Efficient and Effective Prompt Tuning via Prompt Decomposition and Compressed Outer Product

Pengxiang Lan[1]*, Haoyu Xu[1]*, Enneng Yang[1], Yuliang Liang[1], Guibing Guo[1]†, Jianzhe Zhao[1], Xingwei Wang[2]
[1]Software College, Northeastern University, China,
[2]School of Computer Science and Engineering, Northeastern University, China
{pengxianglan, haoyuxu, ennengyang, liangyuliang}@stumail.neu.edu.cn,
{guogb, zhaojz}@swc.neu.edu.cn, wangxw@mail.neu.edu.cn

Abstract

Prompt tuning (PT) offers a cost-effective alternative to fine-tuning large-scale pre-trained language models (PLMs), requiring only a few parameters in soft prompt tokens added before the input text. However, existing PT approaches face two significant issues: (**i**) They overlook intrinsic semantic associations between soft prompt tokens, leading to high discreteness and limited interactions, thus reducing the model's comprehension and effectiveness in complex tasks. (**ii**) Due to the complexity of downstream tasks, long soft prompt is necessitated to improve performance, but prompt length correlates positively with memory usage and computational costs. Achieving high efficiency and performance remains an ongoing challenge. To address these issues, we propose a novel **L**ow-p**A**ra**M**eters **P**rompt Tuning (**LAMP**) method, which leverages prompt decomposition and compressed outer product. Specifically, the prompt decomposition module employs Truncated SVD to reduce training parameters and significantly lower the dimensionality of the soft prompt parameter space. It then utilizes a compressed outer product module to facilitate multiple interactions among prompt tokens, exploring their intrinsic associations to enhance knowledge representation. Finally, LAMP uses average pooling to reduce memory usage and training/inference time. Extensive experiments across six architectures and eight datasets demonstrate that LAMP outperforms state-of-the-art PT-based and LoRA-based methods in performance and efficiency.

1 Introduction

Pre-trained language models (PLMs) possess powerful learning capabilities to extract complex features and patterns from vast amounts of data (Devlin et al., 2019; Radford et al., 2019). In recent years, as the scale of large PLMs has rapidly expanded, computational costs have surged dramatically (Lan et al., 2024). Although full fine-tuning parameters of PLMs yields satisfactory results, it has become impractical, e.g., PaLM has 540B parameters and requires 6144 TPU v4 chips to train for 1200 hours (Chowdhery et al., 2023).

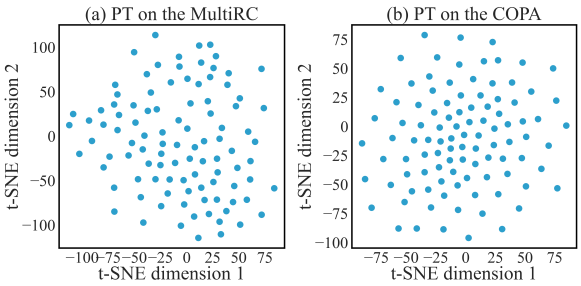

Figure 1: (a) and (b) show the t-SNE clustering visualizations of the original prompt tuning prompts after training on the MultiRC and COPA datasets using the T5-Base model. Source prompt tokens are initialized from sampled vocabulary and length is set to 100.

Parameter-Efficient Fine-Tuning (PEFT) methods attempt to bridge this gap by achieving performance comparable to full fine-tuning with minimal computational resources and time costs (Houlsby et al., 2019; Hu et al., 2021; Lester et al., 2021). Among these methods, prompt tuning (PT) stands out for its efficiency and flexibility. It freezes the model parameters and exclusively trains the soft prompt tokens attached to the model's input, delivering performance on par with full fine-tuning (Lester et al., 2021; Xiao et al., 2023; Razdaibiedina et al., 2023; Lan et al., 2024). Notably, PT's trainable parameters are much lower than other PEFT methods (e.g., Adapter (Houlsby et al., 2019) and LoRA (Hu et al., 2021)) and do not grow exponentially as the scale of PLMs.

Although these methods provide undeniable contributions, existing PT-based methods still suffer

*The first two authors contributed equally.
†Corresponding author.

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 4406–4421
April 29 - May 4, 2025

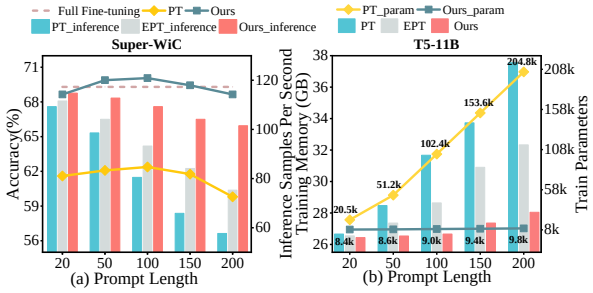

Figure 2: (a) Average performance on the T5 models across the SuperGLUE benchmark. (b) Impact of prompt length on performance and trainable parameters in the WiC dataset of the SuperGLUE benchmark.

from two main challenges: **First**, in various tasks, PT initialization methods exhibit high discreteness, lacking exploration of the intrinsic semantic associations between tokens. The two mainstream prompt initialization methods—random initialization and initialization from sampled vocabulary (i.e., the 5,000 most common tokens)—help the model explore a broader parameter space during training. The sample vocabulary initialization, in particular, is widely utilized in pre-training for transfer learning (Vu et al., 2022; Asai et al., 2022) and multi-task learning (Wang et al., 2022; Zhong et al., 2024) due to its more informative nature. Nevertheless, as shown in Figure 1, we found that although this informative initialization provides prompt tokens with rich knowledge, these tokens remain isolated without establishing semantic connections after training. In NLP tasks, intrinsic semantic associations form the context and meaning of language, helping models better understand and represent complex language structures (Mikolov et al., 2013; Devlin et al., 2019; Vaswani et al., 2017). Prompt tokens leverage semantic knowledge to guide PLMs in producing outputs that better meet task requirements. Clearly, this intrinsic semantic association is essential for PT. This high discreteness overlooks capturing intrinsic semantic associations among soft prompt tokens, limiting the model's knowledge representation capabilities. **Second**, although PT does not require training the parameters of PLMs, adding soft prompts increases the total length of input embeddings. Figure 2 reveals the relationship between computational cost, trainable parameters, and prompt length. Previous research has shown that a long prompt 100 yields optimal PT performance, enabling PLMs to adapt to complex downstream tasks (Lester et al., 2021; Razdaibiedina et al., 2023; Xiao et al., 2023). However, such a length renders PT inefficient. This inefficiency stems from the inherent high complexity of PLMs (e.g., the quadratic complexity of Transformers) (Vaswani et al., 2017) and the fact that the storage of gradients and optimizer states is closely related to the number of trainable parameters (Guo et al., 2021). While some approaches (Xiao et al., 2023; Shi and Lipani, 2024; Lan et al., 2024) attempt to optimize standard PT, they still struggle with inefficiency and suboptimal performance.

To address the aforementioned challenges, we propose a novel efficient and effective low-parameters prompt tuning (**LAMP**) method through prompt decomposition and compressed outer product. Our motivation stems from the soft prompt exhibiting high dispersion and the "intrinsic rank" (Aghajanyan et al., 2021; Hu et al., 2021) in PEFT, which indicates that model fine-tuning can occur in a low intrinsic dimensionality space. Specifically, LAMP first employs Truncated singular value decomposition (SVD) with its inherent structure—two low-dimensional singular vectors and singular values—to transform the loosely related semantic knowledge in PT tokens into a more structured and interrelated form, while simultaneously reducing trainable parameters. It then aggregates the semantic knowledge from the Truncated SVD and leverages the compressed outer product to enable multi-level interactions of intrinsic semantics, enhancing the model's knowledge representation. Finally, LAMP reduces computational load by applying average pooling, which does not increase training parameters. Figure 2 demonstrates that the longer the prompt length, the more significant the reduction in computational cost and memory usage achieved by LAMP.

The main contributions of this paper are:

- Our empirical study reveals that tokens in soft prompts exhibit high dispersion during training, lacking inherent semantic interactions among tokens to assist the model in comprehending and handling complex tasks, thereby limiting the model's knowledge representation capability.
- We propose a novel low-parameter prompt tuning (abbreviated as LAMP) method that captures potential semantic interactions between prompt tokens through prompt decomposition and compressed outer product. LAMP achieves robust performance while significantly reducing computational costs (e.g., training time, memory usage, and trainable parameters).

