Towards Adaptive Prefix Tuning
for Parameter-Efficient Language Model Fine-tuning

Zhen-Ru Zhang, Chuanqi Tan, Haiyang Xu, Chengyu Wang,
Jun Huang, Songfang Huang
Alibaba Group
{zhangzhenru.zzr,chuanqi.tcq,shuofeng.xhy}@alibaba-inc.com
{chengyu.wcy,huangjun.hj,songfang.hsf}@alibaba-inc.com

Abstract

Fine-tuning large pre-trained language models on various downstream tasks with whole parameters is prohibitively expensive. Hence, Parameter-efficient fine-tuning has attracted attention that only optimizes a few task-specific parameters with the frozen pre-trained model. In this work, we focus on prefix tuning, which only optimizes continuous prefix vectors (i.e. pseudo tokens) inserted into Transformer layers. Based on the observation that the learned syntax and semantics representation varies a lot at different layers, we argue that the adaptive prefix will be further tailored to each layer than the fixed one, enabling the fine-tuning more effective and efficient. Thus, we propose Adaptive Prefix Tuning (APT) to adjust the prefix in terms of both fine-grained token level and coarse-grained layer level with a gate mechanism. Experiments on the SuperGLUE and NER datasets show the effectiveness of APT. In addition, taking the gate as a probing, we validate the efficiency and effectiveness of the variable prefix.

1 Introduction

Vanilla fine-tuning strategy usually adjusts all the parameters to adapt the pre-trained language model to downstream tasks. Parameter-efficient learning (He et al., 2022; Houlsby et al., 2019; Lester et al., 2021; Guo et al., 2021; Ben Zaken et al., 2022) is an emerging framework that freezes the pre-trained model and only tunes a few number of task-specific parameters for downstream tasks. For instance, Prefix tuning (Li and Liang, 2021; Liu et al., 2022) prepends length-equivalent pseudo prefix tokens, i.e. continuous task-specific vectors to each layer of the pre-trained model, achieving comparable even superior performance with only 0.1-3% parameters.

In previous works, the length of prefix tokens (or the number of trainable parameters) is usually the same at each layer. However, a potential observation lies in that the structure information and
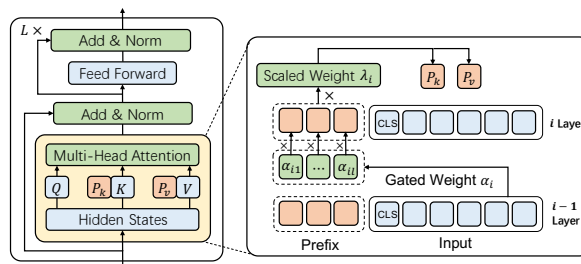

Figure 1: An illustration of the proposed approach APT where the left is the internal structure of Transformer with inserted prefixes, and the right is the schematic of prefix gate mechanism.

representational capacity embedded in each layer are prone to be inconsistent (Jawahar et al., 2019). It is generally considered that the bottom layers of the language model tend to capture concrete and shallow phrase-level features, while the top layers concerns more with abstract semantic information (Tenney et al., 2019). Based on the perspective, we assume adaptive prefix can grab the emphasis more flexibly to adapt to various downstream tasks.

In light of above motivation, we investigate the adaptive prefix in this work. We propose **A**daptive **P**refix **T**uning (APT) with an adaptive gate mechanism at both fine-grained token level and coarse-grained layer level. Specifically, as shown in Figure 1, for fine granularity, APT scores each individual prefix token via gated weight assignment. Then, the scaled weight is utilized to balance the inserted task-specific prefix tokens and original input tokens for current layer at coarse-grained level.

Extensive experiments against prefix tuning on the sentence and token classification tasks in full data and low resources setting validate the effectiveness of APT. In addition, the gate learned from APT could be served as a probing for the number of necessary parameters in different layers, guiding us to directly apply variable prefix to the original prefix tuning. The probing experiment further demonstrates the effectiveness of adaptive prefix.

