Parameter-Efficient Tuning on Layer Normalization for Pre-trained Language Models

Anonymous ACL submission

Abstract

Given the magnitude of the current Pre-trained Language Models (PLMs), conventional fine-tuning becomes increasingly challenging, therefore parameter-efficient tuning is now the focus of cutting-edge research. For PLMs to accomplish transferability, prior techniques in this field added tunable adapters into Multi-Head Attention (MHA) or/and Feed-Forward Network (FFN) of Transformer blocks. However, the ability of Layer Normalization (LayerNorm) for parameter-efficient tuning is disregarded while being a crucial component of Transformer architecture. In this paper, we first propose LN-tuning, which is time-efficient and performs better than BitFit with only half tunable parameters. Moreover, SOTA performance is achieved by the unified framework of combining prefix-tuning and LN-tuning. Lastly, LN-tuning is better understood by an ablation investigation and a visualization experiment of the bias and gain terms.

1 Introduction

Natural language processing (NLP) is presently dominated by the transfer learning from Pre-trained Language Models (PLMs) paradigm (Devlin et al., 2019; Han et al., 2021), which produces superior results in many tasks (Qiu et al., 2020; Peters et al., 2018; Devlin et al., 2019). The typical method used by PLMs to integrate the information they gained during the pre-training stage into downstream tasks is fine-tuning. A copy of the model needs to be retrained and saved for each downstream operation, which could be expensive given the enormous size of modern PLMs. To address the aforementioned issue, parameter-efficient tuning techniques have been proposed, which only modify a small subset of the pre-trained parameters and freeze the majority of them. To make measurable progress in this area, a lot of work has been done. Ziegler et al.; Houlsby et al.; Pfeiffer et al.; He et al. propose several adapter techniques that insert trainable bottleneck layers into the Feed-forward Network layer of each PLM block. Prefix-tuning (Li and Liang, 2021), P-tuning v2 (Qin and Eisner, 2021), and deep prompt tuning are used in MHA to optimize MLP networks and achieve continuous prefix prompt. More recently, research efforts have been made to create a unified framework that simultaneously tunes the representations of MHA and FFN, including those of the MAM adapter (He et al., 2021a) and UniPELT (Mao et al., 2022). By integrating adapter-based approaches that operate on both MHA and FFN, they are able to attain SOTA performance. It is clear from this that earlier approaches in this area included tunable adapters to the MHA or/and FFN of Transformer blocks to provide parameter-efficient tuning. Nevertheless, the power of LayerNorm for parameter-efficient tuning is overlooked while being a crucial component of Transformer-based PLMs. Following the normalization of mean and variance, the gain and bias terms are applied for affine transformation on each input neuron in LayerNorm, acting as a fine-grained adaptive module on the data (Ba et al., 2016). In earlier techniques, it is ignored and kept to be fixed in tuning. However, since LayerNorm enables smoother gradients, faster training and better generalization accuracy with a wide application in deep learning (Xu et al., 2019), we argue it may also help to achieve better data adaptation in parameter-efficient tuning. In this research, we provide a straightforward but efficient technique called LN-tuning with the learnable gain and bias term of LayerNorm. Following are some examples of our contribution:

- We propose LN-tuning, which first explores the potential of LayerNorm for parameter-efficient tuning, achieving comparable performance to prior approaches with a very small number of parameters and a high time efficiency.

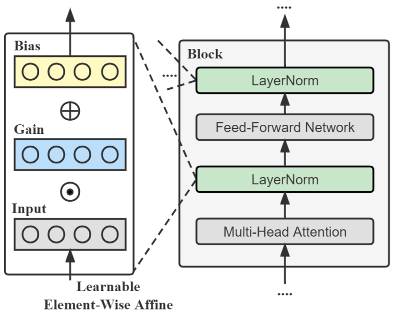


Figure 1: Illustration of our proposed LN-tuning.

- Prefix-tuning combined with LN-tuning leads to SOTA performance, outperforming MAM (*i.e.* the adapter-based unified framework that tunes MHA and FFN simultaneously) by less tunable parameters.
- LN-tuning is better understood thanks to the ablation study of terms, layers, and modules, as well as the visualization experiment of the gain and bias term.