- We comprehensively evaluated LAMP on the SuperGLUE benchmark. Experimental results demonstrate that LAMP outperforms other state-of-the-art PT methods and remains effective in few-shot scenarios. Notably, on the T5-11B model, LAMP improved performance by 5.59% compared to the vanilla PT while also increasing inference speed by 31%, reducing trainable parameters by 91.21%, shortening training time by 23.64%, and lowering memory usage by 24.49%.

2 Method

2.1 Preliminaries

Prompt Tuning. PT can maintain parameter efficiency as model size scales. This approach ensures that trainable parameters does not increase dramatically with model expansion, making it a preferred choice for many applications (Shi and Lipani, 2024; Lan et al., 2024). Let labelled training data $(\boldsymbol{X}, \boldsymbol{Y}) = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N}$ for one target task $\mathcal{T}$, the number of training data is N. The total parameters of PLM is Θ and each input text is $\boldsymbol{x}_i$. The embedding of $\boldsymbol{x}_i$ is represented as $\mathbf{E}_i \in \mathbb{R}^{m \times d}$, where m is maximum sequence length and d is the dimension of input embedding. The target prompt $\mathbf{P} \in \mathbb{R}^{l \times d}$ is initialized, with l as the hyper-parameter determining the length of the soft prompt. This prompt is then concatenated with the fixed embedding $\mathbf{E}_i \in \mathbb{R}^{m \times d}$. $\mathbf{E}_i$ remains unchanged during gradient updates in training, resulting in a new input embedding $[\mathbf{P}; \mathbf{E}_i] \in \mathbb{R}^{(l+m) \times d}$. The formulation for the target task is as follows:

$$\mathcal{L}_p = -\sum_i \log P\left(\mathbf{y}_i | \left[\mathbf{P}; \mathbf{E}_i\right]; \Theta\right) \quad (1)$$

where $\mathcal{L}_p$ is a loss function only optimized with the prompt $\mathbf{P}$. $P(\cdot)$ is maximizing the conditional probability of PT. The overall structure of PT is shown in Figure 3(a).

2.2 LAMP: Low-parameters Prompt Tuning

Prompt initialization. We initialize the source soft prompt $\mathbf{P}$ from sampled vocabulary (Lester et al., 2021; Vu et al., 2022; Razdaibiedina et al., 2023; Asai et al., 2022), a strategy that embeds more semantic richness and prior knowledge than mere random initialization. Inadequate initialization can result in the discovery of suboptimal local minima, thus impairing the model's generalization capabilities.

Prompt decomposition. Our innovative motivation stems from two aspects: (1) Soft prompt tokens exhibit high dispersion (as shown in Figure 1), neglecting the knowledge interactions between tokens; (2) Soft prompt also exhibit low "intrinsic rank" behavior (Xiao et al., 2023). Inspired by these findings and the core idea of Truncated SVD (Hansen, 1987), we employ Truncated SVD to reduce training parameters by decomposing the soft prompt $\mathbf{P} \in \mathbb{R}^{l \times d}$. The original SVD of $\mathbf{P}$ is formulated as follows:

$$\mathbf{P} = \mathbf{U}\text{diag}(\mathbf{Q})\mathbf{V}^\top \quad (2)$$

where $\mathbf{U} \in \mathbb{R}^{l \times min(l,d)}$, $\mathbf{V} \in \mathbb{R}^{d \times min(l,d)}$ are the singular vectors with orthogonal columns, $\mathbf{U}$ and $\mathbf{V}$ transforms highly dispersed tokens from the original PT into interrelated representations. $\mathbf{Q} \in \mathbb{R}^{min(l,d)}$ comprises the singular values arranged in descending order (the larger the singular value, the more information it contains). The operation $\text{diag}(\mathbf{Q})$ converts $\mathbf{Q}$ into a diagonal matrix, and $V^\top$ represents the transpose of $\mathbf{V}$.

Prompt reconstruction. We define the soft prompt's low "intrinsic rank" as r and select the top-r singular values, $\mathbf{Q}_{[:r]}$, which contain a rich amount of information arranged in descending order. The remaining singular values, $\mathbf{Q}_{[r:]}$, are discarded and do not participate in training/inference. Consequently, as the $\mathbf{Q}$ dimension changes, singular values and vectors are redefined as $\left\{\mathbf{U}_{[:r]} \in \mathbb{R}^{l \times r}, \mathbf{Q}_{[:r]} \in \mathbb{R}^{r}, \mathbf{V}_{[:r]} \in \mathbb{R}^{d \times r}\right\}$. We can approximate the original information by storing only r (where $r \ll d$) singular values and their corresponding low-parameters singular vectors, achieving parameter compression, which is also why LAMP adopts Truncated SVD.

The trainable parameters are now reduced from '$l \times d$' to '$l \times r + r + r \times d$'. In experiments, we provide a detailed explanation of hyperparameter r and its impact on model performance. For instance, on the Llama2-7B (Touvron et al., 2023), when the prompt length increases from 100 to 500, traditional PT requires training 2,048k parameters, whereas our method, LAMP, only requires training ($500 \times 8 + 8 + 8 \times 4096 = 36.8$k) parameters. LAMP's advantage becomes more pronounced with longer prompt lengths or larger model scales, as it significantly reduces the computational cost by decreasing trainable parameters. LAMP has established a solid foundation for PT-based methods to excel across various domains.

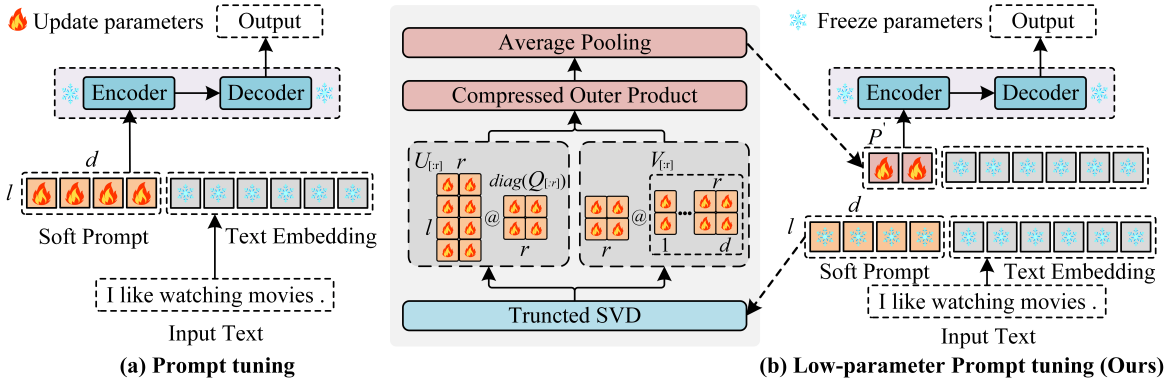

Figure 3: (a) Conventional prompt tuning (Lester et al., 2021). (b) The overview of our proposed LAMP. It decomposes the vanilla prompt to construct a new low-dimensional prompt, captures the intrinsic semantic associations between prompt tokens, and finally reduces computational costs through average pooling.

Compressed outer product. Although using Truncated SVD to reduce the trainable parameters in prompt tuning is promising, directly applying dot products on the decomposed singular values and vectors can only partially capture the intrinsic associations between tokens. Specifically, dot products' inherent linear nature limits their ability to fully express the more complex, nonlinear interactions among prompt tokens. Considering that outer products can mine richer and more complex high-order interactions, we utilize the compressed outer product to further explore the intrinsic semantic associations between tokens in prompt tuning.