2 Related Works

Since fine-tuning the whole model is prohibitively expensive, parameter-efficient language model fine-tuning becomes a lightweight alternative that only optimizes a small number of parameters while keeping most pre-trained parameters frozen (He et al., 2022). Adapter tuning (Houlsby et al., 2019) inserts two tunable task-specific modules after multi-head attention and feed-forward network, achieving comparable performance with only 2-4% of the parameters. Prompt tuning (Lester et al., 2021) and Prefix-Tuning (Li and Liang, 2021) only train soft prompts by adding prefix tokens to the input or hidden states. Recently, Liu et al. (2022) extend the prefix tuning to the natural language understanding tasks, which matches the performance of fine-tuning with only 0.1%-3% tuned parameters.

Furthermore, with an overlap of our motivations that each layer of the pre-trained language model focuses on different aspects of feature for various tasks (Jawahar et al., 2019; Clark et al., 2019b) and extra parameters are probably not necessary for certain tasks (Houlsby et al., 2019; Fan et al., 2020; Rücklé et al., 2021), Adaptable Adapters (Moosavi et al., 2022) selects beneficial adapter layers and learns task-specific activation function for downstream tasks to make adaptor dynamic for each task and layer. In addition to different frameworks (adapter versa prefix tuning), our key difference from their work lies in that we aim to dynamically filter required information at each layer in a soft way, while they choose whether to add trainable modules at the layer level in a hard manner.

3 Methodology

3.1 Prefix Tuning

As prefix tuning is an extension on Transformer (Vaswani et al., 2017), we first recap the structure of Transformer. Transformer is the block consisting of multi-head attention concatenated by multiple single self-attention functions and a fully connected feed-forward network. Formally speaking, the Transformer block is calculated as follows:

$$\text{Attn}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}}\boldsymbol{V}) \quad (1)$$

$$\text{FFN}(\boldsymbol{x}) = \text{ReLU}(\boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2 \quad (2)$$

Prefix tuning prepends pseudo prefix tokens of length l to each layer of the language model, which is implemented by concatenating inserted keys and values matrix with original corresponding items in each multi-head attention. Specifically, let $\boldsymbol{P}_k, \boldsymbol{P}_v \in \mathbb{R}^{l \times d}$ be the keys and values of the engaged prefix separately, where l denotes the length of prefix and d corresponds to the dimension, thus self-attention function can be reformatted as:

$$\text{Attn}(\boldsymbol{Q}, \boldsymbol{K}', \boldsymbol{V}') = \text{softmax}(\frac{\boldsymbol{Q}(\boldsymbol{K}')^T}{\sqrt{d}}\boldsymbol{V}') \quad (3)$$

$$\text{where} \quad \boldsymbol{K}' = [\boldsymbol{P}_k; \boldsymbol{K}], \boldsymbol{V}' = [\boldsymbol{P}_v; \boldsymbol{V}]$$

Here, $[;]$ donates concatenation function.

3.2 Adaptive Prefix Tuning

The length of prefix is usually a manually set hyperparameter for each task and fixed in distinct layers of the model. However, existing work demonstrates each layer of the language model pays attention to different aspects of the input feature. We assume the prefix in fixed length is insufficient to tailor different layers and tasks. To dynamically customize the prefix at each layer, APT performs a gate mechanism via fine-grained gated weight assignment and coarse-grained scaled weight specification.

Specifically, to capture the diversity of information utilization at different layers, we go deep into the token level at the fine-grained granularity. The token-level gate can inspire us on how many trainable parameters (i.e. pseudo tokens in prefix tuning) are required for this layer, which will be discussed in Section 4.4. Thus, APT yields the gated weights of l pseudo tokens at each layer. We use the hidden states to represent the information encoded in the layer and calculate the gated weights $\boldsymbol{\alpha}_i = [\alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{il}]$ for i-th layer as:

$$\boldsymbol{\alpha}_i = \text{sigmoid}(\boldsymbol{h}_{i-1}\boldsymbol{W}_i) \quad (4)$$

Here, $\boldsymbol{h}_{i-1}$ is the d-dimensional hidden states from the previous layer, and $\boldsymbol{W}_i \in \mathbb{R}^{d \times l}$ corresponds to the parameters to be learned.

