Position Really Matters: Towards a Holistic Approach for Prompt Tuning

Xianjun Yang[1*], **Wei Cheng**[2], **Xujiang Zhao**[2], **Wenchao Yu**[2],
Linda Petzold[1], **Haifeng Chen**[2]

[1]University of California, Santa Barbara
{xianjunyang, petzold}@ucsb.edu

[2]NEC Laboratories America
{weicheng, xuzhao, wyu, haifeng}@nec-labs.com

Abstract

Prompt tuning is highly effective in efficiently extracting knowledge from foundation models, encompassing both language, vision, and vision-language models. However, the efficacy of employing fixed soft prompts with a *predetermined position* for concatenation with inputs for all instances, irrespective of their inherent disparities, remains uncertain. Variables such as the position, length, and representations of prompts across diverse instances and tasks can substantially influence the performance of prompt tuning. We first provide a theoretical analysis, revealing that optimizing the position of the prompt to encompass the input can capture additional semantic information that traditional prefix or postfix prompt tuning methods fail to capture. Then, we present a holistic parametric prompt tuning strategy that dynamically determines different factors of prompts based on specific tasks or instances. Experimental results underscore the significant performance improvement achieved by dynamic prompt tuning across a wide range of tasks, including NLP, vision recognition, and vision-language tasks. Furthermore, we establish the universal applicability of our approach under full-data, few-shot, and multitask settings.

1 Introduction

Recently, the research community has fervently dedicated efforts to developing novel methods aimed at achieving parameter-efficient adaptation. Three prominent strategies includes prefix-tuning (PFT) (Li and Liang, 2021), prompt-tuning (PT) (Lester et al., 2021), and Low-Rank Adaptation (LoRA) (Hu et al., 2021). These approaches selectively fine-tune a small subset of parameters, distinct from the original pre-trained model, thereby circumventing the costly process of fine-tuning the entire foundation model. Among those, prompt-tuning involves a minimal amount of parameters, particularly in the context of billion-scale PLMs, a phenomenon commonly referred to as the power of scale (Lester et al., 2021).

The pre-train, prompt, and predict paradigm, as elucidated in (Liu et al., 2023), may be classified into the realms of soft and hard prompts, thereby enabling the seamless optimization and resplendent visualization of this groundbreaking approach. Nonetheless, the majority of previous research has either maintained a static set of optimized prompts (Ma et al., 2022; Liu et al., 2021) across all instances in a task or exclusively added prompts to the beginning of all inputs (Vu et al., 2022; Guo et al., 2022; Gu et al., 2022). Later, (Wu et al., 2022b) proposes to generate the instance-dependent prompts by an Adapter (Houlsby et al., 2019) module for NLU tasks and show noticeable performance improvement. Similarly, (Asai et al., 2022a) dynamically changes the instance-dependent soft prompts via attentional mixtures of prompts learned from multi-tasks. However, they still adopt a fixed position and fixed length for concatenating prompts with inputs, which might be the suboptimal strategy.

To the best of our knowledge, a systematic exploration of the dynamic manipulation of soft prompt position, length, and prompt pools remains absent from the literature. Our endeavor encompasses a comprehensive theoretical analysis that unravels the potential benefits of optimizing the position for concatenating prompts with inputs, thereby capturing additional semantics that conventional prefix or postfix prompt tuning methods fail to encapsulate. Motivated by our analysis, we propose a unified strategy for dynamic prompt (DP) tuning, wherein different factors are dynamically determined based on the specific tasks or instances at hand. An overview of our novel approach is summarized in Figure 1. Specifically, we employ a

[*]Work done during the internship at NEC Laboratories America.

Findings of the Association for Computational Linguistics:
NAACL 2025, pages 8516–8538

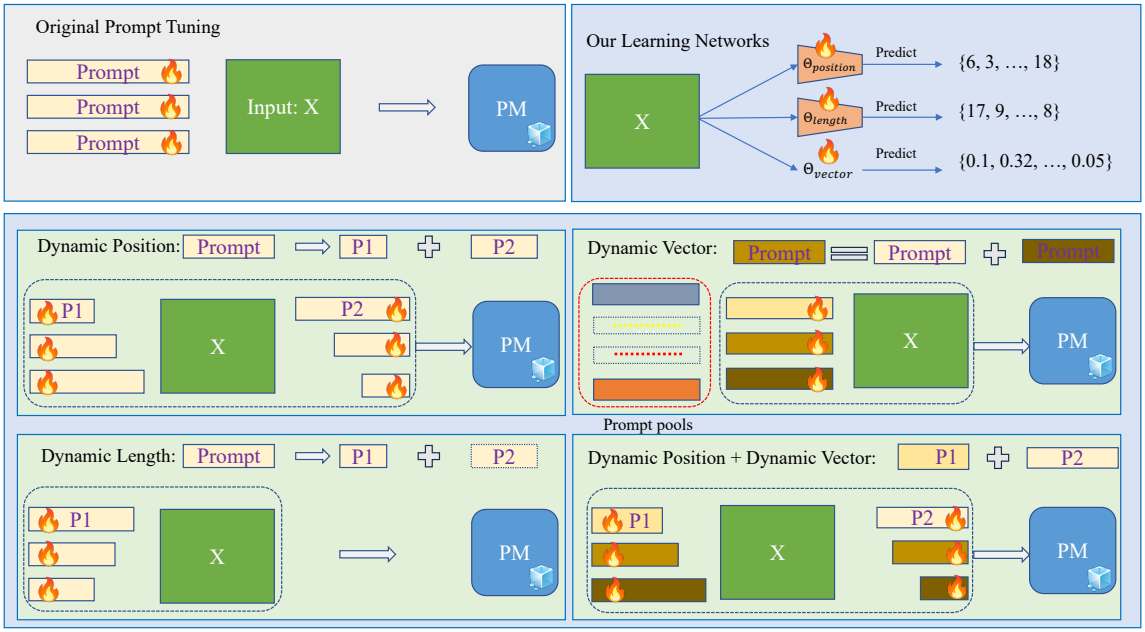


Figure 1: An overview of our approach. The learning networks first predict the task- or instance-dependent prompt position, length, and prompt pools. Then the new soft prompt is concatenated with instances to be fed into the frozen language, vision, or V-L models for prediction. Parameters in the prompts and learning networks are simultaneously updated. Canonical prompt tuning can be seen as a special case of our dynamic prompting, while the same soft prompt is prepended for all instances. The ice symbol means it is frozen during tuning, while the fire represents it is tuned.

one-layer feedforward network in conjunction with the Gumbel-Softmax technique (Jang et al., 2017a) to learn the categorical distribution of position or length, facilitating optimization at both the task and instance levels. This advancement effectively narrows the gap between prompt tuning and traditional fine-tuning, as illustrated in Figure 2.

Our approach serves as a versatile and potent component, capable of seamlessly integrating into a wide array of problem domains to unlock superior performance. Beyond its application in prompt tuning and P-tuning v2 for NLP tasks, as established in prior research (Liu et al., 2021), we extend the reach of our approach to encompass other methodologies. Notably, we successfully apply our framework to vision prompt tuning (VPT) (Jia et al., 2022a) and MaPLe (khattak et al., 2023), catering to the realm of multi-modal prompt learning. The incorporation of our dynamic prompting technique yields additional accuracy gains across these diverse methodologies. Furthermore, our study showcases the effectiveness of dynamic prompting not only in the single-task setting but also in multi-task and few-shot learning scenarios. This broader applicability further amplifies the potential impact of our work and solidifies its relevance in various learning paradigms.

The key contributions of this work include the following:

- We are the first to propose dynamic prompting with instance-dependent prompt position, length, and representation.
- Our innovative research has yielded a comprehensive framework for elucidating the mechanism underlying the superior performance of dynamically adjusted soft prompts in comparison to conventional prompt tuning methods.
- We conduct experiments to validate the efficacy of our methods across a wide range of tasks, including NLP tasks, vision recognition tasks, and vision-language tasks.

2 Related Work

Prompt Tuning. Prompt tuning was introduced by (Lester et al., 2021), a simple yet effective mechanism for learning "soft prompts" to condition frozen language models to perform specific downstream tasks as an alternative to prefix-tuning (Li and Liang, 2021). The parameter-efficient tuning (Liu et al., 2022; Jia et al., 2022b; Chen et al., 2022a; Liu et al., 2021) has shown powerful ability while involving a tiny amount of tunable parameters. Furthermore, (Su et al., 2022; Vu et al., 2022)

investigated the transferability of soft prompt, and (Wei et al., 2021) theoretically proved that prompt tuning obtains downstream guarantees with weaker non-degeneracy conditions. Recently, prompt tuning was also introduced to vision tasks (Jia et al., 2022a; Chen et al., 2022b; Lian et al., 2022), such as vision prompt for continual learning (Wang et al., 2022), and for image inpainting (Bar et al., 2022). Besides, prompt tuning techniques were also proposed for addressing multi-modal applications, such as vision-language applications (khattak et al., 2023; Radford et al., 2021; Zhou et al., 2022a,b; Manli et al., 2022; Jin et al., 2022).

Instance-dependent Prompt Tuning. IDPG (Wu et al., 2022a) proposes an instance-dependent prompt generation method by an up-and-down adapter module. Asai et al. (2022a) improves instance-dependent prompts by an attentional mixture of source multi-task prompts, where the source prompts are pre-trained in a multi-task way, which is also adopted in (Sun et al., 2022; Asai et al., 2022b). Besides, (He et al., 2021) gives a unified view of various parameter-efficient learning methods by looking at the attention and feedforward layers and treating prompt tuning as a simplified prefix tuning (Lester et al., 2021; Liu et al., 2021). But dynamically adjusting the soft prompt for each instance has not been fully explored, and we derive a unified framework to include various dynamic prompting methods.

3 Method

In this section, we first derive a unified view of prompt tuning in Sec. 3.1, then we describe several dynamic prompting strategies: dynamic position for concatenation with inputs, dynamic length, and dynamic representation in Sec. 3.2, as depicted in Figure 1.

3.1 A Unified View

Unlike canonical prompt tuning (Lester et al., 2021), where soft prompts are prepended to inputs, we split the prompts into two parts: *prefix* and *postfix*. Formally, for a sequence $x \in \mathbb{R}^{m\times d}$, the query matrix is $Q = xW^Q \in \mathbb{R}^{m\times d}$, the key and value matrix are $K = xW^K \in \mathbb{R}^{m\times d}$, and $V = xW^V \in \mathbb{R}^{m\times d_v}$, respectively. The soft prompt P with length l is split into two parts, $P = [P_1; P_2]$, where $P_1 \in \mathbb{R}^{l_1\times d}$ and $P_2 \in \mathbb{R}^{l_2\times d}$. The resulting new input becomes $x' = [P_1; x; P_2] \in \mathbb{R}^{(l_1+m+l_2)\times d}$, and the new key and value become $K' = x'W^K \in \mathbb{R}^{(l_1+m+l_2)\times d}$ and $V' = x'W^V \in \mathbb{R}^{(l_1+m+l_2)\times d_v}$. Here, $[;]$ denotes the concatenation operation. By matrix decomposition, we have:

$$Q' = \begin{bmatrix} Q_1 \\ Q \\ Q_2 \end{bmatrix}, K' = \begin{bmatrix} K_1 \\ K \\ K_2 \end{bmatrix}, V' = \begin{bmatrix} V_1 \\ V \\ V_2 \end{bmatrix}, \tag{1}$$

where $Q_1, K_1 \in \mathbb{R}^{l_1\times d}$, $Q_2, K_2 \in \mathbb{R}^{l_2\times d}$ and $V_1 \in \mathbb{R}^{l_1\times d_v}$, $V_2 \in \mathbb{R}^{l_2\times d_v}$.