Firstly, we utilize the dot product of singular values and singular vectors as the initialization input for the compressed outer product module:

$$\mathbf{M} = \mathbf{U}_{[:r]} \operatorname{diag}\left(\mathbf{Q}_{[:r]}\right) \in \mathbb{R}^{l \times r}, \tag{3}$$

$$\mathbf{I} = \operatorname{diag}\left(\mathbf{Q}_{[:r]}\right) \mathbf{V}_{[:r]}^{\top} \in \mathbb{R}^{r \times d}. \tag{4}$$

This approach enables an initial aggregation of knowledge features between tokens, which helps to effectively represent and explore the underlying semantic knowledge associations. Due to compressed outer product can maintains the high-order structure while facilitating multiple layers of intrinsic semantic interactions. This approach effectively restores and enhances the complex information structures that might be lost in Truncated SVD, thereby enriching the knowledge representation capabilities of prompt tokens. The compressed outer product is formulated as follows:

$$\mathbf{C} = \sum_{i=1}^{r} \mathbf{M}_{[:,i]} \otimes \mathbf{I}_{[i,:]} \in \mathbb{R}^{l \times d} \tag{5}$$

where $\mathbf{M}_{[:,i]}$ is the i-th column vector of $\mathbf{M}$, $\mathbf{I}_{[i,:]}$ is the i-th row vector of $\mathbf{I}$, $\otimes$ denotes the outer product of two vectors. $\mathbf{C} \in \mathbb{R}^{l \times d}$ is the resultant prompt after summing all the outer products. The introduction of compressed outer products enhances the representational power of prompt tuning. This approach enables the soft prompt to more effectively adapt to different downstream tasks through deep interactions between prompt tokens.

While introducing compressed outer product does not create new trainable parameters, it does entail a slight increase in computational overhead due to its engagement in more complex higher-order interactions. Additionally, given the transformers' quadratic complexity, the prompt's length is proportional to the training duration. We consider employing average pooling operation to reduce training time:

$$\mathbf{P'}_{i,j} = \frac{1}{p} \sum_{k=0}^{p-1} \mathbf{C}_{i*p+k,j} \tag{6}$$

$\mathbf{P'}_{i,j}$ represents the elements of the tensor $\mathbf{P'} \in \mathbb{R}^{l/p \times d}$ after averaging pooling. This operation effectively compresses the l elements along the first dimension into l/p. It is also noteworthy that we explored a self-attention pooling strategy to dynamically filter prompt tokens in Appendix A.1; however, this strategy was not very effective and introduced additional trainable parameters.

2.3 Training and Inference

Only the parameters of $\mathbf{U}_{[:r]} \in \mathbb{R}^{l \times r}$, $\mathbf{Q}_{[:r]} \in \mathbb{R}^{r}$, and $\mathbf{V}_{[:r]} \in \mathbb{R}^{d \times r}$ are optimized during the training process, while the backbone model (i.e., Θ and $\mathbf{E}_i$) remained frozen as Figure 3(b). The reconstructed prompt $\mathbf{P'}$ is inserted before the input text

embeddings. By $\mathbf{P}'$, Eq.1 is displaced by:

$$\mathcal{L}_{PT} = -\sum_{i} \log P(\mathbf{y}_i|[\mathbf{P}';\mathbf{E}_i];\Theta) \quad (7)$$

where $[\mathbf{P}';\mathbf{E}_i] \in \mathbb{R}^{(l/p+m)\times d}$ is a input embedding of PLMs through the connection of $\mathbf{P}'$ and $\mathbf{E}_i$.

3 Experiments

In this section, we will answer these key research questions by conduct extensive experiments: **RQ1:** How does our LAMP performance compare with other SOTA baselines across different model scales and datasets? **RQ2:** How do few-shot adaptability and hyper-parameters optimization influence the LAMP? **RQ3:** How will the feature space of the soft prompt change after considering the intrinsic semantic associations between tokens?

3.1 Evaluation Datasets and Metrics

Evaluation Datasets: Building upon prior studies in prompt tuning (Xiao et al., 2023), we employ eight NLP tasks from the SuperGLUE (Wang et al., 2019) and GLUE (Wang et al., 2018) benchmark and conduct multi-aspect experiments to evaluate the high efficiency and effectiveness of LAMP. The SuperGLUE benchmark includes more complex and challenging tasks among eight datasets than GLUE (Wang et al., 2018): CB (De Marneffe et al., 2019), WSC (Levesque et al., 2012), COPA (Roemmele et al., 2011), RTE (Giampiccolo et al., 2007), WiC (Pilehvar and Camacho-Collados, 2019), BoolQ (Clark et al., 2019), MultiRC (Khashabi et al., 2018) and ReCoRD (Zhang et al., 2018). MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), SST-2 (Socher et al., 2013) and MRPC(Bill, 2005) in GLUE benchmark. More details about datasets in Appendix A.2.

Metrics: Consistent with previous work (Razdaibiedina et al., 2023; Xiao et al., 2023), the evaluation metric is F1 for MultiRC and ReCoRD, the evaluation metric is Accuracy for other tasks.

3.2 Baselines and Models

We compare LAMP with the following baseline approaches: **Full Fine-tuning**, which updates all parameters of PLMs; **PT-based methods**, where PT (Lester et al., 2021) inserts trainable continuous vectors, known as soft prompt, before the model's input, and its most advanced variants include Residual PT (Razdaibiedina et al., 2023), DePT (Shi and Lipani, 2024), EPT (Lan et al., 2024) and DPT (Xiao et al., 2023); **LoRA-based methods** include PiSSA (Meng et al., 2024), rsLoRA (Kalajdzievski, 2023), LoRA+ (Hayou et al., 2024), DoRA (Liu et al., 2024) and LoRA-GA (Wang et al., 2024) . More details about Baselines can be found in Appendix A.3.

We aim to explore a high-performance PEFT method that minimizes trainable parameters. **Trainable parameters are a crucial factor in our selection of baselines**; hence, methods with more significant trainable parameters and modifying the transformer layers are not included as baselines for comparison. Such as Adapter (Houlsby et al., 2019) (prompt length is 100, 76.8k vs. 1.9M for T5-base) and its variant methods. Notably, EPT (Lan et al., 2024) has demonstrated superior performance compared to these PEFT methods. Furthermore, Xprompt (Ma et al., 2022) underwent rewinding training, and transfer learning (Vu et al., 2022; Asai et al., 2022) and multi-task learning (Wang et al., 2022) require pre-training. These methods are not directly comparable to LAMP.

3.2.1 Models Size

PT tends to underperform in smaller-scale models. Thus, we conducted primary experiments employing three T5 model variants (Raffel et al., 2020) (Small, 60M; Base, 220M; and Large, 770M) and validated the effectiveness of LAMP using T5-11B and Llama2-7B (Touvron et al., 2023).

3.3 Training Details

T5 model as the backbone for our experiments; the hidden dimensions for the T5-base, T5-small, and T5 large are 512, 768, and 1,024, respectively. Following the experimental setup from Xiao et al. (2023), we set the soft prompt length to 100, rank $r = 8$ in Truncated SVD and batch size is 16. The models are trained 100 epochs using the AdamW (Loshchilov and Hutter, 2019) optimizer with an initial learning rate of 0.3. We employ the double quantization operation in QLoRA(Dettmers et al., 2023) for T5-11B and Llama2-7B. Other training details in Appendix A.4.

3.4 Overall Performance Comparison (RQ1)

Table 1 presents a comparison of LAMP against baseline methods on the SuperGLUE benchmark using various T5 model sizes. Notably, LAMP requires the fewest training parameters and demonstrates exceptional average performance across different scales of T5 models. LAMP outperforms