Besides, we also design a coarse-level gate to balance the information brought from task-specific prefix tokens and original input tokens by learning a layer-level weight. A learnable scaled weight λ_i is added to the representation of pseudo prefix tokens at the i-th layer.

With the above strategy, the keys-values pair $\boldsymbol{P}_i = [\boldsymbol{P}_{ik}, \boldsymbol{P}_{iv}]$ derived from pseudo prefix tokens in i-th layer is updated to $\hat{\boldsymbol{P}}_i$ as:

$$\hat{\boldsymbol{P}}_i = \lambda_i \boldsymbol{\alpha}_i \odot [\boldsymbol{P}_{ik}, \boldsymbol{P}_{iv}] \quad (5)$$

Model		SuperGLUE						NER			
		BoolQ	COPA	RTE	WiC	WSC	Avg.	CoNLL03	CoNLL04	OntoNotes	Avg.
BERT-base (110M)	FT	**72.9**	67.0	68.4	<u>71.1</u>	63.5	68.6	-	-	-	-
	PT-2	72.5	<u>67.4</u>	<u>71.3</u>	69.5	<u>65.4</u>	<u>69.2</u>	89.3	82.6	87.1	86.3
	APT	<u>72.6</u>	**70.0**	**72.7**	**71.2**	**66.9**	**70.7**	**89.7**	**84.1**	**87.2**	**87.0**
BERT-large (335M)	FT	**77.7**	69.0	70.4	<u>74.9</u>	68.3	72.1	**92.8**	<u>85.6</u>	**89.2**	**89.2**
	PT-2	75.8	<u>73.0</u>	<u>78.3</u>	**75.1**	68.3	<u>74.1</u>	90.2	84.5	86.4	87.0
	APT	<u>76.0</u>	**79.0**	**79.4**	**75.1**	**70.2**	**75.9**	<u>90.7</u>	**85.8**	<u>88.6</u>	<u>88.4</u>
RoBRETa-large (355M)	FT	**86.9**	**94.0**	86.6	**75.6**	63.5	<u>81.3</u>	92.6	<u>88.8</u>	**89.8**	<u>90.4</u>
	PT-2	<u>84.8</u>	<u>93.0</u>	<u>89.5</u>	73.4	63.5	80.8	**92.8**	88.4	**89.8**	90.3
	APT	<u>84.8</u>	**94.0**	**89.9**	<u>74.6</u>	68.3	**82.3**	<u>92.7</u>	**89.0**	**89.8**	**90.5**
DeBERTa-xlarge (750M)	FT	-	-	-	-	-	-	**93.1**	**89.1**	<u>90.4</u>	**90.9**
	PT-2	-	-	-	-	-	-	**93.1**	86.5	90.4	90.0
	APT	-	-	-	-	-	-	<u>93.0</u>	**89.1**	**90.5**	<u>90.8</u>

Table 1: The results on SuperGLUE development set and NER test set in full data setting. The metric of SuperGLUE is accuracy and other is micro-f1 score. Results for FT and PT-2 on BERT-large, RoBRETa-large and DeBERTa-xlarge are token from (Liu et al., 2022). Results for FT on BERT-base are from (Liu et al., 2021). (FT: vanilla fine-tuning; PT-2: P-Tuning v2; APT: Adaptive Prefix Tuning; **bold**: the best score; <u>underline</u>: the second best)

