DePT: Decomposed Prompt Tuning for Parameter-Efficient Fine-tuning

Zhengxiang Shi
University College London
London, United Kingdom
zhengxiang.shi.19@ucl.ac.uk

Aldo Lipani
University College London
London, United Kingdom
aldo.lipani@ucl.ac.uk

Abstract

Prompt tuning (PT), where a small amount of trainable soft (continuous) prompt vectors is affixed to the input of language models (LM), has shown promising results across various tasks and models for parameter-efficient fine-tuning (PEFT). PT stands out from other PEFT approaches because it maintains competitive performance with fewer trainable parameters and does not drastically scale up its parameters as the model size expands. However, PT introduces additional soft prompt tokens, leading to longer input sequences, which significantly impacts training/inference time and memory usage due to the Transformer's quadratic complexity. Particularly concerning for Large Language Models (LLMs) that face heavy daily querying. To address this issue, we propose **De**composed **P**rompt **T**uning (DePT), which decomposes the soft prompt into a shorter soft prompt and a pair of low-rank matrices that are then optimised with two different learning rates. This allows DePT to achieve better performance while saving over 20% memory and time costs compared to vanilla PT and its variants, without changing trainable parameter sizes. Through extensive experiments on 21 natural language processing (NLP), we demonstrate that DePT outperforms state-of-the-art PEFT approaches, including the full fine-tuning baseline in some scenarios. Additionally, we empirically show that DePT grows more efficient as the model size increases. Code is available at https://github.com/ZhengxiangShi/DePT.

1 Introduction

Prompt Tuning (PT) [21] has emerged as a promising parameter-efficient fine-tuning (PEFT) approach, which appends trainable continuous prompt vectors to the input. PT stands out from other PEFT approaches as it maintains competitive performance with fewer trainable parameters and does not drastically scale up its trainable parameters as the model size expands. While PT has shown promising results across various tasks and models, it has two major limitations: (1) PT often suffers from slow convergence and is sensitive to the initialization [21, 45, 48]; and (2) PT extends the total length of the input sequence, consequently exacerbating the computation demand (*i.e.,* train/inference time and memory cost), due to the quadratic complexity of the Transformer [44]. This is further accentuated given the slow convergence issue. Recent studies [40, 45, 23] have proposed the variants of the vanilla PT to tackle the first issue by initially pre-training soft prompts on a variety of source tasks, which is known as *Parameter-Efficient Transfer Learning* (PETL). Some studies [1, 48] also improve the performance of the PT by jointly training learned prompts from these source tasks on multiple target tasks (referred to as *Multi-task Learning*). However, the issue of increased computational load due to the extension of sequence length remains largely unaddressed. While PETL approaches can reduce the training steps for model convergence, each optimization step remains computationally expensive in terms of time and memory. Most importantly, it does not enhance the efficiency during the inference, which is particularly crucial in the era of Large Language Models (LLMs), considering that the trained models may be queried millions of times per day.

3rd Workshop on Efficient Natural Language and Speech Processing (NeurIPS 2023).

In this work, we propose **De**composed **P**rompt **T**uning (DEPT), which decomposes a trainable soft prompt into a shorter soft prompt and a couple of low-rank matrices, where the multiplication of low-rank matrices is then added element-wise to frozen word embeddings. This shorter soft prompt and the updated word embedding matrix are then optimised using two different learning rates. Experimental results on 21 NLP tasks demonstrate DEPT outperforms the state-of-the-art PEFT approaches, including the full fine-tuning baseline in certain scenarios (§3.1). Our study empirically shows that DEPT largely improves the training efficiency across various model architectures and sizes, saving more than 20% in both training time and memory costs compared to the vanilla PT. Importantly, DEPT becomes increasingly efficient as the model size grows, making it particularly advantageous and suitable for LLMs (§3.2).

2 Method

The decomposition of the soft prompt. Our approach differs from the vanilla PT [21] in the aspect of inputs. We decompose a trainable prompt matrix $\boldsymbol{P} \in \mathbb{R}^{l\times d}$ from the vanilla PT into two components: (1) a shorter trainable prompt matrix $\boldsymbol{P}_s \in \mathbb{R}^{m\times d}$; and (2) a pair of low-rank matrices, $\boldsymbol{A} \in \mathbb{R}^{s\times r}$ and $\boldsymbol{B} \in \mathbb{R}^{r\times d}$, where typically the rank of the matrices $r \ll \min(s, d)$. The first component, the smaller trainable prompt matrix, is appended to the word embedding matrix in a similar manner as in the vanilla PT. The second component uses the multiplication of two low-rank matrices to represent the update of the word embedding through a coordinate-wise sum:

$$\boldsymbol{W}_i^{'} = \boldsymbol{W}_i + \Delta\boldsymbol{W}_i = \boldsymbol{W}_i + \boldsymbol{B}\boldsymbol{A} \in \mathbb{R}^{s\times d}, \tag{1}$$

where $\boldsymbol{W}_i$ is frozen and does not receive gradient updates during the training, whereas $\boldsymbol{A}$ and $\boldsymbol{B}$ are trainable. Following [15], we use a random Gaussian initialization for A and zero for B, so $\Delta W = BA$ is zero when the training starts. The loss function is then optimised as follows:

$$\mathcal{L}_{\text{DEPT}} = -\sum_i \log P(\boldsymbol{y}_i \,|\, [\boldsymbol{P}_s, \boldsymbol{W}_i^{'}] \,; \Theta) \tag{2}$$

