Parameter-Efficient Tuning with Special Token Adaptation

Xiaocong Yang[†][*], **James Y. Huang**[‡], **Wenxuan Zhou**[‡] and **Muhao Chen**[‡]
[†]Tsinghua University; [‡]University of Southern California
yangxc.18@sem.tsinghua.edu.cn;
{huangjam,zhouwenx,muhaoche}@usc.edu

Abstract

Parameter-efficient tuning aims at updating only a small subset of parameters when adapting a pretrained model to downstream tasks. In this work, we introduce PASTA, in which we only modify the special token representations (e.g., `[SEP]` and `[CLS]` in BERT) before the self-attention module at each layer in Transformer-based models. PASTA achieves comparable performance to full finetuning in natural language understanding tasks including text classification and NER with up to only 0.029% of total parameters trained. Our work not only provides a simple yet effective way of parameter-efficient tuning, which has a wide range of practical applications when deploying finetuned models for multiple tasks, but also demonstrates the pivotal role of special tokens in pretrained language models.[1]

1 Introduction

Built upon a pretrained language model (PLM; Devlin et al. 2019; Liu et al. 2019; Yang et al. 2019; Chowdhery et al. 2022), many of the recent NLP systems are developed based on task-specific finetuning. In this way, the PLM effectively leverages the task-agnostic knowledge captured during self-supervised pretraining and adapts itself to downstream tasks. However, full finetuning poses a challenge to model deployment under multi-task, memory-limited scenarios, where we need to train and store a separate full-sized model for each substantially distinct task. As an alternative, parameter-efficient tuning (Ding et al., 2022) aims at only updating a small number of parameters when adapting PLMs to downstream tasks while making most of the model parameters fixed and shared among tasks, thus reducing memory usage.

In this paper, we propose **PA**rameter-efficient tuning with **S**pecial **T**oken **A**daptation (PASTA), where we only add trainable vectors to hidden representations of *special tokens* [2] at each layer before the multi-head attention module in Transformer-based PLMs. Our work is motivated by the role of special tokens in PLMs. First, special tokens such as `[CLS]` collect information from the whole input sequence and are typically regarded as the global text representation (Devlin et al., 2019). For sentence-level tasks such as GLUE (Wang et al., 2018), a common practice is to add a new classifier head based on the `[CLS]` representation in the last model layer. Thus, if trained properly, by updating the `[CLS]` representations, we can approximate the result of the *information collection* process in PLMs. Second, many attention heads in PLMs follow a vertical pattern[3], where the attention scores are mostly allocated to either the `[CLS]` or `[SEP]` token (Clark et al., 2019; Kovaleva et al., 2019), as

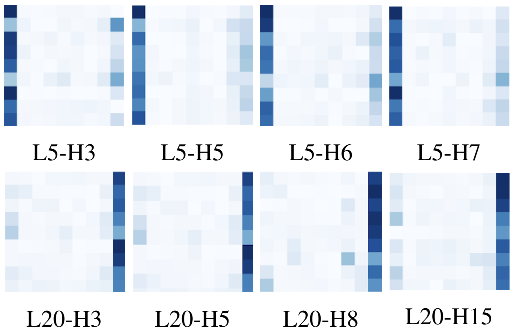

Figure 1: Examples of vertical attention heads in the 5-th and 20-th layer of BERT-large with a random sample from CoLA (Warstadt et al., 2019) as input. Heads in the first row and second row assign most of maximal attention weights to `[CLS]` and `[SEP]` respectively. See Appx. §C for the full attention map.

[*] Work done when visiting USC.

[1] Our code is publicly available at: `https://github.com/luka-group/PASTA/`

[2] WLOG, we use the notation of special tokens `[CLS]` and `[SEP]` in BERT for the convenience of expression, while the method applies in the same way to other paradigms such as `<S>` and `</S>` in RoBERTa (Liu et al., 2019).

[3] Following Voita et al. (2019) and Yao et al. (2021), an attention head is regarded as *vertical* if at least 90% tokens assign maximal attention scores to either `[CLS]` or `[SEP]`.

illustrated in Fig. 1. Therefore, updates to special tokens can also be *disseminated* to other tokens during the forward pass through the vertical attention heads (Elhage et al., 2021), enabling the PLMs to adapt to both sentential and lexical tasks.

