Self-PT: Adaptive Self-Prompt Tuning for Low-Resource Visual Question Answering

Bowen Yuan
yuanbw0925@gmail.com
Nanjing University of Posts and Telecommunications
Nanjing, China

Sisi You
ssyou@njupt.edu.cn
Nanjing University of Posts and Telecommunications
Nanjing, China

Bing-Kun Bao*
bingkunbao@njupt.edu.cn
Nanjing University of Posts and Telecommunications
Nanjing, China
Peng Cheng Laboratory
Shenzhen, China

ABSTRACT

Pretraining and finetuning large vision-language models (VLMs) have achieved remarkable success in visual question answering (VQA). However, finetuning VLMs requires heavy computation, expensive storage costs, and is prone to overfitting for VQA in low-resource settings. Existing prompt tuning methods have reduced the number of tunable parameters, but they cannot capture valid context-aware information during prompt encoding, resulting in **1) poor generalization of unseen answers and 2) lower improvements with more parameters**. To address these issues, we propose a prompt tuning method for low-resource VQA named Adaptive Self-Prompt Tuning (Self-PT), which utilizes representations of question-image pairs as conditions to obtain context-aware prompts. To enhance the generalization of unseen answers, Self-PT uses dynamic instance-level prompts to avoid overfitting the correlations between static prompts and seen answers observed during training. To reduce parameters, we utilize hyper-networks and low-rank parameter factorization to make Self-PT more flexible and efficient. The hyper-network decouples the number of parameters and prompt length to generate flexible-length prompts by the fixed number of parameters. While the low-rank parameter factorization decomposes and reparameterizes the weights of the prompt encoder into a low-rank subspace for better parameter efficiency. Experiments conducted on VQA v2, GQA, and OK-VQA with different low-resource settings show that our Self-PT outperforms the state-of-the-art parameter-efficient methods, especially in lower-shot settings, *e.g.*, 6% average improvements cross three datasets in 16-shot. Code is available at https://github.com/NJUPT-MCC/Self-PT.

CCS CONCEPTS

• **Computing methodologies → Computer vision**.

*Corresponding author

MM '23, October 29-November 3, 2023, Ottawa, ON, Canada

ACM ISBN 979-8-4007-0108-5/23/10…$15.00
https://doi.org/10.1145/3581783.3612222

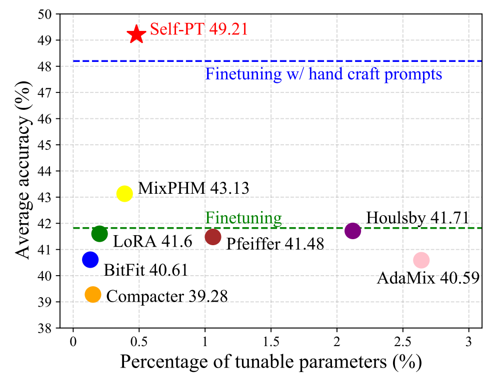

Figure 1: Comparison between parameter-efficient methods for low-resource VQA with 16 training samples. We show the average score across five seeds on VQA v2 and the percentage of tunable parameters w.r.t. pretrained VL-T5. The green dashed line represents direct fine-tuning and the blue dashed line represents fine-tuning method from FewVLM.

KEYWORDS

Low-Resource VQA, Adaptive Self-Prompt Tuning, Parameter-Efficient Tuning

ACM Reference Format:
Bowen Yuan, Sisi You, and Bing-Kun Bao. 2023. Self-PT: Adaptive Self-Prompt Tuning for Low-Resource Visual Question Answering. In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23), October 29-November 3, 2023, Ottawa, ON, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3581783.3612222

1 INTRODUCTION

Visual question answering (VQA) aims to infer a precise answer from the given question-image pair. In recent years, pretraining and finetuning vision-language models (VLMs) have achieved state-of-the-art performance in VQA [4, 5, 20–23, 41, 48, 50]. Due to the large numbers of parameters in VLMs, finetuning VLMs leads to high computational and storage costs, and is prone to overfitting in low-resource settings where training data size is smaller than 1,000 [3, 17, 53].

