Structured Prompt Tuning

Chi-Liang Liu† Hung-yi Lee† Wen-tau Yih‡
†National Taiwan University ‡Meta AI
{r07942083, hungyilee}@ntu.edu.tw scottyih@fb.com

Abstract

We propose structured prompt tuning, a simple and effective method to improve prompt tuning. Instead of prepending a sequence of tunable embeddings to the input, we generate the soft prompt embeddings through a hypernetwork. Our approach subsumes the standard prompt tuning, allows more flexibility in model design and can be applied to both single-task and multi-task training settings. Empirically, structured prompt tuning shows a gain of +1.2∼1.5 points on the GLUE benchmark and is less sensitive to the change of learning rate, compared to standard prompt tuning.

1 Introduction

Prompting large language models has been shown an effective way to avoid expensive full model fine-tuning on a wide range of downstream NLP tasks. This approach prepends either natural language texts or continuous embeddings to the input. For example, Brown et al. (2020) demonstrated that pretrained language models are able to do in-context learning, where the model adapts to a new task simply by prepending a few training examples. In contrast to prompting with textual tokens, soft prompt tuning (Qin and Eisner, 2021; Li and Liang, 2021; Lester et al., 2021), which prepends a few tunable embeddings to the inputs, has also been proposed.

Despite its effectiveness, soft prompt tuning is fundamentally limited in its form of interacting with the language models. Variants of this approach usually differ only in where the soft prompt tokens are placed. For example, Lester et al. (2021) put the soft prompt tokens before the input; Hambardzumyan et al. (2021) insert prompt tokens before, between and after the sentences; Zhong et al. (2021) concatenate soft prompt tokens after the sentence. Adding soft prompt tokens in different Transformer layers has also been attempted (Qin and Eisner, 2021; Li and Liang, 2021).

In this paper, we propose structured prompt tuning, where the soft prompts are generated by a hypernetwork that takes as input a task embedding, as illustrated in Figure 1. Structured prompt tuning generalizes the standard soft prompt tuning method in that with a particular hypernetwork architecture, structured prompt tuning is in fact equivalent to soft prompt tuning. Perhaps more importantly, structured prompt tuning provides additional flexibility in model design, as different hypernetwork architectures impose implicit structures among soft prompt embeddings. Such flexibility can be crucial in better adapting the model to the target task.

With simple hypernetwork architectures, such as linear layer, low rank linear layer or a multilayer perceptron, our structured prompt tuning consistently outperforms standard prompt tuning on various NLU tasks, such as question answering and sentiment classification in the single-task training setting. For the multi-task training setting, our proposed method surpasses not only standard prompt tuning but also the full fine-tuning approach. We additionally find that structured prompt tuning is less sensitive to learning rate change, compared to the standard prompt tuning.
2 Related Work

Transfer learning by adapting large pre-trained language models (Devlin et al., 2019; Liu et al., 2019; Lewis et al., 2020; Raffel et al., 2020) to specific tasks has been the standard approach in NLP. While full fine-tuning is the common adaptation method, it requires storing and updating all model parameters. As a result, more efficient alternatives have been proposed by researchers.

Brown et al. (2020) demonstrated that PLMs can achieve surprising results on downstream tasks by prepending a few training examples to the model input without tuning the model parameters. This method has been referred to as prompting or in-context learning. However, the performance is sensitive to its prompts (Brown et al., 2020; Schick and Schütze, 2021a,b; Shin et al., 2020), which triggers research on designing or learning appropriate prompts. For instance, AutoPrompt (Shin et al., 2020) leverages gradient-guided search over the large discrete space of phrases to construct a prompt. Gao et al. (2021) use T5 (Raffel et al., 2020) to generate template candidates to obtain high-quality prompts guided by the development set performance. Instead of optimizing in the discrete tokens space, “soft prompt” methods that replace discrete tokens with randomly initialized continuous embeddings and update them by gradient descent have been studied. For example, Zhong et al. (2021) and Qin and Eisner (2021) explore the possibility of using cloze prompts to query pre-trained language models for single-word answers. Li and Liang (2021) and Lester et al. (2021) extend the idea to generation tasks and show that soft-prompt tuning is competitive to full model fine-tuning. Concurrent to our work, Levine et al. (2022) introduce input-dependent prompt tuning that also generates prompt tokens using a generator. In contrast to their approach, our generator takes as input a task embedding instead of input tokens.

3 Structured Prompt Tuning

As illustrated in Figure 1, our proposed structured prompt tuning approach is similar to soft prompt in that the origin input is prepended with \( n \) token embeddings. However, these embeddings are not trained directly, but are instead generated by a hypernetwork that takes a task embedding as input. The tuning procedure updates the hypernetwork parameters and the task embedding. We describe our method formally in this section.