In our experiment, we choose the values of m and r to satisfy the equation $l\times d = m\times d + (s+d)\times r$ for maintaining the exact size of trainable parameters as in the vanilla PT. Consequently, m is always less than l when $r > 0$. This design improves memory efficiency and reduces computational expense compared to the vanilla PT, as the shorter input sequence length (*i.e.,* $m + s < l + s$) substantially reduces computation due to the quadratic complexity of the Transformer [44].

Two rates of learning. Our strategy also differs from the vanilla PT method in the aspect of training. We train the shorter trainable prompt matrix, $\boldsymbol{P}_s$, with the learning rate α_1 and the pair of low-rank matrices, $\boldsymbol{A}$ and $\boldsymbol{B}$, with the learning rate α_2, rather than applying a single learning rate as in the vanilla PT. The α_1 is typically much larger than the α_2.

3 Experiments and Results

In this section, we evaluate our proposed method DEPT across 21 NLP (see §3.1) and then assess the train/inference time and memory cost (see §3.2). Please see experimental setup in Appendix §B.

3.1 Main Results

This section shows the empirical evidence supporting the effectiveness of our proposed method DEPT. Our experimental results reveal two key findings: (1) DEPT consistently outperforms the vanilla PT and its PETL variants; and (2) DEPT achieves competitive or even better performance than state-of-the-art PEFT approaches while using fewer trainable parameters. Below we delve deeper with respect to various tasks.

#1. Performance on GLUE and SuperGLUE benchmarks. As shown in Table 3, our experimental result indicates that DEPT outperforms state-of-the-art PEFT approaches, such as Adapter, LoRA and LST on the GLUE and SuperGLUE benchmarks, while using fewer trainable parameters. Remarkably, DEPT also outperforms the full fine-tuning baseline on both benchmarks. In addition, DEPT outperforms vanilla PT and its variants that introduce additional transfer learning and multi-task learning. For example, DEPT surpasses MPT with 0.1% on the GLUE benchmark and 0.4% on the SuperGLUE benchmark, without utilizing additional transfer learning or multi-task learning.

Table 1: Test results on GLUE and SuperGLUE benchmarks, with the corresponding size of trainable parameters. All of the results are based on T5-BASE models. We use Pearson correlation for STS-B, F1 for MultiRC (Multi), and accuracy for other tasks as evaluation metrics.

Method	**#Para**	**GLUE**									**SuperGLUE**					
		MNLI	QQP	QNLI	SST-2	STS-B	MRPC	RTE	CoLA	Mean	Multi	Bool	WiC	WSC	CB	Mean
Single-Task Learning																
Fine-tuning[1]	220M	86.8	91.6	93.0	94.6	89.7	90.2	71.9	61.8	84.9	72.8	81.1	70.2	59.6	85.7	73.9
Adapter[1]	1.9M	86.5	90.2	93.2	93.8	90.7	85.3	71.9	64.0	84.5	75.9	82.5	67.1	67.3	85.7	75.7
AdapterDrop[1]	1.1M	86.3	90.2	93.2	93.6	91.4	86.3	71.2	62.7	84.4	72.9	82.3	68.3	67.3	85.7	75.3
BitFit[1]	280k	85.3	90.1	93.0	94.2	90.9	86.8	67.6	58.2	83.3	74.5	79.6	70.0	59.6	78.6	72.5
LoRA[2]	3.8M	86.3	89.0	93.2	94.3	90.9	90.1	75.5	63.3	85.3	–	–	–	–	–	–
LST[2]	3.8M	85.6	88.8	93.3	94.1	90.7	90.4	71.9	58.1	84.1	–	–	–	–	–	–
PT[4]	76.8k	83.4	90.2	93.1	91.9	90.2	90.1	78.8	60.7	84.8	65.7	63.7	50.8	51.9	67.9	60.0
DEPT (ours)	76.8k	85.0	90.4	93.2	94.2	90.8	90.7	79.1	63.8	85.9	74.3	79.3	68.7	67.3	92.9	76.5
Multi-task Learning																
Fine-tuning (m) [1]	28M	85.7	91.1	92.0	92.5	88.8	90.2	75.4	54.9	83.8	–	–	–	–	–	–
Adapter (m) [1]	1.8M	86.3	90.5	93.2	93.0	89.9	90.2	70.3	61.5	84.4	–	–	–	–	–	–
HyperFormer (m) [1]	638k	85.7	90.0	93.0	94.0	89.7	87.2	75.4	63.7	84.8	–	–	–	–	–	–
HyperDecoder (m) [1]	1.8M	86.0	90.5	93.4	94.0	90.5	87.7	71.7	55.9	83.7	–	–	–	–	–	–
Single-Task Training + Transfer Learning																
SPoT[1]	76.8k	85.4	90.1	93.0	93.4	90.0	79.7	69.8	57.1	82.3	74.0	77.2	67.0	50.0	46.4	62.9
ATTEMPT[1]	232k	84.3	90.3	93.0	93.2	89.7	85.7	73.4	57.4	83.4	74.4	78.8	66.8	53.8	78.6	70.5
MPT[3]	77.6k	85.9	90.3	93.1	93.8	90.4	89.1	79.4	62.4	85.6	74.8	79.6	69.0	67.3	79.8	74.1
Multi-task Learning + Transfer Learning																
ATTEMPT (m) [3]	96k*	83.8	90.0	93.1	93.7	90.8	86.1	79.9	64.3	85.2	74.4	78.5	66.5	69.2	82.1	74.1
MPT (m) [3]	10.5k*	84.3	90.0	93.0	93.3	90.4	89.2	82.7	63.5	85.8	74.8	79.2	70.2	67.3	89.3	76.1