For the new query $x' = [P_1; x; P_2]$, the attention head module becomes:

With the definition above, we can derive a unified view of prompt tuning as shown in the following formula. The detailed derivation is included in Appendix B.

$$\begin{aligned} Head = Attn\Big(&\big[[P_1; x; P_2]W^Q, \qquad (2)\\ &[P_1; x; P_2]W^K, [P_1; x; P_2]W^V\big]\Big) \qquad (3)\\ = \text{softmax}&\left(\frac{Q' * K'^T}{\sqrt{d}}\right)V' \end{aligned}$$

omitting $\sqrt{d}$ for brevity

$$\begin{aligned} &= \big[\text{softmax}(P_1W^Q K'^T)V'; \text{softmax}(xW^Q K'^T)V';\\ &\quad \text{softmax}(P_2W^Q K'^T)V'\big]. \end{aligned} \tag{4}$$

$$\begin{aligned} &Head = Attn(x', K', V') = \\ &\Big[\lambda_1 * \underbrace{Attn(Q_1, K_1, V_1)}_{\textit{prompt tuning}} + \lambda_2 * \underbrace{Attn(Q_1, K_2, V_2)}_{\textit{postfix}}\\ &\quad + (1-\lambda_1-\lambda_2) * \underbrace{Attn(Q_1, K, V)}_{\textit{prompt tuning}};\\ &\beta_1 * \underbrace{Attn(Q, K_1, V_1)}_{\textit{prompt tuning}} + \beta_2 * \underbrace{Attn(Q, K_2, V_2)}_{\textit{postfix}}\\ &\quad + (1-\beta_1-\beta_2) * \underbrace{Attn(Q, K, V)}_{\textit{standard}};\\ &\gamma_1 * \underbrace{Attn(Q_2, K_1, V_1)}_{\textit{postfix}} + \gamma_2 * \underbrace{Attn(Q_2, K_2, V_2)}_{\textit{postfix}}\\ &\quad + (1-\gamma_1-\gamma_2) * \underbrace{Attn(Q_2, K, V)}_{\textit{postfix}}\Big]. \end{aligned} \tag{5}$$

In such a unified formulation, $\{\lambda_i, \beta_i, \gamma_i\}_{i=1,2}$ are normalized weights to control how attention is distributed among the (prefix) *prompt tuning*, *postfix*, and *standard* attention. When P_2 does not exist, postfix weights are equal to 0, so the result is equivalent to standard prompt tuning. By introducing P_2, the overall $Head$ is more flexible to accommodate different query x and potentially provides additional semantics that P_1 can not capture. The theoretical analysis reveals that optimizing the position of the prompt to encompass the input can capture additional semantic information that traditional prefix or postfix prompt tuning methods

fail to capture. Our dynamic prompting is inspired by such a formulation that diversified prompts are expected for coupling with different queries.

3.2 Dynamic Prompting

In this section, we introduce how we use dynamic prompting (DP) to accommodate tuning with respect to the task- or instance-aware insertion position, length, and representation of soft prompt.

Following the ancestral prompt tuning (Lester et al., 2021), given a sequence x of n tokens, $x = \{x_1, x_2, ..., x_n\}$, a pre-trained foundation model, such as the language model T5 (Raffel et al., 2020), generates embedding of the tokens $X \in \mathbb{R}^{n\times d}$ where d is the dimension of the encoded representation. For vision prompt tuning, x is a sequence of visual hidden features (Jia et al., 2022a). The prompt tuning introduces a soft prompt $P \in \mathbb{R}^{l\times d}$ where l is the length of the soft prompt. The next step is to *prepend* the prompt P with actual inputs X into a matrix $X' = [P; X]$, then X' is fed into the model LM for optimization, where only parameters in P is optimized while the backbone LM is frozen.

Dynamic Position for Concatenation with Inputs. As noticed above, the concatenation of soft prompt and inputs is simply a prefix of P into X. However, we assume this kind of concatenation might not be the optimal strategy. Intuitively, the prefix P provides extra information for the input sequence and offers an optimized alternative, but it might not be sufficient. Thus, we propose dynamic position to fill the gap: integer $dpos$ is a parameter to be learned for different tasks or instances, then the original P can be split into two parts $P = [P_{before}, P_{after}]$, where $P_{before} = [P_1, P_2, ..., P_{dpos}]$ and $P_{after} = [P_{dpos+1}, ..., P_l]$. Thus, the new input to LM becomes

$$X' = [P_{before}; X; P_{after}], \tag{6}$$

where $dpos \in [0, l]$ is an integer to be learned and the ancestral prompt tuning is a special case when $dpos$=l. Since $dpos$ is categorical, we use a one-layer network POS_θ and the Gumbel-Softmax (Jang et al., 2017a) to optimize it. Specifically, given the output of POS_θ, $\boldsymbol{\alpha} \in \mathbb{R}^{l+1}$, we need to estimate a binary vector of the same size. A simple way to implement the binarization function is to select the position with a maximum value of $\{\alpha_0, \alpha_1, \cdots, \alpha_l\}$, however, this approach is non-differentiable. There are several ways that allow us to propagate gradients through the discrete nodes (Bengio et al., 2013). In this work, we adopt the Gumbel-Softmax sampling approach (Jang et al., 2017b; Maddison et al., 2017). Thereby, we have

$$logit = \textit{Gumbel-Softmax}(POS_\theta(x), \tau), \tag{7}$$

where τ is the annealing temperature adjusted by the total training steps as detailed in Sec. E.1. The $logit$ is an $(l+1)$-dimensional binary vector where only one element is equal to one and all other elements are zero. A detailed derivation of using Gumbel-Softmax to get the insertion position is included in Appendix C.

Previous research (Lester et al., 2021) has shown that prompt length is crucial for specific models and tasks, while going beyond 20 soft tokens gives marginal gains. We thus adopt l=20 for most experiments. In this way, the parameters of the soft prompt are the same as PT when l is fixed, making the comparison fair. The only additional parameters are brought by the small network of POS_θ with one linear layer, which is with size $d*(l+1)$. We denote this instance-dependant position selection method as *Adaptive position on instance-level*, abbreviated as *adap_ ins_pos*. Notice that for our experiments on learning an optimal position for all instances in a task, we only use a vector $v \in \mathbb{R}^{l+1}$ to learn a global best position for all instances within that task. We refer to this method as the *Adaptive Position on Task-Level* (abbreviated as *adap_ pos*). Then the number of additional parameters is $l+1$.

Dynamic Length. Previous research has shown that prompt length plays a vital role in prompt tuning (Lester et al., 2021), and larger LMs usually require a shorter prompt length. But the effect of prompt length on tasks or instances level has been underexplored. We propose that the prompt length can also be dynamically learned:

$$\begin{aligned} &P \in \mathbb{R}^{l^*\times d}, \\ &l^* = \underset{i}{\operatorname{argmin}}\, loss\Big(LM([\hat{P}_i; X] \mid \hat{P}_i \in \{\hat{P}_1, \cdots, \hat{P}_l\}, \\ &\quad \hat{P}_i \in \mathbb{R}^{i\times d})\Big). \end{aligned} \tag{8}$$

Similarly, $l^* \in [0, l]$ is also categorical and can be optimized by a one-layer network LEN_θ and Gumbel-Softmax. Here, l represents the maximum permissible length for the selection process. Also, the number of additional parameters will be $l+1$ and $d*(l+1)$ for task and instance-level, respectively. Nevertheless, implementing such a mechanism poses a practical challenge since models typically necessitate fixed input matrix dimensions. In

light of this, we employ a surrogate strategy, which is elaborated upon in Appendix D.

Dynamic Vector. Extensive evidence (Wu et al., 2022a; Asai et al., 2022a) supports the advantage of utilizing instance-dependent prompts in downstream tasks. The prompt can either be generated directly through an adapter module, as demonstrated in the study conducted by (Wu et al., 2022b), or by employing attentional mixtures during multi-task training, as illustrated in the research conducted by (Asai et al., 2022a). We propose a novel and streamlined method for generating dynamic prompts using prompt pools. This approach simplifies the process and allows for the seamless generation of dynamic prompts. Specifically, suppose there are a set of prompt pools $Pool = \{P^{(1)}, ..., P^{(k)}\}$, where k is the size of the pool. Then given any input x, we learn a small network Po_θ to get the attention score of every prompt $P^{(k)}$ with respect to x, finally the new soft prompt become:

$$P_{new} = \sum_{i=1}^{k} \beta_i * P^{(i)}, \boldsymbol{\beta} = \text{softmax}(Po_\theta(x)). \tag{9}$$

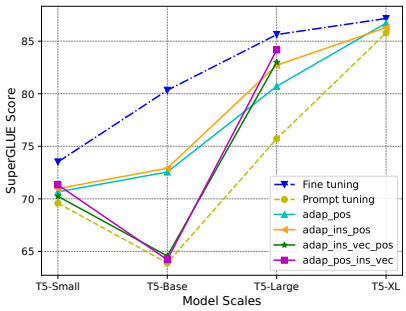
In practice, k controls the size of the prompt pool and increased parameters. Since the P_{new} depends on a specific input instance, we denote this setting as *Adaptive vector on instance-level*.

Combination. Notice that the previously mentioned methods can be combined to unleash the power of dynamic prompting further. For example, we can simultaneously update dynamic position and prompt pool together, which we denote as *Adaptive instance-vector-position*, shortened as *adap_ins_vec_pos*. Alternatively, we first use dynamic position to learn the best task-level position and update the instance-level prompt pool, denoted as *Adaptive position-instance-vector*, shortened as *adap_pos_ins_vec*. We leave more combinations for future work.

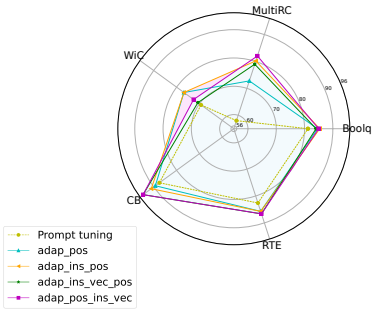
4 Experiments

Models. For language tasks, we use the OpenPrompt (Ding et al., 2022) framework for implementing our experiments, which is built on Huggincaface[1] and Pytorch [2]. We use the T5-adaptive-LM[3] version for its superiority for prompt tuning. In all our experiments, we freeze the backbone LMs and only optimize the soft prompts and learning networks for acquiring dynamic information. We choose the initial learning rate *lr* from $[0.1, 0.2, 0.3]$, weight decay to be $1e{-}5$, and Adafactor (Shazeer and Stern, 2018) as the optimizer. Besides, we use the default settings for prompt templates and verbalizers from OpenPrompt[4]. Unless otherwise stated, we keep the soft tokens to 20 for all experiments. We evaluate the validation set every 500 steps. For all fine-tuning experiments, we keep the same setting as our prompt tuning, except for initializing the *lr* to $1e{-}5$, removing the added soft prompts, and tuning the whole LMs.

