G-Tuning: Improving Generalization of Pre-trained Language Models with Generative Adversarial Network

Rongxiang Weng
Soochow University, China
miHoYo AI, China
wengrongxiang@gmail.com

Wensen Cheng
miHoYo AI, China
vinson7973@gmail.com

Min Zhang
Soochow University, China
minzhang@suda.edu.cn

Abstract

The generalization ability of pre-trained language models (PLMs) in downstream tasks is heavily influenced by fine-tuning. The objective of fine-tuning is to transform the latent representation of PLMs from a universal space to a target space, allowing the model to be applied to downstream tasks with the capability of generalizing to unseen samples. However, the effect of PLMs will be diminished when the training data coverage is insufficient, in which fine-tuning is inadequate to learn the complete mapping. In this study, we propose a new fine-tuning framework, referred to as G-Tuning, that aims to preserve the generalization ability of PLMs in downstream tasks. Specifically, we integrate a generative adversarial network into the fine-tuning process to aid in the transformation of the latent representation in the entire space. Empirical evaluations on the GLUE benchmark, as well as two additional demanding scenarios involving domain and language generalization, demonstrate that G-Tuning can accurately map the universal representation to the target space, thus effectively enhancing the generalization performance of PLMs across various downstream tasks.

1 Introduction

Large-scale pre-trained language models (PLMs) have demonstrated substantial achievements in natural language processing (NLP) recently (Qiu et al., 2020; Han et al., 2022). Generally, fine-tuning PLMs with the task-specific training data can get significant improvements compared to training a model from scratch (Devlin et al., 2019; Lample and Conneau, 2019; Radford et al., 2018; Ouyang et al., 2022). Fine-tuning aims at transforming the representation from PLMs in the universal space to the target space, thereby enabling the model to generalize to a wider range of samples (Pan and Yang, 2010; Liu et al., 2021; Wei et al., 2021).

However, the generalization capability of PLMs is largely affected by the task-specific data when using fine-tuning to further train the model (Patel et al., 2022). As noted by Wu et al. (2022), fine-tuning is susceptible to memorizing the training data when the capacity of the PLM exceeds that of the downstream task data. Furthermore, the advantage of PLMs over random initialization is lost when the coverage of task-specific data is low, resulting in a large gap between training and test data (Zoph et al., 2020; He et al., 2019). For instance, in domain generalization, PLMs fine-tuned with in-domain training sets often fail to perform well on out-of-domain test sets, even when data from the test sets is used for pre-training (Yang et al., 2022). Therefore, a crucial challenge for unlocking the potential of PLMs is how to learn a complete and accurate mapping from the universal space to the target space with limited training data.

Previous studies have presented some promising approaches to address this problem. Fang et al. (2020) proposed a self-teaching method to use a fine-tuned PLM to get soft labels of the unlabeled data and train another PLM by these synthetic data, which brings considerable improvements in the cross-lingual transfer scenario. Li and Zhang (2021) presented regularized self-labeling to correct mislabeled data points and reweight less confident data points to regularize PLMs. Li et al. (2022) proposed an ensemble learning method for domain generalization, which can dynamically dispatch proper PLMs to predict each test sample. Wu et al. (2022) proposed a noisy tuning method to add matrix-wise perturbation to different parameter matrices to overcome the outfitting problem of fine-tuning, which indirectly improve the generalization ability of PLMs. Lu et al. (2022) presented stochastic weight averaging to improve generalization by encouraging convergence of the model to a flatter minimum. Furthermore, compared to fine-tuning, in-context learning which doesn't tune the parameter of PLMs, we leave this to future work (Li and Liang, 2021; Brown et al., 2020; Vu et al., 2022).

