Evaluating Parameter-Efficient Tuning Methods in Low-Data Regimes

Haoming(Hammond) Liu
NYU Shanghai
hl3797@nyu.edu

Xiaochen(Nigel) Lu
NYU Shanghai
xl3139@nyu.edu

Wenbin(Jim) Qi
NYU Shanghai
wq372@nyu.edu

Abstract

Numerous parameter-efficient tuning methods have been proposed to reduce the computation and storage burden of standard fine-tuning. However, these methods are generally evaluated on large-scale benchmarks, which may not secure their robustness in low-data regimes. This work further evaluates the effectiveness and efficiency of several representative methods when different amounts of training samples are provided. The experiments have been conducted on various downstream tasks, which could provide empirical guidance on the choice of methods. The code is available at: https://github.com/hmdliu/MLLU-S22.

1 Introduction

Pre-trained language models (PLMs) have achieved great success in processing natural languages. *Fine-tuning*, though found effective by some predominant works like BERT (Devlin et al., 2019) and GPT (Brown et al., 2020), needs to store a model copy for each downstream task and update all the parameters for adaption, which is not efficient in terms of storage and computational resources.

To alleviate this issue, there have been many works exploring how to adapt PLMs to downstream tasks effectively and efficiently (Houlsby et al., 2019; Zaken et al., 2021; Hu et al., 2021; Li and Liang, 2021; Lester et al., 2021). In general, these methods tune a small set of adaptive parameters (inherently in the model or additionally introduced) instead of the whole PLM - such a paradigm is termed as *delta tuning* by Ding et al. (2022).

The evaluation of *delta tuning* methods are usually conducted on benchmarks (*e.g.,* GLUE, SuperGLUE) or datasets (*e.g.,* E2E, XSUM), where the amount of training samples are generally fixed and sufficient for the downstream task adaption (Wang et al., 2018, 2019; Novikova et al., 2017; Narayan et al., 2018). However, downstream tasks are likely to have limited training data, so the effectiveness and efficiency of different *delta tuning* methods may vary a lot depending on the number of training samples provided.

To address this potential vulnerability, we set up a unified testing framework to compare the performance and convergence speed of different *delta tuning* methods while varying the scale of training data. The experiments are conducted across various downstream tasks, including sentiment analysis, natural language inference, machine reading comprehension, and multi-choice question answering. We hope this work can provide empirical guidance for method selection.

2 Related Works

This section introduces the categorization of *delta tuning* methods and some representative methods to be evaluated. As Figure 1 illustrates, *delta tuning* methods can be categorized as addition-based, specification-based, and reparameterization-based methods, which are differentiated by the usage of adaptive parameters (Ding et al., 2022).

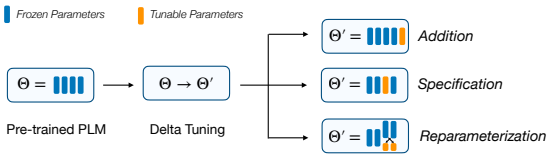

Figure 1: Illustrations of *delta tuning* categorization, where the pre-trained parameter set (Θ) is mapped to the well-tuned parameter set (Θ') with a split of frozen and tunable parameters. The figure is drawn from Ding et al. (2022) for demonstration purpose.

2.1 Addition-based Method

Addition-based methods introduce new tunable modules or some set of trainable parameters to the original model architecture (Ding et al., 2022).

Houlsby et al. (2019) proposes to insert small and tunable *Adapter* modules between layers of a pre-trained network. The *Adapter* module passes the features through a down-projection, a non-linearity, and an up-projection to adapt the PLM representations to downstream tasks.

2.2 Specification-based Method

Specification-based methods select a subset of parameters from the pre-trained model and make them trainable while freezing the remaining parameters in the model (Ding et al., 2022).

Zaken et al. (2021) tunes all the bias terms in the original PLM, which means the thresholds of non-linear activation are thus adjusted for downstream task adaption. Though the bias terms are merely an extremely small portion, this method still surprisingly well, especially on small-scale models.