Setting	Method	BoolQ	COPA	RTE	WiC	WSC	Avg.
BERT-base (16-shot)	FT	$47.2_{7.5}$	$54.0_{6.5}$	$49.4_{2.7}$	$50.3_{2.3}$	$46.2_{6.8}$	49.4
	PT-2	$52.4_{7.2}$	$54.2_{3.3}$	$50.8_{3.1}$	$48.2_{3.3}$	$48.5_{4.3}$	50.8
	APT	$\mathbf{55.7}_{6.5}$	$\mathbf{57.4}_{2.7}$	$\mathbf{53.1}_{4.4}$	$\mathbf{53.7}_{2.2}$	$\mathbf{55.2}_{3.8}$	**55.0**
BERT-large (16-shot)	FT	$\mathbf{57.3}_{9.7}$	$52.0_{2.4}$	$49.5_{2.7}$	$50.0_{0.0}$	$38.7_{2.2}$	49.5
	PT-2	$50.3_{5.7}$	$58.2_{5.3}$	$49.9_{3.4}$	$49.3_{2.2}$	$48.1_{4.2}$	51.2
	APT	$51.7_{3.5}$	$\mathbf{60.0}_{6.3}$	$\mathbf{53.9}_{4.6}$	$\mathbf{51.8}_{4.8}$	$\mathbf{55.4}_{2.3}$	**54.6**
BERT-base (32-shot)	FT	$48.1_{9.4}$	$52.2_{6.4}$	$49.5_{2.7}$	$49.4_{0.9}$	$\mathbf{60.4}_{3.8}$	51.9
	PT-2	$50.1_{5.5}$	$55.0_{3.2}$	$53.8_{3.4}$	$52.0_{4.1}$	$51.5_{4.6}$	52.5
	APT	$\mathbf{53.5}_{5.3}$	$\mathbf{57.6}_{2.2}$	$\mathbf{56.5}_{1.6}$	$\mathbf{54.8}_{3.9}$	$54.6_{6.5}$	**55.4**
BERT-large (32-shot)	FT	$47.6_{11.9}$	$45.0_{3.6}$	$48.4_{2.2}$	$50.0_{0.0}$	$47.3_{13.2}$	47.6
	PT-2	$45.5_{5.1}$	$57.4_{6.9}$	$51.3_{2.3}$	$53.3_{2.1}$	$46.0_{7.1}$	50.7
	APT	$\mathbf{49.9}_{5.9}$	$\mathbf{62.0}_{5.0}$	$\mathbf{55.5}_{3.6}$	$\mathbf{54.9}_{2.8}$	$49.0_{4.4}$	**54.3**

Table 2: The mean$_{std}$ experimental results within 5 random seeds on SuperGLUE development set in 16-shot and 32-shot setting where all metrics are accuracy. **bold**: the best score.

$\odot$ is the element-wise multiplication. Accordingly, the calculation of the self-attention function in APT is similar to Eq.(3) without further elaboration.

4 Experiments

4.1 Experimental Setup

We conduct 5 NLU tasks on SuperGLUE (Wang et al., 2019) benchmark including BoolQ (Clark et al., 2019a), COPA (Roemmele et al., 2011), RTE (Wang et al., 2018), WiC (Pilehvar and Camacho-Collados, 2019) and WSC (Levesque et al., 2012) as well as 3 Named Entity Recognition (NER) tasks including CoNLL03 (Tjong Kim Sang and De Meulder, 2003), CoNLL04 (Carreras and Màrquez, 2004), and OntoNotes 5.0 (Weischedel et al., 2013). With BERT-base / large (Devlin et al., 2019) and RoBERTa-large (Liu et al., 2019) instantiated by HuggingFace Transformers (Wolf et al.,

2020), we compare APT with vanilla fine-tuning and P-Tuning v2 (Liu et al., 2022) which is an implementation of the prefix tuning, configured with hyper-parameters public in the released code[1]. We also verify our method with DeBERTa-xlarge (He et al., 2020) on NER tasks following P-Tuning v2.

4.2 Results

We report the main results in Table 1. For BERT-base, we can observe that APT achieves 1.5% and 0.7% improvements over P-Tuning v2 on Super-GLUE and NER tasks, respectively. For BERT-large, APT outperforms P-Tuning v2 by 1.8% on SuperGLUE tasks and 1.4% on NER tasks. For RoBERTa-large, APT surpasses P-Tuning v2 by 1.5% on SuperGLUE tasks and 0.2% on NER tasks. On NER tasks with DeBERTa-xlarge, APT is supe-

[1] https://github.com/THUDM/P-tuning-v2

Setting	SuperGLUE						NER			
	BoolQ	COPA	RTE	WiC	WSC	Avg.	CoNLL03	CoNLL04	OntoNotes	Avg.
APT	**72.6**	**70.0**	**72.7**	**71.2**	**66.9**	**70.7**	**89.7**	**84.1**	**87.2**	**87.0**
w/o token-level α	72.6	69.0	69.9	70.8	65.8	69.6	89.5	83.7	87.2	86.8
w/o layer-level λ	72.1	67.4	71.3	69.6	65.4	69.1	89.0	82.6	86.9	86.2
w/o hidden states h	72.0	68.8	68.7	70.2	64.6	68.9	89.1	83.6	87.1	86.6

Table 3: Ablation study on BERT-base for two different level gate mechanisms and the hidden states from the previous layer. **bold**: the best score.