Recently, parameter-efficient tuning methods have been proposed to tune VLMs by adjusting lightweight trainable parameters while keeping most pretrained parameters frozen [13, 14, 17, 19, 25, 32, 42, 47]. Prompt tuning is one of the favorite paradigms in parameter-efficient tuning, which concatenates trainable prompt tokens and the inputs of each block to enable few-shot learning in downstream tasks, *e.g.,* natural language understanding and generation [6, 25] and image classification [16]. The general prompt tuning methods [25] concatenate static prompt tokens and all inputs. Recent studies show that directly updating the trainable tokens leads to unstable optimization and performance drops [24, 45]. To solve the above issues, they thus leverage a prompt encoder, e.g., an MLP [24, 25], to reparameterize the token embeddings. However, existing static prompt methods would cause two main issues: **1) poor generalization of unseen answers, 2) lower improvements with more parameters.** Firstly, existing methods tend to implicitly correlate prompts and answers that have been observed, making the static prompts overfit to those seen answers. To be specific, the static prompts always forget essential general answers unseen in the training data, called catastrophic knowledge forgetting, thus leading to a poor generalization of unseen answers. Secondly, in low-resource settings, the static prompts cannot capture the full complexity of the task to learn a robust task-level prompt, *e.g.,* "Answer the following question: [Question]," thus the embedding capacity of the prompt encoder is underutilized. Therefore, more parameters to the prompt encoder cannot learn more valid information, which results in lower improvements due to the underutilized prompt.

To address the above issues, a feasible idea is constructing robust context-aware prompts in low-resource VQA, which has the following advantages: **1) strong generalization of unseen answers, 2) well utilization of prompt embedding**. Firstly, context-aware prompts can adapt the model to unseen samples in the image classification tasks [43, 44, 53]. Therefore, we can use the instance-level context as a condition to encode proper conditional prompts for unseen answers, thus improving the generalization ability of prompts. Secondly, context-aware prompts can capture the complex relationships between question-image pairs and answers to construct well-utilized prompt embeddings. Moreover, low-rank methods and hyper-networks can be utilized to achieve higher improvements with lower parameters based on the well-utilized prompt embeddings, which improves the parameter efficiency of prompt tuning. Therefore, the context-aware prompts can improve the generalization of unseen answers and enable parameter-efficient prompt tuning, which should be well explored in low-resource VQA.

To construct robust context-aware prompts, we propose an Adaptive Self-Prompt Tuning (Self-PT) method to learn the dynamic context information, as illustrated in Fig 1. To enhance the generalization of unseen answers, Self-PT utilizes instance-level representations of question-image pairs as conditions to obtain context-aware prompts, which are free from implicit correlations between static prompts and seen answers. Specifically, Self-PT uses learnable key-value pairs to search proper prompts for given conditions, enabling the adaptation ability to provide accurate answers for unseen samples. To reduce parameters and improve performance, we utilize hyper-network and low-rank parameter factorization to make Self-PT more flexible and efficient. The hyper-network allows simple prompt index information as input to provide weights for encoding each prompt token so that it can decouple the number of parameters from prompt length. The low-rank parameter factorization decomposes and reparameterizes the embedding layer weights into a low-rank subspace, which yields better performance with fewer parameters. With these design considerations, our Self-PT capitalizes context-aware prompts with dramatically fewer tunable parameters for more precise answers in low-resource VQA.

Our contributions are summarized as follows:

- We propose Adaptive Self-Prompt Tuning (Self-PT) which utilizes instance-level multimodal representations as conditions to obtain context-aware prompts. Moreover, Self-PT provides appropriate instructions to unseen samples, thus improving the generalization ability.
- We explore low-rank and parameter-reused strategies to construct a parameter-efficient Self-PT method. Specifically, we employ hyper-network to decouple the number of parameters and the prompt length, making Self-PT more flexible. We utilize low-rank parameter factorization to decompose and reparameterize the weights, making Self-PT more parameter-efficient.
- Experiments conducted on VQA v2, GQA, and OK-VQA with different low-resource settings show that our Self-PT outperforms the state-of-the-art parameter-efficient methods, especially in lower-shot settings, *e.g.,* 6% average improvements cross three datasets in 16-shot.

2 RELATED WORK

2.1 Vision-Language Pretraining and Finetuning

Vision-language pretraining has gained popularity as it can learn generalized multimodal representations, thus significantly improving downstream task performance [4, 5, 20–23, 41, 48, 50]. Recent multimodal pretraining methods use the encoder-decoder framework to unify different tasks into a sequence-to-sequence paradigm [4, 21–23, 40]. Specifically, they employ generative modeling objectives and use task-specific prompts in the pretraining or finetuning stage, such as "vqa:" [4, 18, 38]. Therefore, VQA can be considered a generative task that generates answers based on images and questions. Some recent studies in VQA have observed that VLMs with unified encoder-decoder architectures exhibit better generalization ability [21, 40]. By designing appropriate prompts to instruct a unified multimodal pretrained model, we can significantly reduce computation costs compared to conventional finetuning. Specifically, we utilize instance-level multimodal representations as conditions to obtain context-aware prompts for low-resource VQA.