In this paper, we propose a novel fine-tuning framework (*named* G-Tuning) to preserve the generalization capability of PLMs when training with task-specific data. We adopt parameter efficient fine-tuning (PEFT) as our backbone, which only tunes a lightweight adapter network connected behind PLMs (Houlsby et al., 2019)[1], and incorporate a generative adversarial network (Goodfellow et al., 2020; Arjovsky et al., 2017; Gulrajani et al., 2017) into it to learn the representation mapping. Specifically, we first train a discriminator to aim of discriminating the representation that is not mapped correctly from the target space. Then, besides predicting the ground-truth label, the model is seen as a generator requested to generate the representation hard to be discriminated. We conduct experiments on the GLUE and two more challenging scenarios, *i.e.*, domain and language generalizations. Experimental results show that G-Tuning can improve generalization capability by effectively mapping the universal representation to the target space.

2 Methodology

2.1 Preliminaries

Pre-trained Language Model. Given a large-scale unlabeled data-set $\boldsymbol{M}$, the widely-used training function of PLMs is

$$\mathcal{L}_{\mathcal{P}}(\theta) = -\mathbb{E}_{\boldsymbol{x}^u\sim\boldsymbol{M}}[\log P(\boldsymbol{x}^u|m(\boldsymbol{x}^u);\theta)], \quad (1)$$

where $\boldsymbol{x}^u$ is an input sequence and $m(\cdot)$ is a perturbation function, which masks tokens in the $\boldsymbol{x}^u$ by a certain rule (Devlin et al., 2019; Radford et al., 2018). The θ is the parameter set of the PLM, which generally adopts the Transformer structure (Vaswani et al., 2017).

Parameter Efficient Fine-tuning. Given a training set $\boldsymbol{B}$ from a downstream task, we can summarize the loss function of PEFT (Houlsby et al., 2019; He et al., 2022a) as:

$$\mathcal{L}_{\mathcal{T}}(\phi) = -\mathbb{E}_{\{\boldsymbol{x}^t,\boldsymbol{y}^t\}\sim\boldsymbol{B}}[\log P(\boldsymbol{y}^t|\boldsymbol{x}^t;\theta,\phi)], \quad (2)$$

where $\boldsymbol{x}^t$ is the input and $\boldsymbol{y}^t$ is the corresponding label. PEFT only trains an adapter parameterized by ϕ, and the parameter θ of the PLM is fixed.

2.2 The Proposed G-Tuning

Before elaborating on the G-Tuning, we first define the composition of the PLM and the adapter

[1]Compared to fine-tuning the whole model, PEFT is efficient and stable in our experiments (see Appendix A).

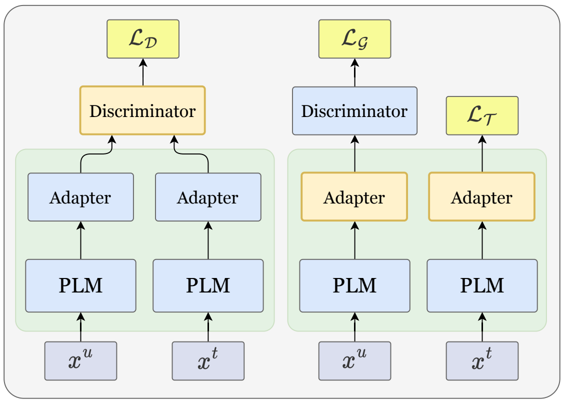

Figure 1: An illustration of the training process of the proposed G-Tuning.

as $f(\cdot)$. Given an input, the output of $f(\cdot)$ is a vector or the mean pooling of a matrix according to different tasks. Inspired by the Wasserstein GAN (WGAN) (Arjovsky et al., 2017; Gulrajani et al., 2017), we train a discriminator $D(\cdot)$ as follows:

$$\begin{aligned}\mathcal{L}_{\mathcal{D}} = &-\mathbb{E}_{\boldsymbol{x}^t\sim\boldsymbol{B},\boldsymbol{x}^u\sim\boldsymbol{M}}[D(f(\boldsymbol{x}^t)) - D(f(\boldsymbol{x}^u))] \\ &+ \lambda\mathbb{E}_{\boldsymbol{z}\sim\boldsymbol{N}}[||\nabla D(\boldsymbol{z})||_2 - 1]^2. \quad (3)\end{aligned}$$

Here, the second term is gradient penalty, which is used to smooth the weight of $D(\cdot)$ (Gulrajani et al., 2017). The coefficient λ is set as 10 and the latent representation $\boldsymbol{z}$ is computed by:

$$\boldsymbol{z} = \epsilon f(\boldsymbol{x}^u) + (1-\epsilon)f(\boldsymbol{x}^t), \epsilon \sim \mathcal{N}(0,1). \quad (4)$$

We consider the representation from $f(\boldsymbol{x}^t)$ obeys the real distribution, and from $f(\boldsymbol{x}^u)$ obeys the generated distribution. The aim of $D(\cdot)$ is to identify the representation that did not correctly map from the universal space to the target space.