To assess the efficacy of dynamically learning the optimal position of prompts to comprehensively cover the input, as discussed in Sec. 3.1, we have incorporated our dynamic optimization method into various methodologies, namely P-tuning v2(Liu et al., 2021), vision prompt tuning



(a) Average acc. across different pre-trained model sizes



(b) Performance comparison across different SuperGLUE datasets on T5-Large

Figure 2: Standard prompt tuning achieves suboptimal scores on SuperGLUE. Our dynamic prompting (*adap_***) consistently yields superior results. Fine-tuning results on T5-XL are reported in (Aribandi et al.).

[1] https://huggingface.co/

[2] https://pytorch.org/

[3] https://github.com/google-research/text-to-text-transfer-transformer/blob/main/released_checkpoints.md

[4] https://github.com/thunlp/OpenPrompt/blob/main/tutorial/1.4_soft_template.py

(VPT)(Jia et al., 2022a), and MaPLe (khattak et al., 2023) for multi-modal prompt learning. Notably, since OpenPrompt solely supports prompt addition at the input layer, we extended our experimentation beyond T5-series pretrained models to include BERT-Large and Roberta-Large models using the P-tuning v2 approach. Moreover, we have verified the effectiveness of our approach in the domain of visual recognition tasks, employing the vision prompt tuning (VPT) framework (Jia et al., 2022a) with ViT-B backbone("sup_vitb16_imagenet21k"), as well as in the context of multi-modal prompt learning for the novel class generalization task, utilizing MaPLe (khattak et al., 2023) as the underlying backbone. For vision-language model validation, we conduct the generalization from the Base-to-Novel classes task. Specifically, we evaluate the generalizability of MaPLe with our dynamic prompt insertion technique and follow a zero-shot setting where the datasets are split into base and novel classes. The model is trained only on the base classes in a few-shot setting and evaluated on base and novel categories. The backbone model is the ViT-B/16 CLIP. For further details on experimental configurations, please refer to Appendix E.

Datasets. Following previous work (Ding et al., 2022), we evaluate our approach on five SuperGLUE (Wang et al., 2019) datasets to test the language understanding ability, namely BoolQ (Clark et al., 2019), MultiRC (Khashabi et al., 2018), CB (De Marneffe et al., 2019), RTE (Giampiccolo et al., 2007), and WiC (Pilehvar and Camacho-Collados, 2019). We use the default train/dev split and report the default metric on the validation set since the test set is not directly available. For comparison with P-tuning v2, we use eight SuperGLUE datasets. For the vision prompt tuning setting, we follow (Jia et al., 2022a) and use the well-known FGVC datasets consisting of 5 benchmarked datasets. For the vision-language setting, we follow MaPLe (khattak et al., 2023) and use 11 different datasets. Details on the datasets are included in Appendix E.6.

5 Results

To demonstrate the efficiency of our method, in this section, we show that our proposed simple yet powerful approach leads to substantial accuracy gains across various methodologies, including Prompt tuning, P-tuning v2 (Liu et al., 2021), vision prompt tuning (VPT) (Jia et al., 2022a), and MaPLe (khattak et al., 2023) for multi-modal prompt learning, across a wide range of tasks, such as NLP tasks, vision recognition tasks, and vision-language tasks. More experiments, such as the case and ablation study (F), parameter sensitivity analysis (G.6), and additional results (G) are included in the Appendix.

Adaptive Position. As presented in Table 1, we compare two variations of adaptive position: $adap\_pos$ represents the dynamically learned position for all instances in a single task, while $adap\_ins\_pos$ indicates that an optimal position is expected to exist for each instance. The experiments are conducted using the T5-LM-Adapt version (Small, Base, Large, and XL), and we report the best results in the table. We can see a general trend that *adap_ins_pos* > *adap_pos* > *fixed_pos* on almost all five datasets[5]. On average, T5-Large demonstrates substantial improvements of approximately 5 and 7 points compared to the fixed position PT. These improvements are less pronounced for smaller LMs, aligning with the findings of (Lester et al., 2021) that larger models are better suited for prompt tuning. Considering that the number of additional parameters for *adap_pos* is merely 20, while several thousand are required for *adap_ins_pos*, we can conclude that the fixed position is suboptimal for prompt tuning, and adaptive position consistently provides gains.

Adaptive Length. As mentioned in Sec. 3.2, we only use a surrogate strategy for length adjustment. Table 3 shows the adaptive length results. For simplicity, we only test $adap\_length$ on T5-base and $adap\_ins\_length$ on T5-large. Overall, adjusting length on task or instance-level helps, compared with fixed prompt length. However, compared with the adaptive position strategy in Table 1, the performance gain is lower, which might be caused by the difficulty of tuning. Thus, we recommend this strategy for quickly locating the proper length for different models instead of a greedy search.

Adaptive Prompt. By adaptively adjusting the synthesized prompt from the prompt pool, the soft prompts are expected to more efficiently utilize the frozen LMs. The results are reported in Table 2, and we also give the histogram comparison in Figure 3. In general, compared with adaptive position only, adding an adaptive prompt vector increases the performance. But when both position and prompts are

[5]The T5-Base sometimes demonstrate suboptimal results, as also reported in (Asai et al., 2022a)

Table 1: Three strategies for the dynamic position of soft prompts. Fixed Position is the default prompt tuning. Adaptive Position means the position is learned for every task but fixed for all instances within a task, while Adaptive ins_position learns a dynamic position for each instance.

Dataset	T5-LM-Small			T5-LM-Base			T5-LM-Large			T5-LM-XL		
	Fixed Position	Adaptive Position	Adaptive Ins_Position	Fixed Position	Adaptive Position	Adaptive Ins_Position	Fixed Position	Adaptive Position	Adaptive Ins_Position	Fixed Position	Adaptive Position	Adaptive Ins_Position
Boolq	67.31	67.55	**67.61**	62.35	**69.88**	69.17	81.20	84.60	**85.35**	89.02	88.89	**89.16**
MultiRC	68.68	68.89	**69.29**	57.42	70.19	**71.08**	58.00*	72.77	**80.20**	**84.49**	84.31	84.41
WiC	62.69	66.14	**68.34**	53.61	64.42	**64.89**	69.30	**71.20**	**71.20**	**72.57**	71.22	70.91
CB	**83.93**	**83.93**	**83.93**	78.57	**87.50**	**87.50**	87.50	89.29	**91.07**	94.64	**98.21**	96.43
RTE	65.34	**66.79**	65.70	67.51	70.75	**71.93**	82.60	**85.71**	**85.71**	88.21	**90.94**	90.58
Avg.	69.59	70.66	**70.97**	63.89	72.55	**72.91**	75.72	80.71	**82.71**	85.79	**86.72**	86.30

Table 2: Compared with only adjusting position in Table 1, combining together with the adaptive vector can further close the gap between fine-tuning.

Dataset	T5-LM-Small			T5-LM-Base			T5-LM-Large		
	Adaptive Ins_vec_pos	Adaptive Pos_ins_vec	Fine tuning	Adaptive Ins_vec_pos	Adaptive Pos_ins_vec	Fine Tuning	Adaptive Ins_vec_pos	Adaptive Pos_ins_vec	Fine Tuning
Boolq	67.40	**68.04**	71.02	**62.51**	62.39	81.33	84.07	**84.98**	87.25
MultiRC	68.92	**69.12**	69.58	**57.96**	57.65	77.91	78.96	**82.03**	85.85
WiC	66.30	**66.61**	65.25	60.97	**64.29**	69.18	70.69	**72.57**	73.82
CB	82.14	**85.71**	92.86	**80.36**	75.00	94.62	**94.64**	**94.64**	94.64
RTE	66.42	**67.15**	68.84	61.01	**61.73**	78.62	**86.64**	**86.64**	86.59
Avg.	70.24	**71.33**	73.51	**64.56**	64.21	80.33	83.00	**84.17**	85.63

Table 3: Fixed length PT v.s. adaptive length.

Dataset	T5-LM-Base		T5-LM-Large	
	Fixed Length	Adaptive Length	Fixed Length	Adaptive Ins_Length
Boolq	62.35	**67.28**	81.20	**83.46**
MultiRC	**57.41**	57.34	58.00	**66.30**
WiC	53.61	**60.50**	69.30	**71.47**
CB	78.57	**80.36**	**87.50**	84.32
RTE	67.51	**68.32**	**82.60**	79.78
Avg.	63.89	**66.76**	75.72	**77.07**

Table 4: Few-shot results on T5-LM-Large.

Dataset	T5-LM-Large				
	Fixed Position	Adaptive Position	Adaptive Ins_vec	Adaptive Ins_vec_pos	Adaptive Pos_ins_vec
Boolq	54.37	60.24	60.18	**61.07**	60.64
MultiRC	53.49	55.26	54.52	56.13	**56.79**
WiC	52.66	53.29	53.92	55.33	**55.33**
CB	76.79	78.57	73.21	80.36	**87.50**
RTE	51.26	55.96	54.43	54.87	**57.40**
Avg.	58.55	60.66	59.05	61.55	**63.53**

optimized for each instance, we see slightly lower results in most cases, which might be caused by the increasing difficulty of optimization. We leave the work of better optimization methods for future work. Additional results are in Appendix G.1.

Few-Shot. We illustrate the few-shot results in Table 4 on T5-Large. As we can see from all datasets, our dynamic prompting consistently improves the results given only 32-shot training examples, demonstrating the broad generalization ability under the low-resource regime.

Multi-Task. To furthermore demonstrate that multi-task tuning could benefit across tasks for learning a better shared prompt pool, we also show multi-task results in Table 6. Here we randomly sample 10% or 30% samples from all five datasets and report the average performance. The results confirm that sharing a prompt pool across multiple tasks brings universal benefits.

P-tuning V2. In the pursuit of pushing the boundaries of prompt tuning, (Liu et al., 2021) delved deeper into the inner workings of language models (LMs) by attaching a prompt to each transformer layer, transcending the conventional practice of simply affixing a soft prompt to the original input sequence. Remarkably, their efforts yielded

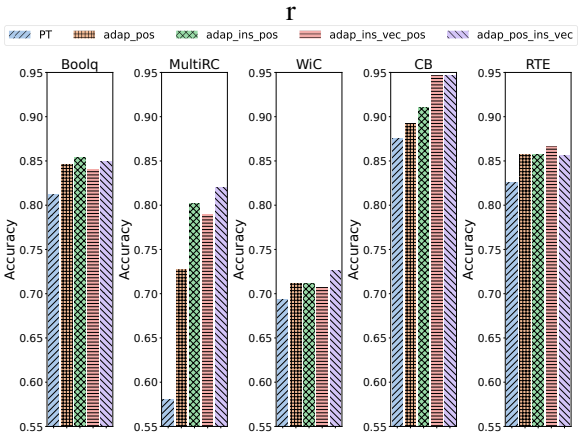
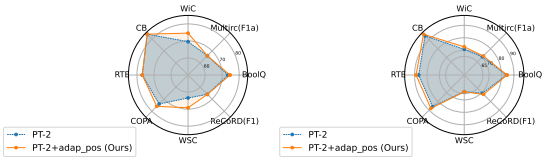


Figure 3: Results on SuperGLUE with T5-Large.