2.2 Parameter-Efficient Tuning

While finetuning large pretrained models on downstream tasks can significantly improve performance, it is computationally intensive and requires expensive storage costs. To address this issue, researchers in natural language processing (NLP) have proposed parameter-efficient tuning methods [13, 14, 19, 25, 32, 42, 47] that can tune lightweight trainable parameters while keeping most of the pretrained parameters frozen. These methods can be split into two groups, depending on whether new trainable parameters are introduced: (1) tuning partial parameters of VLMs, such as BitFit [47] and FISH Mask [37], and (2) tuning additional parameters, such as prompt-tuning [25, 26], adapter [13, 19, 32, 42], and low-rank methods [14, 51]. All of these methods have shown effectiveness in various NLP tasks.

Inspired by these methods in NLP, recent studies [28, 36, 45, 52] have introduced these techniques to tune pretrained VLMs for multimodal tasks. VL-Adapter [36] explores shared and unshared adapters for vision-language multitask learning, while Yang et al. [45] investigate prompt tuning methods for generative VLMs. Uniadapter [28] proposes unified unimodal and multimodal adapters for video QA and retrieval tasks. HyperPELT [52] proposes a unified parameter-efficient framework that uses a shared hyper-network [29] to prepare weights for lightweight additional modules. These methods demonstrate the ability to approach or even exceed finetuning in most multimodal tasks. Lately, MixPHM [17] propose an adapter where up- and down-sample layers are implemented by multiple PHM linear layers [49] in a mixture of experts manner for low-resource VQA. MixPHM surpasses finetuning in all varieties of low-resource settings with few tunable parameters.

However, prompt tuning methods only achieve better performance with sufficient samples [9], while cannot address low-resource VQA.

2.3 Class/Instance-Level Prompting

Recent class- and instance-level prompt tuning methods for multimodal pretrained models are mainly used for CLIP-based [33] image classification. CoOp [54] attempts to design class-specific prompts at CLIP's language branch and finds them useful for fine-grained classification. CoCoOp [53] embeds the image features and adds them with learnable text prompts case-by-case to enhance the generalization ability in classifying unseen classes. FedAPT [35] assigns a unique key to each client to adapt prompts across domains for cross-domain federated image classification. DualPrompt [43] and L2P [44] create a prompt pool and select the Top$k$ prompts for insertion into the model for class-incremental continual learning. Another trend explores prompting visual concepts for frozen language models, enabling language models to handle multimodal tasks. Frozen [39] and PICa [46] prompt multimodal image representations and descriptions respectively to large-scale pretrained language models and obtain the few-shot learning ability in VQA. Meanwhile, Song et al. [34] use a pretrained language model to generate question-aware templates. They select the answer with a higher CLIP contrastive score between images and these templates that are filled with candidate answers.

However, the above methods with frozen language models are effective in VQA but require additional pretained prompt encoders, leading to expensive computation and inefficient storage. We utilize instance-level representations from VLM as conditions to generate prompts. Moreover, we employ hyper-networks and low-rank parameter factorization to construct a parameter-efficient prompt encoder for low-resource VQA.

3 METHODOLOGY

In this section, we first briefly overview the vision-language framework for VQA and our proposed Self-PT. We then introduce our Self-PT in detail with adaptive self-prompt embedding and parameter-efficient construction, to show how we adapt VLM for low-resource VQA. The overall architecture is illustrated in Fig 2.

3.1 Overview

VQA needs to infer an answer y based on a given image I and question Q pair. Following recent work [4, 18], we formulate VQA as a generative modeling task, generating free-form textual answers for a given question. We utilize a unified generative VLM $\mathcal{M}(\cdot)$, *i.e.*, VL-T5 [4], as our frozen backbone, which consists of a multimodal encoder and an auto-regressive decoder. We focus on tuning the VLM by prompts in a parameter-efficient principle for low-resource VQA. Existing prompt tuning methods [24, 25] generally concatenate tunable prefix vectors to the original input X, where $X = \text{concat}(I, Q)$. However, general prompt tuning methods cannot capture valid context-aware information during prompt encoding, resulting in 1) poor generalization of unseen answers and 2) lower improvements with more parameters. Hence, we propose Self-PT to solve the above issues by generating instance-level prompts conditioned on the given input X. Specifically, the formulation of Self-PT is demonstrated below:

$$y = \mathcal{M}(F(A(X)), X) \tag{1}$$

where $A(\cdot)$ is adaptive self-prompt embedding module (Section 3.2), $F(\cdot)$ includes two parameter-efficient designs conducted on $A(\cdot)$ (Section 3.3). In the following, we will describe the detail of these two modules.

3.2 Adaptive Self-Prompt Embedding

In this section, we would introduce the adaptive self-prompt embedding module $A(\cdot)$. Firstly, we analyze the bias that exists in general prompt tuning methods. Then, we propose adaptive self-prompt embedding to reduce the impact of the bias.