2 Method

Layer normalization (LayerNorm) is a technique to normalize the distributions of intermediate layers. It enables smoother gradients, faster training, and better generalization accuracy (Xu et al., 2019). As Eq. 1 shows, LayerNorm involves two stages: (1) normalize $\boldsymbol{x}$ by mean and variance (2) forward by the scale and shift operations consisting of the gain term $\boldsymbol{g}$ and bias term $\boldsymbol{b}$, respectively.

Our proposed LN-Tuning keeps parameters in the gain term (for scale operation) and bias term (for shift operation) trainable, which are initialized from the pre-training stage, while fixing other parameters of PLMs. The scale and shift operation in LN-tuning is a unique, sped-up FFN that only conducts projection on a single neuron, as opposed to linear aggregation between input layer neurons.

$$\mathrm{LayerNorm}(\boldsymbol{x}) = \frac{\boldsymbol{g}}{\sigma} \odot (\boldsymbol{x} - \mu) + \boldsymbol{b} \qquad (1)$$

where

$$\mu = \frac{1}{H}\sum_{i=1}^{H} \boldsymbol{x}_i \qquad \sigma = \sqrt{\frac{1}{H}\sum_{i=1}^{H} (\boldsymbol{x}_i - \mu)^2}$$

3 Experiments

We validate the effectiveness of the proposed method on 11 benchmark datasets and seven types of downstream tasks, including both NLU and NLG ones, with the presence of six state-of-the-art baselines.

3.1 General Setup

Task Setup. To evaluate the proposed LN-tuning comprehensively, we conduct *cross-task*, *cross-PLM-architecture*, and *cross-PLM-scale* experiments. For cross-task validation, we conduct both NLU and NLG tasks.

Baseline Methods. We compare our methods with six state-of-the-art tuning methods including full-tuning, scaled parallel adapter-tuning (Pfeiffer et al., 2021; He et al., 2021a), prefix-tuning (Liu et al., 2022), LoRA (Hu et al., 2021), MAM adapter (He et al., 2021a), BitFit (Zaken et al., 2022) and 3V[1] (Yang et al., 2022). For brevity, we agree to use adapter, prefix, MAM to represent scaled parallel adapter-tuning, prefix-tuning, and MAM Adapter respectively in all tables of this paper.

More implementation details can be found in section A of Appendix.

3.2 Main Results

Method	#Para.	CN04	Twiiter	SICK	SNLI	SST-2	CB	CSQA	SocIQA	Avg.
				BERT-Large						
FT	100%	**85.2**	75.8	86.2	**85.4**	92.8	80.4	**69.8**	63.4	**79.9**
Adapter	0.33%	82.8	76.3	86.4	85.0	93.0	74.1	62.6	65.3	78.2
Prefix	0.33%	81.4	76.2	86.3	85.3	93.4	75.0	63.2	65.4	78.3
LoRA	0.33%	82.3	77.1	86.4	85.2	93.4	74.6	62.7	65.1	78.4
MAM	0.66%	83.0	**78.1**	**86.6**	85.2	93.1	77.6	63.2	**65.5**	79.0
3V	0.0006%	68.1	73.6	81.3	82.8	89.1	70.2	-	-	-
BitFit	0.07%	79.2	74.2	77.8	81.6	92.6	70.5	59.7	62.8	74.8
LN	0.03%	78.9	76.9	85.8	83.8	89.8	70.5	59.6	63.3	76.1
Prefix+LN	0.36%	84.2	77.2	**86.6**	**85.4**	**93.8**	**81.2**	64.0	**65.5**	79.8
				BERT-Base						
FT	100%	**87.2**	75.3	84.5	84.2	90.9	**82.7**	50.2	55.0	76.3
Adapter	0.28%	72.5	75.7	83.7	84.4	91.5	73.8	**60.6**	61.6	75.5
Prefix	0.28%	77.9	75.9	84.2	84.0	**91.9**	76.8	60.4	61.6	76.6
LoRA	0.33%	74.2	75.5	83.8	84.2	91.3	73.1	60.3	61.4	75.5
MAM	0.56%	80.3	76.3	84.8	84.5	91.6	73.8	60.4	**61.8**	76.7
3V	0.0014%	67.2	70.7	**85.0**	82.2	88.1	72.0	-	-	-
BitFit	0.08%	80.9	71.5	74.4	79.9	89.9	68.5	55.3	57.6	72.2
LN	0.04%	79.1	**76.7**	74.0	82.4	91.4	73.8	58.5	58.8	74.3
Prefix+LN	0.32%	80.7	76.1	84.5	**84.6**	**91.9**	74.1	**60.6**	61.7	**76.8**