(a) BERT-Large (b) RoBERTa-Large

Figure 4: Performance comparison of PT-2 and PT-2 with adaptive position (ours) on SuperGLUE with different PLMs.

comparable results to fine-tuning, highlighting the remarkable potential of prompt tuning. We thus embarked on our investigation to explore the full potential of dynamic prompting. We sought to ascertain the extent to which our dynamic manipulation of soft prompts, reaching deeper into the LM, could enhance accuracy. To ensure a fair comparison, we adopted the identical setup employed by (Liu et al., 2021), employing the backbone models BERT-Large(Devlin et al., 2018) and RoBERTa-Large(Liu et al., 2019) on the SuperGlue datasets. The results, as depicted in Figure 4, demonstrate the efficacy of our adaptive prompt position approach. Across most datasets, we observe performance improvements. Our technique achieves an impressive absolute average gain of 1.74% on BERT-Large and 1.34% on the RoBERTa-Large over P-tuning

Table 5: Comparison between basic VPT model and VPT with the adaptive position.

	Dataset	CUB	NABirds	Flowers	Dogs	Cars	Avg.
VPT-Shallow	prompt length	100	50	100	100	100	
	VPT	85.42	75.11	98.29	90.42	54.60	80.77
	VPT+*adap_pos*	86.26	75.56	98.44	91.27	57.02	81.71
		+0.84	+0.45	+0.15	+0.85	+0.42	+0.94
	VPT+*adap_ins_pos*	86.31	76.63	98.52	91.39	58.13	82.20
		+0.89	+1.52	+0.23	+0.97	+3.53	+1.43
VPT-Deep	prompt length	10	50	5	5	100	
	VPT	87.81	81.43	98.91	90.57	82.99	88.34
	VPT+*adap_pos*	88.06	82.98	98.99	91.27	83.26	88.91
		+0.25	+1.55	+0.08	+0.70	+0.27	+0.57
	VPT+*adap_ins_pos*	**88.15**	**83.02**	**99.01**	**91.32**	**83.42**	88.98
		+0.34	+1.59	+0.10	+0.75	+0.43	+0.64

Table 6: Multi-task results comparing prompt tuning (PT) and *adap_ins_vec* on T5-Large under few-shot setting.

Methods	**k=8**	
	10%-shot	30%-shot
PT	67.15	69.79
adap_ins_vec	**68.59**	**72.56**

Table 7: Comparison of MaPLe with and without dynamic prompt position averaged over 11 datasets.

Method	Base Acc.	Novel Acc.	HM (Base+Novel)
MaPLle	83.74	73.64	77.08
MaPLe+*adap_pos* (Ours)	83.96	75.68	79.25
	+0.22	+2.04	+2.17

V2. Theoretically, as expounded upon in Sec. 3.1, our technique enables soft prompts to encompass the input, capturing additional semantic information that traditional prefix or postfix prompt tuning methods fail to capture. The effectiveness of our adaptive position becomes increasingly apparent as we manipulate prompts across more transformer layers. More results are included in Appendix G.3.

Vision Prompt Tuning (VPT). In the realm of adapting large pre-trained Transformers for downstream vision tasks, a remarkable piece of work known as VPT (Jia et al., 2022a) has emerged. VPT integrated additional parameters into the input sequence of each Transformer layer, simultaneously learned alongside a linear head during the fine-tuning process. Our methodology takes a step further by embedding our approach within the model, allowing for adaptive optimization of the prompt position. When applying VPT in a deep setting, we maintain the prompt position of the deep transformer layers consistent with that of the input layer. The results, as depicted in Table 5, demonstrate the performance gains achieved by intelligently optimizing the prompt position. The instance-aware prompt position selection further improves accuracy. Remarkably, these benefits manifest across both shallow and deep settings of VPT, underscoring the robustness and efficacy of our approach. More results are included in Appendix G.4.

Table 8: Study on the learned parameters on different tasks.

Methods	**T5-LM-Small**					**T5-LM-Large**				
	CB	RTE	Boolq	WiC	MultiRC	CB	RTE	Boolq	WiC	MultiRC
Learned position	4	2	3	4	9	2	3	3	6	4

Vision-Language Model. The Vision-language (V-L) model, such as the remarkable CLIP (Radford et al., 2021), has garnered widespread acclaim for its exceptional ability to align language and vision modalities. The pioneering work of MaPLe (khattak et al., 2023) introduced a coupling function to effectively condition vision prompts based on their language counterparts, bridging the gap between the two modalities. Inspired by these advancements, we incorporate our adaptive prompt position approach into the text input layer, leveraging the power of dynamic prompt manipulation. We outline the detailed experimental settings in the Appendix. The compelling results, as summarized in Table 7, substantiate the potency of our approach. Remarkably, by incorporating adaptive prompt position into MaPLe, we achieve an impressive absolute average gain of 2.04% on novel classes and 2.17% on the harmonic mean, surpassing the state-of-the-art method MaPLe (khattak et al., 2023). This performance improvement serves as a compelling testament to the effectiveness of our dynamic prompting methodology, firmly establishing its efficacy in the realm of V-L models. More detailed results are included in Appendix G.5.

Case Study on the Learned Position. to evaluate the efficacy of the learned positions and lengths in our method across various tasks. The learned parameters are applied universally across all input instances within a given dataset. The findings in Table 8 reveal that the optimal length or position is uniquely influenced by the specific input task. Additionally, the model size proves to be a significant factor affecting the results. Consequently, there are no universally applicable strategies for selecting these parameters; each dataset needs to align with the characteristics of the specific model employed. In Appendix F, we also show a case study of learning input-specific information. These results substantiate the effectiveness of our proposed strategy.

6 Conclusion

In this work, we first derive a unified view of prompt tuning and then present a novel dynamic

prompting approach that can significantly improve the performance of prompt tuning while adding only a few additional parameters. The key contributions of this work include exploring the effectiveness of the dynamic position, length, and prompt representation in improving traditional prompt tuning and systematically exploring dynamic prompting under the combination of different dynamic methodologies in various scenarios. Comprehensive experiments on a broad spectrum of datasets validate that dynamic prompting consistently achieves superior results across a diverse range of tasks, including language understanding, vision recognition, and vision-language tasks, with varying model sizes. We also demonstrate that dynamic prompting is effective in multi-task and few-shot settings. Overall, our work can further unleash the power of prompt tuning across various modalities to close the gap between fine-tuning.

Limitations

One potential limitation of our method is that it inevitably increases the complexity, compared with original prompt tuning. However, we argue that our work focuses on exploring the ultimate power of prompt tuning. And to make our work reproducible, we also upload our codes together with our submission.

Ethics Statement

We believe our work is conformant to the ACL Code of Ethics. All models and datasets used in our experiments are open-sourced. We did not see any obvious ethical impact of this work.

References

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. Ext5: Towards extreme multi-task scaling for transfer learning. In *International Conference on Learning Representations*.

Akari Asai, Mohammadreza Salehi, Matthew E. Peters, and Hannaneh Hajishirzi. 2022a. Attempt: Parameter-efficient multi-task tuning via attentional mixtures of soft prompts. In *Conference on Empirical Methods in Natural Language Processing*.

Akari Asai, Mohammadreza Salehi, Matthew E. Peters, and Hannaneh Hajishirzi. 2022b. Attentional mixtures of soft prompt tuning for parameter-efficient multi-task knowledge sharing. *ArXiv*, abs/2205.11961.

Amir Bar, Yossi Gandelsman, Trevor Darrell, Amir Globerson, and Alexei A. Efros. 2022. Visual prompting via image inpainting. *Advances in Neural Information Processing Systems (NeurIPS)*.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Guanzheng Chen, Fangyu Liu, Zaiqiao Meng, and Shangsong Liang. 2022a. Revisiting parameter-efficient tuning: Are we really there yet? In *Conference on Empirical Methods in Natural Language Processing*.

Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. 2022b. Adaptformer: Adapting vision transformers for scalable visual recognition. *36th Conference on Neural Information Processing Systems (NeurIPS)*.

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 the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Haitao Zheng, and Maosong Sun. 2022. OpenPrompt: An open-source framework for prompt-learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 105–113, Dublin, Ireland. Association for Computational Linguistics.

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

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2022. 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. Association for Computational Linguistics.

Xu Guo, Boyang Albert Li, and Han Yu. 2022. Improving the sample efficiency of prompt tuning with domain adaptation. In *Conference on Empirical Methods in Natural Language Processing*.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*.

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

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685.

Eric Jang, Shixiang Gu, and Ben Poole. 2017a. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Eric Jang, Shixiang Gu, and Ben Poole. 2017b. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. 2022a. Visual prompt tuning. In *European Conference on Computer Vision (ECCV)*.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge J. Belongie, Bharath Hariharan, and Ser Nam Lim. 2022b. Visual prompt tuning. *ArXiv*, abs/2203.12119.

Woojeong Jin, Yu Cheng, Yelong Shen, Weizhu Chen, and Xiang Ren. 2022. A good prompt is worth millions of parameters: Low-resource prompt-based learning for vision-language models. In *Association for Computational Linguistics (ACL)*.

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 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. Association for Computational Linguistics.

Muhammad Uzair khattak, Hanoona Rasheed, Muhammad Maaz, Salman Khan, and Fahad Shahbaz Khan. 2023. Maple: Multi-modal prompt learning. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

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.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. 2022. Scaling & shifting your features: A new baseline for efficient model tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *ArXiv*, abs/2205.05638.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *ArXiv*, abs/2110.07602.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Fang Ma, Chen Zhang, Lei Ren, Jingang Wang, Qifan Wang, Wei Yu Wu, Xiaojun Quan, and Dawei Song. 2022. Xprompt: Exploring the extreme of prompt tuning. In *Conference on Empirical Methods in Natural Language Processing*.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. *International Conference on Learning Representations*.

Shu Manli, Nie Weili, Huang De-An, Yu Zhiding, Goldstein Tom, Anandkumar Anima, and Xiao Chaowei. 2022. Test-time prompt tuning for zero-shot generalization in vision-language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word

representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*.

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. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR.

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. 2022. 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*, pages 3949–3969, Seattle, United States. Association for Computational Linguistics.

Tianxiang Sun, Zhengfu He, Qinen Zhu, Xipeng Qiu, and Xuanjing Huang. 2022. Multi-task pre-training of modular prompt for few-shot learning. *ArXiv*, abs/2210.07565.

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 the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5039–5059, Dublin, Ireland. Association for Computational Linguistics.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. 2022. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149.

Colin Wei, Sang Michael Xie, and Tengyu Ma. 2021. Why do pretrained language models help in downstream tasks? an analysis of head and prompt tuning. *Advances in Neural Information Processing Systems*, 34:16158–16170.

Zhuofeng Wu, Sinong Wang, Jiatao Gu, Rui Hou, Yuxiao Dong, VG Vydiswaran, and Hao Ma. 2022a. Idpg: An instance-dependent prompt generation method. *arXiv preprint arXiv:2204.04497*.

Zhuofeng Wu, Sinong Wang, Jiatao Gu, Rui Hou, Yuxiao Dong, V.G.Vinod Vydiswaran, and Hao Ma. 2022b. IDPG: An instance-dependent prompt generation method. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5507–5521, Seattle, United States. Association for Computational Linguistics.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. 2022a. Conditional prompt learning for vision-language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. 2022b. Learning to prompt for vision-language models. *International Journal of Computer Vision (IJCV)*.

Appendix

A Broader Impacts and Limitations

Due to the extensive workload of experiments, we only test our methods for classification tasks from the SuperGLUE benchmark. Additional text generation tasks could be exploited in future work. Also, our conclusions are drawn from the encoder-decoder architecture such as T5, BERT and RoBERTa models, vision pre-trained model ViT-B and a pre-trained ViT-B/16 CLIP model. And it is worth investigating whether dynamic prompting still holds for decoder-only GPT. Besides, our approach introduces additional parameters like anneal temperature or learning network, which could increase the difficulty of optimization.