In general prompt tuning, two sets of prefix vectors P^k, $P^v \in \mathbb{R}^{\ell \times d}$ are concatenated with the original key-value sequence $C \in \mathbb{R}^{m \times d}$. Then, for a query vector $x \in \mathbb{R}^d$ in X, multi-head attention[1] is performed on the combined keys and values:

$$\begin{aligned} x_h =& \text{Attn}(xW_q, \text{concat}(P^k, CW_k), \text{concat}(P^v, CW_v)), \\ &\text{Attn}(Q, K, V) = \text{softmax}\left(QK^T/\sqrt{d_k}\right)V \end{aligned} \tag{2}$$

where W_q, W_k, and $W_v \in \mathbb{R}^{d \times d}$ are the weights used to project inputs to queries, keys, and values. Attention for given query Q, key K, and value V is also detailed in Eq. (2).

Following [11], Eq. (2) can be decomposed as:

$$\begin{aligned} x_h =& (1 - \lambda(x, P^k))\text{Attn}(xW_q, CW_k, CW_v) \\ &+ \lambda(x, P^k)\text{Attn}(xW_q, P^k, P^v) \end{aligned} \tag{3}$$

where the first term is the standard attention without prompts, and the second term modifies each x by the prompts. Note that $\lambda(x)$ is a scalar that represents the sum of normalized attention weights on the prompts:

$$\lambda(x, P^k) = \frac{\sum_i exp(xW_q(P_i^k)^T)}{\sum_i exp(xW_q(P_i^k)^T) + \sum_j exp(xW_qW_k^TC_j^T)} \tag{4}$$

Static prompts exert a direct effect on the query token x through the second term in Eq. (3). However, due to the scarcity of samples in low-resource VQA, it is hard to learn static prompts P^k and P^v

[1]Multi-head attention performs the attention mechanism in parallel over n heads, which we omit here for simplicity.

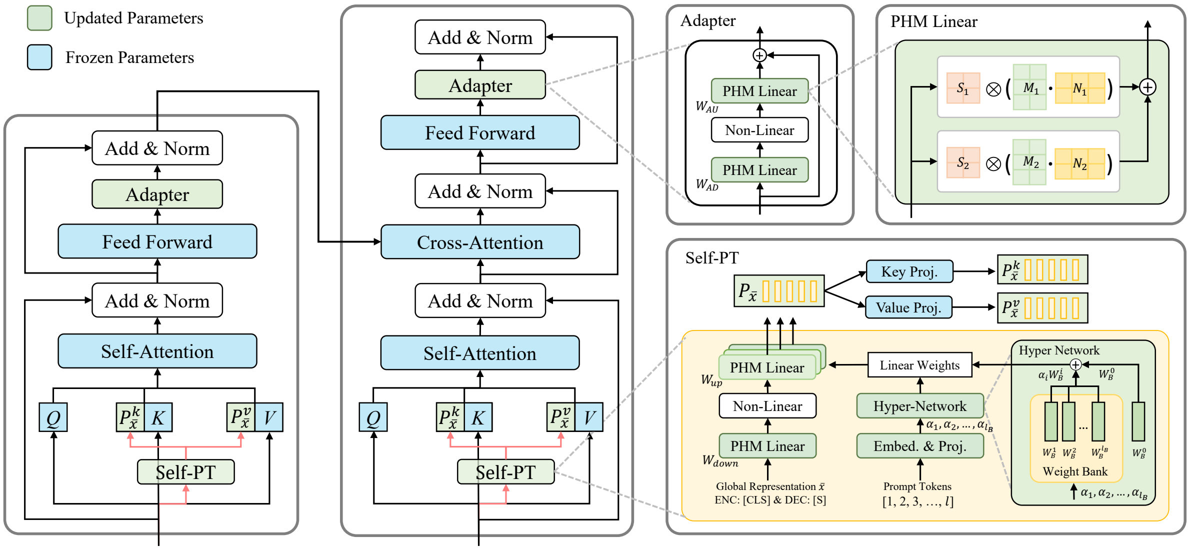

Figure 2: Overview of Adaptive Self-Prompt Tuning (Self-PT).

that are generalized well for all question types. In contrast, the fixed P^k and P^v only serve for adaptation to seen answers, incurring the bias for unseen answers, *e.g.*, prompting irrelevant seen answers.

To mitigate the above bias, we propose Self-PT to generate adaptive, context-aware prompts conditioned on the input representations from the self-attention layer. We construct the prompts conditioned on input question-image pair representations for several reasons. Firstly, by leveraging pretrained VLMs, the input representations provide sufficient context-aware information. Secondly, these representations can be used to retrieve prompts that are most relevant to the current sample. Bias from those irrelevant seen answers would be diminished due to decreased correlations between the instance-level prompts and the irrelevant seen answers.