Method	Params.	CB Acc.	WSC Acc.	COPA Acc.	RTE Acc.	WiC Acc.	BoolQ Acc.	MultiRC F1	ReCoRD F1	Average (%)
T5-Small										
Fine-Tuning†	60M	89.28	67.94	59.00	72.56	68.18	77.06	66.98	55.64	69.58
Prompt Tuning	51K	71.43	59.62	58.33	66.91	63.95	66.12	63.31	50.11	62.47
Residual PT†	462K	72.02	63.14	56.66	67.02	60.96	**73.35**	65.12	53.08	63.91
DePT	51K	75.00	**67.31**	52.66	65.22	62.70	66.00	61.94	**56.71**	63.44
EPT	51K	78.57	**67.31**	55.66	**71.01**	**67.39**	69.17	64.46	54.14	65.96
DPT	6K	78.85	60.53	59.33	70.40	64.26	72.17	64.61	53.60	65.47
LAMP(ours)	5K	**83.93**	**67.31**	**60.66**	69.31	66.46	72.97	**66.25**	53.80	**67.59**
T5-Base										
Fine-Tuning‡	220M	91.70	81.70	60.00	84.50	69.30	82.30	76.90	80.90	78.41
Prompt Tuning	77K	78.57	61.54	55.00	67.63	62.38	77.00	72.37	71.32	68.27
Residual PT†	693K	77.37	67.94	56.66	81.70	66.87	80.00	72.11	72.21	71.86
DePT	77K	82.14	67.31	54.33	73.91	65.20	79.02	**72.70**	70.80	70.68
EPT	77K	85.71	67.31	56.00	78.99	67.71	79.14	72.62	71.15	72.33
DPT	9K	78.56	67.30	56.66	79.42	68.49	**80.28**	72.50	**72.56**	71.97
LAMP(ours)	7K	**94.64**	**68.42**	**58.66**	**83.39**	**70.06**	80.24	72.72	72.55	**75.09**
T5-Large										
Fine-Tuning‡	770M	94.30	88.50	87.00	90.60	73.50	88.30	85.40	89.20	87.10
Prompt Tuning	102K	82.35	65.38	57.33	88.45	70.69	84.28	76.37	74.36	74.90
Residual PT†	925K	73.21	70.50	**62.66**	88.92	72.25	85.04	76.46	84.36	76.67
DePT	102K	85.71	67.31	50.66	83.33	68.97	83.24	75.76	74.03	73.63
EPT	102K	89.29	68.30	54.00	86.33	71.79	84.77	76.62	73.94	75.63
DPT	11K	89.29	65.79	**62.66**	88.45	71.63	84.53	76.72	84.35	77.93
LAMP(ours)	9K	**98.21**	**78.95**	57.33	**90.61**	**73.35**	**85.11**	**76.94**	**84.56**	**80.63**

Table 1: For the performance comparison on the SuperGLUE benchmark, all experimental results are based on the T5-Small, T5-Base, and T5-Large. All scores represent the mean across three runs using distinct random seeds. † sourced from Xiao et al. (2023). ‡ sourced from Aribandi et al. (2021). The best result is marked in bold. The second-highest result is indicated by an underline.

the original PT by 7.58%, 9.08%, and 7.11% on T5-Small, T5-Base, and T5-Large, respectively. Detailed information on the standard deviation of LAMP can be found in Appendix A.5. LAMP's performance improvement is attributed to enhancing the model's knowledge representation by uncovering the intrinsic semantic interactions between soft prompt tokens. LAMP achieves outstanding performance while significantly reducing training parameters and computational costs.

From the perspective of trainable parameters, LAMP and DPT are more efficient than other baselines; even though EPT outperforms DPT, EPT requires more trainable parameters. LAMP clearly outperforms DPT in several ways. First, DPT relies on randomly generated initial prompts that lack semantic richness, whereas LAMP leverages sample vocabularies to better aid the model in understanding complex language structures. Additionally, while DPT reduces trainable parameters, its prompt length remains at 100 when input into model, leading to inefficiency. In contrast, LAMP employs average pooling operations that neither harm performance nor increase trainable parameters, making it an efficient and effective novel prompt tuning method. **LAMP demonstrates superior performance and requires significantly fewer training parameters than the latest LoRA-based PEFT methods,** with detailed results provided in Appendix A.6.

3.5 Ablation Experiment Analysis (RQ2)

Few-shot adaptation. Following the few-shot experimental setup of Xiao et al. (2023), we randomly

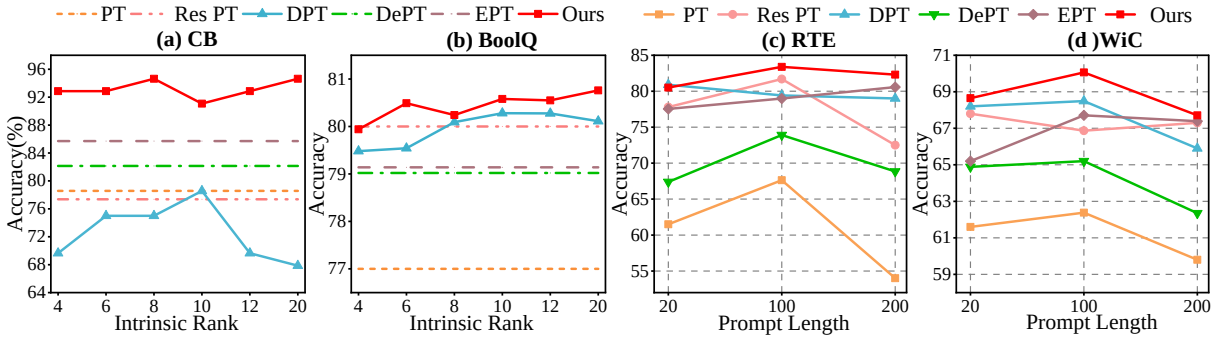

Figure 4: (a) and (b), the performance of all baselines with the number of inherent ranks $r \in \{4, 6, 8, 10, 12, 20\}$ on the SuperGLUE benchmark. (c) and (d), the performance of different baselines varies with the prompt length $l \in \{20, 100, 200\}$. All results represent the average of three runs conducted with a different random seed.

	K-Shot		
Model	**8**	**16**	**32**
Prompt Tuning	48.23	49.83	50.85
Residual PT	52.95	57.57	58.50
DePT	49.83	50.31	49.93
EPT	50.42	50.71	53.51
DPT	56.26	55.60	57.72
LAMP (ours)	**57.21**	**58.14**	**59.25**

Table 2: Few-shot adaptation results with $k = \{8, 16, 32\}$ on SuperGLUE benchmark.

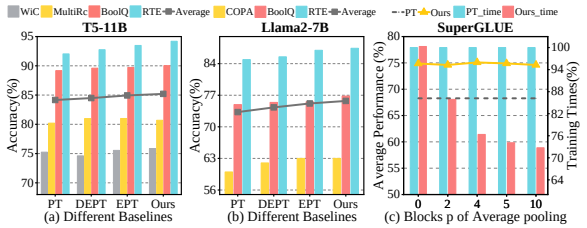

Figure 5: (a) and (b), the performance changes of different methods at various datasets on the T5-11B and Llama2-7B. (c), the variation of training time and performance in different average pooling blocks p.

sampled 8, 16, and 32 training examples. Table 2 presents the results of all baselines on the SuperGLUE benchmark. All results are averaged over three runs with different random seeds on the T5-base model. We found that LAMP outperforms other baselines on most datasets in the few-shot setting, demonstrating its effectiveness. The few-shot performance of various methods across different datasets is detailed in Appendix A.7.

Sensitivity of Rank Size. The intrinsic rank r is the primary factor influencing the total number of trainable parameters in LAMP. We analyzed the impact of $r \in \{4, 6, 8, 10, 12, 20\}$ on LAMP performance using the T5-base model on the CB and BoolQ datasets within the SuperGLUE benchmark. As shown in Figure 4(a) and Figure 4(b), despite the minimal differences in trainable parameters of LAMP and DPT across different r values, LAMP consistently outperforms DPT and other baseline methods in most scenarios. This demonstrates the effectiveness of incorporating semantic knowledge into LAMP. The details of how the intrinsic rank r affects the changes in training parameters can be found in the Appendix A.8.

Effect of Prompt Length. We conduct analyses using the RTE and WiC datasets within the SuperGLUE benchmark. To understand the impact of prompt length on the LAMP's performance, we maintained an "intrinsic rank" r of 8 for the soft prompt on the T5-Base model, varying the prompt lengths $\in \{20, 100, 200\}$. Figure 4(c) and Figure 4(d) illustrates that LAMP consistently outperforms other baselines at prompt lengths of 20, 100, and 200. LAMP achieves optimal performance when the prompt length is set to 100, consistent with previous findings that 100 is the optimal hyperparameter for prompt length (Lester et al., 2021; Razdaibiedina et al., 2023). The experimental details of other datasets in the Appendix A.9.