We think of $f(\cdot)$ as the generator which can be optimized by the following loss function:

$$\mathcal{L}_{\mathcal{G}}(\phi) = -\mathbb{E}_{x^u\sim\boldsymbol{M}}[D(f(\boldsymbol{x}^u))]. \quad (5)$$

Finally, different from the original WGAN, we combine the loss functions from Eq. 2 and Eq. 5 in a multi-task learning paradigm (Sener and Koltun, 2018) to optimize the adapter network:

$$\mathcal{L}(\phi) = \alpha\mathcal{L}_{\mathcal{T}}(\phi) + \beta\mathcal{L}_{\mathcal{G}}(\phi), \quad (6)$$

where the coefficient α and β are used to control the loss function, which we set as 1 and 0.5, respectively.[2] Here, we utilize $\mathcal{L}_{\mathcal{G}}$ to learn the representation mapping. To avoid the deviation of the target space, we further use $\mathcal{L}_{\mathcal{T}}$ to keep it consistent. An illustration of the G-Tuning is shown in Figure 1.

[2]The overall training algorithm is shown in Appendix B.

Model	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STSB	Avg.
XLNET†	89.8	93.9	91.8	83.8	95.6	89.2	63.6	91.8	87.4
RoBERTa†	90.2	94.7	92.2	86.6	96.4	90.9	68.0	92.4	88.9
HyperPrompt†	90.3	95.0	87.0	**87.7**	96.7	**93.6**	57.5	91.9	87.5
RoBERTa	90.5	95.1	92.3	85.5	96.5	90.9	68.3	92.0	88.9
+ NoisyTune	90.8	95.7	91.8	86.4	96.3	91.3	68.2	92.7	89.2
+ Self-Teach	88.6	95.3	91.7	84.9	96.2	88.9	67.4	91.7	88.1
+ Adapter	89.9	92.7	90.5	85.3	96.0	91.5	67.9	93.4	88.4
+ G-Tuning	**90.9**	**95.9**	**92.8**	87.1	**96.9**	92.3	**69.8**	**93.1**	**89.9**

Table 1: Results on the GLUE. "Avg." is the average score of all tasks. "†" means the results come from their paper.

3 Experiments

3.1 Implementation Detail

Data-set. We first experiment on the GLUE benchmark (Wang et al., 2018). In consideration of the labels of the test sets are not released, we report results on the validation sets. Further, we report the result of MNLI-matched for the MNLI task and do not evaluate the WNLI task due to problems with the data. Then, we conduct experiments in two more challenging scenarios: domain generalization and language generalization. In domain generalization, following Yang et al. (2022), we reorganize the date set of each task in GLUE (*named* GLUE$_{\text{ood}}$) and use the same metric to evaluate the model. In language generalization, we adopt XTREME benchmark (Hu et al., 2020) to evaluate our approach, in which we use English training data to tune the model and transfer it to other languages. In addition, more details of GLUE and XTREME benchmarks can refer to their papers (Wang et al., 2018; Hu et al., 2020). The evaluation metric of all tasks and the data statistic of the GLUE$_{\text{ood}}$ data-set are shown in Appendix C.