Self-PT uses a straightforward module called adaptive self-prompt embedding to construct instance-level context-aware prompts. As depicted in Fig. 2, given the input question-image pair, we extract the [CLS] token [2] as the global multimodal representation denoted by $\bar{x}$. Adaptive self-prompt embedding employs ℓ prompt encoders to get ℓ context-aware prompt tokens. Specifically, prompts with length ℓ can be formulated as:

$$P_{\bar{x}} = W_{\text{up}} \cdot \delta(W_{\text{down}} \cdot \bar{x}) \tag{5}$$

where $P_{\bar{x}} \in \mathbb{R}^{d\times\ell}$, $W_{\text{up}} \in \mathbb{R}^{d_{mid}\times d\times\ell}$ and $W_{\text{down}} \in \mathbb{R}^{d\times d_{mid}}$ represent the up- and down-projection in prompt encoder, d denotes the dimension size of VLMs, d_{mid} denotes the middle size of the prompt encoder, $\delta(\cdot)$ is the non-linear activation, W_{down} is shared to embed the global multimodal representation $\bar{x}$ for each prompt. The prompt encoders are constructed like a feed-forward layer and the W_{down} and W_{up} serve as a set of key-value memory tokens [7, 12]. Therefore, the prompt embeddings can be adjusted adaptively according to their relevance to the global representation $\bar{x}$, enabling better adaptation of VLM to low-resource VQA.

[2]In the decoder, we use the start token [s]. In addition, the global multimodal representations can also be obtained by avg/max pooling, which would be discussed in the experiment section.

Moreover, since the generated prompts are conditioned on the input global representation $\bar{x}$, we utilize the key and value projections in the frozen VLM to obtain key and value prompts, instead of generating them separately:

$$P^k_{\bar{x}} = W^*_k \cdot P_{\bar{x}}, P^v_{\bar{x}} = W^*_v \cdot P_{\bar{x}} \tag{6}$$

where the W^*_k and W^*_v are frozen key and value projections. Leveraging the key and value projections in frozen VLMs can further utilize the existing knowledge in the pre-training models and no more need to learn the mapping from prompt embedding space to the key and value representation spaces.

By substituting $P^k_{\bar{x}}$ and $P^v_{\bar{x}}$ from Eq. (6) into Eq. (3), we get:

$$\begin{aligned} x_h =& (1-\lambda(x, P^k_{\bar{x}}))\text{Attn}(xW_q, CW_k, CW_v) \\ &+ \lambda(x, P^k_{\bar{x}})\text{Attn}(xW_q, P^k_{\bar{x}}, P^v_{\bar{x}}) \end{aligned} \tag{7}$$

Due to decreased correlations between current questions and those irrelevant seen answers, bias to those irrelevant seen answers is diminished. Therefore, $P^k_{\bar{x}}$ and $P^v_{\bar{x}}$ are prompts that are appropriate to instruct each sample. They let the second item of Eq. (7) provide appropriate instructions to VLMs instead of misleading them, thus mitigating overfitting to seen answers. Moreover, adaptive self-prompt embedding further obtains implicit information from the relative relationship between different types of questions, which enables better utilization of the prompt embedding capacity.

3.3 Parameter-Efficient Self-PT

The adaptive self-prompt embedding in Self-PT can solve the overfitting issue, but brings the following problems: 1) the parameters in adaptive self-prompt embedding are linearly related to the number of prompt tokens, which is not flexible to generate prompts with any length. 2) adaptive self-prompt embedding needs large numbers of parameters to generate instance-level context-aware prompts. Therefore, we use hyper-networks $F_H(\cdot)$ and low-rank parameter factorization $F_L(\cdot)$ to make Self-PT flexible and parameter-efficient.

Hyper-Network for Prompt Embedding. The general idea of hyper-networks [10, 12, 52] is to learn a parametric task-specific hyper-embedding for each task. The hyper-embedding is fed to a hyper-network which generates task-specific parameters for other networks. Different from existing methods, we focus on decoupling the number of parameters for prompt embedding and the prompt length ℓ while capturing the shared knowledge across prompt tokens.