[1] from [1]. [2] from [41]. [3] from [48]. [4] we reproduce and substantially increase the performance of the vanilla PT reported in the prior work [1]. * These values are obtained after amortizing over 8 tasks, and the minimal number of parameters to perform a single task remains 232k and 77.6k for ATTEMPT and MPT. (m) represents additional multi-task training.

Table 2: Test results on MRQA 2019 Shared Task and other datasets using the T5-BASE model. We report the F_1 for MRQA tasks and accuracy for other datasets across three seeds, with standard deviations in subscripts. All baseline results are directly quoted from [48].

Method	**#Para**	**MRQA**					**Others**				
		NQ	HP	SQA	News	Mean	WG	Yelp	SciTail	PAWS	Mean
Fine Tuning	220M	75.1	77.5	81.1	65.2	74.7	61.9	96.7	95.8	94.1	87.1
Adapters	1.9M	74.2	77.6	81.4	65.6	74.7	59.2	96.9	94.5	94.3	86.2
BitFit	280K	70.7	75.5	77.7	64.1	72.0	57.2	94.7	94.7	92.0	84.7
PT	76.8K	67.9	72.9	75.7	61.1	69.4	49.6	95.1	87.9	55.8	72.1
SPoT	76.8K	68.2	74.8	75.3	58.2	69.1	50.4	95.4	91.2	91.1	82.0
ATTEMPT	232K	70.4	75.2	77.3	62.8	71.4	57.6	96.7	93.1	92.1	84.9
MPT	77.6K	$72.0_{0.1}$	$75.8_{0.1}$	$77.2_{0.1}$	$63.7_{0.1}$	72.2	$56.5_{0.9}$	$96.4_{0.0}$	$95.5_{0.1}$	$93.5_{0.1}$	85.5
DEPT (ours)	76.8K	$73.2_{0.1}$	$76.8_{0.3}$	$77.6_{0.2}$	$64.4_{0.1}$	73.0	$59.0_{0.2}$	$96.8_{0.1}$	$95.6_{0.2}$	$93.7_{0.1}$	86.3

#2. Performance on MRQA 2019 Shared Task and other NLP datasets. Table 2 presents the performance of various PEFT approaches, including DEPT, on the MRQA 2019 Shared Task and four other datasets. We observe that DEPT improves the average performance of the vanilla PT by a substantial margin of +3.6% on MRQA and +14.2% on the other datasets. DEPT exceeds the performance of the PT variants that leverage additional transfer and multi-task learning, without introducing extra trainable parameters to the vanilla PT or relying on any PETL approaches. While DEPT improves over the vanilla PT and its variants are promising, there remains a disparity in performance when compared to the full fine-tuning baseline.

3.2 Time and Memory Efficiency

This section shows the empirical evidence supporting the efficiency of our proposed method DEPT, spanning over diverse model architectures of varying scales on the GLUE benchmark. To ensure a fair comparison, we consistently keep the number of trainable parameters in DEPT the same as that in the vanilla PT ($l = 100$). As a result, once we choose the length of the soft prompt m in DEPT, the rank of the low-rank matrices r becomes deterministic. Below we elaborate on two key findings.