Impact of Model Scale. Based on quantification, we conducted experiments on T5-11B and Llama2-7B using randomly selected datasets and compared LAMP against baselines that initialize prompts with semantic knowledge from samples. As shown in Figure 5(a) and Figure 5(b), across different model architectures with more than 7B parameters (T5 with an encoder-decoder structure and Llama2 with a decoder-only structure), LAMP consistently helps models adapt to various down-

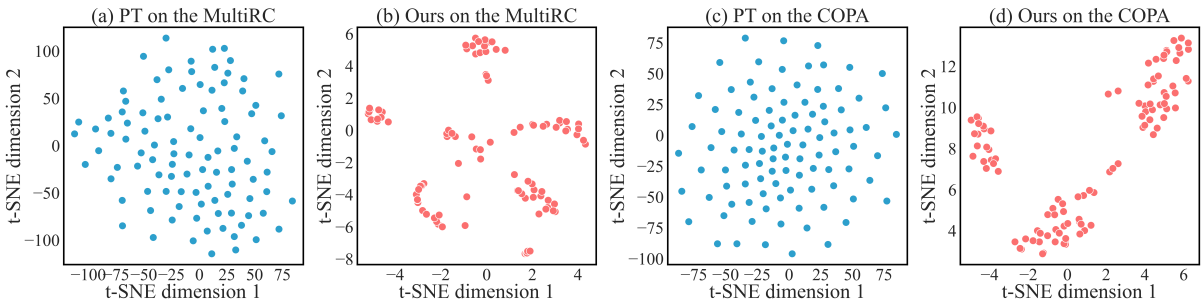

Figure 6: The comparison of the dispersion of PT tokens on the MultiRC and COPA datasets before and after considering the intrinsic semantic associations between soft prompt tokens on T5-Base.

stream tasks, achieving superior performance. The experimental details of T5-3B can be found in the Appendix A.10. Meanwhile, Appendix A.11 details the changes in training parameters and memory usage associated with the model scale.

Blocks of Average Pooling. Figure 5(c) illustrates the changes in model training time and performance on the SuperGLUE benchmark as the number of average pooling blocks increases. The larger the pooling block p, the shorter the prompt length input to the model, resulting in a more noticeable reduction in training time. Furthermore, we were pleasantly surprised to find that average pooling had minimal impact on performance, yet it provided significant advantages for PT. This finding offers a promising mentality for extending PT to various domains.

3.6 Interpretability (RQ3)

Figure 6 compares general PT before (i.e., blue points) and after (i.e., red points) extracting the intrinsic semantic associations between soft prompt tokens on the MultiRC and COPA datasets. To more intuitively reflect the discreteness of PT, we did not standardize the dimensions for comparison. The more extensive the x and y-axis ranges, the higher the degree of discreteness. The features extracted by LAMP exhibit more distinct clustering, indicating LAMP's superior ability to capture and represent the intrinsic structure and patterns of the data. By considering the interactions between prompt tokens, LAMP uncovers the intrinsic semantic associations among tokens to enhance knowledge representation. This enables PLMs to better grasp the semantic content of textual data. The comparative results of discreteness between LAMP and original prompt tuning (PT) across other datasets are detailed in Appendix A.12.

4 Related Work

Parameter-efficient Fine-tuning. Parameter-efficient fine-tuning achieves strong results by training a small subset of parameters, thereby reducing computational costs and improving efficiency. AdapterDrop (Rücklé et al., 2020) improves efficiency by removing unimportant adapters for a given task in each layer of the Transformer. BitFit (Zaken et al., 2021) only updates the bias terms while freezing most of the pre-trained model's parameters. LST (Sung et al., 2022) reduces training memory by running a small ladder network alongside the pre-trained network. LoRA (Hu et al., 2021) re-parameterizes incremental matrices through simple low-rank decomposition. KronA (Edalati et al., 2022) replaces the low-rank decomposition in LoRA with Kronecker product decomposition; PISSA (Meng et al., 2024) initializes the low-rank matrices with the weights of the pre-trained model, enhancing performance and efficiency. However, prompt tuning (Lester et al., 2021) stands out from the rest by achieving good results with training very few parameters.

Prompt-based Fine-tuning. Unlike other PEFT methods, prompt-based fine-tuning methods sustain a controlled increase in trainable parameters, even with substantial model scaling. Prompt tuning (Lester et al., 2021) only adds the soft prompt to the input embedding layer of the model. DPT(Xiao et al., 2023) employs a re-parameterization strategy, using two low-rank matrices to replace the original soft prompt. DPT relies solely on random number generation for the soft prompt, resulting in weaker generalization and higher sensitivity to initialization. DePT(Shi and Lipani, 2024) decomposes the soft prompt into shorter prompts and pairs of low-rank matrices, which are then used to update the model's weights. EPT (Lan et al.,

2024) leverages multi-space projection and prompt fusion to refine soft prompt knowledge, enhancing flexibility and balancing accuracy with computational efficiency for diverse tasks. Nevertheless, these PT-based methods need more efficiency and task-specific knowledge richness when dealing with long soft prompt. LAMP provides the ability to tailor prompts more precisely and effectively to the specific requirements of various tasks.

5 Conclusions

In this work, we observed that soft prompt tokens initialized randomly from the vocabulary lack intrinsic semantic associations to enhance knowledge representation. Additionally, PT-based methods face challenges balancing knowledge richness and computational cost in different tasks. Based on these issues, we approximate soft prompt by proposing a Low-parameter Prompt Tuning (LAMP) method, which utilizes two singular vectors and singular values. LAMP facilitates semantic knowledge interaction, allowing the soft prompt to incorporate more task-specific knowledge. It can serve as an efficient plugin for various PT-based tasks. Experimental results across three model scales (T5-Small, T5-Base, T5-Large) demonstrate that LAMP achieves high effectiveness and robustness with fewer trainable parameters.

Acknowledgements

This work is partially supported by the National Natural Science Foundation of China under Grant (No. 62032013, 62102074), the Science and Technology projects in Liaoning Province (No. 2023JH3/10200005).

Limitations

While our method significantly reduces trainable parameters in NLP, its potential applications extend far beyond this domain. Its evaluation in areas beyond NLP and with other advanced large language models remains a work in the future. While our approach significantly reduces trainable parameters, further quantification of model parameters will also be explored in future research. The intrinsic semantic interactions between soft prompt tokens can be more effectively mined without increasing the number of trainable parameters. In future work, we will explore methods to enhance knowledge representation for PT.

References

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of ACL*, pages 7319–7328.

Vamsi Aribandi, Yi Tay, Tal Schuster, Jinfeng Rao, Huaixiu Steven Zheng, Sanket Vaibhav Mehta, Honglei Zhuang, Vinh Q Tran, Dara Bahri, Jianmo Ni, et al. 2021. Ext5: Towards extreme multi-task scaling for transfer learning. In *Proceedings of ICLR*.

Akari Asai, Mohammadreza Salehi, Matthew E Peters, and Hannaneh Hajishirzi. 2022. Attempt: Parameter-efficient multi-task tuning via attentional mixtures of soft prompts. In *Proceedings of EMNLP*, pages 6655–6672.

DOLAN William Bill. 2005. Automatical ly constructing a corpus of sentential paraphrases. In *Proceedings of IWP*, pages 9–16.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of NAACL*, pages 2924–2936.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. In *SuB*, volume 23, pages 107–124.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: efficient finetuning of quantized llms. In *Proceedings of NeurIPS*, pages 10088–10115.

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

Ali Edalati, Marzieh Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. 2022. Krona: Parameter efficient tuning with kronecker adapter. *arXiv preprint arXiv:2212.10650*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL*, pages 1–9.

Demi Guo, Alexander M Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. In *Proceedings of ACL*, pages 4884–4896.

Per Christian Hansen. 1987. The truncated svd as a method for regularization. *BIT Numerical Mathematics*, 27:534–553.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. Lora+: Efficient low rank adaptation of large models. In *Proceedings of ICML*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *Proceedings of ICML*, pages 2790–2799.

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 *Proceedings of ICLR*.

Damjan Kalajdzievski. 2023. A rank stabilization scaling factor for fine-tuning with lora. *arXiv preprint arXiv:2312.03732*.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of NAACL*, pages 252–262.

Pengxiang Lan, Enneng Yang, Yuting Liu, Guibing Guo, Linying Jiang, Jianzhe Zhao, and Xingwei Wang. 2024. Efficient prompt tuning by multi-space projection and prompt fusion. *arXiv preprint arXiv:2405.11464*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of EMNLP*, pages 3045–3059.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Proceedings of KR*.

Shih-yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation. In *Proceedings of ICML*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *Proceedings of ICLR*.