Table 1: Results with $\mathrm{BERT}_{\mathrm{large}}$ and $\mathrm{BERT}_{\mathrm{base}}$. We report the average score with the standard deviation as the subscript. The **best** and 2nd best methods on each dataset are in bold and underlined, respectively.*3V can not be applied into these two QA tasks and thus is omitted to calculate the average values and rank metric.

In Table 1, we present the comparison results for the NLU tasks on $\mathrm{BERT}_{\mathrm{large}}$ and $\mathrm{BERT}_{\mathrm{base}}$. It is clear from this that full-tuning and MAM adapter may typically achieve superior performance. Better performance is expected because more recently introduced parameters and multiple PLM modules

[1]we name it 3V in our paper for clarity and brevity.

Method	Para.	E2E BLEU	E2E NIST	E2E MET	E2E R-L	E2E CIDEr	Samsum R-1	Samsum R-2	Samsum R-L	WebNLG BLEU	WebNLG MET	WebNLG TER↓	WebNLG Mover	WebNLG BERT	WebNLG BLEURT	Rank↓
FT	100%	65.07	**8.61**	43.42	67.90	2.38	**44.70**	**20.37**	**41.57**	**39.43**	**0.34**	0.55	**0.65**	**0.93**	**0.39**	**2.43**
Adapter	0.13%	64.93	8.46	**44.21**	68.63	**2.39**	43.23	18.67	40.17	38.40	0.33	0.56	0.64	**0.93**	0.38	3.79
Prefix	0.13%	**65.27**	8.55	43.70	68.27	2.37	43.70	19.97	40.83	38.87	0.33	**0.54**	**0.65**	**0.93**	0.38	3.50
LoRA	0.13%	64.91	8.47	43.36	68.60	2.36	43.38	18.65	40.13	38.51	0.33	0.55	**0.65**	**0.93**	0.38	4.93
MAM	0.26%	64.80	8.46	43.90	**68.67**	2.36	43.50	19.40	40.33	38.87	0.33	0.55	**0.65**	**0.93**	0.38	4.43
BitFit	0.09%	64.27	8.54	41.80	67.63	2.17	39.27	15.23	36.17	35.33	0.30	0.61	0.62	0.92	0.32	7.14
LN	0.03%	64.07	8.34	43.63	67.97	2.35	42.77	18.80	39.53	38.47	0.33	0.55	0.64	**0.93**	0.36	6.36
Prefix+LN	0.16%	65.24	8.57	43.75	68.43	**2.39**	43.88	20.03	41.07	39.16	**0.34**	**0.54**	**0.65**	**0.93**	0.38	3.43

Table 2: Results with GPT-2_{medium}. We report the average score with the standard deviation as the subscript. The **best** and 2nd best methods on each dataset are in bold and underlined respectively. 3V can not be applied into NLG tasks and thus is omitted as a baseline here.

need to be tuned. Compared to other earlier approaches, 3V and BitFit performs the poorest with less parameters. Under the tunable parameter alignment setting, the performance of prefix tuning and adapter tuning is comparable to one another.

The performance of LN-tuning is then examined. Comparing approaches whereas the ratio of the tunable parameters is more significant than 0.3%, LN-tuning is inferior to them by tuning only 0.03%–0.04% of parameters. By using almost half the tunable parameters of BitFit, LN-tuning performs much better than BitFit. LN-tuning outperforms 3V in terms of performance and is also applicable to a wider variety of NLP tasks than 3V, including QA tasks for NLU and NLG tasks.