B Derivation of the Unified View of Prompt Tuning

For the new query $x' = [P_1; x; P_2]$, the attention head module becomes:

$$\textit{Head} = \textit{Attn}\left([P_1; x; P_2]W^Q,\ [P_1; x; P_2]W^K, [P_1; x; P_2]W^V\right) \tag{10}$$

$$= \text{softmax}\left(\frac{Q' * K'^T}{\sqrt{d}}\right)V' \tag{11}$$

omitting $\sqrt{d}$ for brevity

$$= \left[\text{softmax}(P_1W^QK'^T)V'; \text{softmax}(xW^QK'^T)V'; \text{softmax}(P_2W^QK'^T)V'\right], \tag{12}$$

where

$$\text{softmax}(P_1W^QK'^T)V'$$

$$= \text{softmax}\left(P_1W^Q[K_1^T; K^T; K_2^T]\right)\begin{bmatrix}V_1\\V\\V_2\end{bmatrix} \tag{13}$$

$$= \lambda_1 * \text{softmax}(Q_1K_1^T)V_1 + \lambda_2 * \text{softmax}(Q_1K_2^T)V_2 + (1-\lambda_1-\lambda_2) * \text{softmax}(Q_1K^T)V \tag{14}$$

$$= \lambda_1 * Attn(Q_1, K_1, V_1) + \lambda_2 * Attn(Q_1, K_2, V_2) + (1-\lambda_1-\lambda_2) * Attn(Q_1, K, V), \tag{15}$$

where λ_1 and λ_2 are normalized weights:

$$\lambda_1 = \frac{\sum_i exp(Q_1K_1^T)_i}{\sum_{j=1}^2\sum_i exp(Q_1K_j^T)_i + \sum_i exp(Q_1K^T)_i}, \quad \lambda_2 = \frac{\sum_i exp(Q_1K_2^T)_i}{\sum_{j=1}^2\sum_i exp(Q_1K_j^T)_i + \sum_i exp(Q_1K^T)_i}.$$

$$\text{softmax}(xW^QK'^T)V'$$

$$= \text{softmax}\left(xW^Q[K_1^T; K^T; K_2^T]\right)\begin{bmatrix}V_1\\V\\V_2\end{bmatrix} \tag{16}$$

$$= \beta_1 * \text{softmax}(QK_1^T)V_1 + \beta_2 * \text{softmax}(QK_2^T)V_2 + (1-\beta_1-\beta_2) * \text{softmax}(QK^T)V \tag{17}$$

$$= \beta_1 * Attn(Q, K_1, V_1) + \beta_2 * Attn(Q, K_2, V_2) + (1-\beta_1-\beta_2) * Attn(Q, K, V), \tag{18}$$

where β_1 and β_2 are normalized weights:

$$\beta_1 = \frac{\sum_i exp(QK_1^T)_i}{\sum_{j=1}^2\sum_i exp(QK_j^T)_i + \sum_i exp(QK^T)_i}, \quad \beta_2 = \frac{\sum_i exp(QK_2^T)_i}{\sum_{j=1}^2\sum_i exp(QK_j^T)_i + \sum_i exp(QK^T)_i}.$$

$$\text{softmax}(p_2 W^Q K'^T) V'$$

$$= \text{softmax}\left(p_2 W^Q [K_1^T; K^T; K_2^T]\right) \begin{bmatrix} V_1 \\ V \\ V_2 \end{bmatrix} \tag{19}$$

$$= \gamma_1 * \text{softmax}(Q_2 K_1^T) V_1 + \gamma_2 * \text{softmax}(Q_2 K_2^T) V_2 + (1 - \gamma_1 - \gamma_2) * \text{softmax}(Q_2 K^T) V \tag{20}$$

$$= \gamma_1 * Attn(Q_2, K_1, V_1) + \gamma_2 * Attn(Q_2, K_2, V_2) + (1 - \gamma_1 - \gamma_2) * Attn(Q_2, K, V), \tag{21}$$

where γ_1 and γ_2 are normalized weights:

$$\gamma_1 = \frac{\sum_i exp(Q_2 K_1^T)_i}{\sum_{j=1}^2 \sum_i exp(Q_2 K_j^T)_i + \sum_i exp(Q_2 K^T)_i}, \quad \gamma_2 = \frac{\sum_i exp(Q_2 K_2^T)_i}{\sum_{j=1}^2 \sum_i exp(Q_2 K_j^T)_i + \sum_i exp(Q_2 K^T)_i}.$$

Finally, we can write them together:

$$\begin{aligned} Head &= Attn(x', K', V') \\ &= \Bigg[\lambda_1 * \underbrace{Attn(Q_1, K_1, V_1)}_{\textit{prompt tuning}} + \lambda_2 * \underbrace{Attn(Q_1, K_2, V_2)}_{\textit{postfix}} + (1 - \lambda_1 - \lambda_2) * \underbrace{Attn(Q_1, K, V)}_{\textit{prompt tuning}}; \\ &\beta_1 * \underbrace{Attn(Q, K_1, V_1)}_{\textit{prompt tuning}} + \beta_2 * \underbrace{Attn(Q, K_2, V_2)}_{\textit{postfix}} + (1 - \beta_1 - \beta_2) * \underbrace{Attn(Q, K, V)}_{\textit{standard}}; \\ &\gamma_1 * \underbrace{Attn(Q_2, K_1, V_1)}_{\textit{postfix}} + \gamma_2 * \underbrace{Attn(Q_2, K_2, V_2)}_{\textit{postfix}} + (1 - \gamma_1 - \gamma_2) * \underbrace{Attn(Q_2, K, V)}_{\textit{postfix}} \Bigg]. \end{aligned} \tag{22}$$

C Dynamic Insertion Position with Gumbel-Softmax

We use the Gumbel-Max trick (Maddison et al., 2017) to dynamically decide insertion position for soft prompts. Specifically, to decide $dpos$, we have $(l + 1)$ positions to choose. Let $\{\alpha_0, \cdots, \alpha_l\}$ represent the log probabilities $\{\log(p_0), \cdots, \log(p_l)\}$ of different insertion positions. $\boldsymbol{\alpha}$ is the output logit of the network POS_θ. Thus, we can draw samples in the following way: we first draw i.i.d samples $\{g_0, \cdots, g_l\}$ from a Gumbel distribution, i.e., $g = -\log(-\log(z)) \sim Gumbel$, where $z \sim$ Uniform(0, 1). Then we produce the discrete sample by adding g to introduce stochasticity:

$$dpos = \arg\max_i [\alpha_i + g_i], i \in \{0, \cdots, l\}. \tag{23}$$

$$\mu_i = \exp\left((\alpha_i + g_i)/\tau\right) / \left(\sum_{\hat{i}=0}^{l} \exp\left((\alpha_{\hat{i}} + g_{\hat{i}})/\tau\right)\right), i \in \{0, \cdots, l\} \tag{24}$$

The $\arg\max$ operation is non-differentiable, but we can use the softmax as a continuously differentiable approximation to it (Eq. (24)). τ is the temperature to control the discreteness. Thus, we use the $\arg\max$ to make the discrete selection on the forward pass, while approximating it with softmax on the backward pass, which is called the straight-through estimator (Jang et al., 2017b).

D Dynamic Length

The huggingface transformers [6] implemented the attention mask mechanism by giving infinite minus value to padded tokens so that the calculated attention score will reach zero.

$$Attention = \text{softmax}\left(\frac{Q * K^T}{\sqrt{d_k}} + M\right) * V, \tag{25}$$

[6] https://huggingface.co/

where

$$M = \begin{cases} 0, & \text{mask=1} \\ -\infty, & \text{mask=0.} \end{cases} \tag{26}$$

It seems natural to just treat truncated prompts as padding tokens with $mask = 0$. However, when the prompt length l_i is dynamically updated in input instance x_i, the logits returned by Gumbel-Softmax can not be directly applied to the attention mask matrix M since M can not provide gradients. Therefore, we adopt a surrogate strategy by

$$P_{new} = [0 * P_{before}; P_{after}], \tag{27}$$

where $P_{before} \in \mathbb{R}^{(l-l_i)\times d}$ and $P_{after} \in \mathbb{R}^{l_i \times d}$. In this way, although the attention score after $\mathrm{softmax}$ is $\frac{e^0}{\sum_i (q_i * k_i)}$ rather than 0, the corresponding value in V is 0. Such implementation is not optimal, but we stick to it for simplicity.

E Experimental Settings

E.1 Implementation details

We find the initialization of prompts could have influential effects on the final performance. Initializing from the vocabulary of LMs almost always gives better results. We thus follow the default setting in OpenPrompt using the list of embeddings in front of the token vocabulary as the initialization of soft prompt vectors. Setting warm-up steps to 500 yields consistent gains for the small, base, and large models. However, for T5-XL, a warm-up step of 500 does not lead to convergence, and we reduce it to 10 steps. Also, we only run limited experiments with T5-XL due to the computational resource constraint.

We empirically find the annealing temperature is very important for Gumbel-Softmax (Jang et al., 2017a) to behave well. We follow (Jang et al., 2017b), and adjust the annealing temperature by

$$\tau = max(\tau * exp(-\frac{\gamma * iterations}{step}), 0.5), \tag{28}$$

where initial τ is set to 1.0 and the annealing rate $\gamma \in \{3e^{-7}, 3e^{-5}, 3e^{-3}\}$ and the $step$ is picked from $[0.1, 1, 10, 30, 100, 200, 600]$. In practice, the Gumbel-Softmax simulates more closely to the true categorical distribution when τ is reaching 0.5.

Hardware. We use NVIDIA A40 48 GB for T5-XL and RTX 6000 24 GB for all other models.

E.2 Model hyperparameters for experiments on T5-series models

For all one-layer networks for learning instance-dependent dynamic position, length, or representation information, we adopt a single linear layer followed by ReLU activation. The input embedding dimension d is 512, 768, 1024, 2048 for T5-small, base, large, and XL, respectively. We illustrate the details of our additional parameters in Table 9. The batch size is 32 and 16 for dynamic prompting and finetuning. All inputs are truncated to a maximum of 480 tokens. For each method, we tune the learning temperature via grid search in the range $\{10^{-0.5}, 10^{-1}, 10^{-1.5}, 10^{-2}, 10^{-2.5}, 10^{-3}\}$ to obtain the best performances. k is set to 8 for all settings. In all settings except for comparison on varying length l, we set l=20.

Table 9: Tuned parameters in our Dynamic Prompting methods. PTs are soft prompts and Θ_{NN} is the learning network. d is the input vocabulary embedding size of LMs. l is the prompt length. k is the number of prompts in the prompt pool.

Methods	**Number of parameters**			
	Fixed Position	Adaptive pos	Adaptive ins_pos	Adaptive ins_vec
PTs	$l * d$	$l * d$	$l * d$	$(l+1) * d * k$
Θ_{NN}	0	$l + 1$	$d * (l + 1)$	$d * k$

Table 10: Parameter setting for P-tuning V2 related evaluation.