2.3 Reparameterization-based Method

Reparameterization-based methods convert the existing parameters to a parameter-efficient form via some reparameterization tricks (Ding et al., 2022).

Hu et al. (2021) hypothesizes that the change of weights during model adaptation has a low intrinsic rank. Accordingly, we can inject trainable low-rank decomposition matrices to replace the self-attention weight matrices. This significantly reduces the number of trainable parameters and retains performance that is comparable to fine-tuning.

3 Method

3.1 Evaluation Methodology

In a recent paper, Ding et al. (2022) conduct a comprehensive study on different aspects of *delta tuning*. We largely adopt their unified testing framework and extend their study in low-data regimes, which basically investigates the variations of the performance and efficiency when different portions of training samples are provided.

Specifically, we sample subsets from each dataset by a log scale (*i.e.,* 0.1%, 1%, 10%, 100%) to construct a series of training set for evaluation. Then we apply the selected *delta tuning* methods on these subsets to observe the average metrics and the speed of convergence, which would demonstrate the robustness of different methods with limited data. Notably, the module or structure of each *delta tuning* method follows the default settings in the original paper; the configurations of training and hyper-parameter search are also unified. More implementation details are discussed in Section 4.1.

We hypothesize that the performance and convergence speed of a *delta tuning* method is proportional to the scale of its tunable parameters.

3.2 Pre-trained Language Model

We use `T5-Base` as the default PLM backbone for evaluation (Raffel et al., 2020). Moreover, we use the checkpoints released by Lester et al. (2021), which conducts an additional 100k steps of LM adaption. As T5 (Raffel et al., 2020) modulates all downstream tasks in a text-to-text format and pre-trains on a span corruption objective, these extra training steps were shown effective for boosting performance and convergence speed.

3.3 Dataset

To evaluate the selected *delta tuning* methods comprehensively, we use four large and representative datasets targeting distinctive downstream tasks.

MultiNLI. Multi-Genre Natural Language Inference corpus is a crowd-sourced collection of 433k sentence pairs annotated with textual entailment information of three categories (*i.e.,* neutral, entailment, or contradiction) (Williams et al., 2018).

RACE. RACE is a reading comprehension dataset with roughly 28k passages and 100k multiple-choice questions; the dataset is collected from English examinations for middle school and high school students in China (Lai et al., 2017).

SQuAD. Stanford Question Answering Dataset is a reading comprehension dataset that requires text-based answers; we use SQuAD v1.1 for this work, which consists of more than 100k question-answer pairs on over 500 articles (Rajpurkar et al., 2016).

Yelp Polarity. Yelp Polarity is a sentiment analysis dataset with a set of 560k highly polarized yelp reviews for training, and 38k for testing. The dataset is constructed by Zhang et al. (2015) based on the Yelp Dataset Challenge 2015 data[1].

4 Experiment & Analysis

4.1 Implementation Details

The codebase is implemented based on the Hugging Face library (Lhoest et al., 2021). The sequence-to-sequence trainer is adopted from Mahabadi et al. (2021) and the implementation of *delta*

[1]The challenge web page is defunct, the Yelp dataset is now available at: `https://www.yelp.com/dataset`

tuning methods is based on the OpenDelta library by Ding et al. (2022).

We use the AdamW optimizer (Loshchilov and Hutter, 2019) and apply a random search with 8 trials to find a fair training setting for each method. As shown in Table 1, the hyper-parameters includes learning rate, batch size, and max steps. As RACE and SQuAD have long inputs, we shrink their batch size search space and enlarge the max input length to 512 and 384 respectively (Rajpurkar et al., 2016; Lai et al., 2017). For MultiNLI and Yelp Polarity, we search the batch size between 16 and 32 and set the max input length to 128 (Zhang et al., 2015; Williams et al., 2018). To accommodate the low-data regimes, we apply early stopping with a patience of 3, and the evaluation is done every 2k steps (due to the job time limit on GCP).