In Eq. (5), the number of parameters in W_{up} is directly related to the prompt length ℓ. Specifically, it scales linearly with prompt length ℓ as $O(d \cdot d_{mid} \cdot \ell)$, which is not as flexible as general prompt tuning methods [24, 25]. Hence, to achieve higher flexibility and parameter efficiency, we employ hyper-networks to generate parameters for W_{up}. We introduce a set of embeddings $\{e_i\}_{i=1}^{\ell}$ and a weight bank $\{W_B^i\}_{i=1}^{\ell_{\text{B}}}$ to construct the hyper-network, where $e_i \in \mathbb{R}^{d_e}$ only specifies the prompt index, the dimension $d_e \ll d$, and $W_B^i \in \mathbb{R}^{d_{\text{mid}} \times d}$ including ℓ_{B} trainable weights. Hence, if we consider W_{up} as ℓ numbers of prompt encoder $\{W_{\text{up}}^i\}_{i=1}^{\ell}$, the prompt index information can be used to generate weights, *i.e.*, Linear Weights in Fig. 2, specific for the corresponding prompt encoder:

$$W_{\text{up}}^i = \sum_{j=1}^{\ell_B} \text{LN}(W_e \cdot e_i) \cdot W_B^j + W_B^0 \tag{8}$$

where $W_e \in \mathbb{R}^{d_e \times \ell_{\text{in}}}$ is a lightweight mapping, $\text{LN}(\cdot)$ is layer normalization, W_B^0 is an additional shared weight. Hence, for any prompt length ℓ, Self-PT decouples the number of the parameters for prompt embedding as $O(d \cdot d_{mid} \cdot \ell_{\text{B}})$, in which the number of the parameters is directly related to the predefined width of weight bank ℓ_{B} instead of prompt length ℓ. Moreover, hyper-network enables knowledge sharing across prompt tokens in each layer while maintaining a low parameter cost during the end-to-end training.

Low-Rank Parameter Factorization. We employ low-rank parameter factorization to reparameterize the weight of each linear layer in Self-PT with much fewer parameters while maintaining the performance. The parameterized hypercomplex multiplication (PHM) layers [49] are first used to construct a parameter-efficient transformer. Recent studies [17, 19] show the effectiveness of the PHM layer in adapter-based parameter-efficient tuning. We further explore the low-rank parameter factorization method in prompt tuning for better parameter efficiency. In Eq. (8) and Eq. (6), the $W_{\text{down}} \in \mathbb{R}^{d \times d_{mid}}$ and the weights in the weight bank $\{W_B^i\}_{i=0}^{\ell_{\text{B}}} \in \mathbb{R}^{d_{mid} \times d}$ are firstly decomposed in the low-dimensional matrices by the Kronecker products like PHM layer:

$$W_{\text{down}} = \sum_{j=1}^{n} S_{\text{down}}^j \otimes T_{\text{down}}^j, W_B^i = \sum_{j=1}^{n} S_B^{ij} \otimes T_B^{ij} \tag{9}$$

where $S_{\text{down}}^j, S_B^{ij} \in \mathbb{R}^{n \times n}$, $T_{\text{down}}^j \in \mathbb{R}^{\frac{d}{n} \times \frac{d_{\text{mid}}}{n}}$, $T_B^{ij} \in \mathbb{R}^{\frac{d_{\text{mid}}}{n} \times \frac{d}{n}}$. The $\otimes$ indicates the Kronecker product, which is a special outer product between matrices. For example, given $S \in \mathbb{R}^{m \times k}$ and $T \in \mathbb{R}^{p \times q}$, $S \otimes T \in \mathbb{R}^{mp \times kq}$ is a block matrix as follows:

$$S \otimes T = \begin{bmatrix} s_{11}T & s_{12}T & \cdots & s_{1k}T \\ s_{21}T & s_{22}T & \cdots & s_{2k}T \\ \vdots & \vdots & \ddots & \vdots \\ s_{m1}T & s_{m2}T & \cdots & s_{mk}T \end{bmatrix} \tag{10}$$

To be more parameter-efficient, the matrix T_{down}^j and T_B^{ij} are further factorized into two low-rank matrices by:

$$T_{\text{down}}^j = M_{\text{down}}^j \cdot (N_{\text{down}}^j)^T, T_B^{ij} = M_B^{ij} \cdot (N_B^{ij})^T \tag{11}$$

where $M_{\text{down}}^j \in \mathbb{R}^{\frac{d}{n} \times r}$, $N_{\text{down}}^j \in \mathbb{R}^{\frac{d_{\text{mid}}}{n} \times r}$, $M_B^{ij} \in \mathbb{R}^{\frac{d_{\text{mid}}}{n} \times r}$, $N_B^{ij} \in \mathbb{R}^{\frac{d}{n} \times r}$, r is the predefined rank of these matrices. Finally, W_{down} and W_B^i can be reparameterized by:

$$\begin{aligned} W_{\text{down}} &= \sum_{j=1}^{n} S_{\text{down}}^j \otimes \left(M_{\text{down}}^j \cdot (N_{\text{down}}^j)^T\right), \\ W_B^i &= \sum_{j=1}^{n} S_B^{ij} \otimes \left(M_B^{ij} \cdot (N_B^{ij})^T\right) \end{aligned} \tag{12}$$

After low-rank parameter factorization, parameters of each weight W in Self-PT, *i.e.*, W_{down} and W_B^i, is reduced from the original $d \cdot d_{\text{mid}}$ to $\frac{r}{n}(d + d_{\text{mid}}) + n^3$. With the mild condition that the rank $r \ll d$ and d_{mid}, it can reduce the parameters to r/nd_{mid} compared with the original numbers of parameters at most.