The methods' overall performance in NLG tasks is similar to that in NLU tasks, With a few limited differences. First, prefix-tuning outperforms MAM adapter. Second, our LN-tuning exhibits a performance closer to that of adapter-based approaches, such as adapter and MAM adapter, compared to the NLU task.

3.3 Efficiency Analysis

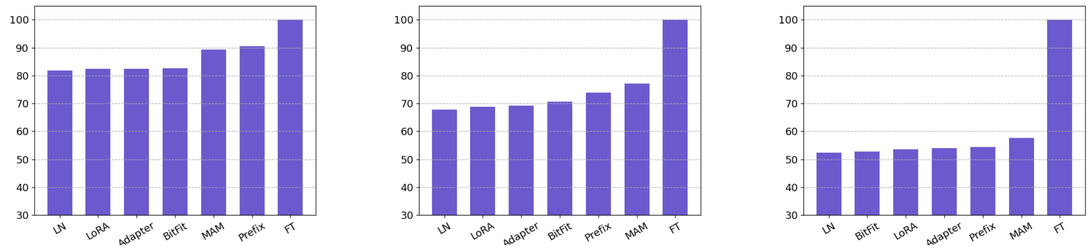
Setup. In order to compare the training and inference time efficiency between our method and earlier ones, we generate statistics from running logs. Then, in comparison to Full-Tuning (FT), we report the relative training and inference times. This includes the average time costs for three NLG datasets for GPT-2 and eight NLU datasets for BERT. The time cost of FT is normalized to 100.

Result. As shown in Fig. 2, our proposed LN-tuning takes the least time in all PLM architectures for training process. LN-tuning, along with BitFit and FT, costs the similar least time for inference process as expected. The above results on both training and inference show the significant superiority of our method in time efficiency comparing previous adapter-based methods. More details of analysis can be found in Section B of Appendix.

4 Visualization of Gain and Bias Term.

Setup. We visualize the change of the gain and bias term on each layer of PLMs to give a further understanding about LN-tuning. Specifically, following BitFit, we use $\frac{1}{\dim(\boldsymbol{t})}\|\boldsymbol{t}_o - \boldsymbol{t}_f\|_1$ to measure the amount of change for terms, where $\boldsymbol{t}$ represents the gain term $\boldsymbol{g}$ or the bias term $\boldsymbol{b}$ of LayerNorm, which means the average absolute change, across its dimensions, between the initial LM values $\boldsymbol{t}_o$ and its fine-tuned values $\boldsymbol{t}_f$. We choose five datasets which covers all type of NLU tasks in Sec. 3.1 and use $\text{BERT}_{\text{large}}$ for the experiment.

Result. As shown in Fig. 3, there can be oberseved that the terms of layers close to the output, i.e. layer 15 to 24, changes more than those close to input, whether the gain or bias. Meanwhile, in those layers close to output, the gain term change more than bias term (This doesn't mean that the gain term is more important than the bias term in LN-tuning). Comparing results between tasks, the task complexity and the dataset scale may affect the extent of terms' change. Firstly, comparing SST-2 and the other two datasets of binary classification tasks, there is a greater change in terms of LN-tuning. This may be because that there are larger solution spaces (greater task complexity) for the QA (CSQA) and NER (Twitter) task than binary classification tasks such as sentiment analysis (SST-2), Paraphrase Identification (SICK) or Natural Language Inference (CB) task. There needs greater variation in the terms of LN-tuning in CSQA and Twitter dataset. Secondly, the order of term variation in binary classification tasks is SST-2 > SICK > CB, which is the same as the order of their data scale: SST-2 (67,349 items) > SICK (4,439 items) > CB (250 items). A reasonable explanation for this different degree of variation is



(a) BERT-base Training (b) BERT-large Training (c) GPT-2 Medium Training

Figure 2: Time Efficiency Comparison of Training.