Setting. Depending on the type of task, we use the large setting of RoBERTa (Liu et al., 2019) and XLM-R (Conneau et al., 2020) as the foundation model. We set the batch size as 32 and the number of gradient accumulations as 2. The training epoch of all models is 10. We compose the data-set M by randomly sample the Similar to previous work (Xu et al., 2022; Zaken et al., 2022), we search learning rate from {5e-6,1e-5,5e-5,1e-4,5e-4}. The optimization frequency of the discriminator is {3,5,7,9} times than the generator. We use Adam (Kingma and Ba, 2014) as the optimizer for our method. We use a three layers transformer structure as the discriminator and the adapter, respectively. We first fine-tune the PLM for 5 epochs; then, we use the fine-tuned model to train the discriminator and employ the proposed method with another 5 epochs. For the sentence pair task, we compute the representation for each sentence individually and combine them before the output layer. For the structure prediction task, we use the average of all the outputs of all tokens as input. We report the average score of 3 runs with different seeds. Experiments are performed on 4 NVIDIA A100 GPUs.

3.2 Main Results

Results on GLUE. The results of the GLUE benchmark are presented in Table 1. We first report the results for several widely-used PLMs, *i.e.*, XLNET (Yang et al., 2019) and RoBERTa (Liu et al., 2019), as well as a prompt-based method, HyperPrompt (He et al., 2022b). To ensure fair comparison, we implement the self-teaching (Self-Teach) (Fang et al., 2020), noisy tuning (Noisy-Tune) (Wu et al., 2022) and the standard adapter-based method (as trained by Eq. 2) with the same structure of our model. Our G-Tuning approach gets a 1.0 absolute improvement compared to fine-tuning RoBERTa directly. Additionally, our model consistently outperforms previous work.

Subsequently, we evaluate the generalization capabilities of our approach in domain generalization and language generalization. While G-Tuning gets considerable improvements on the GLUE benchmark, it is important to consider whether it effectively transforms the entire representation space or only a neighborhood surrounding the training data.

Results on Domain Generalization. The results are presented in Table 2. Similar to Yang et al. (2022), we evaluate our approach by exploiting out-of-domain (OOD) data as test sets. Compared to the standard dev sets on the GLUE, the performance of all methods exhibits a notable decline on the OOD test sets. Moreover, the results of fine-tuning the whole model are inferior to those of

Model	MNLI/SICK	RTE/HANS	SST/IMDB	CoLA/ColAood	STSB/SICK	Avg.
RoBERTa	72.3	65.4	85.9	28.2	84.7	67.3
+ Self-Teach	75.6	62.3	83.4	29.2	86.5	67.4
+ NoisyTune	75.3	65.7	89.3	28.6	87.1	69.2
+ Adapter	75.7	66.8	88.1	27.2	86.3	68.8
+ G-Tuning	**82.1**	**69.9**	**92.3**	**44.3**	**90.2**	**75.8**

Table 2: Results on the GLUE_{ood} (Domain Generalization). "Avg." is the average score. "$*/*$" is training/test set.

Model	Sent Pair		Struct Pred		Sent Retrieval		Question Answering			Avg.
	XNLI	PAWS-X	POS	NER	BUCC	Tatoeba	XQuAD	MLQA	TyDiQA	
XLM-R†	79.0	86.3	72.7	62.3	79.2	76.0	76.2	71.4	65.0	74.2
NoisyTune†	79.3	86.5	73.5	63.2	79.9	76.8	76.7	71.9	65.4	74.8
InfoXLM	80.6	87.1	**74.3**	64.1	80.4	**77.5**	75.3	72.9	**66.8**	75.4
XLM-R	79.3	86.7	73.3	64.9	79.6	76.3	76.4	71.2	64.3	74.7
+ NoisyTune	80.1	87.4	72.4	64.5	78.8	77.7	76.7	72.5	64.9	75.0
+ Self-Teach	79.4	85.8	74.1	65.4	79.1	75.5	75.9	72.1	65.2	74.7
+ Adapter	79.9	86.1	73.5	65.6	79.5	76.8	77.1	71.5	63.8	74.9
+ G-Tuning	**80.9**	**87.5**	74.2	**66.3**	**80.8**	77.3	**77.7**	**73.4**	66.5	**76.1**

Table 3: Results on the XTREME benchmark (Language Generalization). "Sent" is short for sentence and "Struct Pred" is structural prediction. "Avg." is the average score of all tasks. "†" means the results come from their paper.