Model	SuperGLUE						NER			
	BoolQ	COPA	RTE	WiC	WSC	Avg.	CoNLL03	CoNLL04	OntoNotes	Avg.
PT-2	72.5	67.4	71.3	69.5	65.4	69.2	89.3	82.6	87.1	86.3
PT-2*	**72.6**	**68.8**	**71.9**	**70.0**	**65.8**	**69.8**	89.3	**83.0**	**87.2**	**86.5**
PT-2+	**72.8**	65.4	69.1	71.1	65.8	68.8	89.4	83.2	87.1	86.6
APT	72.6	**70.0**	**72.7**	**71.2**	**66.9**	**70.7**	**89.7**	**84.1**	**87.2**	**87.0**

Table 4: Comparison between PT-2 and PT-2*, PT-2$^+$ and APT on BERT-base. (PT-2: P-Tuning v2; PT-2*: PT-2 with variable prefix; PT-2$^+$: PT-2 with enlarged prefix)

rior to P-Tuning v2 by an average of 0.8%. Compared with vanilla fine-tuning, APT is comparable or even better on part of tasks. In addition, we explore the experimental performance under low resource settings on SuperGLUE benchmark. As shown in Table 2, APT is a better few-shot learner than P-Tuning v2, which exceeds 4.2%, 3.4% in 16-shot setting, and 2.9%, 3.6% in 32-shot setting for BERT-base and BERT-large, respectively.

4.3 Ablation Study

We conduct an ablation study in order to explore the separate effect of token-level gated weight α, layer-level scaled weight λ and the hidden states h from the previous layer which is used to calculate token-level gated weight α in Eq.(4). As shown in Table 3, it can be found that removing any strategy hurts the performance to varying degrees, demonstrating that they are all advantageous. Specifically, the beneficial effect of λ for APT is slightly greater than α overall. Besides, it is effective and meaningful to introduce the context (i.e. the hidden states h from the previous layer) when obtaining the gated weight, especially for SuperGLUE tasks.

4.4 Discussion

What is prefix weight distribution learned by APT? The gate mechanism for prefix serves as the key strategy of the proposed APT, where the learned prefix weight distribution turns out to be a critical point. Figure 2 illustrates the gate weights of the pseudo prefix token for COPA and CoNLL04,
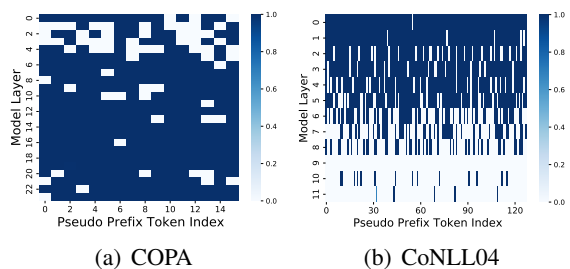

(a) COPA   (b) CoNLL04

Figure 2: Visualization of the learned weights of the prefix token for SuperGLUE task COPA on BERT-large and NER task CoNLL04 on BERT-base, with darker colors indicating higher weights.

respectively. It can be found that CoNLL04 is concerned with bottom layers in the language model which are regarded as phrase-level features, while COPA pays more attention to the higher layers, indicating semantic information. The observation is consistent with the characteristics of corresponding tasks. NER is a token-level task while COPA is a causal reasoning task sensitive to the semantics of sentences, which reminds us that it is worth placing various prefix tokens on specific layers according to the task properties.

Does *variable* prefix work better than *fixed* one? To verify the effectiveness of adaptive prefix under the proposed architecture, we wonder if the learned ratio at each layer can be directly transferred to P-Tuning v2. Taking the gate as a probing indicator, we reset the prefix length of P-Tuning v2 from fixed to variable in different layers based on the ob-

servation of the learned ratio (e.g. the distribution shown in Figure 2). From the comparison between PT-2 and PT-2* in Table 4, we demonstrate that the variable prefix with less trainable parameters surprisingly outperforms the original implementation in fixed prefix. Nonetheless, it is also worth noting that there is still a gap between P-Tuning v2 with variable prefix and APT, where the latter continuously adjusts the weight of prefix during the training phase while the former only initializes with a one-time mask probing.