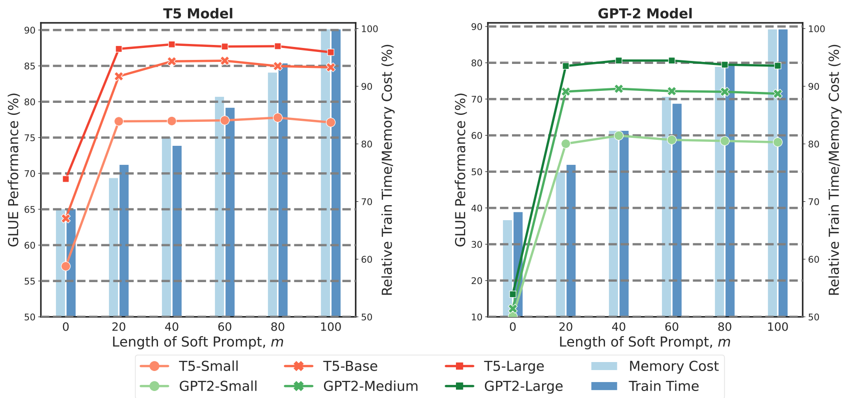

Figure 1: Performance on the GLUE benchmark for different soft prompt lengths m in DEPT, associated with corresponding relative train time and memory cost. The time and memory are averaged over different model sizes using batch size as 16. DEPT consistently uses the same number of trainable parameters as the vanilla PT (m=100).

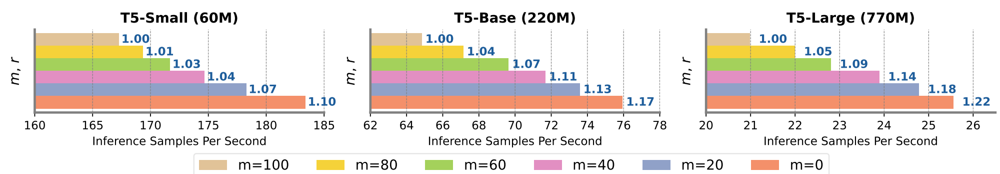

Figure 2: Average inference speed on GLUE benchmark using varying soft prompt length m and the rank of low-rank matrices r, keeping the total number of trainable parameters constant. Small texts in blue indicate the speed relative to the vanilla PT (represented by brown) (m=100).

 # 1. DEPT improves time and memory efficiency up to more than 20%. Figure 1 presents the mean performance of DEPT, associated with average training time and memory, on the GLUE benchmarks, against different lengths of soft prompt m. The average training time and memory costs are computed across 8 tasks on the GLUE benchmark and three different model sizes. The study reveals that decomposing the soft prompt ($l = 100$) into a small soft prompt and low-rank matrics delivers comparable or even better performance while substantially enhancing the efficiency of training and reducing memory utilization. Specifically, using a soft prompt length of 20 in DEPT with the T5 model leads to a better average performance on the GLUE benchmark to vanilla PT, while improving the efficiency of training and reducing memory utilization by approximately 25%. This observation is also applicable when the GPT model is used as the backbone model.

 # 2. DEPT grows more efficient as the model size increases. Figure 2 represents the inference speed, measured by the average number of samples evaluated per second on the GLUE benchmark using a single RTX 3090 GPU. The inference time is computed using the Huggingface Trainer Class. We observe that the relative improvement in the number of inference samples per second over vanilla PT grows as the model size increases. For example, when using the T5-SMALL model, the vanilla PT evaluates 167.3 samples per second, while DEPT ($m = 20$) evaluates 178.3 samples per second, resulting in a 6.5% boost in inference speed. In contrast, when the T5-LARGE is utilized, the vanilla PT evaluates 21.0 samples per second and DEPT ($m = 20$) evaluates 24.8 samples per second, resulting in an 18.1% increase in inference speed, a substantial rise from the previous 6.5%. This indicates that DEPT is particularly beneficial and more applicable in the context of LLMs.

4 Conclusion

In this work, we propose Decomposed Prompt Tuning (DEPT), which substantially improves the efficiency of the vanilla PT (up to over 20%) in terms of time and memory while delivering competitive or even superior performance compared to the state-of-the-art PEFT methods. Remarkably, DEPT efficiency amplifies with increasing model sizes, making it exceptionally apt for LLMs.

References

[1] Akari Asai, Mohammadreza Salehi, Matthew Peters, and Hannaneh Hajishirzi. ATTEMPT: Parameter-efficient multi-task tuning via attentional mixtures of soft prompts. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6655–6672, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[2] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 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, May 2022. Association for Computational Linguistics.

[3] Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[4] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 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, June 2019. Association for Computational Linguistics.

[5] Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, volume 23, pages 107–124, 2019.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.

[7] William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[8] Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*, 2017.

[9] Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. MRQA 2019 shared task: Evaluating generalization in reading comprehension. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 1–13, Hong Kong, China, November 2019. Association for Computational Linguistics.

[10] Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, Prague, June 2007. Association for Computational Linguistics.

[11] Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. PPT: Pre-trained prompt tuning for few-shot learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8410–8423, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[12] Demi Guo, Alexander Rush, and Yoon Kim. 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, August 2021. Association for Computational Linguistics.

[13] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online, July 2020. Association for Computational Linguistics.

[14] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799, 2019.

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

[16] Hamish Ivison and Matthew Peters. Hyperdecoders: Instance-specific decoders for multi-task NLP. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1715–1730, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[17] Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. 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 565–576, Online, August 2021. Association for Computational Linguistics.