Dataset	BERT-Large					RoBERTa-Large				
	batch_size	lr	dropout	prompt length	#epoch	batch_size	lr	dropout	prompt length	#epoch
BoolQ	32	5.00E-03	0.1	40	100	16	7.00E-03	0.1	8	100
MultiRC (F1a)	16	5.00E-03	0.1	40	100	16	7.00E-03	0.1	8	100
WiC	16	1.00E-04	0.1	20	80	32	1.00E-02	0.1	8	50
CB	32	5.00E-03	0.1	40	100	16	7.00E-03	0.1	8	100
RTE	16	1.00E-02	0.1	20	60	32	5.00E-03	0.1	128	100
COPA	16	1.00E-02	0.1	16	80	8	9.00E-03	0.1	8	120
WSC	16	5.00E-03	0.1	20	80	16	1.00E-02	0.1	8	10
ReCoRD (F1)	20	5.00E-03	0.1	40	100	16	7.00E-03	0.1	8	100

E.3 Model hyperparameters for experiments on P-tuning V2(Liu et al., 2021)

Since we intend to test if our approach can lead to accuracy gains when our dynamic insertion position, as discussed in Sec. 3.1, is embedded in different prompt learning frameworks, we use the default setting in the GitHub repo[7] of P-tuning V2(Liu et al., 2021). In the repo, the scripts for datasets cb, multirc, and record are missing. The detailed setting is summarized in Table 10.

E.4 Model hyperparameters for experiments on VPT(Jia et al., 2022a)

For all datasets, we use the default setting in the repo[8]. The pre-trained backbone we used is ViT-B ("sup_vitb16_imagenet21k"). SGD with momentum 0.9, base learning rate 0.25 and weight_decay 0.001 is used as the optimizer. The batch size is set to 32. The random seed is 0.

E.5 Model hyperparameters for experiments on MaPLe(khattak et al., 2023)

We follow the setting in (khattak et al., 2023) repo[9] and use a few-shot training strategy in all experiments at 16 shots which are randomly sampled for each class. For each dataset, we run three times using seed values 1, 2, and 3, and report the average accuracy. We apply prompt tuning on a pretrained ViT-B/16 CLIP model. We set prompt depth J to 9 and the language and vision prompt lengths to 4. All models are trained for 12 epochs with a batch size of 4 and a learning rate of 0.0035 via SGD optimizer. We initialize the language prompts of the first layer with the pretrained CLIP word embeddings of the template category "for sure This is a photo of <category>". This is different from the original setting in MaPLe which use only prompt length 2 with initialization using "a photo of a <category>", as we need to leave room for our algorithm to select a good insertion position. The setting for the MaPLe baseline also follows the same setting for a fair comparison. Since the task depends on the calculation of similarity between text and image embeddings, and there is no instance-dependent information, we only validate the $adap\_pos$ setting.

E.6 Details of datasets

Following previous work (Ding et al., 2022), we evaluate our approach on five SuperGlue (Wang et al., 2019) datasets to test the natural language understanding ability, namely BoolQ (Clark et al., 2019), MultiRC (Khashabi et al., 2018), CB (De Marneffe et al., 2019), RTE (Giampiccolo et al., 2007), and WiC (Pilehvar and Camacho-Collados, 2019). We use the default train/dev/test split and report the default metric on the validation set since the test set is not directly available. For comparison with P-tuning V2, we also use SuperGlue datasets. For the vision prompt tuning setting, we follow (Jia et al., 2022a) and use the well-known FGVC benchmark datasets consisting of 5 benchmarked Fine-Grained Visual Classification tasks including CUB-200-2011, NABirds, Oxford Flowers, Stanford Dogs, and Stanford Cars. For the vision-language setting, we follow MaPLe (khattak et al., 2023) and use 11 datasets including Caltech101, OxfordPets, StanfordCars, Flowers102, Food101, FGVCAircraft, SUN397, UCF101, DTD, EuroSAT, and ImageNet-R.

[7] https://github.com/THUDM/P-tuning-v2
[8] https://github.com/KMnP/vpt
[9] https://github.com/muzairkhattak/multimodal-prompt-learning

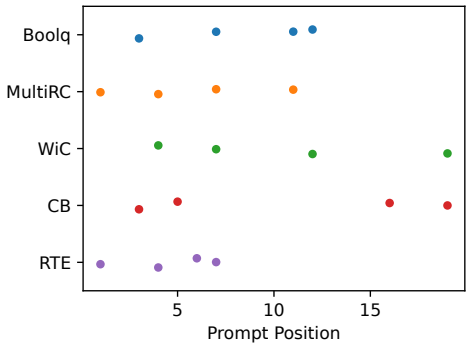


Figure 5: Comparison of learned best position on 5 datasets under 4 random seeds.

Table 11: Ablation study on the effects of sentence representation.

Methods	T5-LM-Small		T5-LM-Large	
	CB	RTE	CB	RTE
T5-LMs	83.93	66.79	89.29	85.71
Vocabulary	83.39	66.53	88.91	85.29

F Case and Ablation Study

In this section, we conduct a case study to explore how the learned position differs from each other and an ablation study to investigate the influence of original prompt length and instance representation.

Adaptive Position. We first look at how the learned dynamic position differs from each other across five datasets. We run three experiments on T5-small and record the final optimal position on each task, where the initial prompt length is fixed at 20. Figure 5 shows that the most suitable position varies across tasks and individual runs. This suggests that the optimal position depends on the specific tasks and prompt representations. There is no one-for-all solution.

Influence of Input Representation. Since our instance-dependent dynamic prompting requires generating the sentence representation, where we run a forward of the LMs to get $LM(X)$. In IDPG (Wu et al., 2022a), the authors use the Glove (Pennington et al., 2014) embedding as the sentence representation and obtain close prompt tuning performance. And they also suggest caching for applications in downstream tasks. But we believe those methods add additional complexity for deployment. To overcome the drawback of feeding one instance twice into the LMs, we try an alternative method: use the initial vocabulary representation from T5 of the input sentence as the input for the learning networks to generate dynamically learned prompts. We perform $adap\_ins\_pos$ experiments on RTE and CB datasets in T5-small and T5-large. The results are shown in Table 11. As we can see, the performance is almost maintained for all cases. Our operation does not introduce additional models or complexity, thus suitable for various downstream tasks.

Influence of Prompt Length. We conduct an ablation study to investigate whether our dynamic position approach still works when the total prompt length varies. As shown in Table 12 and Appendix G.2, we first perform a greedy search over $L = [2, 4, 8, 16, 20, 32]$ to find the optimal prompt length. Then based on the searched best length, we run different dynamic position experiments. The results show that our adaptive position consistently increases the performance when L is set to 32. Besides, combining a prompt pool (adaptive vector) can further improve the results. In particular, when the L is set to only 4 for T5-Large, our adaptive position can still maintain superiority on most datasets. These results show that our dynamic position can universally improve dynamic prompting for varying prompt lengths, even if L is extremely small. Recall the unified view in Sec. 3.1, and these results validate the hypothesis that prepended prompts are not enough, and a slight change leads to significant improvements.

Case Study of Learning Input-specific Information. We also conduct a case study on two different examples from MultiRC with $adapt\_ins\_pos\_vec$, using T5-large-LM. We set the number of prompt

pools to be 4. We illustrate the learned scores of each prompt in the prompt pool by P_i score, $i \in \{1, 2, 3, 4\}$. We also report the learned position for each prompt. For those two different inputs, as shown in Figure 6, we can clearly see that the model learnt input-specific information.

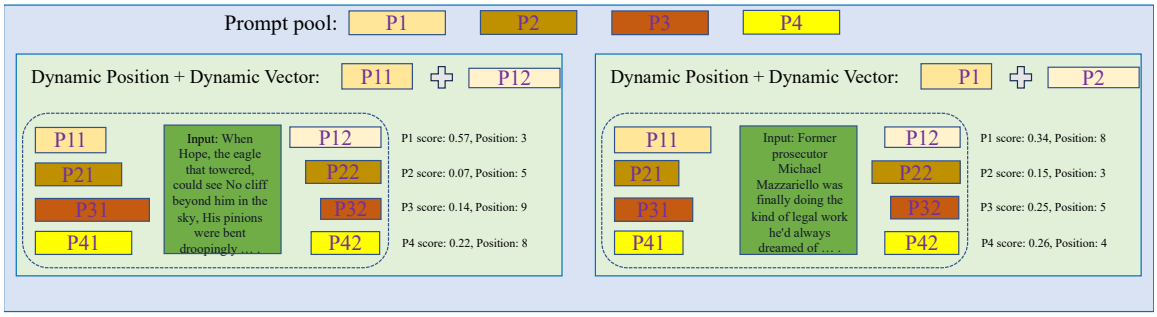


Figure 6: A case study on two different examples from MultiRC with adaptive_instance_position_vector, using T5-large-LM.

G Additional Results

G.1 Additional comparison of adaptive vector

Here we show the additional results for T5-Small in Figure 7, and T5-Base in Figure 8. We observe a similar trend as T5-Large in Sec. 5, but T5-Base demonstrates worse results.

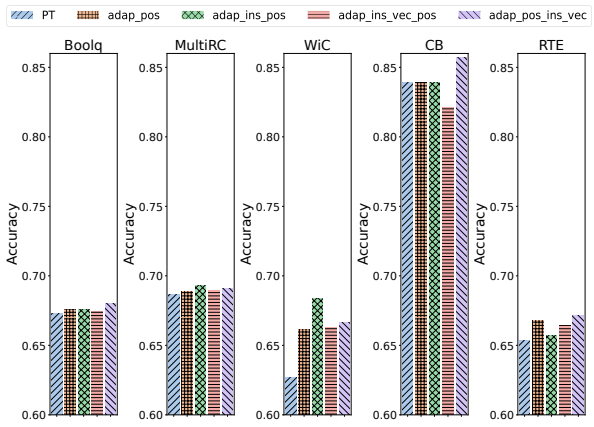


Figure 7: Results on SuperGLUE with T5-Small.

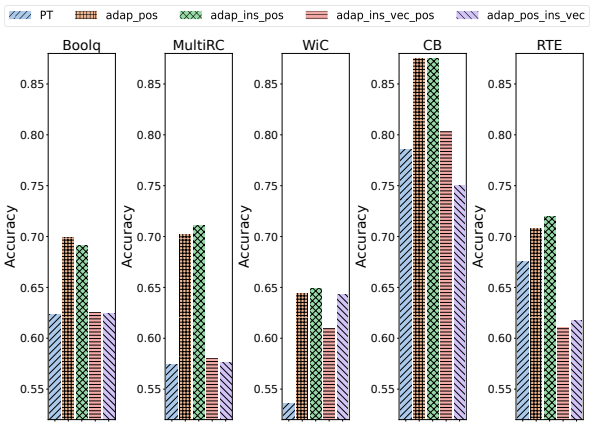


Figure 8: Results on SuperGLUE with T5-Base.

G.2 Additional influence of prompt length

In this section, we present the comprehensive results obtained from T5-Small (Table 12), T5-Base (Table 13), and T5-Large (Table 14) models, respectively. To ensure a thorough analysis, we conducted experiments using different prompt lengths, namely 2, 4, 8, 16, 20, and 32. Our aim was to identify the optimal prompt length for each model size across the five SuperGlue datasets and validate the effectiveness of our technique over the best run of baseline prompt tuning. For T5-Small, we discovered that a prompt length of 32 consistently yielded the highest average accuracy across the SuperGlue datasets. Consequently, for this particular model size, we proceeded to employ our approaches, adapting the prompt position and representation in an adaptive manner. The prompt lengths of 8 and 4 were adopted for T5-Base and T5-Large models respectively.