By tuning as few as up to 0.029% of the total parameters, PASTA achieves competitive performance on par with full finetuning and BitFit (Zaken et al., 2022) on GLUE (§4.2). It also outperforms P-tuning v2 (Liu et al., 2022) by 0.6% on CoNLL2003 with 20× fewer additional parameters (§4.3). The ablation study shows that we can further reduce trainable parameters to 0.009% with only a slight performance drop (§4.4), showing the merit of adapting special token representations.

2 Related Work

A recent survey (Ding et al., 2022) categorizes three types of *parameter-efficient tuning* methods. *Addition methods* (Houlsby et al., 2019; Lester et al., 2021; Liu et al., 2022) introduce a small number of additional trainable parameters while keeping those in the PLM unchanged. *Specification methods* (Zaken et al., 2022; Guo et al., 2021; Zhao et al., 2020) update a portion of parameters in the PLM while keeping others frozen. *Reparameterization methods* (Aghajanyan et al., 2021; Hu et al., 2021; Qin et al., 2021) modify PLMs' structures to parameter-efficient forms. Our method belongs to the addition-based methods and follows the basic settings of P-tuning v2 (Liu et al., 2022), where newly initialized hidden representations of tokens are inserted into each Transformer layer. Different from most prompt tuning methods that introduce new tokens, we add the introduced vectors to the hidden states of special tokens and keep the sequence length unchanged.

Previous works use probing tasks (Wu et al., 2020) and pruning methods (Prasanna et al., 2020) to study the roles of different modules inside BERT. It has been shown that functional specialization exists in BERT self-attention heads (Clark et al., 2019), and vertical attention heads[3] take up a large portion (Yao et al., 2021). Kovaleva et al. (2019) find that vertical attention heads are almost exclusively associated with attention to `[SEP]` or `[CLS]` tokens, and Clark et al. (2019) conclude that heads in early layers often attend to `[CLS]` while in middle layers attend to `[SEP]`. In this work, we demonstrate that adapting hidden representations of special tokens is sufficient to bring the performance of PLMs to the level of full finetuning.

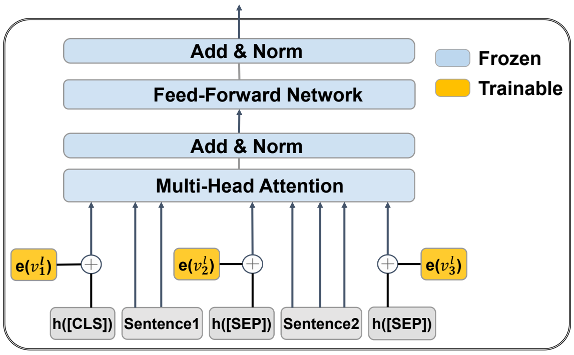

Figure 2: Architecture of PASTA layer in Transformer. Skip-connections in Transformers are not shown for brevity. At layer l we add a trainable vector $\mathbf{e}(\mathbf{v}_p^l) \in \mathbb{R}^d$ to the hidden representation of the p-th special token in the input sequence, and freeze the weights of the PLM.

3 PASTA

Given a large PLM, our goal is to develop a parameter-efficient tuning method that only updates a small set of parameters when adapting to a downstream task. To this end, we propose a simple yet effective method called PASTA, in which we train a hidden vector for every special token at each Transformer layer, along with a task-specific classifier, while freezing the parameters of the PLM.

3.1 Special Token Adaptation

The special token adaption is illustrated in Fig. 2. Although these adaptations are not directly applied to non-special tokens, changes in special token hidden states can be effectively disseminated to other tokens via self-attention during forward passes, thanks to the prevalence of vertical attention heads[3] in PLMs.