[18] Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[19] Tushar Khot, Ashish Sabharwal, and Peter Clark. Scitail: A textual entailment dataset from science question answering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018.

[20] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

[21] Brian Lester, Rami Al-Rfou, and Noah Constant. 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, November 2021. Association for Computational Linguistics.

[22] Hector Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2012.

[23] Junyi Li, Tianyi Tang, Jian-Yun Nie, Ji-Rong Wen, and Xin Zhao. Learning to transfer prompts for text generation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3506–3518, Seattle, United States, July 2022. Association for Computational Linguistics.

[24] Xiang Lisa Li and Percy Liang. 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, August 2021. Association for Computational Linguistics.

[25] Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. Scaling down to scale up: A guide to parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.15647*, 2023.

[26] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 1950–1965. Curran Associates, Inc., 2022.

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

[28] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[29] Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. UniPELT: A unified framework for parameter-efficient language model tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6253–6264, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[30] Mohammad Taher Pilehvar and Jose Camacho-Collados. 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)*. Association for Computational Linguistics, June 2019.

[31] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), jan 2020.

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

[33] Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. AdapterDrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, November 2021.

[34] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, aug 2021.

[35] Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Association for Computational Linguistics, April 2021.

[36] Zhengxaing Shi and Aldo Lipani. Don't stop pretraining? make prompt-based fine-tuning powerful learner. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[37] Zhengxiang Shi, Yue Feng, and Aldo Lipani. Learning to execute actions or ask clarification questions. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2060–2070, Seattle, United States, July 2022. Association for Computational Linguistics.

[38] Zhengxiang Shi, Francesco Tonolini, Nikolaos Aletras, Emine Yilmaz, Gabriella Kazai, and Yunlong Jiao. Rethinking semi-supervised learning with language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5614–5634, Toronto, Canada, July 2023. Association for Computational Linguistics.

[39] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural*

Language Processing, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[40] Yusheng Su, Xiaozhi Wang, Yujia Qin, Chi-Min Chan, Yankai Lin, Huadong Wang, Kaiyue Wen, Zhiyuan Liu, Peng Li, Juanzi Li, Lei Hou, Maosong Sun, and Jie Zhou. On transferability of prompt tuning for natural language processing. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, July 2022.

[41] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. LST: Ladder side-tuning for parameter and memory efficient transfer learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[42] Yi-Lin Sung, Varun Nair, and Colin A Raffel. Training neural networks with fixed sparse masks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 24193–24205. Curran Associates, Inc., 2021.

[43] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[45] Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou', and Daniel Cer. SPoT: Better frozen model adaptation through soft prompt transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5039–5059, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[46] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In *arxiv*, 2019.

[47] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 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, November 2018. Association for Computational Linguistics.

[48] Zhen Wang, Rameswar Panda, Leonid Karlinsky, Rogerio Feris, Huan Sun, and Yoon Kim. Multitask prompt tuning enables parameter-efficient transfer learning. In *The Eleventh International Conference on Learning Representations*, 2023.

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

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

[51] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on EMNLP*, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[52] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 649–657, Cambridge, MA, USA, 2015. MIT Press.

[53] Yuan Zhang, Jason Baldridge, and Luheng He. PAWS: Paraphrase adversaries from word scrambling. 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 1298–1308, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Appendix Overview

The appendix is structured as follows:

Appendix §A discusses the related works.

Appendix §B describes the experimental setup.

Appendix §C provides a visualization of the model performance against the number of trainable parameters on the GLUE and SuperGLUE benchmarks.

Appendix §D provides a brief description of all datasets used in this work.

Appendix §E provides implementation details and hyperparameters for all comparison methods used in our experiments.

A Related Works

Parameter-efficient Fine-tuning. In contrast to standard fine-tuning and prompt-based fine-tuning [6, 35, 36] where full parameters are updated, parameter-efficient fine-tuning (PEFT) approaches have demonstrated remarkable performance across a wide range of tasks [47, 37] while updating only a limited number of parameters. Adapters [14], along with its variants, HyperFormer [17] and Compacter [28], add new trainable modules (adapters) to each transformer block of the T5 model [31]. BitFit [2] limits updates only to the bias parameters, while this method tends to underperform on larger networks [25]. Prefix-tuning [24] adds a soft prompt, parameterized by a feed-forward network, to the model input. Diff pruning [12] learns a sparse update of a neural network's weights at the cost of more memory usage. FishMask [42] also performs sparse updates, but it is computationally intensive and inefficient on contemporary deep learning hardware [25]. LoRA [15] employs a straightforward low-rank matrix decomposition to parameterise the weight update. $(IA)^3$ [26] scales activations by learned vectors for few-shot learning. LST [41] operates a small transformer network on the side of the pre-trained network, aiming to decrease the training memory. Prompt Tuning (PT) [21] appends a trainable soft prompt to the model input embeddings. In comparison to the above-mentioned PEFT approaches, PT uses fewer trainable parameters, which do not proliferate as the model size expands. [29] introduces a method that combines Prefix-tuning, Adapters, and LoRA through a gating mechanism. DEPT is also applicable to this method and can be easily integrated with other PEFT approaches.