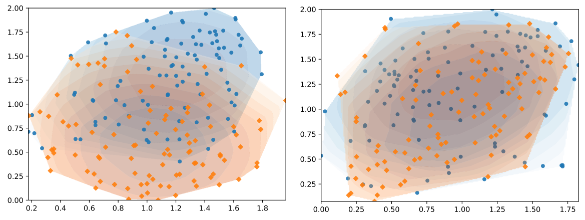

Figure 2: The scatter figure of the representation of in-domain (Orange Square) and out-of-domain (Blue Circle) data generated by fine-tuning (Left) and G-Tuning (Right), respectively.

other methods, suggesting that training PTMs with in-domain data will lead to a severe decline in generalization ability. In comparison to the strongest baseline, G-Tuning achieves an average improvement of 6.4 in the domain generalization scenario.

Results on Language Generalization. The overall results on XTREME benchmark are shown in Table 3. Compared to fine-tuning XLM-R, our method achieves an average improvement of 1.4. However, the improvement is lower than that observed in the domain generalization. We posit that the characteristics of different languages have an impact on language generalization. Here, we do not utilize any bilingual parallel data, which makes it challenging to learn the alignment of these characteristics, resulting in limited improvements.

3.3 Analysis

We sampled 200 sentences each from in-domain and out-of-domain data and used the model fine-tuned and G-Tuned to generate the representations. We then normalized the representations and reduce the dimensionality using t-SNE. The visualization is shown in Figure 2. We also included contour lines based on sample density in the figure. It is apparent that the density centers of different domains are nearly coincident in our model, whereas fine-tuning results in a significant gap between different domains. Empirically, the representation in a task-specific space should be centralized. The gap from the fine-tuning method leads to incorrect label predictions for some data, *e.g.*, the samples in the upper right corner of the left figure.

4 Conclusion

In this work, we elucidate a drawback of the fine-tuning strategy on PLMs, which is that the representation from PLMs is not fully mapped to the target space when the training data is insufficient. The generalization ability of PLMs in the downstream tasks will be diminished in this situation. To address this issue, we present G-Tuning, a framework that aims to preserve the generalization ability of PLMs in the downstream tasks. The proposed G-Tuning utilizes a generative adversarial network to transform the representation that is not covered by training data into the target space. Extensive experiments on the GLUE and two additional scenarios show that G-Tuning accurately maps the universal representation to the target space, getting substantial improvements in generalization performance.

Limitations

In this section, we will summarize several limitations of our work. The first one is we only apply the proposed method to the natural language understanding (NLU) tasks. It is uncertain how to extend our approach to the natural language generation (NLG) tasks and whether it can bring considerable improvement. Then, the training process of GAN is sensitive to hyper-parameters, leading to us not simply using the default setting when extending to other tasks. Finally, this paper does not include a theoretical explanation and proof of how our method works, which we will further study in future work.

Acknowledgments

We would like to thank the reviewers for their insightful comments. This work is supported by the National Natural Science Foundation of China (Grant No. 62036004).

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *ICML*, pages 214–223.

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. *NeurIPS*, 33:1877–1901.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *ACL*, pages 8440–8451.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

Yuwei Fang, Shuohang Wang, Zhe Gan, Siqi Sun, and Jingjing Liu. 2020. Filter: An enhanced fusion method for cross-lingual language understanding. *arXiv:2009.05166*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. *NeurIPS*, 30.

Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. 2022. A survey on vision transformer. *IEEE TPAMI*.

Kaiming He, Ross Girshick, and Piotr Dollár. 2019. Rethinking imagenet pre-training. In *ICCV*, pages 4918–4927.

Xuehai He, Chunyuan Li, Pengchuan Zhang, Jianwei Yang, and Xin Eric Wang. 2022a. Parameter-efficient fine-tuning for vision transformers. *arXiv preprint arXiv:2203.16329*.

Yun He, Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, Yaguang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng, and Ed H. Chi. 2022b. HyperPrompt: Prompt-based task-conditioning of transformers. In *ICML*, volume 162, pages 8678–8690.

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 *ICML*, pages 2790–2799. PMLR.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *ICML*, volume 119, pages 4411–4421.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv:1901.07291*.