Specifically, denote the inputs to the l-th Transformer layer as $\mathbf{H}^l = \{\mathbf{h}_i^l\}_{i=1}^N, \mathbf{h}_i^l \in \mathbb{R}^d$, where N is the number of input tokens, d is the hidden size, PASTA modifies the inputs as follows:

$$\begin{aligned} \mathbf{H}_{\text{mod}}^l &= \{\mathbf{h}_i^l + \mathbf{m}_i^l\}_{i=1}^N, \\ \mathbf{H}^{l+1} &= \text{Trm}^l(\mathbf{H}_{\text{mod}}^l), \end{aligned}$$

where Trm^l is the l-th Transformer layer, $\mathbf{m}_i^l \in \mathbb{R}^d$ is our special token adaptation defined as follows:

$$\mathbf{m}_i^l = \begin{cases} \mathbf{0} & \text{if token } i \text{ is not a special token} \\ \mathbf{e}(\mathbf{v}_p^l) & \text{if token } i \text{ is the } p\text{-th special token} \end{cases}$$

	# Param	Parameter Consistency
Adapter	$\mathcal{O}(L \times d \times r)$	✓
P-tuning v2	$\mathcal{O}(L \times d \times T)$	✓
BitFit	$\mathcal{O}(L \times (d + m))$	✓
Diff-Prune	-	✗
PASTA	$\mathcal{O}(L \times d)$	✓

Table 1: Parameter complexity of PASTA and baselines. Here L and d refer to the number of layers and hidden size of the PLM. m and r refer to the intermediate size of FFN modules in Transformers and Adapters, respectively. T is the prompt length. Parameter consistency refers to whether the set of trainable parameters is consistent across different tasks (Zaken et al., 2022).

where $\mathbf{e}(\mathbf{v}_p^l) \in \mathbb{R}^d$ is the trainable vector added to the hidden representation of the p-th special token in the input sequence. During downstream task training, only those introduced hidden vectors for special tokens and the task-specific classifier are optimized, and the rest of model parameters are frozen.

3.2 Parameter Efficiency and Consistency

As shown in Tab. 1, PASTA achieves $\mathcal{O}(L \times d)$ parameter complexity[4] and updates as few as 0.015%-0.029% of the parameters compared to a full PLM when using BERT-large or RoBERTa-large as backbone. Unlike Adapter (Houlsby et al., 2019) that learns the transformation of all input tokens using a shared FFN, PASTA only learns the task-specific update of special token representations as a bias term, which significantly reduces the parameter capacity needed for adaptation on downstream tasks.

Meanwhile, the set of parameters introduced by PASTA is consistent across different tasks, making it efficient for hardware-based deployment (Zaken et al., 2022). On the contrary, in Diff-Prune, the parameter update is considered as a term of the loss function (Guo et al., 2021), resulting in different sets of updated parameters in distinct tasks.

4 Experiments and Results

We hereby study the downstream performance of PASTA and analyze the properties of introduced hidden vectors.

4.1 Experimental Setup

Baseline Methods. We compare PASTA with the following parameter-efficient tuning methods in prior studies. **Adapter** (Houlsby et al., 2019) introduces new feed-forward modules in Transformer layers while keeping original parameters of the PLM frozen. **BitFit** (Zaken et al., 2022) updates all bias terms in the PLM during finetuning. **Diff-Prune** (Guo et al., 2021) introduces L_0-norm penalty on the updated parameters to encourage sparsity of tuned parameters. **P-tuning v2** (Liu et al., 2022) prepends trainable hidden vectors before the input sequence at each layer while keeping the original PLM parameters frozen. **LoRA** (Hu et al., 2021) uses low-rank decomposition matrices to model the parameter updates.

Model Configuration. We conduct our experiments using BERT-large and RoBERTa-large (We also report experiments with BERT-base in Appx. §A). To facilitate comparison with baseline works, we take most of the experimental results from their original papers which are reported with either BERT-large or RoBERTa-large. Note that multiple `[SEP]` tokens in a single sequence (e.g., in sentence pair tasks like MNLI) are treated as different special tokens and have separate sets of trainable parameters, and the number of trainable parameters varies among downstream tasks according to the number of special tokens added. Details of training and hyperparameters settings are shown in Appx. §B.