An in-depth analysis of the results reveals a remarkable trend across all T5 models. Our dynamic prompting techniques consistently outperformed the best run achieved by traditional prompt tuning with a fixed length. This compelling observation unequivocally illustrates the effectiveness and prowess of our approach. Furthermore, as the model size increased from T5-Small to T5-Base and T5-Large, we

Table 12: The left column shows the results of prompt tuning under different lengths, and L=32 performs the best. The right column shows different dynamic position strategies by using the L=32.

Dataset	T5-LM-Small						T5-LM-Small			
	Fixed L=2	Fixed L=4	Fixed L=8	Fixed L=16	Fixed L=20	Fixed L=32	Adaptive Position	Adaptive Ins_pos	Adaptive Ins_vec_pos	Adaptive Pos_ins_vec
Boolq	62.20	62.96	65.23	67.25	67.31	**67.61**	67.00	68.35	67.83	**67.95**
MultiRC	66.17	67.57	68.42	68.30	**68.69**	68.67	68.83	68.38	69.27	**69.41**
WiC	61.29	61.76	63.95	63.48	62.69	**64.11**	65.20	66.14	65.83	**68.85**
CB	75.00	83.93	82.14	83.93	83.93	**87.50**	**91.07**	85.71	91.07	89.29
RTE	61.37	64.98	66.79	**68.95**	65.34	68.23	67.15	**67.87**	66.79	66.43
Avg.	65.21	68.24	69.31	70.38	69.59	**71.22**	71.85	71.29	72.16	**72.39**

Table 13: The left column shows the results of prompt tuning under different lengths, and L=8 performs the best. The right column shows different dynamic position strategies by using the L=8.

Dataset	T5-LM-Base						T5-LM-Base			
	Fixed L=2	Fixed L=4	Fixed L=8	Fixed L=16	Fixed L=20	Fixed L=32	Adaptive Position	Adaptive Ins_position	Adaptive Ins_vec_pos	Adaptive Pos_ins_vec
Boolq	62.32	65.02	62.20	**70.18**	62.35	62.26	67.95	68.04	**68.41**	66.18
MultiRC	**58.07**	57.49	56.93	57.88	57.43	57.32	69.25	68.13	65.08	**69.35**
WiC	61.12	60.50	60.19	**63.64**	53.61	52.66	65.20	**66.14**	65.36	63.32
CB	85.71	78.57	**94.64**	80.36	78.57	82.14	87.50	87.50	**92.86**	89.29
RTE	59.57	57.04	**68.23**	67.15	67.51	55.60	**68.59**	68.59	60.29	67.51
Avg.	65.36	63.72	**68.44**	67.84	63.89	62.00	**71.70**	71.68	70.40	71.13

Table 14: The left column shows the results of prompt tuning under different lengths, and $L = 4$ performs the best. The right column shows different dynamic position strategies by using the $L = 4$.

Dataset	T5-LM-Large						T5-LM-Large			
	Fixed L=2	Fixed L=4	Fixed L=8	Fixed L=16	Fixed L=20	Fixed L=32	Adaptive Position	Adaptive Ins_position	Adaptive Ins_vec_pos	Adaptive Pos_ins_vec
Boolq	79.08	81.87	82.57	75.11	81.20	**84.80**	81.83	80.15	81.50	**81.87**
MultiRC	67.66	76.13	75.64	**79.19**	58.00	68.52	**79.08**	70.15	75.39	73.64
WiC	57.84	67.87	65.05	**70.22**	69.30	69.75	67.87	67.40	**71.47**	69.75
CB	**96.43**	89.29	80.36	82.14	87.50	82.14	87.50	87.50	**98.21**	96.43
RTE	74.73	77.62	82.31	**84.48**	82.60	82.67	80.86	77.26	**84.12**	81.23
Avg.	75.15	**78.55**	77.18	78.23	75.72	77.58	79.43	76.49	**82.14**	80.58

observed a corresponding increase in the accuracy gain facilitated by our dynamic prompting techniques. This empirical evidence reaffirms the potency of our methodology as model complexity grows.

The findings presented here provide substantial evidence of the superiority of our dynamic prompting techniques over traditional fixed-length prompt tuning. This knowledge empowers researchers and practitioners alike to leverage the full potential of dynamic prompts, unlocking new avenues for improved performance in various natural language processing tasks.

G.3 Additional results on P-tuning V2

To ensure a fair comparison, we adopted the identical setup employed by Tang et al. (Liu et al., 2021), employing the backbone models BERT-Large(Devlin et al., 2018) and RoBERTa-Large(Liu et al., 2019) on the SuperGlue datasets. The detailed results of the two models on the SuperGlue dataset are depicted in Table 15. The obtained results serve as a testament to the efficacy of our adaptive insertion position approach. Remarkably, significant performance improvements are observed across the majority of datasets. The underlying theoretical foundation, as elaborated in Sec. 3.1, sheds light on the essence of our technique. By enabling soft prompts to encompass the input, we are able to capture supplementary semantic information that conventional prefix or postfix prompt tuning methods fail to capture.

As we delve deeper into the manipulation of prompts across multiple transformer layers, the effec-

Table 15: Comparison of PT-2 with Adaptive Position and original PT-2 on SuperGLUE datasets.

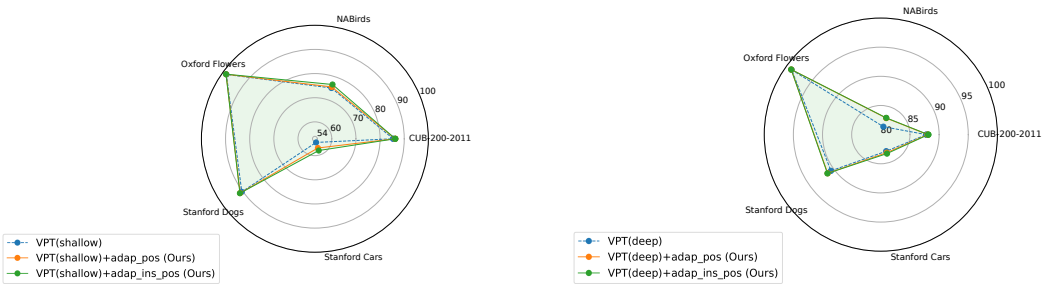
	BERT-Large		RoBERTa-Large	
	PT-2	PT-2+*adap_pos* (Ours)	PT-2	PT-2+*adap_pos* (Ours)
BoolQ	73.46	74.62	84.07	84.46
MultiRC (F1a)	66.05	66.05	70.72	71.76
WiC	69.59	74.61	70.69	72.73
CB	83.93	83.93	94.64	96.43
RTE	76.90	76.90	86.64	88.81
COPA	74.00	76.00	86.00	88.00
WSC	63.46	69.23	63.46	63.46
ReCoRD(F1)	66.02	66.05	70.72	72.01
Avg.	71.68	73.42 +1.74	78.37	79.71 +1.34

tiveness of our dynamic insertion position approach becomes increasingly apparent. This observation highlights the inherent power and adaptability of our methodology, as it successfully exploits the intricacies and hierarchies within the transformer architecture to enhance performance. By dynamically optimizing the insertion position of prompts, we can tap into additional layers of contextual understanding, enabling our approach to surpass traditional methods.

Overall, the results demonstrate that our dynamic insertion position approach is a promising avenue for enhancing the performance of downstream tasks. It offers a novel perspective on prompt adaptation, leveraging the strengths of pre-trained transformers to capture a more comprehensive representation of the input data.

G.4 Additional results on vision prompt tuning (VPT)

In the realm of adapting large pre-trained Transformers for downstream vision tasks, a remarkable piece of work known as Vision Prompt Tuning (VPT) has emerged. We thus further include our methods in the VPT framework. We follow the same setting as in (Jia et al., 2022a) and optimize insertion position on both shallow and deep settings. The results are depicted in Figure 9, which demonstrates the performance gains achieved by dynamically optimizing the insertion position for prompts. Remarkably, these benefits manifest across both shallow and deep settings of VPT, underscoring the robustness and efficacy of our approach.



(a) VPT-shallow (b) VPT-deep

Figure 9: Comparison of basic VPT model and VPT with adaptive position (ours) on different datasets.

G.5 Additional results on vision language modeling.

Vision-language (V-L) models, such as the remarkable CLIP, have drawn significant attention recently. As the pioneering work of MaPLe (khattak et al., 2023) introduced a coupling function to effectively condition vision prompts based on their language counterparts, it bridges the gap between the vision and text modalities. Here, we also incorporate our adaptive insertion position approach into the text input layer, leveraging the power of dynamic prompt manipulation. We summarize our results in detail in Table 16 and Table 17, which substantiates the potency of our approach. By incorporating adaptive insertion position into MaPLe, we achieve an impressive absolute average gain of 2.04% on novel classes' average accuracy

Table 16: Comparison of vision language model prompt tuning results between basic MaPLe model and MaPLe with adaptive position (ours) average over 11 datasets.

Dataset	MaPLe					MaPLe + *adap_pos* (Ours)				
	Base		Novel		HM	Base		Novel		HM
	Acc.	Macro_F1	Acc.	Macro_F1	(base+novel Acc.)	Acc.	Macro_F1	Acc.	Macro_F1	(base+novel Acc.)
stanford_cars	76.43	75.70	71.70	69.97	73.99	76.50	75.87	72.83	71.37	74.62
caltech101	98.37	96.77	93.57	93.43	95.91	98.40	96.83	94.00	93.63	96.15
oxford_pets	95.23	95.23	97.17	97.17	96.19	95.47	95.47	97.43	97.43	96.44
oxford_flowers	97.53	97.37	71.00	65.50	82.17	97.07	96.90	72.53	67.47	83.02
food-101	89.87	89.83	90.50	90.50	90.18	90.03	90.03	90.83	90.80	90.43
fgvc-aircraft	39.83	37.53	24.47	21.13	21.54	40.17	37.70	33.40	28.70	36.34
SUN397	81.50	81.27	76.83	76.03	79.09	81.57	81.37	77.13	76.30	79.29
DTD	79.97	79.77	52.10	50.87	62.94	81.07	81.00	55.43	53.53	65.71
eurosat	91.40	91.40	69.27	65.33	78.76	92.27	92.27	73.50	71.30	81.63
UCF101	84.20	83.17	78.37	76.20	81.12	84.27	83.20	78.67	76.53	81.35
imagenet-r	86.80	85.03	85.10	85.03	85.94	86.73	85.03	86.73	85.03	86.73
Avg.	83.74	83.01	73.64	71.92	77.08	**83.96**	**83.24**	**75.68**	**73.83**	**79.25**

Table 17: Comparison with MaPLe on base-to-novel generalization. The adaptive position of the prompt on MaPLe will improve generalization performance over existing methods on 11 recognition datasets. Absolute gains over basic MaPLe are indicated in blue.