In addition to reducing the parameters of each weight W in Self-PT, low-rank parameter factorization is also used to reduce the parameters of the adapter. To adapt feed-forward layer for low-resource VQA, we also employ adapters [13, 19] that are added after the feed-forward layer in VLMs. As depicted in Fig. 2, the adapter layer consists of a down-projection $W_{\text{AD}} \in \mathbb{R}^{d \times d_{mid}}$ and an up-projection $W_{\text{AU}} \in \mathbb{R}^{d_{mid} \times d}$, where d is the input dimension and d_{mid} is the bottleneck dimension for the adapter layer, for input x, adapter layer can be defined as:

$$x_h = x + W_{\text{AU}} \cdot \delta(W_{\text{AD}} \cdot x) \tag{13}$$

The same as Self-PT, we reparameterize the W_{AD} and W_{AU} by low-rank parameter factorization similar to Eq. (12) to reduce the number of parameters in adapters while maintaining the performance.

4 EXPERIMENT

4.1 Experimental Setup

Datasets and Evaluation Metrics. Our experimental evaluation involves three widely used datasets in the field of visual question answering (VQA): VQA v2 [8], GQA [15], and OK-VQA [30]. We follow the approach of Chen et al. [3] and consider training data sizes, denoted as $N_{\mathcal{D}}$, smaller than 1,000. To achieve a more practical low-resource learning scenario, we adopt the true few-shot learning analysis [6, 31] and use the development set $\mathcal{D}_{\text{dev}}$ of the same size as the training set $\mathcal{D}_{\text{train}}$ (*i.e.*, $|\mathcal{D}_{\text{train}}| = |\mathcal{D}_{\text{dev}}| = N_{\mathcal{D}}$), instead of a large-scale validation set, for best model selection and hyper-parameter tuning. Following recent study [17], our experiments cover $N_{\mathcal{D}}$ values of 16, 32, 64, 100, 500, and 1,000. To create the $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{dev}}$ sets for these three datasets, we randomly sample $2N_{\mathcal{D}}$ samples from its training set and divide them equally between $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{dev}}$. The accuracy of low-resource VQA tasks is measured using the VQA-Score metric [1].

Implementation details. We implement all methods using PyTorch on an NVIDIA Tesla V100 GPU. we utilize the pretrained VLM, *i.e.*, VL-T5 [4], as our baseline for low-resource VQA. We consider VQA as a generation task for parameter-efficient tuning and do not introduce additional parameters from VQA heads. We