4.2 GLUE Tasks

Task Setup. We evaluate PASTA on the widely used GLUE benchmark[5] (Wang et al., 2018). For the convenience of direct comparison, we use the same metrics as were used in baseline works (Devlin et al., 2019; Liu et al., 2019). For experiments with BERT, MRPC and QQP are evaluated using F1 score, STS-B is evaluated using Spearman's correlation coefficient, CoLA is evaluated using Matthew's Correlation, and the other tasks are evaluated using accuracy. For experiments with RoBERTa, STS-B is evaluated using Pearson's correlation coefficient, CoLA is evaluated using Matthew's Correlation, and the other tasks are evaluated using accuracy.

Results. Tabs. 2 and 3 report the performance of PASTA on GLUE benchmark with BERT-large and RoBERTa-large respectively. PASTA with RoBERTa-large achieves the same average score

[4] The number of special tokens are invariant to the scale of models, and are usually very small.

[5] Following previous work (Houlsby et al., 2019; Guo et al., 2021; Zaken et al., 2022), we exclude WNLI since BERT underperforms the majority class baseline (Devlin et al., 2019).

	%Param	**RTE** acc.	**CoLA** mcc.	**STS-B** Spearman	**MRPC** F1	**SST-2** acc.	**QNLI** acc.	**MNLI(m/mm)** acc.	**QQP** F1	**Avg.**
Full Finetuning*	100%	70.1	60.5	86.5	89.3	**94.9**	92.7	**86.7/85.9**	**72.1**	**81.6**
Adapter**	3.6%	71.5	59.5	**86.9**	89.5	94.0	90.7	84.9/85.1	71.8	81.1
Diff-Prune†	0.5%	70.6	61.1	86.0	**89.7**	94.1	**93.3**	86.4/86.0	71.1	81.5
P-tuning v2	0.29%	70.1	60.1	86.8	88.0	94.6	92.3	85.3/84.9	70.6	81.0
BitFit‡	0.08%	**72.0**	59.7	85.5	88.9	94.2	92.0	84.5/84.8	70.5	80.9
PASTA	**0.015%-0.022%**	70.8	**62.3**	86.6	87.9	94.4	92.8	83.4/83.4	68.6	80.9

Table 2: BERT-large model performance on GLUE benchmark test set. Lines with * and ** are results from Devlin et al. (2019) and Houlsby et al. (2019), and lines with † and ‡ are from Guo et al. (2021) and Zaken et al. (2022) respectively. We reimplement experiments of P-tuning v2 on GLUE benchmark with a prompt length of 20.

	%Param	**RTE** acc.	**CoLA** mcc.	**STS-B** Pearson	**MRPC** acc.	**SST-2** acc.	**QNLI** acc.	**MNLI(overall)** acc.	**QQP** acc.	**Avg.**
Full Finetuning*	100%	86.6	68.0	92.4	**90.9**	96.4	94.7	90.2	**92.2**	88.9
LoRA†	0.24%	**87.4**	68.2	**92.6**	**90.9**	96.2	94.9	**90.6**	91.6	**89.0**
PASTA	**0.015%-0.029%**	86.6	**69.7**	91.8	**90.9**	**96.8**	**95.1**	90.4	89.9	88.9

Table 3: RoBERTa-large model performance on GLUE benchmark. Lines with * are results from Liu et al. (2019), and lines with † are from Hu et al. (2021). We follow the metric settings of baselines and also report results on GLUE development set for the convenience of direct comparison.

	CoLA	**RTE**	**MRPC**	**STS-B**	**CoNLL2003**
PASTA	**65.4**	**76.2**	89.7	**90.8**	**94.0**
- w/o [CLS]	58.8	72.6	91.4	90.2	93.7
- w/o [SEP]	64.5	71.1	91.9	90.3	93.7
- shared vector	64.7	74.7	**92.1**	90.0	93.9
- classifier only	36.5	54.2	81.5	64.9	77.4

Table 4: Performance of ablation study with BERT-large on GLUE and CoNLL2003 development sets.

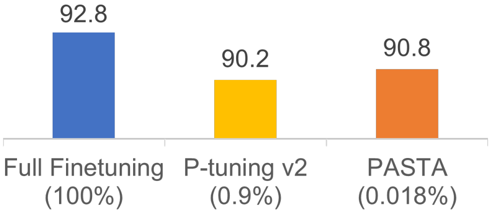

Figure 3: NER results on CoNLL03 with BERT-large (F1 score percentages are marked over the bars). Each method is labeled with the percentage of trainable parameter sizes with regard to full finetuning in parentheses.

as that of full finetuning over GLUE tasks. PASTA with BERT-large achieves an average score on par with BitFit using over 3× fewer trainable parameters and comparable results to other higher parameter complexity baselines. The results demonstrate that by leveraging the pivotal role of special tokens in PLMs, PASTA is able to effectively adapt the model to sentence-level tasks with significantly fewer parameters tuned than previous methods.