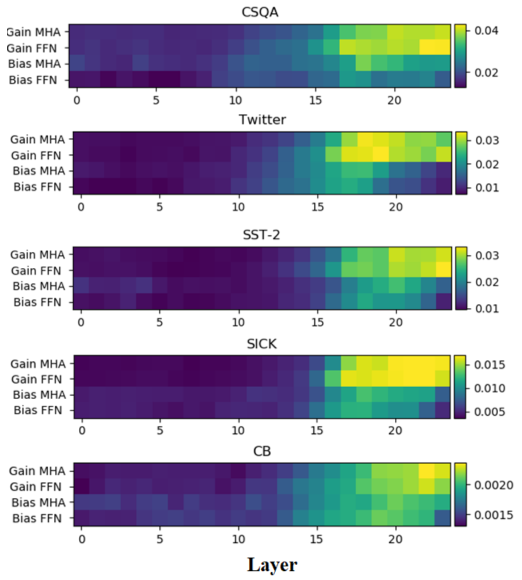


Figure 3: Change in gain and bias term on five type of NLU tasks. 'Gain MHA' means the gain term of LayerNorm module after MHA in each layer of PLMs, and so forth for other labels of Y-axis.

that larger data sizes require a more significant term variation to accommodate a variety of data samples from a wider range of domains.

5 Ablation Study

Setup. To explore whether LN-tuning may be enhanced to be more parameter-efficient, we undertake an ablation study from three aspects: terms, modules, and layers. Specifically, for terms, we only keep one option of the gain or the bias term trainable. For layers, we keep vectors of LayerNorm of only the half layers close to input or output trainable, i.e. from layer 1 to 12 or from 13 to 24, if using $\text{BERT}_{\text{large}}$. The same way is for using $\text{BERT}_{\text{base}}$. For modules, since there are two LayerNorm modules in each block of Transformer, where one is after MHA and the other is after FFN, we keep vectors trainable of only one module in each Transformer block. We use both $\text{BERT}_{\text{large}}$ and $\text{BERT}_{\text{base}}$ for the experiment in this section.

Result. As shown in Table 5 of appendix, comparing to full LN-tuning method, all ablated techniques obtain a performance drop, which validates no extraneous components for LN-tuning. Further, the influence of layers seems more critical than that of modules due to a larger performance decrease comparing ablated layer methods and ablated module methods. For term ablation type, the method with only the bias term performs better than that with only the gain term, whether in $\text{BERT}_{\text{large}}$ or $\text{BERT}_{\text{base}}$, which indicates that the bias term plays a more critical role than the gain term in LN-tuning. The added MHA learnable module looks more relevant for module ablation type than the added FFN learnable module. For layer ablation type, the layers adjacent to input seems to be more importan than that close to output in $\text{BERT}_{\text{base}}$, however, the outcome is the opposite for $\text{BERT}_{\text{large}}$. This shows that the importance of layers is quite different in different size of PLMs in LN-tuning and those layers close to output can play a more significant role in larger size of PLMs.

6 Conclusions

In this paper, we first propose *LN-Tuning*, which only tunes the bias and gain term of LayerNorm to enable parameter-efficient transferring for PLMs. Later, we investigate a unified framework for merging LN-tuning with earlier parameter-efficient techniques and discover that SOTA performance can be achieved by combining prefix-tuning with LN-tuning. Finally, the ablation study of terms, layers, and modules, as well as the visualization experiment of the gain and bias term further understand LN-tuning.

Limitation

While prefix-tuning and LN-tuning operate together to attain SOTA performance and LN-tuning has a high time efficiency with very few tunable parameters, there are still worthwhile areas for additional research. First, take note that the LN-tuning approach for tuning gain and bias term is a novel tuning technique that can be used after any PLM output vector. Exist any undiscovered techniques to perform SOTA by only learnable modules in LN-tuning? Further investigation can be done in future work to determine why the unified framework of integrating LN-tuning and Prefix-tuning (MHA+LN) can perform better than earlier adapter-based techniques (MHA+FFN).

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Anja Belz and Ehud Reiter. 2006. Comparing automatic and human evaluation of nlg systems. In *11th conference of the european chapter of the association for computational linguistics*, pages 313–320.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Xavier Carreras and Lluís Màrquez. 2004. Introduction to the conll-2004 shared task: Semantic role labeling. In *CoNLL 2004*, pages 89–97.

Leon Derczynski, Kalina Bontcheva, and Ian Roberts. 2016. Broad twitter corpus: A diverse named entity recognition resource. In *COLING 2016*, pages 1169–1179.