(a) stanford_cars

Method	Base Acc.	Novel Acc.	HM (Base+Novel)
MaPLe	76.43	71.70	73.97
MaPLe+*adap_pos*	76.50	72.83	74.61
	+0.07	+1.13	+0.64

(b) caltech101

Method	Base Acc.	Novel Acc.	HM (Base+Novel)
MaPLe	98.37	93.57	95.91
MaPLe+*adap_pos*	98.40	94.00	96.15
	+0.03	+0.43	+0.24

(c) oxford_pets

Method	Base Acc.	Novel Acc.	HM (Base+Novel)
MaPLe	95.23	97.17	96.19
MaPLe+*adap_pos*	95.46	97.43	96.44
	+0.23	+0.26	+0.25

(d) oxford_flowers

Method	Base Acc.	Novel Acc.	HM (Base+Novel)
MaPLe	97.53	71.00	82.17
MaPLe+*adap_pos*	97.07	72.53	83.02
	-0.46	+1.53	+0.85

(e) food-101

Method	Base Acc.	Novel Acc.	HM (Base+Novel)
MaPLe	89.87	90.50	90.18
MaPLe+*adap_pos*	90.03	90.83	90.43
	+0.16	+0.33	+0.25

(f) fgvc_aircraft

Method	Base Acc.	Novel Acc.	HM (Base+Novel)
MaPLe	39.83	24.47	21.54
MaPLe+*adap_pos*	40.17	33.40	36.34
	+0.34	+8.93	+14.80

(g) SUN397

Method	Base Acc.	Novel Acc.	HM (Base+Novel)
MaPLe	81.50	76.83	79.09
MaPLe+*adap_pos*	81.57	77.13	79.29
	+0.07	+0.30	+0.20

(h) DTD

Method	Base Acc.	Novel Acc.	HM (Base+Novel)
MaPLe	79.97	52.10	62.94
MaPLe+*adap_pos*	81.07	55.43	65.71
	+1.10	+3.33	+2.77

(i) eurosat

Method	Base Acc.	Novel Acc.	HM (Base+Novel)
MaPLe	91.40	69.27	78.76
MaPLe+*adap_pos*	92.27	73.50	81.63
	+0.87	+4.23	+2.87

(j) UCF101

Method	Base Acc.	Novel Acc.	HM (Base+Novel)
MaPLe	84.20	78.37	81.12
MaPLe+*adap_pos*	84.27	78.67	81.35
	+0.07	+0.30	+0.23

(k) imagenet-r

Method	Base Acc.	Novel Acc.	HM (Base+Novel)
MaPLe	86.80	85.10	85.94
MaPLe+*adap_pos*	86.73	86.73	86.73
	-0.07	+1.63	+0.79

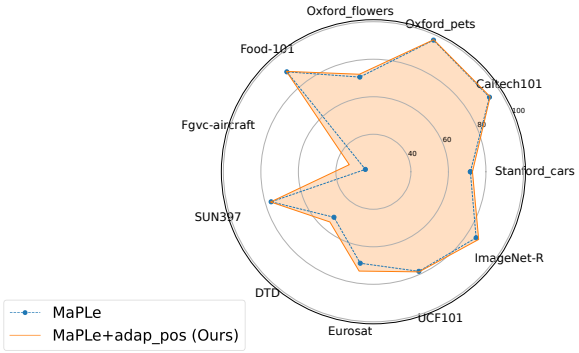


Figure 10: Comparison of MaPLe with dynamic prompt position and original MaPLe average over 11 datasets on the vision-language pretrained model prompt tuning for novel class generalization task. MaPLe with our adaptive position surpasses the original MaPLe on 11 diverse image recognition datasets for novel class generalization tasks.

and 2.17% on the harmonic mean (HM) averaged over 3 runs (seeds) of both base and novel classes respectively. This substantial enhancement in performance stands as a compelling testament, providing evidence of the effectiveness and potency of our dynamic prompting methodology. It firmly establishes our approach as a powerful ingredient in the realm of Vision-Language (V-L) models, significantly elevating their capabilities and pushing the boundaries of what can be achieved.

G.6 Parameter Sensitivity Analysis

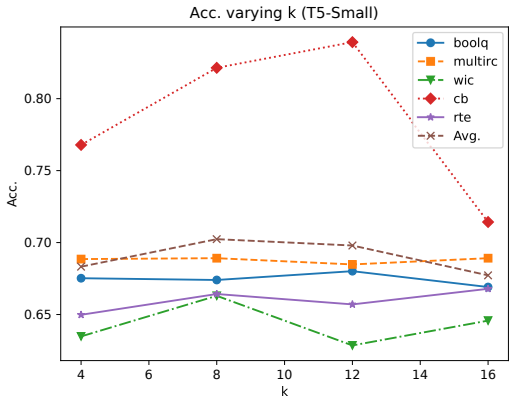


Figure 11: Parameter Sensitivity of k.

We also analyzed the hyperparameters of our approach in this section. Basically, our approach only has one hyper-parameter, i.e., k for *adap_pos_ins_vec* and *adap_ins_pos_vec*. Different choices of the number of mixture soft-prompts do not significantly impact the model performance. Using T5-Small as an example, we present the accuracy with respect to k in Figure 11. The figure shows that our algorithm is not very sensitive to hyperparameter k. In our setting, we thus empirically set $k = 8$ for all experiments.

G.7 Different Optimizers and Learning Rates

We adopt the default setting on learning rate (lr) and optimizer in openprompt, that is Adafactor with lr 0.3 for SuperGlue experiments. For PV2, VPT and MaPLe experiments, we follow their original settings for fair comparison. As an example of the sensitivity of learning rate and optimizer, we use the T5-Large on Boolq as an example, the results are shown in Table 18 and Table 19.

G.8 Standard Deviation

All the reported results in the paper, including both baseline models and our approaches, represent the average accuracy over 20 individual runs. In Table 1 and Table 5, to ensure reproducibility, we adhered to the default settings of openprompt including using a fixed random seed of 144 and initializing prompts

lr	0.05	0.1	0.2	0.3
adap pos	82.19	83.21	84.72	84.6
adap Ins_Length	82.65	83.59	84.12	83.46

Table 18: Adafactor.

lr	0.05	0.1	0.2	0.3
adap pos	82.89	83.98	84.12	84.52
adap Ins_Length	82.12	82.92	83.01	83.11

Table 19: AdamW.

with the embeddings of the top indexed tokens in the vocabulary. The Standard Deviation (SD) scores are very small as shown in Table 20 and 21.

	T5-Small			T5-Base			T5-Large			T5-XL		
	Fixed Pos	Adap Pos	Adap Ins_Pos	Fixed Pos	Adap Pos	Adap Ins_Pos	Fixed Pos	Adap Pos	Adap Ins_Pos	Fixed Pos	Adap Pos	Adap Ins_Pos
Boolq	0.03	0.02	0.04	0.05	0.12	0.15	0.06	0.08	0.11	0.02	0.01	0.04
MultiRC	0.01	0.02	0.03	0.06	0.11	0.13	0.04	0.07	0.08	0.03	0.02	0.03
WiC	0.03	0.04	0.05	0.06	0.09	0.11	0.03	0.08	0.08	0.02	0.04	0.03
CB	0.01	0.01	0.01	0.08	0.14	0.15	0.06	0.06	0.09	0.03	0.04	0.05
RTE	0.02	0.02	0.03	0.04	0.06	0.09	0.05	0.06	0.07	0.01	0.02	0.02

Table 20: Standard Deviation (SD) Score of Table 1.

	Data	**CUB1**	**NABirds**	**Flowers**	**Dogs**	**Cars**
VPT-Shallow	length	100	50	100	100	100
	VPT	0.02	0.02	0.01	0.01	0.02
	VPT+adap_pos	0.04	0.02	0.01	0.02	0.04
	VPT+adap_ins_pos	0.05	0.06	0.02	0.03	0.07
VPT-Deep	length	10	50	5	5	100
	VPT	0.01	0.03	0.01	0.01	0.01
	VPT+adap_pos	0.02	0.04	0.01	0.02	0.01
	VPT+adap_ins_pos	0.03	0.04	0.02	0.02	0.02

Table 21: Standard Deviation (SD) Score of Table 5.

G.9 Comparison to Adaptor Approaches

Below we demonstrate the comparison with (Houlsby et al., 2019; Wu et al., 2022a) on three datasets from SuperGLUE. We only use T5-base as a testbed and follow the setting in (Wu et al., 2022a) for implementations. As we can see from Table 22, ours achieve substantial improvements over (Wu et al., 2022a). Besides, although the Adapter(Houlsby et al., 2019) outperforms ours, it requires million-level parameters to be tuned, compared with only several thousand parameters in prompt tuning. So overall, our novel design shows obvious advantages and can potentially be combined with those methods for further improvements.

Method	**MultiRC**	**WiC**	**CB**	**Avg.**
Adapter(Houlsby et al., 2019)	75.9	67.1	85.7	76.2
IDPG(Wu et al., 2022a)	70.5	64.4	84.9	73.2
Ours	71.1	64.9	87.5	74.5

Table 22: Comparison to Adaptor Approaches.

G.10 Comparison to Grid-search and Hyper-parameter Optimization Approaches

Our approach is a holistic framework rooted in theoretical analysis. Specifically, the instance-aware insertion position and length selection cannot be considered traditional hyperparameters suitable for Bayesian optimization or grid search. Each instance optimizes its unique insertion position, making it unsuitable as a hyperparameter. Still, We would like to elaborate on some comparison results with

Bayesian optimization and grid search approaches treating the insertion position as a hyperparameter in this section. As an example, we only conduct experiment with T5-Base. Similar observations can be found in other backbone models. For the traditional grid search, combined with our unique position selection idea, we keep the length to 20 and adjust the position to be 1, 4, 8, 12, 16, and 19. For Bayesian optimization, we use Bayesian with Gaussian Process. The comparison results are shown in Table 23. As evident from the table, our adoption of the Gumbel-softmax approach has demonstrated superior performance compared to both grid search and Bayesian optimization. However, it is crucial to reiterate that our primary contribution lies in validating the efficacy of insertion position in various scenarios, including task-dependent and instance-dependent situations. We aim to showcase the effectiveness of dynamically manipulating soft prompts, both theoretically and empirically. In essence, we propose a comprehensive prompt tuning framework. Besides, when the prompt length is very large such as 50 or 100 (like that in VPT experiments), the traditional optimization methods that treat insertion position as an additional hyper-parameter require significantly more time and resources. On the contrary, our adopted Gumbel-softmax does not have such a limit.

Table 23: Comparison to Grid-Search and Hyper-parameter Optimization Approaches (T5-Base).

Dataset	**baseline(fixed position)**	**grid search**	**Bayesian**	**Ours**($adapt\_pos$)
Boolq	62.35	64.79	68.21	**69.88**
MultiRC	57.42	66.87	65.74	**70.19**
WiC	53.61	61.29	63.96	**64.42**
CB	78.57	85.20	86.02	**87.50**
RTE	67.51	68.58	68.61	**70.75**

G.11 Running Time Comparison

We also validate the computational time of the proposed approaches. As illustrated in Table 9 in the appendix, the number of additional parameters is very small. For example, $adap\_pos$ only introduces l (typically < 20) new parameters compared with the vallina prompt tuning. Therefore, the computational costs are very similar to vanilla prompt tuning. Taking the experiments on Maple for example, we report the running time for training and testing in Table 24 (Quadro RTX 6000 with 24576MiB, 1 GPU. #Epoch=12).

Table 24: Running Time (Seconds) Comparison with MaPLe and MaPLe with Adaptive Position.

	MaPLe		MaPLe+adap_pos	
	train	test	train	test
caltech101	89.23	24.45	93.11	25.32
food101	214.34	79.47	220.65	82.12
ucf101	106.54	31.11	113.29	33.18