4.3 Named Entity Recognition

Task Setup. We experiment with the NER task on CoNLL2003 (Tjong Kim Sang and De Meulder, 2003). Following Devlin et al. (2019), we formulate NER as a token classification problem.

Results. As shown in Fig. 3, PASTA with BERT-large achieves an F1 score of 90.8% on CoNLL2003 test set, outperforming P-tuning v2 (Liu et al., 2022) by 0.6% with 20× fewer trainable parameters, while falling behind full finetuning by 2.0%. Nevertheless, the strong performance of PASTA compared to P-tuning v2 indicates that even though PASTA only directly adapts special tokens, the representations of all input tokens can still be properly tuned, supporting our hypothesis that vertical attention heads are able to disseminate adaptations in special token hidden states to other tokens.

4.4 Analysis

Ablation on choices of special tokens. To understand the effect of tuning different combinations of special tokens on downstream tasks, we further limit the additional parameter capacity of PASTA by only adapting either [CLS] or [SEP], or share a common vector across all special tokens. Tab. 4 shows the performance of three ablated variants and a baseline that only tunes the classification head on top of a fixed BERT-large. In general, we observe

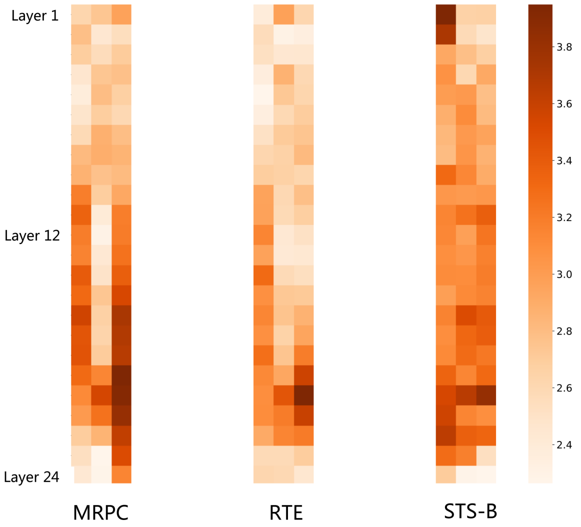

Figure 4: The distribution map of norms of introduced hidden vectors on MRPC, RTE and STS-B tasks with BERT-large. In each subgraph, the first column shows the norms of introduced vectors added to `[CLS]` at each layer, and the second and third columns are introduced vector norms at two `[SEP]` tokens respectively.

a decrease in performance on most tasks except on MRPC for three PASTA variants, and performance degrades significantly if we do not adapt any special tokens. These results demonstrate the vital role of introduced hidden vectors for special tokens in PASTA, while the best choice of special tokens to be adapted may vary depending on the task.

Norm distribution of introduced hidden vectors. Fig. 4 shows the norm distribution of introduced vectors on downstream tasks. The introduced hidden vectors learn the difference of special tokens between pretrained and adapted models, and thus norms of those vectors indicate the magnitude of parameter change at different layers. Similar to the pattern of parameter change during full finetuning (Kovaleva et al., 2019), PASTA generally has larger norms of hidden vectors at layers closer to the output.

5 Conclusion

We present PASTA, a parameter-efficient tuning method that only modifies special token representations at each Transformer layer when adapting to downstream tasks. Our approach is motivated by the observation that PLMs have a large amount of vertical attention heads that heavily attend to special tokens, and these heads disseminate value updates from special tokens to all of the other tokens. Experiments show that PASTA achieves strong performance comparable to full finetuning on sentential and lexical tasks with high parameter efficiency. Our work not only provides an effective solution for parameter-efficient tuning, but also demonstrates the pivotal role of special tokens in PLMs.