Whether the adaptive structure benefits the fine-tuning? Compared to P-Tuning v2, APT learns extra gated and scaled weights. To figure it out whether the improvement of APT is brought from more trainable parameters or the adaptive model structure, we adjust the hyper-parameter, i.e., enlarge the prefix length of P-Tuning v2 by 1.5 times to align the number of parameters with our APT. As shown in the comparison between PT-2$^+$ and APT of Table 4, we observe that APT still outperforms enlarged P-Tuning v2 with 1.9%, 0.4% on average for SuperGLUE and NER tasks respectively, validating the superiority of the gate mechanism.

5 Conclusion

In this paper, we investigate prefix tuning and assume that adaptive prefix is probably more efficient and effective than fixed prefix. Firstly, we propose APT that leverages the token-level and the layer-level gate mechanism which achieves an improvement of performance over original prefix tuning. Then, we illustrate the weight distribution learned by APT and take it as a probe, which validates the variable prefix can work better than the fixed one. The above experiments and analysis demonstrate that the adaptive prefix can be served as a promising strategy for parameter-efficient fine-tuning.

Limitations

The proposed approach in this paper also suffers from certain limitations, i.e. we adapt APT on the encoder model and lack design for the other architectures such as decoder-only and encoder-decoder. In addition, it is better to generalize the key idea to other parameter-efficient learning approaches. A unified solution for existing work may be worth exploring in the future.

References

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland. Association for Computational Linguistics.

Xavier Carreras and Lluís Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, pages 89–97, Boston, Massachusetts, USA. Association for Computational Linguistics.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019a. BoolQ: Exploring the surprising difficulty of natural yes/no questions. 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 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019b. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

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.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*.

Demi Guo, Alexander Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. 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 4884–4896, Online. Association for Computational Linguistics.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

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.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*.

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.

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 *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland. Association for Computational Linguistics.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. Gpt understands, too. *arXiv:2103.10385*.

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*.

Nafise Sadat Moosavi, Quentin Delfosse, Kristian Kersting, and Iryna Gurevych. 2022. Adaptable Adapters. In *Proceedings of the 2022 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Seattle, WA, USA. Association for Computational Linguistics.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. 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 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI spring symposium: logical formalizations of commonsense reasoning*, pages 90–95.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021. AdapterDrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7930–7946, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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. In *Advances in Neural Information Processing Systems*, volume 32. 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.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2013. OntoNotes Release 5.0. Abacus Data Network.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

A Experimental Details

Datasets In the full data setting, all train-dev-test splits follow P-Tuning v2 (Liu et al., 2022). For low resources setting, to generate k-shot ($k = 16, 32$) datasets on SuperGLUE, the fixed set of random seed [11,21,42,87,100] is utilized to sample instances in training and development set, while the entire development set is treated as test set, where the average performance is reported in Table 2.

Experimental Setting We grid search the learning rate over [5e-3, 7e-3, 1e-2, 1e-4], training epoch over [20, 40, 60, 80, 100, 120], batch size over [8, 16, 32], and random seeds over [11, 21, 42, 87, 100]. For a fair comparison, the prefix length utilized by APT is consistent with P-Tuning v2. In low resources setting, the batch size we used is 2. In Eq.(4), we take the hidden states of the first input token as representation in previous layer.

Experimental Computation We use the pretrained model BERT-base with 110M parameters, BERT-large with 335M parameters, RoBERTa-large with 355M parameters and DeBERTa-xlarge with 750M parameters. We conduct experiments on NVIDIA V100 or A100 GPUs for each task.