Fang Ma, Chen Zhang, Lei Ren, Jingang Wang, Qifan Wang, Wei Wu, Xiaojun Quan, and Dawei Song. 2022. Xprompt: Exploring the extreme of prompt tuning. In *Proceedings of EMNLP*, pages 11033–11047.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. Pissa: Principal singular values and singular vectors adaptation of large language models. *arXiv preprint arXiv:2404.02948*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of NAACL*, pages 1267–1273.

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

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 EMNLP*, pages 2383–2392.

Anastasiia Razdaibiedina, Yuning Mao, Madian Khabsa, Mike Lewis, Rui Hou, Jimmy Ba, and Amjad Almahairi. 2023. Residual prompt tuning: improving prompt tuning with residual reparameterization. In *Proceedings of ACL*, pages 6740–6757.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *Proceedings of AAAI*.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2020. Adapterdrop: On the efficiency of adapters in transformers. *arXiv preprint arXiv:2010.11918*.

Zhengxiang Shi and Aldo Lipani. 2024. Dept: Decomposed prompt tuning for parameter-efficient fine-tuning. In *Proceedings of ICLR*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, pages 1631–1642.

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. *Advances in Neural Information Processing Systems*, 35:12991–13005.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NeurIPS*.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2022. Spot: Better frozen model adaptation through soft prompt transfer. In *Proceedings of ACL*, pages 5039–5059.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Superglue: a stickier benchmark for general-purpose language understanding systems. In *Proceedings of NeurIPS*, pages 3266–3280.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of EMNLP*, pages 353–355.

Shaowen Wang, Linxi Yu, and Jian Li. 2024. Lora-ga: Low-rank adaptation with gradient approximation. *arXiv preprint arXiv:2407.05000*.

Zhen Wang, Rameswar Panda, Leonid Karlinsky, Rogerio Feris, Huan Sun, and Yoon Kim. 2022. Multitask prompt tuning enables parameter-efficient transfer learning. In *Proceedings of ICLR*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of NAACL*, pages 1112–1122.

Yao Xiao, Lu Xu, Jiaxi Li, Wei Lu, and Xiaoli Li. 2023. Decomposed prompt tuning via low-rank reparameterization. In *Proceedings of EMNLP*.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*.

Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. 2024. Panda: Prompt transfer meets knowledge distillation for efficient model adaptation. *TKDE*.

A Appendix

A.1 Self-Attention Pooling

We propose utilizing a self-attention mechanism for the pooling operation to allow the soft prompt to adaptively assign different weights to prompt tokens —emphasizing key tokens while ignoring less important ones. The formula is expressed as follows:

$$\begin{aligned} \mathbf{K} &= \mathbf{C}W_{sa} \\ \mathbf{A}_{weight} &= \mathrm{Softmax}(\mathbf{K}) \\ \mathbf{P}' &= \mathbf{A}_{weight}^{\top}\mathbf{C} \end{aligned} \tag{8}$$

where, $W_{sa} \in \mathbb{R}^{d\times l/p}$ is a learnable initialization weight matrix, and $A_{weight} \in \mathbb{R}^{l\times l/p}$ represents the attention weights. By applying A_{weight} to $\mathbf{C} \in \mathbb{R}^{l\times d}$, we achieve adaptive selection and pooling operation of prompt tokens, resulting in $\mathbf{P}' \in \mathbb{R}^{l/p\times d}$. The self-attention pooling operation introduces additional trainable parameters $W_{sa} \in \mathbb{R}^{d\times l/p}$, and its performance is suboptimal. We hypothesize that this may be due to the compressed outer product effectively capturing the intrinsic semantic associations between prompt tokens. The dynamic weight assignment might disrupt these previously captured associations.

A.2 Dataset Details

Table 3 provides detailed information on the 8 datasets we used in SuperGLUE benchmark. The processing of all datasets follows the approach of Xiao et al. (2023).

A.3 Baselines Details

Apart from full fine-tuning, due to the significant advantages of PT, all other baselines are variants based on PT. Descriptions of all **PT-based** baselines are as follows:

- **Full Fine-tuning:** Updating all model parameters in the T5-models (Raffel et al., 2020) on each downstream task. It is the most fundamental method for comparing PEFT methods' performance and trainable parameters.
- **Prompt tuning** (Lester et al., 2021): PT stands out in the PEFT approaches because it freezes the parameters of PLMs and only trains the attached soft (continuous) prompt vectors to the input text.
- **Residual Prompt tuning** (Razdaibiedina et al., 2023): A PT-based variant (named Res PT) that utilizes a residual network to increase the flexibility of model selection for soft prompt token representations and improve the convergence rate.
- **DePT** (Shi and Lipani, 2024): Decomposing the prompt into a shorter prompt and low-rank

SuperGLUE Benchmark					
Dataset	**#Train**	**#Dev**	**Type**	**Domain**	**#Metric**
CB	250	56	Natural Language Inference	various	accuracy
WSC	259	104	Common Sense Reasoning	fiction books	accuracy
COPA	400	100	Question Answering	blogs, etc.	accuracy
RTE	2,490	277	Natural Language Inference	News, Wikipedia	accuracy
Wic	5,428	638	Word Sense Disambiguation	lexical databases	accuracy
BoolQ	9,427	3,270	Question Answering	Wikipedia	accuracy
MulticRC	27,243	4,848	Question Answering	various	F1
ReCoRD	100,730	10,000	Common Sense Reasoning	news (CNN, Daily Mail)	F1
MNLI	392,702	19,647	NLI	various	accuracy
QNLI	103,743	6,463	NLI	Wikipedia	accuracy
SST-2	66,349	1872	Sentiment	Movie Reviews	accuracy
MRPC	3,668	408	Paraphrase	news	accuracy

Table 3: The details of the 8 datasets in SuperGLUE benchmark utilized in our experiment.

Model	**Params.**	CB	WSC	COPA	RTE	WiC	BoolQ	MultiRC	ReCoRD
				T5-Small					
LAMP_S	6K	1.54	2.26	0.47	0.78	0.71	0.23	1.12	0.25
				T5-Base					
LAMP_B	9K	0.84	2.15	1.25	0.51	0.34	0.13	0.32	0.05
				T5-Large					
LAMP_L	11K	1.37	2.48	0.47	0.31	0.29	0.07	0.08	0.07

Table 4: We report standard deviation of three runs for our method LAMP, where _S is T5-Small, _B is T5-Base and _L is T5-Large.

matrix pairs to reduce training time. It utilizes low-rank matrix pairs to update the input embedding.

- **EPT** (Lan et al., 2024): This method utilizes multi-space projection and prompt fusion modules to enhance the soft prompt knowledge, making it more adaptable to various downstream tasks while balancing accuracy and efficiency.
- **DPT** (Xiao et al., 2023): A novel prompt initialization method involves replacing the prompt with two randomly initialized low-dimensional matrices.

Descriptions of all **LoRA-based** baselines are as follows:

- **LoRA** (Hu et al., 2021): A parameter-efficient approach focuses on updating only the low-rank matrices within the model.
- **PiSSA** (Meng et al., 2024): It performs Truncated Singular Value Decomposition (SVD) on the model's weight matrix and uses the resulting low-rank matrices as the initialization for the low-rank matrices A and B.
- **rsLoRA** (Kalajdzievski, 2023): introduces a new scaling factor to stabilize LoRA's parameter scaling.
- **LoRA+** (Hayou et al., 2024): It utilizes two different learning rates to control the updates of the low-rank matrices A and B.
- **DoRA** (Liu et al., 2024): It decomposes the pre-trained weight into two components, $magnitude$ and $direction$, for fine-tuning.
- **LoRA-GA** (Wang et al., 2024): It aligns low-rank gradient products with full fine-tuning gradients at the first step.