Limitations

In this work we hypothesize that the vertical attention heads could play a role as "information disseminator" based on the theoretical analysis of Transformers (Elhage et al., 2021). However, we still have no direct approaches such as probing tasks and reverse engineering to prove this assumption. And since PASTA relies on adaptation of special tokens, it cannot be applied to language models which do not pad special tokens to input sequences such as GPT-2 (Radford et al., 2019). For the empirical results, we choose GLUE benchmark and CoNLL2003 to study the performance on language understanding tasks. The effectiveness of PASTA on language generation tasks has not been tested in this work due to limited bandwidth. Finally, similar to other parameter-efficient tuning methods, PASTA suffers from a higher computational cost compared to full finetuning.

Acknowledgement

We appreciate the anonymous reviewers for their insightful comments and suggestions. Xiaocong Yang, Wenxuan Zhou and Muhao Chen are supported by the National Science Foundation of United States Grant IIS 2105329. This material is supported in part by the DARPA MCS program under Contract No. N660011924033 with the United States Office Of Naval Research, an Amazon Research Award, a Cisco Research Award and a subaward from NSF Cloudbank 1925001.

References

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. 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 7319–7328, Online. Association for Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek

Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. 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.

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

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

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.

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 *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, Hong Kong, China. 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.

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.

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.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization.

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Wen-tau Yih, and Madian Khabsa. 2021. Unipelt: A unified framework for parameter-efficient language model tuning.

Sai Prasanna, Anna Rogers, and Anna Rumshisky. 2020. When BERT Plays the Lottery, All Tickets Are Winning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3208–3229, Online. Association for Computational Linguistics.

Yujia Qin, Xiaozhi Wang, Yusheng Su, Yankai Lin, Ning Ding, Jing Yi, Weize Chen, Zhiyuan Liu, Juanzi Li, Lei Hou, Peng Li, Maosong Sun, and Jie Zhou.

2021. Exploring universal intrinsic task subspace via prompt tuning.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

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 *International Conference on Learning Representations*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. 2020. Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4166–4176, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*.

Xingcheng Yao, Yanan Zheng, Xiaocong Yang, and Zhilin Yang. 2021. Nlp from scratch without large-scale pretraining: A simple and efficient framework.

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.

Mengjie Zhao, Tao Lin, Fei Mi, Martin Jaggi, and Hinrich Schütze. 2020. Masking as an efficient alternative to finetuning for pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2226–2241, Online. Association for Computational Linguistics.

A Performance of BERT-base

Tab. 5 reports the performance of PASTA and a part of baselines with BERT-base as backbones on GLUE development sets. We do not put the result table in the main body of this work because most baselines did not report GLUE test set scores using BERT-base, and only three of them reported GLUE development set scores with BERT-base in their works. PASTA slightly underperforms baselines on average, while it outperforms other models on small datasets such as RTE and MRPC.

B Implementation Details

Our model is implemented based on Huggingface's Transformers. We optimize models with AdamW (Loshchilov and Hutter, 2017). We set the maximum input length to 128 and use a fixed random seed of 42 for all tasks. Experiments are done on NVIDIA RTX A5000 for averagely 3 hours per task, and distributed training is used for most tasks. Tab. 6 reports the best hyperparameters for model training. For hyperparameter search, we select learning rate from {5e-4, 1e-3, 2e-3, 2.5e-3, 3e-3, 4.5e-3, 5e-3, 7e-3} and the number of epochs from {50, 80, 100, 150}.

C Full Attention Map

Fig. 5 illustrates the attention maps of all heads in BERT-large. With a random sample from CoLA dataset as input (*"Fred watered the plants flat."*), there are 112 heads out of 384 in total being vertical heads[3].