Dongyue Li and Hongyang Zhang. 2021. Improved regularization and robustness for fine-tuning in neural networks. *NeurIPS*, 34:27249–27262.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, pages 4582–4597.

Ziyue Li, Kan Ren, Xinyang Jiang, Bo Li, Haipeng Zhang, and Dongsheng Li. 2022. Domain generalization using pretrained models without fine-tuning. *arXiv*.

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*.

Ziquan Liu, Yi Xu, Yuanhong Xu, Qi Qian, Hao Li, Xiangyang Ji, Antoni B Chan, and Rong Jin. 2021. Improved fine-tuning by better leveraging pre-training data. In *NeurIPS*.

Peng Lu, Ivan Kobyzev, Mehdi Rezagholizadeh, Ahmad Rashid, Ali Ghodsi, and Philippe Langlais. 2022. Improving generalization of pre-trained language models via stochastic weight averaging. *arXiv*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *arXiv*.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Arkil Patel, Satwik Bhattamishra, Phil Blunsom, and Navin Goyal. 2022. Revisiting the compositional generalization abilities of neural sequence models. In *ACL*, pages 424–434.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *SCTS*, pages 1872–1897.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *CoRR*.

Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. *NIPS*, 31.

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 *NIPS*, volume 30.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2022. Spot: Better frozen model adaptation through soft prompt transfer. In *ACL*, pages 5039–5059.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. In *ICLR*.

Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2022. Noisytune: A little noise can help you finetune pretrained language models better. In *ACL*, pages 680–685.

Runxin Xu, Fuli Luo, Baobao Chang, Songfang Huang, and Fei Huang. 2022. S4-tuning: A simple cross-lingual sub-network tuning method-tuning: A simple cross-lingual sub-network tuning method. In *ACL*, pages 530–537.

Linyi Yang, Shuibai Zhang, Libo Qin, Yafu Li, Yidong Wang, Hanmeng Liu, Jindong Wang, Xing Xie, and Yue Zhang. 2022. Glue-x: Evaluating natural language understanding models from an out-of-distribution generalization perspective. *arXiv*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv:1906.08237*.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *ACL*, pages 1–9.

Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin Dogus Cubuk, and Quoc Le. 2020. Rethinking pre-training and self-training. *NeurIPS*, 33:3833–3845.

A The Comparison of Fine-tuning and PEFT

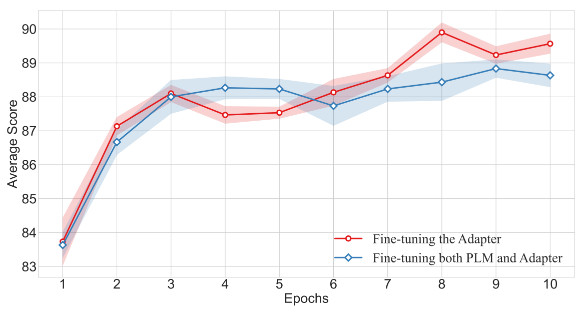

Figure 3: The curve of the average score on the GLUE benchmark of the proposed G-Tuning during training.

In this section, we compare the performance of fine-tuning the adapter (PEFT) and both PLM and adapter (FT) in the proposed G-Tuning. The curve of the average score on the GLUE is shown in Figure 3. In the first five epochs, in which we do not employ the loss from the WGAN, FT outperforms PEFT. However, when employing the proposed approach, FT becomes unstable and cannot obtain comparable performance to PEFT. Here, we think when the task-specific training objective trains the whole model, G-Tuning is challenging to adjust continuously. On the other hand, the difficulty of optimizing the adapter is much less than the whole model, which makes the training more stable and effective.

B The Overall Process of G-Tuning

The overall process of the proposed G-Tuning is shown in Algorithm 1. Compared to traditional fine-tuning or PEFT, G-Tuning requires extra training costs for the discriminator. Fortunately, due to the size of the labeled data B being usually small, our method will not affect the efficiency of the fine-tuning stage.