B Further Ablation Results

We demonstrate further ablation results on BERT-large and RoBERTa-large as shown in Table 5. It can be found that the beneficial impact of the three strategies and the observation is consistent with BERT-base in Section 4.3 in general.
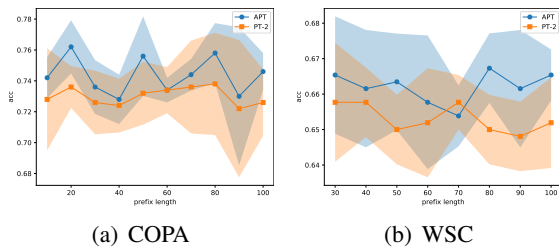

(a) COPA (b) WSC

Figure 3: The performance of APT and PT-2 on COPA and WSC in a range of prefix length on BERT-large.

C Prefix Length

The prefix length is an important hyper-parameter for prefix tuning and APT. Figure 3 illustrates the performance of APT and P-Tuning v2 with different prefix lengths over a range. It can be observed that APT is superior to P-Tuning v2 in most prefix length settings, indicating that APT has a relatively wider range of prefix length to achieve better performance.

D Scientific Artifacts

We use datasets involving SuperGLUE (Wang et al., 2019) benchmark including BoolQ (Clark et al., 2019a), COPA (Roemmele et al., 2011), RTE (Wang et al., 2018), WiC (Pilehvar and Camacho-Collados, 2019) and WSC (Levesque et al., 2012) as well as 3 Named Entity Recognition (NER) tasks including CoNLL03 (Tjong Kim Sang and De Meulder, 2003), CoNLL04 (Carreras and Màrquez, 2004), and OntoNotes 5.0 (Weischedel et al., 2013). The pre-trained model we used are BERT-base / large (Devlin et al., 2019), RoBERTa-large (Liu et al., 2019) and DeBERTa-xlarge (He et al., 2020). We use HuggingFace Transformers (Wolf et al., 2020) and P-Tuning v2 (Liu et al., 2022) as the codebase implemented by PyTorch [2]. They are all open-source and we only use for academic research which is consistent with their intended use.

---

[2]https://pytorch.org/

Model	Setting	SuperGLUE						NER			
		BoolQ	COPA	RTE	WiC	WSC	Avg.	CoNLL03	CoNLL04	OntoNotes	Avg.
BERT-large	APT	**76.0**	**79.0**	**79.4**	**75.1**	**70.2**	**75.9**	90.7	**85.8**	**88.6**	**88.4**
	w/o token-level α	75.8	77.0	77.3	74.8	68.3	74.6	91.1	84.4	88.5	88.0
	w/o layer-level λ	75.4	74.0	76.9	74.6	68.3	73.8	90.7	83.7	88.4	87.6
	w/o hidden states h	74.7	76.0	75.8	74.6	68.3	73.9	**91.2**	84.0	88.6	87.9
RoBERTa-large	APT	**84.8**	**94.0**	**89.9**	**74.6**	**68.3**	**82.3**	**92.7**	**89.0**	**89.8**	**90.5**
	w/o token-level α	84.3	88.0	88.1	73.0	65.4	79.8	92.2	88.7	89.5	90.1
	w/o layer-level λ	84.7	88.0	86.3	72.1	64.4	79.1	92.0	88.7	89.8	90.2
	w/o hidden states h	83.9	91.0	87.0	72.9	64.4	79.8	92.2	88.7	89.4	90.1

Table 5: Ablation experiments on BERT-large and RoBERTa-large for two different level gate mechanisms and the hidden states from the previous layer. **bold**: the best score.

A For every submission:

☑ A1. Did you describe the limitations of your work?
section limitations

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
section abstract and section 1 introduction

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

B ☑ Did you use or create scientific artifacts?

section 4.1

☑ B1. Did you cite the creators of artifacts you used?
section 4.1

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
section D Scientific Artifacts

☑ 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)?
section D Scientific Artifacts

☒ 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?
We use open-source datasets and do not change datasets for a fair comparison.

☐ 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.
It can be found in the cited paper.

C ☑ Did you run computational experiments?

section 4 Experiments

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Table 1 and section appendix A Experimental Computation

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
section appendix A Experimental Details

☑ 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?
Table 2 report the mean and std results.

☒ 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.)?
We follow the existing work and keep consistent with them.

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.?
No response.

☐ 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)?
No response.

☐ 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?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.