Transfer Learning for PT. Several recent works aim to enhance the performance of PT through parameter-efficient transfer learning (PETL). PPT [11] strives to improve the performance of PT [21] by further pre-training [13, 38], which necessitates a set of hand-crafted, task-specific designs and considerable computational cost. [40] improves PT via prompt transfer across different tasks and models. SPoT [45] adopts a single prompt, chosen based on a similarity measure at the cost of a massive search. ATTEMPT [1] employs an attention mechanism over the source prompts to initialize the prompt for target tasks at the cost of extra parameters. MPT [48] applies a shared soft prompt across different tasks, while its effectiveness for a broad range of source tasks remains untested - it is debatable whether a diverse range of tasks can utilise a single prompt to share all knowledge effectively. Previous works [1, 48] might have overemphasized the importance of transfer learning and multi-task learning in boosting model performance when extensive labelled datasets are accessible. It is worth noting that the primary benefit of PETL for PT is in accelerating training convergence and improving the model performance, particularly in the context of few-shot learning [11].

B Experimental Setup

Datasets and tasks. We evaluate our proposed method DEPT on 21 NLP tasks. For NLP tasks, we follow the previous works [45, 41, 1, 48] and use various datasets sourced from: (1) GLUE

Table 3: Test results on GLUE and SuperGLUE benchmarks, with the corresponding size of trainable parameters. All of the results are based on T5-BASE models. We use Pearson correlation for STS-B, F1 for MultiRC (Multi), and accuracy for other tasks as evaluation metrics.

Method	#Para	GLUE									SuperGLUE					
		MNLI	QQP	QNLI	SST-2	STS-B	MRPC	RTE	CoLA	Mean	Multi	Bool	WiC	WSC	CB	Mean
Single-Task Learning																
Fine-tuning[1]	220M	86.8	91.6	93.0	94.6	89.7	90.2	71.9	61.8	84.9	72.8	81.1	70.2	59.6	85.7	73.9
Adapter[1]	1.9M	86.5	90.2	93.2	93.8	90.7	85.3	71.9	64.0	84.5	75.9	82.5	67.1	67.3	85.7	75.7
AdapterDrop[1]	1.1M	86.3	90.2	93.2	93.6	91.4	86.3	71.2	62.7	84.4	72.9	82.3	68.3	67.3	85.7	75.3
BitFit[1]	280k	85.3	90.1	93.0	94.2	90.9	86.8	67.6	58.2	83.3	74.5	79.6	70.0	59.6	78.6	72.5
LoRA[2]	3.8M	86.3	89.0	93.2	94.3	90.9	90.1	75.5	63.3	85.3	–	–	–	–	–	–
LST[2]	3.8M	85.6	88.8	93.3	94.1	90.7	90.4	71.9	58.1	84.1	–	–	–	–	–	–
PT[4]	76.8k	83.4	90.2	93.1	91.9	90.2	90.1	78.8	60.7	84.8	65.7	63.7	50.8	51.9	67.9	60.0
DEPT (ours)	76.8k	85.0	90.4	93.2	94.2	90.8	90.7	79.1	63.8	85.9	74.3	79.3	68.7	67.3	92.9	76.5
Multi-task Learning																
Fine-tuning (m) [1]	28M	85.7	91.1	92.0	92.5	88.8	90.2	75.4	54.9	83.8	–	–	–	–	–	–
Adapter (m) [1]	1.8M	86.3	90.5	93.2	93.0	89.9	90.2	70.3	61.5	84.4	–	–	–	–	–	–
HyperFormer (m) [1]	638k	85.7	90.0	93.0	94.0	89.7	87.2	75.4	63.7	84.8	–	–	–	–	–	–
HyperDecoder (m) [1]	1.8M	86.0	90.5	93.4	94.0	90.5	87.7	71.7	55.9	83.7	–	–	–	–	–	–
Single-Task Training + Transfer Learning																
SPoT[1]	76.8k	85.4	90.1	93.0	93.4	90.0	79.7	69.8	57.1	82.3	74.0	77.2	67.0	50.0	46.4	62.9
ATTEMPT[1]	232k	84.3	90.3	93.0	93.2	89.7	85.7	73.4	57.4	83.4	74.4	78.8	66.8	53.8	78.6	70.5
MPT[3]	77.6k	85.9	90.3	93.1	93.8	90.4	89.1	79.4	62.4	85.6	74.8	79.6	69.0	67.3	79.8	74.1
Multi-task Learning + Transfer Learning																
ATTEMPT (m) [3]	96k*	83.8	90.0	93.1	93.7	90.8	86.1	79.9	64.3	85.2	74.4	78.5	66.5	69.2	82.1	74.1
MPT (m) [3]	10.5k*	84.3	90.0	93.0	93.3	90.4	89.2	82.7	63.5	85.8	74.8	79.2	70.2	67.3	89.3	76.1

[1] from [1]. [2] from [41]. [3] from [48]. [4] we reproduce and substantially increase the performance of the vanilla PT reported in the prior work [1]. * These values are obtained after amortizing over 8 tasks, and the minimal number of parameters to perform a single task remains 232k and 77.6k for ATTEMPT and MPT. (m) represents additional multi-task training.