Algorithm 1 The overall process of G-Tuning.

```
Input: Unlabeled set $\boldsymbol{M}$; Labeled set $\boldsymbol{B}$; PLM with adapter $f(\cdot) = h \circ g(\cdot)$; Training step $N$; Discriminator $D(\cdot)$; Update Frequency $T$
Output: The fine-tuned model $f(\cdot)$
 1: Randomly initialize the parameter of $h(\cdot)$
 2: Initialize the parameter of $g(\cdot)$ from PLM
 3: for _ to $\frac{N}{2}$ do
 4:     Sample $\{\boldsymbol{x}^t, \boldsymbol{y}^t\} \sim \boldsymbol{B}$
 5:     Optimize $f(\cdot)$ by the Eq. 2
 6: end for
 7: for _ to $\frac{N}{2}$ do
 8:     Sample $\{\boldsymbol{x}^t\} \sim \boldsymbol{B}$, $\{\boldsymbol{x}^u\} \sim \boldsymbol{M}$
 9:     Optimize $D(\cdot)$ by the Eq. 3
10: end for
11: for _ to $\frac{N}{2}$ do
12:     Sample $\{\boldsymbol{x}^t, \boldsymbol{y}^t\} \sim \boldsymbol{B}$, $\{\boldsymbol{x}^u\} \sim \boldsymbol{M}$
13:     Optimize $f(\cdot)$ by the Eq. 6
14:     for _ to $T$ do
15:         Sample $\{\boldsymbol{x}^t\} \sim \boldsymbol{B}$, $\{\boldsymbol{x}^u\} \sim \boldsymbol{M}$
16:         Optimize $D(\cdot)$ by the Eq. 3
17:     end for
18: end for
19: return $f(\cdot)$
```

C Evaluation Metric and Data Statistic

We summarize the evaluation metric in the GLUE and XTREME benchmarks used in our experiments. The summary is shown in Table 4. Compared to the Wang et al. (2018) and Hu et al. (2020), our work has several differences. First, we do not evaluate the MNLI-mismatch set in the MNLI task. Then, we choose the Pearson Correlation to evaluate the STSB task and F1 score to evaluate the XQuAD, MLQA and TyDiQA tasks.

Then, following Yang et al. (2022), we select several data-sets as the out-of-domain test sets to evaluate the domain generalization ability. The data statistic is shown in Table 5. For the MNLI and STSB, we concat the training and test set from SICK as the test set. We randomly sample 6000 samples from HANS and IMDB as test sets for the RTE and SST2, respectively. All data-sets mentioned above can be downloaded on the Hugging Face[3]. Moreover, the CoLA$_{\text{ood}}$ is collected by the Yang et al. (2022). We randomly select 6000 samples from the original set as the test sets.

[3]https://huggingface.co/datasets

Task	Metric	Task	Metric
MNLI	Accuracy	XNLI	Accuracy
QNLI	Accuracy	PAWS-X	Accuracy
QQP	Accuracy	POS	F1 Score
RTE	Accuracy	NER	F1 Score
SST	Accuracy	BUCC	Accuracy
MRPC	Accuracy	Tatoeba	Accuracy
CoLA	Matthews Correlation	XQuAD	F1 Score
STSB	Pearson Correlation	MLQA	F1 Score
–	–	TyDiQA	F1 Score

Table 4: The evaluation metric used in our experiment in the GLUE and XTREME benchmarks.

Training Set		Test Set	
MNLI	39270	SICK	4906
RTE	2490	HANS	6000
SST2	67349	IMDB	6000
CoLA	8551	CoLA$_{\text{ood}}$	6000
STSB	5749	SICK	4906

Table 5: The data statistic of the GLUE$_{\text{ood}}$ data-set.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
The limitations of this work is shown in the last section.

☒ A2. Did you discuss any potential risks of your work?
Left blank.

☒ A3. Do the abstract and introduction summarize the paper's main claims?
Left blank.

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☐ Did you use or create scientific artifacts?

Not applicable. Left blank.

☐ B1. Did you cite the creators of artifacts you used?
Not applicable. Left blank.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Not applicable. Left blank.

C ☒ Did you run computational experiments?

Left blank.

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 3

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 3

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 3

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 3

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.