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

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *EMNLP-IJCNLP 2019*, page 70.

Wenjuan Han, Bo Pang, and Ying Nian Wu. 2021. Robust transfer learning with pretrained language models through adapters. In *ACL/IJCNLP 2021*, pages 854–861.

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

Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jia-Wei Low, Lidong Bing, and Luo Si. 2021b. On the effectiveness of adapter-based tuning for pretrained language model adaptation. In *ACL/IJCNLP 2021*, pages 2208–2222.

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 2019*, pages 2790–2799.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685.

Alon Lavie and Abhaya Agarwal. 2007. METEOR: an automatic metric for MT evaluation with high levels of correlation with human judgments. In *WMT@ACL 2007*, pages 228–231.

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

Chin-Yew Lin. 2004. Rouge: a package for automatic evaluation of summaries. In *Workshop on Text Summarization Branches Out, Post-Conference Workshop of ACL 2004, Barcelona, Spain*, pages 74–81.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *ACL 2022*, pages 61–68.

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. 2022. Unipelt: A unified framework for parameter-efficient language model tuning. In *ACL 2022*, pages 6253–6264.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223.

Linyong Nan, Dragomir R. Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Mutethia Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. 2021. DART: open-domain structured data record to text generation. In *NAACL-HLT 2021*, pages 432–447.

Jekaterina Novikova, Ondrej Dusek, 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, Saarbrücken, Germany, August 15-17, 2017*, pages 201–206.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT 2018*, pages 2227–2237.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. Adapterfusion: Non-destructive task composition for transfer learning. In *EACL 2021*, pages 487–503.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulic, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. Adapterhub: A framework for adapting transformers. In *EMNLP 2020*, pages 46–54.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying lms with mixtures of soft prompts. In *NAACL-HLT 2021*.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. In *EMNLP-IJCNLP 2019*, pages 4463–4473.

Mathew Snover, Bonnie Dorr, Richard Schwartz, John Makhoul, Linnea Micciulla, and Ralph Weischedel. 2005. A study of translation error rate with targeted human annotation. Technical report, Technical Report LAMP-TR-126, CS-TR-4755, UMIACS-TR-2005-58, University of ....

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *NAACL 2019*, pages 4149–4158.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *CVPR, 2015*, pages 4566–4575.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS 2019*, pages 3261–3275.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR 2019*.

Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. 2019. Understanding and improving layer normalization. In *NeurIPS 2019*, pages 4383–4393.

Haoran Yang, Piji Li, and Wai Lam. 2022. Parameter-efficient tuning by manipulating hidden states of pretrained language models for classification tasks. *CoRR*, abs/2204.04596.

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

Zachary M. Ziegler, Luke Melas-Kyriazi, Sebastian Gehrmann, and Alexander M. Rush. 2019. Encoder-agnostic adaptation for conditional language generation. *CoRR*, abs/1908.06938.

A Implementation Details for Experiments

Specifically, for NLU tasks, we choose seven type datasets: **(1) Named-Entity Recognization (NER)**, including CoNLL2004 (Carreras and Màrquez, 2004) and Twitter (Derczynski et al., 2016); **(2) Natural Language Inference (NLI)**, including SNLI (Bowman et al., 2015) and CB (Wang et al., 2019a); **(3) Paraphrase Identification (PI)**, including SICK (Marelli et al., 2014); **(4) Sentiment Analysis (SA)**, including SST-2 (Wang et al., 2019b); **(5) Question Answering (QA)**, including CSQA (Talmor et al., 2019) and SocIQA (Sap et al., 2019); **(6) Table-to-Text Generation**, including E2E (Novikova et al., 2017) and DART (Nan et al., 2021); **(7) Dialogue Summarization**, including Samsum (Gliwa et al., 2019) .

The cross-PLM-architecture validation requires approaches to be verified on both encoder-only (BERT (Devlin et al., 2019)) and decoder-only (GPT-2 (Radford et al., 2019)) Transformer architecture. The cross-PLM-scale validation requires approaches to be verified on PLMs of different scales. Specifically, the same experiments for NLU are conducted on both $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$, while $\text{GPT-2}_{\text{medium}}$ is for NLG.