[47] benchmark, including MNLI [50], QQP[1], QNLI [32], SST-2 [39], STS-B [3], MRPC [7], RTE [10] and CoLA [49]; (2) SuperGLUE benchmark [46], including MultiRC [18], BoolQ [4], WiC [30], WSC [22], and CB [5]; (3) MRQA 2019 Shared Task [9], including Natural Questions [20], HotpotQA [51], SearchQA [8] and NewsQA [43]; (4) other datasets, including WinoGrande [34], Yelp-2 [52], SciTail [19] and PAWS-Wiki [53].

Baselines. We compare DEPT with a variety of baselines: (1) fine-tuning (FT), where all the model parameters are tuned during adaptation on each downstream task; (2) the vanilla PT [21], where target prompt vectors are initialized by randomly sampled top vocabularies, and its variants using additional transfer and multi-task learning, including SPoT [45], ATTEMPT [1], and MPT [48]; (3) state-of-the-art PEFT approaches including Adapters [14], AdapterDrop [33], BitFit [2], HyperFomer [17], HyperDecoder [16], P-tuning [27], LoRA [15], LST [41], and their multi-task learning variants. For a fair comparison, we directly quote performance metrics from published papers [28, 17, 1, 48, 41] for a fair comparison, where all these baselines using the T5-BASE as the backbone and adhere to the train, validation and test splits used by [17, 28] for NLP tasks.

Implementation details. In our study, we mainly experiment using the T5-BASE model with 220M parameters [31]. We consistently set the number of virtual tokens l as 100 across all tasks for the vanilla PT and adjust the hyper-parameters of DEPT accordingly to maintain the equivalent number of trainable parameters. For instance, the vanilla PT contains $l \times d$ trainable parameters where the hidden size d is 768 for the T5-BASE, and DEPT can configure the number of virtual tokens m as 40 and the rank of low matrices r as 45, resulting in $m \times d + (s + d) \times r$ trainable parameters. This yields a total of $76,800$ trainable parameters, aligning with the vanilla PT.

We also extend our evaluation to include T5-SMALL (60M), T5-LARGE (770M), GPT2-SMALL (110M), GPT2-MEDIUM (345M), and GPT2-LARGE (774M) models.

[1] https://www.quora.com/q/quoradata/

C Model performance against the parameter-efficiency

We visualize the experimental results in Table 3, as shown in Figure 3. The visualization shows that our proposed method DEPT outperforms other PEFT approaches and full fine-tuning baselines on the GLUE and SuperGLUE benchmark (y-axis) while updating only a small number of trainable parameters (x-axis).

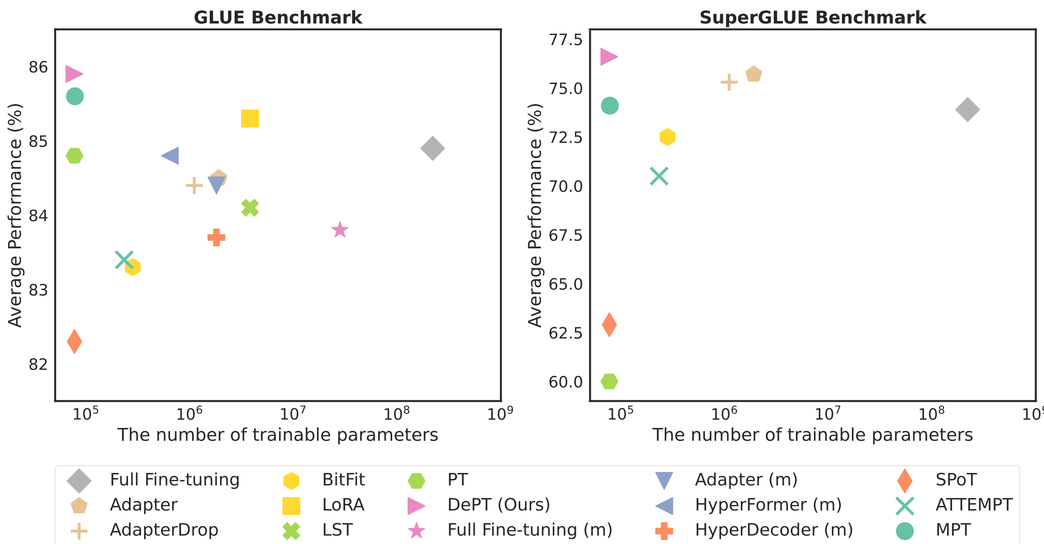

Figure 3: The average performance against the number of trainable parameters on the GLUE and SuperGLUE benchmark using the T5-BASE model.