3.1 Method

Following T5 (Raffel et al., 2020) and prompt tuning (Lester et al., 2021), we reduce any downstream task to a seq2seq text generation task. Given a series of input tokens \( X \) and a sequence of tokens \( Y \) that represents a class label for classification tasks or a ground truth sequence for generation tasks, our goal is to maximize the conditional probability \( P_{LM}(Y|X) \), where \( P_{LM} \) is a language model.

Prompt Tuning Following Lester et al. (2021), we prepend \( n \) prompt representations \( r_1 \ldots r_n \) to the input, where \( r_i \in \mathbb{R}^d \) and \( d \) is dimension of the embeddings. The new conditional probability is thus calculated by \( P_{LM}(Y|r_1 \ldots r_n, X) \). Notice that unlike full model fine-tuning, the parameters in the original model, \( P_{LM} \), are frozen. Only the prompt representations \( r_1 \ldots r_n \) are tunable.

Structured Prompt Tuning Standard prompt tuning learns the prompt representations \( r_1 \ldots r_n \) directly. In contrast, we generate these representations via a hypernetwork \( H \):

\[
R = H(e_t),
\]

where \( R = [r_1; r_2; \ldots; r_n] \in \mathbb{R}^{n \times d} \) is the prompt representations and \( e_t \in \mathbb{R}^k \) is the task embedding for task \( t \). The hypernetwork \( H \) and the task embedding \( e_t \) are the only trainable parameters. The design of task embedding \( e_t \) makes our prompt tuning method applicable to both single- and multi-task learning scenarios. When there are more than one task to be considered, each task is represented by a different embedding \( e_t \), but \( H \) is shared by all the tasks. Note that standard prompt tuning is a special case of structure prompt tuning with a \( 1 \times nd \) matrix as the hypernetwork, and \( k = 1 \) for the task embedding dimension.

3.2 Hypernetwork Architecture

We consider three different architectures for the hypernetwork. To simplify the description below, we introduce a matricization\(^1\) notation \( M : \mathbb{R}^{nd} \to \mathbb{R}^{n \times d} \), which constructs a matrix from a vector.

Linear We linearly project the shared vector to different representation subspaces.

\[
H(e_t) = M(We_t + b) \tag{2}
\]

where \( W \in \mathbb{R}^{nd \times k} \) and \( b \in \mathbb{R}^{nd} \).

\(^1\text{Equivalent to \text{reshape(vector, (batch_size, n_tokens, dimension)) in numpy.}}\)
Low-Rank Linear To strengthen the relationship between generated prompts, we introduce a low-rank constraint to the hypernetwork: matrix $W$ is derived via low-rank factorization.

$$H(e_t) = M(W e_t + b) = M(C F e_t + b), \quad (3)$$

where $C \in \mathbb{R}^{nd \times r}, F \in \mathbb{R}^{r \times k}$ and $r$ is a hyperparameter that specifies the rank of $W$.

Multilayer Perceptron (MLP) We also try an MLP model that consists of two linear transformations with a GeLU activation $\phi$ in between.

$$H(e_t) = M(W_2 \phi(W_1 e_t + b_1) + b_2), \quad (4)$$

where $W_1 \in \mathbb{R}^{h \times k}, W_2 \in \mathbb{R}^{nd \times h}, b_1 \in \mathbb{R}^{h}$ and $b_2 \in \mathbb{R}^{nd}$.

4 Experiments

4.1 Datasets We evaluate our models and baselines, such as standard soft prompt tuning and full fine-tuning, on the GLUE (Wang et al., 2019) and SQuAD 1.1 (Rajpurkar et al., 2016) benchmarks. GLUE covers multiple tasks such as paraphrase detection (MRPC, QQP), sentiment classification (SST-2) and natural language inference (MNLI, RTE, QNLI). SQuAD is an extractive question answering dataset, where the answer to an input question is a span in the given context. Since GLUE and SQuAD do not offer publicly available test sets, for small datasets (less than 10,000 samples), we use half of the validation data as the development set and the other half as the test data; for larger datasets, we keep the original validation set as the test set and sample 1,000 examples from the training set for validation.

4.2 Pre-trained Language Model Following Lester et al. (2021), we use T5 (Raffel et al., 2020) for the PLM. We primarily use the adapted version of T5-v1.1\textsuperscript{2} small and base with 60M, 220M parameters, respectively as our frozen PLM. While we do not apply our method to other varying sizes (Large, XL, XXL) due to the computing resource constraint, the experiments can be easily extended to other sizes. For a fair comparison, the full fine-tuning method is also updated from the same adapted version of T5-v1.1.