We conduct experiments on two NVIDIA GeForce RTX 3090 GPUs. The results are evaluated by different measures as suggested by different tasks. To reduce the interference of randomness, we repeat the experiments for three times and the average scores (for NLU) or the rank (for NLG) is returned as results. According to the recorded experience (Houlsby et al., 2019; Pfeiffer et al., 2020; Li and Liang, 2021; He et al., 2021a), the common hyper-parameters are adjusted according to the statistical characteristics of datasets.

For NLU tasks, we set the training epoch 30, with an early stopping strategy of 10 non-decrease validation loss. The batch size setting can be found in Table 3. For LN-tuning, we adjust the learning rate from the priority order in {1e-2, 1e-3, 2e-4} [2]. We adjust the learning rate from the priority order in {1e-3, 2e-4} for other methods.

For NLG tasks, we set the training epoch 20. The batch size setting can be found in Table 4, and the learning rate is 2e-4 for all methods. The E2E dataset contains about 50K examples whose average output length is 22.9. We use the official evaluation script [3] to calculate BLEU (Papineni et al., 2002), NIST (Belz and Reiter, 2006), METEOR (Lavie and Agarwal, 2007), ROUGE-L (Lin, 2004), and CIDEr (Vedantam et al., 2015). The Samsum dataset contains about 15K examples, whose average output length is 23.7. We use the standard python package `rouge` to calculate ROUGE-1, ROUGE-2, ROUGE-L (Lin, 2004). The DART dataset consists of 82K examples, whose average output length is 27.3. We use the official evaluation script [4] to calculate BLEU, METEOR, and TER (Snover et al., 2005). We use $\text{GPT-2}_{\text{medium}}$ (Radford et al., 2019) as the experimental PLM, where the max generation length is set to [35, 35, 45] for [E2E, Samsum, DART], respectively.

We align the tunable amount of additional parameters of different methods to ensure a fair comparison, which is accomplished by setting hyperparameters. Specifically, for prefix-tuning, the hyperparameter to be adjusted is its prefix length l, where we set $l = 16$ for $\text{BERT}_{\text{base}}$, $l = 24$ for $\text{BERT}_{\text{large}}$, and $l = 16$ for $\text{GPT-2}_{\text{medium}}$. For adapter, we adjust the intermediate dimension d_b, where we set $d_b = 16$ for $\text{BERT}_{\text{base}}$, $d_b = 24$ for $\text{BERT}_{\text{large}}$, $d_b = 16$ for $\text{GPT-2}_{\text{medium}}$. For MAM adapter, we adjust the both, keeping $d_b = l = 8$ for $\text{BERT}_{\text{base}}$, $d_b = 16, l = 8$ for $\text{BERT}_{\text{large}}$, and $d_b = 8, l = 8$ for $\text{GPT-2}_{\text{medium}}$.

Methods	CN04	Twitter	SICK	SNLI	SST-2	CB	CSQA	SociQA
				BERT-Base				
FT	128	128	512	512	256	48	48	48
MAM	128	128	512	512	392	64	64	48
Others	128	128	512	512	392	64	64	64
				BERT-Large				
FT	32	32	256	256	128	24	16	12
MAM	48	48	256	256	256	32	24	24
Others	48	48	256	256	256	32	32	24

Table 3: Batch size setting for NLU tasks.

Method	Samsum	E2E	WebNLG
FT	32	48	40
Others	36	96	84

Table 4: Batch size setting for NLG tasks.

The detailed batch size settings for NLU and NLG tasks are displayed in the Table 3 and able 4 respectively. In order to conduct a fair comparison, we

[2]We empirically find that LN-tuning needs larger learning rate than other approaches in some datasets.

[3]https://github.com/tuetschek/e2e-metrics

[4]https://github.com/Yale-LILY/dart

make full use of the GPUs' VRAM capacity and work to make sure the batch size parameters for each approach are identical. We decrease the value of batch size to prevent a "CUDA Out Of Memory" problem because full-tuning and MAM Adapter have more tunable parameters.

