGNEWS, HELPFULNESS, IMDB), we report the  $F_1$  score from a single random seed for each task. Here we sweep the learning rate in  $\{1e-4, 3e-4, 6e-4\}$ , the number of epochs in  $\{10, 20\}$ , and the patience factor in  $\{3, 5\}$ .

Dataset	Baseline RoBERTa	TAPT	Hyper-parameters (LR, Epochs, Patience)
CHEMPROT	<b>82.8</b> <sub>0.9</sub>	82.62 <sub>0.5</sub>	3e-5, 20, 5
RCT	86.89 <sub>0.1</sub>	<b>87.4</b> <sub>0.2</sub>	2e-5, 10, 3
ACL-ARC	69.24 <sub>2.6</sub>	<b>70.08</b> <sub>2.3</sub>	3e-5, 20, 5
SCIERC	80.59 <sub>0.9</sub>	<b>81.28</b> <sub>1.2</sub>	2e-5, 20, 5
HYPER	<b>94.53</b> <sub>2.0</sub>	86.17 <sub>1.3</sub>	3e-5, 10, 3
AGNEWS	93.9 <sub>0.2</sub>	<b>94.05</b> <sub>0.1</sub>	2e-5, 10, 3
HELPFUL	69.63 <sub>0.6</sub>	<b>71.28</b> <sub>0.8</sub>	2e-5, 20, 5
IMDB	94.93 <sub>0.1</sub>	<b>95.33</b> <sub>0.1</sub>	2e-5, 20, 5
Average $F_1$	<b>84.06</b>	83.52	

Table 8: Validation performance of Baseline RoBERTa and TAPT experiments that corresponds to Table 3. Each score is averaged over 5 random seeds.

Dataset	Original Results Baseline RoBERTa	Original Results TAPT	Our Results Baseline RoBERTa	Our Results TAPT
CHEMPROT	81.9 <sub>1.0</sub>	82.6 <sub>0.4</sub>	81.64 <sub>0.8</sub>	82.58 <sub>0.5</sub>
RCT	87.2 <sub>0.1</sub>	87.7 <sub>0.1</sub>	86.89 <sub>0.1</sub>	87.4 <sub>0.2</sub>
ACL-ARC	63.0 <sub>5.8</sub>	67.4 <sub>1.8</sub>	64.12 <sub>5.5</sub>	66.11 <sub>4.6</sub>
SCIERC	77.3 <sub>1.9</sub>	79.3 <sub>1.5</sub>	78.89 <sub>2.7</sub>	79.94 <sub>0.7</sub>
HYPER	86.6 <sub>0.9</sub>	90.4 <sub>5.2</sub>	85.03 <sub>6.0</sub>	91.56 <sub>2.5</sub>
AGNEWS	93.9 <sub>0.2</sub>	94.5 <sub>0.1</sub>	93.72 <sub>0.2</sub>	94.05 <sub>0.1</sub>
HELPFULNESS	65.1 <sub>3.4</sub>	68.5 <sub>1.9</sub>	69.2 <sub>1.4</sub>	71.24 <sub>0.7</sub>
IMDB	95.0 <sub>0.2</sub>	95.5 <sub>0.1</sub>	95.15 <sub>0.1</sub>	95.33 <sub>0.1</sub>
Average $F_1$	81.3	83.24	81.83	83.53

Table 9: Reproducing Baseline RoBERTa and TAPT Results, average  $F_1$  Scores with standard deviation.  $F_1$  score is averaged over 5 random seeds. We use the same hyper-parameters in [Gururangan et al., 2020](#).