Hyper-parameter	Search Space
Learning Rate	loguniform(1e-5, 1e-3)
Batch Size	{4, 8} or {16, 32}
Max Steps	{10k, 20k, 40k}

Table 1: Settings for hyper-parameter search.

Following OpenDelta implementation (Ding et al., 2022), we insert Adapter modules (Houlsby et al., 2019) with `SiLU` activations and a bottleneck dimension of 64; for BitFit (Zaken et al., 2021), we tunes all the bias terms by default; for LoRA (Hu et al., 2021), we use decompositions of rank 4 to reparameterize the attention module method.

4.2 Results & Analysis

We report the average metrics (%) under different downstream tasks and training data ratio in Table 2 and visualize the results in Figure 2. We use the exact match and F-1 scores for SQuAD and a single accuracy metric for the others.

Besides, some experiment details regarding the datasets are addressed as follows: 1) SQuAD has closed the test server for v1.1, so we evenly split the original development set as our new development and test set before all the experiments; 2) Yelp Polarity doesn't have a public development set, so we randomly chop off 38,000 samples from the training set for validation purpose, which has the same size as the public test set; 3) For MultiNLI, we were unable to evaluate our models on the test server for time reason, so the average metrics we reported in Table 2 are the accuracy evaluated on a development set with `validation-matched` and `validation-mismatched` concatenated.

Dataset	AP	BF	LR	FT
Tunable Ratio	2.38%	0.10%	0.38%	100%
Training Data Ratio: 100%				
MultiNLI	87.18	84.34	84.81	**87.41**
RACE	70.35	62.12	69.14	**74.09**
SQuAD v1.1	75.93	73.75	74.85	**76.84**
Yelp Polarity	96.27	95.82	95.99	**96.47**
Training Data Ratio: 10%				
MultiNLI	**83.11**	81.04	80.65	82.55
RACE	54.68	52.12	53.53	**57.15**
SQuAD v1.1	**70.79**	67.97	68.77	70.48
Yelp Polarity	**95.40**	95.08	94.87	95.37
Training Data Ratio: 1%				
MultiNLI	**77.44**	74.12	73.98	77.18
RACE	35.52	35.58	31.18	**40.72**
SQuAD v1.1	**60.83**	55.78	56.28	59.93
Yelp Polarity	94.14	93.47	93.61	**94.18**
Training Data Ratio: 0.1%				
MultiNLI	66.72	56.78	62.62	**68.12**
RACE	26.21	26.29	25.82	**26.76**
SQuAD v1.1	**36.59**	29.77	30.55	31.89
Yelp Polarity	92.18	88.87	91.81	**92.78**

Table 2: Average metrics (%) under different training data ratio. **AP** refers to Adapter (Houlsby et al., 2019); **BF** refers to BitFit (Zaken et al., 2021); **LR** refers to LoRA (Hu et al., 2021); **FT** refers to standard fine-tuning. The best performed method is shown in **bold**, and the worst performed method is underlined.

Several interesting results can be observed from the table and plots. First of all, almost every *delta tuning* method shows a decrease in performance testing against 4 datasets. Secondly, fine-tuning (**FT**) produces the highest scores almost in every setting except a few. This is consistent with our preliminary hypothesis since **FT** adjusts all the parameters. It's surprising that adapter tuning (**AP**) performs better than **FT** under a few settings, especially low data settings of SQuAD. One possible reason for this phenomenon is that **FT** has a relatively low convergence speed and it couldn't reach its full potential within the given max training steps. BitFit (**BF**) and LoRA (**LR**) show slightly worse performance compared to **FT** and **AP**. The plots show that either **BF** or **LR** is responsible for the

Figure 2: Comparisons of average metrics (%) under different downstream tasks and training data ratio.

lowest score in every setting. However, it's worth mentioning that **BF** is still a parameter-efficient *delta tuning* method and it presents enough competence even with such few parameters to tune.