D Dataset

In this work, we use 23 popular datasets from previous few-shot learning and PEFT research. We limit the maximum training data number of Yelp-2 to 100k samples. We train MNLI with longer steps, 200k steps in total. For the GLUE dataset, we use the HuggingFace dataset[2]. For the Super GLUE dataset, we use the HuggingFace dataset[3]. For MRQA 2019 Shared Task and other datasets, we use the HuggingFace dataset[4].

E Implementation Details

Our code is implemented using Pytorch[5], Huggingface Transformers[6], and Huggingface PEFT[7]. Below, we provide a comprehensive list of the hyperparameters used in our code. For prompt tuning and DEPT, as shown in Table 5, we conduct a grid search for learning rates. For the soft prompt, we search the learning rate within the set {3e-1, 4e-1, 5e-1}. For the low-rank matrice pairs, we search the learning rate within the set {1e-04, 5e-4, 1e-03}. We choose a batch size of 16. We typically use the max sequence length as 256 except the SuperGLUE-MultiRC where the max sequence length is set as 348. In each trial, we train the model for 30,000 steps, evaluate performance every 1,000 steps, and select the best checkpoint based on optimal performance on the evaluation set. For the large dataset with more than 100,000 training example, we follow the prior work [45] to train the vanilla PT and our proposed method DEPT with up to 300,000 steps. Training more steps is helpful for improving the performance of the vanilla PT for the large dataset. The best performance is

[2] https://huggingface.co/datasets/glue
[3] https://huggingface.co/datasets/super_glue
[4] https://huggingface.co/lucadiliello
[5] https://pytorch.org/
[6] https://github.com/huggingface/transformers
[7] https://github.com/huggingface/peft

GLUE Benchmark						
Dataset	**Source**	**Target**	**#Train**	**#Valid**	**#Test**	**Type**
MNLI	31.8	1.0	392,702	9,832	9,815	NLI
QQP	24.1	1.0	362,846	1,000	40,431	Paraphrase
QNLI	38.4	1.0	103,743	1,000	5,463	NLI
SST-2	10.4	1.0	66,349	1,000	872	Sentiment
STS-B	21.9	1.0	5,749	750	750	Sent. Similarity
MRPC	45.9	1.0	3,668	204	204	Paraphrase
RTE	54.4	1.0	2,490	138	139	NLI
CoLA	8.7	1.0	8,551	521	522	Acceptability
SuperGLUE Benchmark						
Dataset	**Source**	**Target**	**#Train**	**#Valid**	**#Test**	**Type**
MultiRC	286.1	1.0	27,243	2,424	2,424	Question Answering
BoolQ	108.3	1.0	9,427	1,635	1,635	Question Answering
WiC	18.4	1.0	5,428	319	319	Word Sense Disambiguation
WSC	28.1	1.0	554	52	52	Common Sense Reasoning
CB	64.6	1.0	250	28	28	NLI
ReCoRD	210.7	1.5	137,484	1,370	15,176	Common Sense Reasoning
MRQA 2019 Shared Task						
Dataset	**Source**	**Target**	**#Train**	**#Valid**	**#Test**	**Type**
NaturalQuestions	242.7	4.5	103,071	1,000	12836	Question Answering
HotpotQA	225.7	2.6	71,928	1,000	5,901	Question Answering
SearchQA	942.8	2.0	116,384	1,000	16,980	Question Answering
NewsQA	615.5	5.1	73,160	1,000	4,212	Question Answering
Other Datasets						
Dataset	Source	Target	**#Train**	**#Valid**	**#Test**	**Type**
WinoGrande	23.8	1.0	39,398	1,000	1,267	Common Sense Reasoning
YelpPolarity	134.0	1.0	100,000	1,000	38,000	Sentiment
SciTail	30.8	1.0	23,596	652	652	NLI
PAWS	44.7	1.0	4,9401	8,000	8,000	Sent. Similarity

Table 4: The datasets evaluated in this work. Source indicates the average length of the source sentences in the training set. Target indicates the average length of the target sentences in the training set. STS-B is a real-valued regression task over the interval $[0, 5]$). Note that we only sample examples from the original training set in our few-shot experiments.

determined by the relevant evaluation metric. We train the T5 model from the original checkpoint rather than the LM-adapted 1.1 version [21].

Hyperparameter	**Assignment**
number of steps	30,000 steps (evaluate every 1,000 steps)
batch size	16
maximum learning rate (α_1)	3e-1, 4e-1, 5e-1
maximum learning rate (α_2)	1e-04, 5e-4, 1e-03
length of the soft prompt (m)	20, 40, 60, 80
maximum sequence length	256
learning rate optimizer	AdamW
Adam epsilon	1e-6
Adam beta weights	0.9, 0.98
learning rate scheduler	Warmup linear
Weight decay	0.01
Warmup proportion	0.06

Table 5: Hyperparameters for Prompt Tuning and DEPT.