Model	Param.	MNLI 393K	QNLI 105K	SST-2 67K	MRPC 3.7K	Mean (%)
Fine-Tuning	220M	86.33	93.19	94.75	84.56	89.71
LoRA	3.8M	85.30	92.96	94.04	68.38	85.17
PiSSA	3.8M	85.75	93.15	94.07	76.31	87.32
rsLoRA	3.8M	85.73	93.12	94.19	52.86	81.48
LoRA+	1.6M	85.81	93.14	93.85	74.43	86.81
DoRA	3.8M	85.67	93.04	94.04	68.08	85.21
LoRA-GA	3.8M	85.70	93.18	94.11	85.29	89.57
DEPT	76.8K	85.12	93.20	94.19	88.71	90.31
EPT	76.8K	85.63	93.15	94.21	89.20	90.55
DPT	9K	85.34	93.15	94.50	88.47	90.37
LAMP	7K	**85.82**	**93.32**	**94.50**	**90.20**	**90.96**

Table 5: The performance comparison between LAMP and the latest LoRA-based PEFT methods on the GLUE benchmark, all experimental results are based on the T5-Base model. The LoRA-based baseline results are derived from LoRA-GA (Wang et al., 2024).

A.4 Implementation Details

Weight decay of $1e-5$, and the maximum sequence length for the model is typically configured at 256. LAMP is implemented by the Python library of PyTorch 2.0.0 [1], Huggingface Transformers 4.30.0 [2]. All of our experiments were conducted with 8 GPUs, with 48 GB memory each.

A.5 Standard Deviation

We present the standard deviation across three runs for our method on T5-Small, T5-Base and T5-Large. The outcomes are provided in Table 4.

A.6 Performance Comparison with LoRA Variants

Table 5 presents a performance comparison between LAMP and LoRA, along with its variants on the GLUE benchmark. LAMP outperforms all other novel LoRA-based baselines. Notably, LAMP requires significantly fewer trainable parameters than all baselines, representing a unique advantage among PEFT methods.

A.7 Few-shot Adaptions Details

In Table 6, we present the results of all baseline models across all SuperGLUE datasets, using T5-Base as the benchmark. The subscripts represent the standard deviation of our method LAMP across different K-shot settings.

[1] https://pytorch.org/

[2] https://github.com/huggingface/transformers

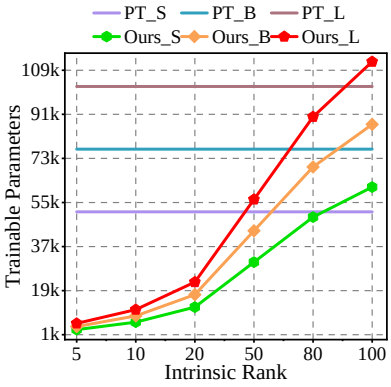

Figure 7: Trainable parameters with different intrinsic ranks on the T5 models, where _S is T5-Small, _B is T5-Base and _L is T5-Large.

A.8 Change in LAMP Parameters from Intrinsic Rank

As shown in Figure 7, the impact of intrinsic rank $r \in \{5, 10, 20, 50, 80, 100\}$ on LAMP trainabel parameters is illustrated across T5 models (T5-Small, T5-Base, and T5-Large). As r increases, LAMP's trainable parameters gradually approach that of the vanilla PT. When all ranks in the Truncated SVD are retained, the trainable parameters of PT are fewer than those of LAMP. We downplay the impact of prompt length on the number of trainable parameters. The number of trainable parameters in LAMP can be further reduced by adjusting r.

A.9 Impact of Rank Size and Prompt Length

As shown in Figure 8, we present the effects of intrinsic rank and prompt length across various datasets. The experimental results indicate that

K-Shot	Method	SuperGLUE								
		CB	WSC	COPA	RTE	WiC	BoolQ	MultiRC	ReCoRD	Average
8	PT	58.57	32.69	40.66	49.64	53.61	53.94	50.17	46.55	48.23
	Res PT	60.55	28.95	55.00	47.29	44.51	61.35	57.78	68.13	52.95
	DePT	61.43	36.69	56.66	48.92	51.72	47.83	47.45	47.94	49.83
	EPT	57.14	42.31	45.33	51.80	50.47	53.39	56.91	46.04	50.42
	DPT	62.50	39.47	55.66	56.68	50.00	62.17	60.12	63.49	56.26
	LAMP	$62.86_{1.53}$	$44.74_{1.24}$	$59.00_{0.94}$	$53.43_{0.2}$	$50.00_{0.00}$	$62.17_{0.26}$	$59.98_{0.00}$	$62.35_{1.15}$	**57.21**
16	PT	57.14	32.69	44.00	51.08	53.29	58.10	56.02	46.34	49.83
	Res PT	68.88	50.00	55.00	47.65	53.13	62.42	55.24	68.22	57.57
	DePT	50.00	37.69	48.66	55.40	49.53	57.98	52.43	50.77	50.31
	EPT	42.86	42.31	48.00	51.80	56.11	57.43	57.73	49.47	50.71
	DPT	44.64	55.26	55.66	53.07	51.41	62.17	59.99	62.57	55.60
	LAMP	$62.21_{2.66}$	$47.37_{2.48}$	$59.00_{1.89}$	$54.51_{0.85}$	$54.23_{0.59}$	$62.29_{0.06}$	$59.88_{0.01}$	$70.72_{0.44}$	**58.14**
32	PT	57.14	32.69	46.33	54.68	55.49	59.94	51.39	49.15	50.85
	Res PT	69.21	47.37	58.66	51.99	53.34	63.09	56.15	68.15	58.50
	DePT	53.57	32.69	52.33	50.36	52.98	58.72	49.89	48.91	49.93
	EPT	67.85	40.38	56.33	51.80	55.80	60.12	49.74	46.05	53.51
	DPT	62.50	52.63	55.00	53.79	52.35	62.17	59.95	63.34	57.72
	LAMP	$63.14_{2.23}$	$47.37_{1.54}$	$60.00_{1.41}$	$54.87_{0.34}$	$56.74_{0.44}$	$62.17_{0.04}$	$60.08_{0.08}$	$67.80_{1.71}$	**59.25**

Table 6: Few-shot adaptation results (%) with $k = \{8, 16, 32\}$ on SuperGLUE benchmark. All results are presented as the average of three runs and subscripts indicate standard deviation, each with different random seeds on T5-Base. The best result is marked in bold.

Method	T5-3B			
	PT	DEPT	EPT	LAMP
MultiRC	78.00	77.95	78.51	78.83
WiC	71.16	71.16	73.67	74.92
RTE	91.31	92.75	92.75	93.48
BoolQ	87.22	87.65	87.89	87.83
Mean	81.93	82.38	83.21	83.77

Table 7: The performance changes of different methods at various datasets on the T5-3B.

LAMP consistently performs strongly across different intrinsic ranks r and prompt lengths l.

A.10 Performance in T5-3B

For baseline selection, we similarly chose baselines initialized from sampled vocabulary. Table 7 shows that LAMP performs best on the T5-3B model, further validating the importance of incorporating intrinsic semantic associations between prompt tokens to enhance knowledge representation capacity.

A.11 Change in LAMP Parameters

A.11.1 Change in LAMP Parameters from Model Size

With intrinsic rank $r = 8$, Figure 9 illustrates the effect of model size on LAMP trainable parameters. As the model size increases, trainable parameters in original PT increases significantly, whereas LAMP mitigates this issue. Table 8 shows that when the prompt length l and intrinsic rank r remain constant, the higher the model size, the more significant the reduction in trainable parameters achieved by LAMP. LAMP also mitigates the impact of the hidden dimension on trainable parameters.

A.11.2 Variation of LAMP Parameters

Table 8 presents the variation of LAMP's trainable parameters with prompt length $l \in \{20, 100, 1000, 5000, 10000\}$ with $r = 8$. The longer the prompt length, the more LAMP downplays the impact of prompt length on trainable parameters, making LAMP's advantages more evident. For instance, in the Llama2-7B model, when the prompt length is set to 10,000, the number of

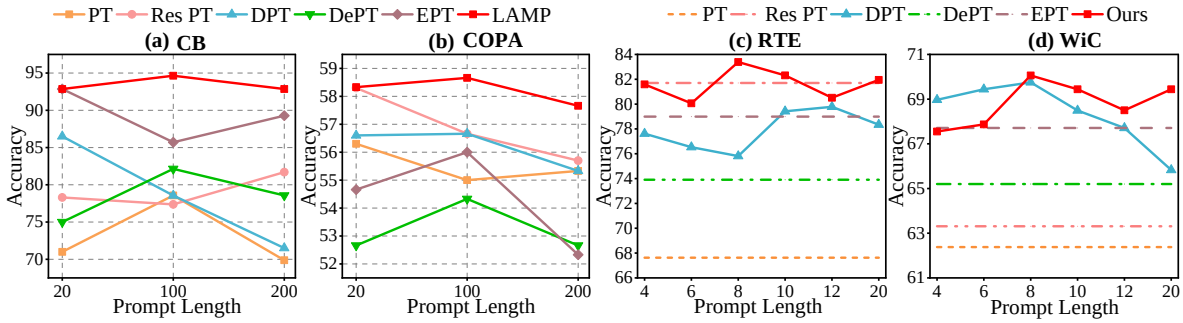

Figure 8: (a) and (b), performance of different baselines varies with the prompt length $l \in \{20, 100, 200\}$ on the CB and COPA datasets. (c) and (d), compare the performance of all baselines with the number of inherent ranks $r \in \{4, 6, 8, 10, 12, 20\}$ on the RTE and WiC datasets. All results represent the average of three runs conducted with a different random seed.