Figure 3 (placed in the appendix due to page limit) plots the average metrics on the development set versus training steps, we compare the convergence speed under different downstream tasks and training data ratio. According to the results, all the methods converge quite fast when the training data ratio is extremely low, and the convergence gets slower as we increase the training data ratio.

Ranking by the performance: **FT** $\gtrsim$ **AP** $>$ **LR** $\gtrsim$ **BF**; whereas for the convergence speed: **FT** $\gtrsim$ **AP** $>$ **LR** $>$ **BF**. Notably, such results are highly consistent with our hypothesis in Section 3.1.

Additionally, we've also tested another *delta tuning* method called prefix-tuning (**PT**) (Li and Liang, 2021). However, **PT** gives unstable performance on some preliminary experiments, so we discard it from the main experiments. More details about **PT** are discussed in the appendix.

5 Conclusion

This work investigates the performance of three representative parameter-efficient tuning methods (Adapter, BitFit, and LoRA) in low-data regimes. All three methods show competitive results under limited data, whose performance is comparable to standard fine-tuning with merely a small portion of tunable parameters. As a takeaway, we would generally recommend the Adapter method for its outstanding performance and fast convergence speed across various tasks and data ratios. In comparison, BitFit and LoRA are more memory-friendly (*i.e.,* $24\times$ and $6\times$ times less tunable parameters compared to the Adapter method) but with a slight drop on the overall performance.

Ethical Considerations

Parameter-efficient tuning methods aim to reduce the computation and storage burden of standard fine-tuning, which reduces the resources for model tuning. Such methods are environment friendly and thus should be further promoted to the communities in both academia and industry.

All these methods (including fine-tuning) essentially do the same job (*i.e.,* adapt a pre-trained model to some downstream tasks). Though all of them are proposed to facilitate our lives in various ways (*e.g.,* grammar checking, auto-filling, etc.), we still need to be extremely cautious while applying such methods (*e.g.,* avoid using it for malicious purpose or causing privacy concerns).

Collaboration Statement

All group members attended the regular project meetings, completed the literature review, ran the experiments, and wrote up the final report together. Additionally, Hammond set up the codebase for evaluation; Jim analyzed the experiment results; Nigel visualized the experiment results. All the experiments were conducted on NYU HPC (GCP). This work is supervised by Professor Samuel R. Bowman, Arka Talukdar, and Eugene Choi during the DS-UA 203: Machine Learning for Language Understanding course (Spring 2022).

Acknowledgements

We thank Jason Phang for advising on this work and Shenglong Wang for the help on HPC usage.