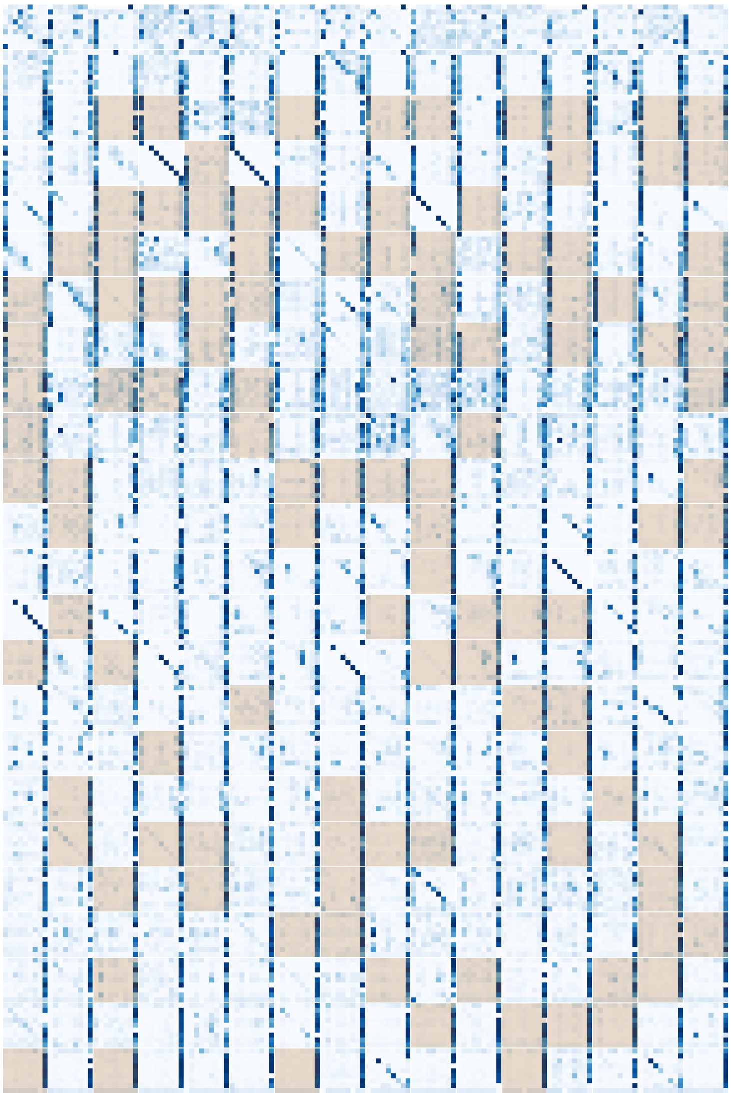

Figure 5: Full attention map of BERT-large pretrained with a random sample from CoLA as input. Rows and columns represent model layers and heads respectively, and darker color indicates larger weights. Vertical attention heads are highlighted in orange.

	%Param	**RTE**	**CoLA**	**STS-B**	**MRPC**	**SST-2**	**QNLI**	**MNLI(m/mm)**	**QQP**	**Avg.**
Full Finetuning *	100%	66.4	**62.1**	**89.8**	90.9	91.6	90.0	**83.2/ -**	**87.4**	82.7
Adapter*	0.81%	71.8	61.5	88.6	89.9	91.9	**90.6**	83.1/ -	86.8	**83.0**
BitFit†	0.8%	72.3	58.8	89.2	90.4	**92.1**	90.2	81.4/ -	84.0	82.3
PASTA	**0.015%-0.022%**	**73.6**	57.9	88.7	**91.5**	91.2	89.7	77.8/78.8	80.8	81.4

Table 5: PASTA with BERT-base model performance on GLUE benchmark development set. Lines with * and † refer to results from Mao et al. (2021) and Zaken et al. (2022) respectively.

	RTE	**CoLA**	**STS-B**	**MRPC**	**SST-2**	**QNLI**	**MNLI**	**QQP**	**CoNLL2003**
Learning rate	4.5e-3	5e-3	2e-3	2.5e-3	7e-3	2e-3	5e-4	5e-3	3e-3
Batch size	32×4	32×1	32×3	32×4	64×3	32×4	32×1	32×4	16×1
Number of adapted tokens	3	2	3	3	2	3	3	3	2
Training epochs	150	100	150	150	100	80	50	100	100
Best dev performance	76.2	65.4	90.8	89.7	93.9	92.2	83.7	87.9	94.1
Best epochs	121	63	109	136	99	42	49	97	89

Table 6: PASTA with BERT-large training details for GLUE and CoNLL2003 tasks. Distributed training on multiple GPUs is used when avaiable for less training time.