4.3 Training Details We follow most of the hyperparameters and details in (Lester et al., 2021) except the optimizer, where we use AdamW (Loshchilov and Hutter, 2019) instead of Adafactor (Shazeer and Stern, 2018). Although Adafactor reduces the memory requirements for the square gradients from square to sublinear, it introduces new hyperparameters: decay rate, scale parameter and relative steps. To reduce the hyperparameter search space, we use AdamW and do not observe significant performance differences. All of our models are reimplemented with

\textsuperscript{2}T5 v1.1 LM adapted models are initialized from T5 v1.1 and then trained for 100K additional steps using the LM objective. The checkpoints are available at https://github.com/google-research/text-to-text-transfer-transformer/blob/main/released_checkpoints.md
Table 1 presents the main results, the model per-

transformers library (Wolf et al., 2020). During

multi-task training results on GLUE. Table 2:

fine-tuning on multiple tasks and has a higher aver-

views much better than the standard prompt tuning.

prompt tuning (the low rank variant) generally per-

vtee, regardless of which task an example

contrast, the standard prompt tuning treats all exam-

is the hypernetwork, while allowing some flexi-

structured prompt tuning, the hypernetwork

sampled from the training set of each task. For

multi-task learning setting on GLUE. Each batch

Multi-Task Learning

and full finetuning, our structured prompt tuning

GLUE. Although there are still some significant

+1

method in almost every setting, with

sustains outperforms the standard prompt tuning

Structured Prompt Tuning (denoted as SPT) con-

Performance on the GLUE and SQuAD benchmarks.

Figure 2: The box plot of prompt tuning with differ-

ent learning rates (0.01, 0.1, 1). Structured prompt tuning

(low-rank matrix variant) is much less sensitive to

learning rate than the standard prompt tuning. Each

learning rate has 3 runs with different random seeds.

4.4 Results

Table 1 presents the main results, the model perfor-

ance on the GLUE and SQuAD benchmarks. Structured

Prompt Tuning (denoted as SPT) consistently outperforms

the standard prompt tuning method in almost every setting, with +1.5/+1.2-

point gains with T5-v1.1 small/base in average on

GLUE. Although there are still some significant performance differences between prompt tuning

and full finetuning, our structured prompt tuning methods have successfully reduced the gap.

Multi-Task Learning

We also compare structured prompt tuning with other baselines in the

multi-task learning setting on GLUE. Each batch

consists of roughly the same number of examples

sampled from the training set of each task. For

structured prompt tuning, the hypernetwork \( H \) is

shared across all tasks but each task has its own

unique task embedding \( e_t \). This design aims for

preserving the shared knowledge of multiple tasks in

the hypernetwork, while allowing some flexi-

bility for adapting to specific tasks through \( e_t \).

In contrast, the standard prompt tuning treats all exam-

ples the same, regardless of which task an example

belongs to. As shown in Table 2, our structured

prompt tuning (the low rank variant) generally per-

forms much better than the standard prompt tuning.

Perhaps more surprisingly, it also outperforms full

fine-tuning on multiple tasks and has a higher aver-

age score overall.

Learning Rate Sensitivity

Li and Liang (2021) point out that prompt tuning is very sensitive to

the change of learning rate. However, we do not observe similar issues on structured prompt tuning.

To show this, we test three different learning rates (0.01, 0.1, 1) for both the standard and our

structured prompt tuning (with the low-rank matrix hypernetwork) methods on SST-2 and SQuAD,

where each learning rate has 3 runs with different random seeds. As can be seen clearly in Figure 2,

the variance of standard prompt tuning is quite large, while different learning rates have almost no

effect on structured prompt tuning. We leave more analysis on this phenomenon for future work.

5 Conclusion

In this work, we present structured prompt tuning, strengthening the relationship between soft

tokens via generating them with task embedding. To show this, we test three different learning

rates (0.01, 0.1, 1). Structured prompt tuning performs both standard prompt tuning and the full

fine-tuning method. We also found that structured
prompt tuning is much less sensitive to the learning rate change.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.

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.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. 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 3816–3830, Online. Association for Computational Linguistics.

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. WARP: Word-level Adversarial ReProgramming. 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 4921–4933, Online. 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.

Yoav Levine, Itay Dalmedigos, Ori Ram, Yoel Zeldes, Daniel Jannai, Dor Muhlgay, Yoni Osin, Opher Lieber, Barak Lenz, Shai Shalev-Shwartz, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2022. Standing on the shoulders of giant frozen language models.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7871–7880, Online. Association for Computational Linguistics.

Xiang Li 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.

Yinhan Liu, Myle Ot, 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. 2019. Decoupled weight decay regularization. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pages 8024–8035.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5203–5212, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. 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. Association for Computational Linguistics.
Timo Schick and Hinrich Schütze. 2021a. 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, pages 255–269, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021b. It’s not just size that matters: Small language models are also few-shot learners. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2339–2352, Online. Association for Computational Linguistics.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, volume 80 of Proceedings of Machine Learning Research, pages 4603–4611. PMLR.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 4222–4235, Online. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net.

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.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [MASK]: Learning vs. learning to recall. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5017–5033, Online. Association for Computational Linguistics.