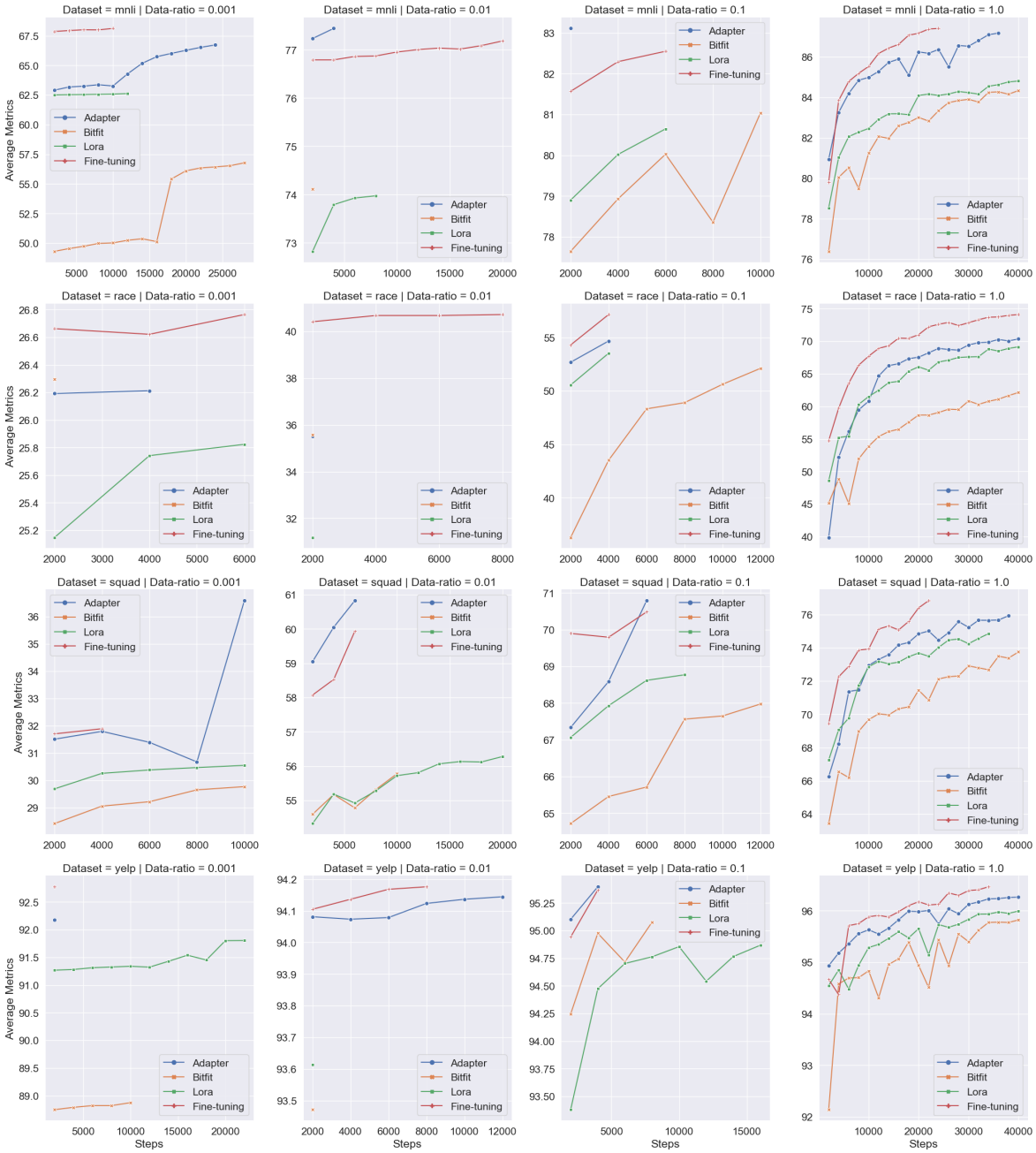

Figure 3: Comparisons of convergence speed under different downstream tasks and training data ratio.

A Comparisons of Convergence Speed

Figure 3 plots the average metrics on the development set versus training steps, where the evaluation is conducted every 2k steps. In fact, decreasing the evaluation interval would better reflect the convergence patterns, but we were unable to do so due to the job time limit on GCP.

B Prefix-tuning (discarded method)

Li and Liang (2021) proposes to prepend a small continuous task-specific vector (termed *prefix*) to the input and hidden states at each Transformer layer. Moreover, the *prefix* is reparameterized by a MLP to stabilize training.

In some preliminary experiments, we found that prefix-tuning was extremely unstable to train even with 8 search trials, so we finally discard it.

Method	MNLI	RACE	SQuAD	Yelp
FT	87.41	74.09	76.84	96.47
PT	77.13	26.76	29.70	92.57

Table 3: Average metrics (%) under full training data. **FT** refers to fine-tuning, **PT** refers to prefix-tuning.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models.

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, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, 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. 2021. Lora: Low-rank adaptation of large language models. *ArXiv preprint*, abs/2106.09685.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *ArXiv preprint*, abs/2104.08691.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

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

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. *ArXiv preprint*, abs/2106.04647.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany. Association for Computational Linguistics.

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.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

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.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

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

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.