B Details of Efficiency Analysis

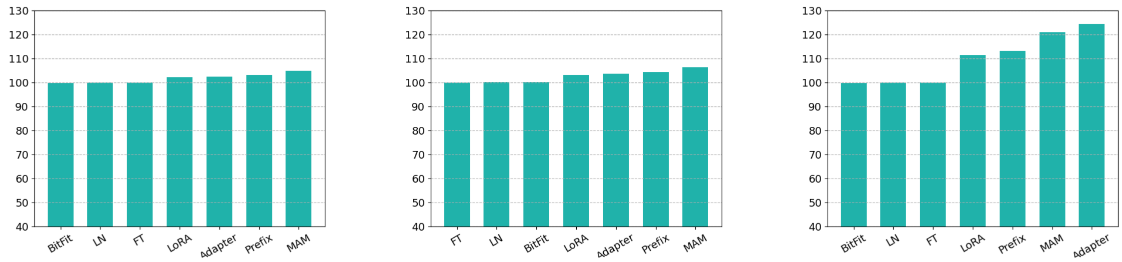
In Fig.2(a), all parameter-efficient methods require training times that are less than 90% of those of FT in $\text{BERT}_{\text{base}}$ and less than 80% of those of FT in $\text{BERT}_{\text{large}}$, demonstrating that parameter-efficient methods can train PLMs of greater scales more quickly. From Fig. 2(a), Fig. 2(b) and Fig. 2(c), we can observe that parameter-efficient methods show higer time efficiency in training in NLG tasks than in NLU tasks comparing with FT. However, whether in training or inference, MAM adapter typically has the lowest time efficiency, demonstrating that the unified methods of both tuning MHA and FFN require a significant investment in computational resources despite being able to produce better performance. Further, adapter-tuning shows higher time efficiency than prefix-tuning in training and inference, except for the NLG inference process.

C Details of Ablation Study

Ablation Type	Method	CN04	Twitter	SICK	SNLI	SST-2	CB	CSQA	SociQA	Avg
					BERT-Large					
-	*Full**	*80.2*	*77.2*	*84.9*	*84.0*	*91.9*	*74.1*	*60.5*	*63.2*	*77.0*
Term	Only Gain	69.5	69.5	76.3	80.9	91.6	71.4	53.3	57.9	71.3
	Only Bias	79.8	72.6	77.0	81.2	91.8	73.2	55.8	60.9	74.0
Module	Only FFN	77.3	76.5	82.2	81.9	92.6	72.8	55.6	61.0	**75.0**
	Only MHA	75.8	77.4	82.0	81.6	92.2	72.3	56.2	58.8	<u>74.6</u>
Layer	Only Layer 1-12	73.2	75.1	82.4	78.4	91.8	73.1	51.7	56.4	72.8
	Only Layer 13-24	73.8	75.7	82.4	78.6	93.2	72.9	53.8	56.6	73.4
					BERT-Base					
-	*Full**	*79.8*	*76.4*	*81.0*	*83.3*	*91.4*	*70.2*	*57.9*	*59.1*	*74.9*
Term	Only Gain	72.9	68.8	67.5	76.7	87.7	73.2	50.0	52.9	68.7
	Only Bias	76.5	67.8	77.5	76.3	89.7	71.4	51.1	53.4	70.5
Module	Only FFN	79.1	76.6	81.5	77.0	91.6	76.2	53.3	53.8	**73.6**
	Only MHA	78.4	76.5	81.8	77.2	91.2	75.0	52.6	54.0	<u>73.3</u>
Layer	Only Layer 1-6	78.2	76.0	67.9	74.1	90.7	74.4	50.8	50.6	70.3
	Only Layer 7-12	71.3	74.9	68.2	73.9	90.8	73.8	50.3	50.3	69.2

Table 5: Results of ablation study about terms, layers and modules with $\text{BERT}_{\text{large}}$ and $\text{BERT}_{\text{base}}$. *We use italic font to show results of the full LN-tuning, which is as a standard for comparison.

.



(a) BERT-base Inference

(b) BERT-large Inference

(c) GPT-2 Medium Inference

Figure 4: Time Efficiency Comparison of Inference.