Method	**T5-Large** Prompt Length					**Llama2-7B** Prompt Length				
	20	100	1000	5000	10000	20	100	1000	5000	10000
	#Trainable Params					#Trainable Params				
Full FT	0.7B	0.7B	0.7B	0.7B	0.7B	7B	7B	7B	7B	7B
PT	0.02M	0.1M	1.02M	5.12M	10.24M	0.08M	0.41M	4.10M	20.48M	40.96M
LAMP	**0.008M**	**0.009M**	**0.016M**	**0.048M**	**0.088M**	**0.033M**	**0.034M**	**0.041M**	**0.073M**	**0.113M**
Ratio	2.45	11.38	63.21	106.22	116.10	2.49	12.20	100.45	281.41	363.20

Table 8: Trainable parameters of different baselines varies with the prompt length. “Ratio" denotes the multiple of trainable parameters in vanilla prompt tuning (PT) relative to LAMP.

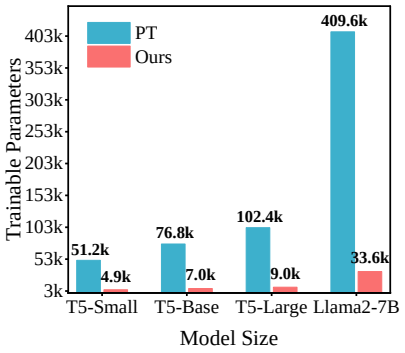

Figure 9: Trainable parameters with different model size.

trainable parameters is reduced to just one-three-hundred-sixtieth of the original PT, significantly boosting computational efficiency. In summary, influenced by intrinsic rank r, prompt length l , and model size, the LAMP's efficiency compared to PT and other PEFT methods can be further expanded.

A.11.3 Change in LAMP memory usage from Model Size

We visualized the memory usage of LAMP across different model scales, with the T5-11B results reflecting quantization. As shown in Figure 10, LAMP consistently has the lowest memory usage and highest computational efficiency across various model sizes.

A.12 Visualization of Intrinsic Semantic Associations

We visualized the comparison of general PT before and after extracting the intrinsic semantic associations between soft prompt tokens on other datasets within the SuperGLUE benchmark. Figure 11 shows that LAMP also exhibits apparent clustering on these datasets, enhancing semantic knowledge representation.

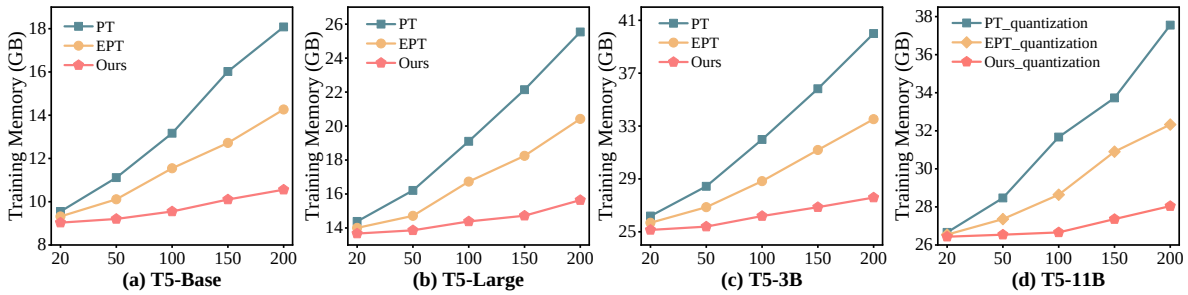

Figure 10: Comparison of memory usage using different methods on various model scales. We leverage quantization operation on T5-11B.

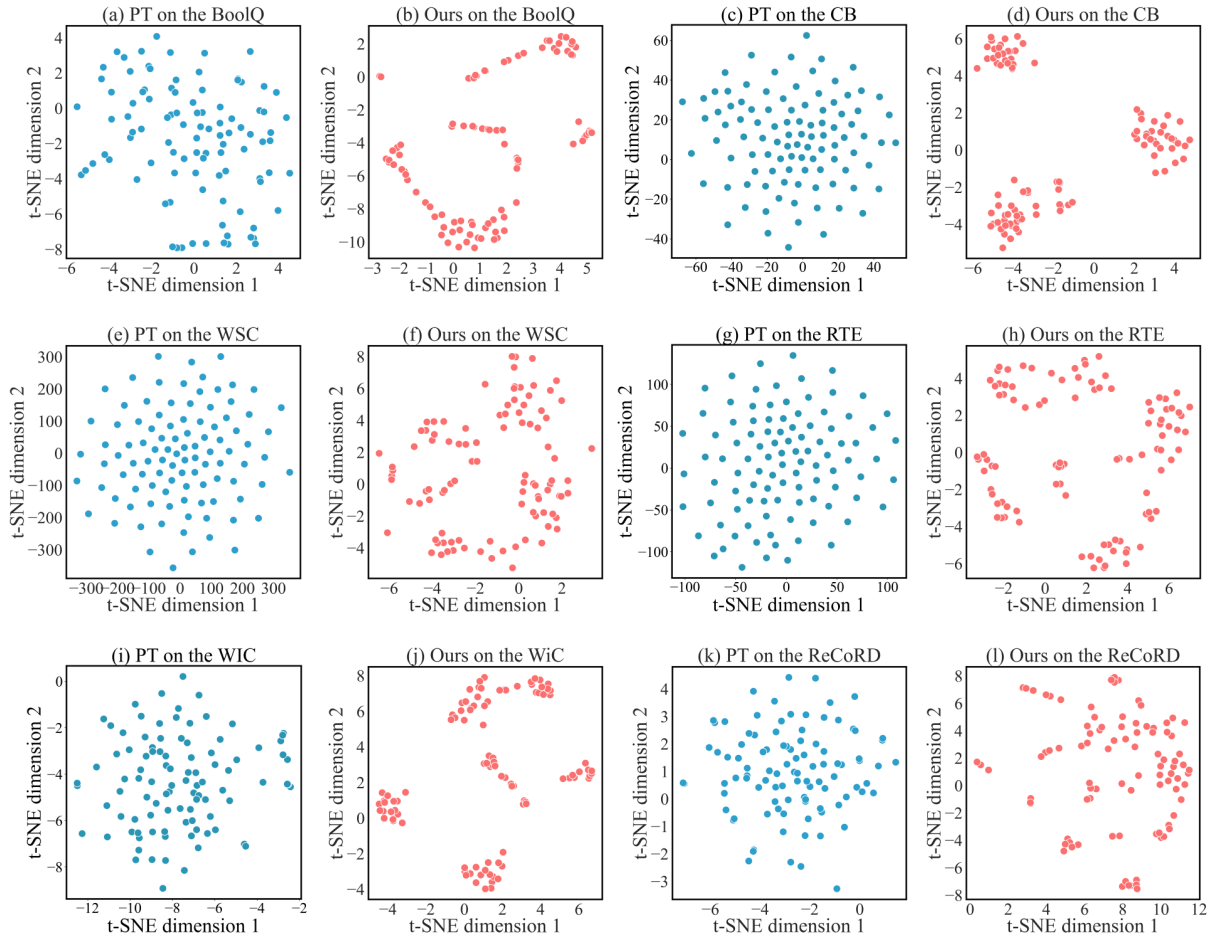

Figure 11: The comparison of the dispersion of PT tokens on the other datasets in SuperGLUE benchmark before and after considering the intrinsic semantic associations between soft prompt tokens on T5-Base.