Dataset	Method	#Param (M)	(%)	#Sample 16-shot	32-shot	64-shot	100-shot	500-shot	1,000-shot
VQA v2 [8]	Finetuning	224.54	100%	41.82±1.58	43.09±3.10	46.87±0.57	48.12±0.87	53.46±0.41	55.56±0.13
	BitFit [47]	0.29	0.13%	40.61±4.15	43.86±2.19	46.14±1.00	47.53±0.67	51.91±0.40	53.18±0.58
	LoRA [14]	0.44	0.20%	41.60±2.27	42.62±2.41	45.36±1.66	47.57±0.91	51.93±0.38	54.15±0.45
	Compacter [19]	0.34	0.15%	39.28±1.87	42.47±2.76	44.91±1.27	46.28±1.37	51.21±0.90	53.39±0.54
	Houlsby [13]	4.76	2.12%	41.71±2.16	44.01±2.09	45.11±1.40	47.71±0.78	52.27±1.05	54.31±0.34
	Pfeiffer [32]	2.38	1.06%	41.48±1.86	44.18±2.13	45.93±1.11	47.42±1.15	52.35±0.52	53.98±0.38
	AdaMix [42]	5.92	2.64%	40.59±2.05	43.42±2.08	46.70±1.32	47.34±0.91	51.72±1.05	54.12±0.63
	MixPHM [17]	0.87	0.39%	43.13±1.78	45.97±2.01	48.26±0.56	49.91±0.76	**54.30±0.33**	56.11±0.40
	Self-PT	1.08	0.48%	**49.21±2.21**	**49.77±2.44**	**50.31±0.84**	**50.76±0.78**	**54.30±0.44**	**56.25±0.34**
GQA [15]	Finetuning	224.54	100%	28.24±2.08	30.80±2.49	34.22±0.59	36.15±0.99	41.49±0.54	43.04±0.57
	BitFit [47]	0.29	0.13%	26.13±2.83	29.00±4.81	34.25±1.16	35.91±1.22	40.08±0.42	41.84±0.15
	LoRA [14]	0.44	0.20%	26.89±2.74	30.40±2.27	34.40±0.99	36.14±1.10	40.20±1.02	42.06±1.12
	Compacter [19]	0.34	0.15%	23.70±2.10	27.18±2.61	32.70±1.30	35.28±1.45	38.68±0.50	41.17±0.95
	Houlsby [13]	4.76	2.12%	25.13±2.32	28.34±1.17	33.23±0.94	35.88±1.79	40.85±0.48	41.90±0.72
	Pfeiffer [32]	2.38	1.06%	25.08±1.81	29.18±1.32	32.97±0.84	35.08±1.01	40.30±0.40	41.39±0.27
	AdaMix [42]	5.92	2.64%	24.62±2.34	28.01±1.33	32.74±0.96	35.64±0.94	40.14±0.42	41.97±0.86
	MixPHM [17]	0.87	0.39%	28.33±2.63	32.40±2.52	**36.75±0.55**	37.40±0.87	41.92±0.55	**43.81±0.50**
	Self-PT	1.08	0.48%	**34.72±2.13**	**35.62±2.32**	36.27±0.80	**37.77±1.17**	**41.96±0.55**	43.45±0.53
OK-VQA [30]	Finetuning	224.54	100%	11.66±2.08	14.20±0.78	16.65±1.02	18.28±0.67	24.07±0.40	26.66±0.72
	BitFit [47]	0.29	0.13%	11.29±1.79	13.66±1.49	15.29±0.57	16.51±0.53	22.54±0.57	24.80±0.63
	LoRA [14]	0.44	0.20%	10.26±1.53	12.46±1.82	15.95±0.38	17.03±0.82	23.02±0.41	25.26±0.53
	Compacter [19]	0.34	0.15%	9.64±2.73	11.04±1.39	13.57±1.07	15.92±1.18	22.20±0.89	24.52±0.59
	Houlsby [13]	4.76	2.12%	9.79±1.71	12.25±2.13	15.04±1.25	16.58±0.65	22.67±0.77	25.04±0.44
	Pfeiffer [32]	2.38	1.06%	9.06±0.53	11.39±0.79	14.23±1.54	16.34±0.79	22.90±1.03	26.70±0.71
	AdaMix [42]	5.92	2.64%	8.39±1.20	11.55±1.37	13.66±2.29	16.27±0.92	23.20±0.78	26.34±0.88
	MixPHM [17]	0.87	0.39%	13.87±2.39	16.03±1.23	18.58±1.42	20.16±0.97	26.08±0.88	28.53±0.85
	Self-PT	1.08	0.48%	**19.67±2.41**	**20.43±0.71**	**21.52±0.82**	**23.08±1.16**	**26.41±0.31**	**29.54±0.57**

Table 1: Performance comparison on low-resource VQA with pretrained VL-T5. The average VQA-Scores with standard deviation across 5 seeds are evaluated on VQA v2 validation set, GQA test-dev, and OK-VQA test set. The best and second best parameter-efficient tuning methods are highlighted.

Method	#Param Total	Tuned	Dataset VQAv2	GQA	OK-VQA
Frozen [39]	7B	-	38.2	12.6	-
PICa-Base [46]	175B	-	54.3	-	43.3
PICa-Full [46]	175B	-	**56.1**	-	**48.0**
VL-T5$_{\text{no-vqa}}$[4]	224M	100%	31.8	19.6	12.7
FewVLM [18]	224M	100%	48.2	32.2	15.0
MixPHM [17]	225M	0.39%	43.13	28.33	13.87
Self-PT	225M	0.48%	**49.21**	**34.72**	**19.67**

Table 2: Comparison with the few-shot methods on VQA v2. All methods are tested on 5 different seeds with 16 randomly selected samples for each seed.

use the weights released by FewVLM [18], which re-trained VL-T5 without the overlapping samples. All reported results are averaged over five seeds $\{13, 21, 42, 87, 100\}$. The learning rate is set to $1e$-4. The batch size and number of epochs are set to 16 and 400, respectively. For optimization, we employ the AdamW optimizer [27] and an early stopping strategy with the patience of 200 non-increasing epochs, where the stopping metric is the VQA-Score on development set $\mathcal{D}_{\text{dev}}$ for each dataset. We added the adapters to adapt feed-forward layer for low-resource VQA which performs similarly to the compactor [19] with 0.16% tunable parameters.

4.2 Comparative Evaluation

We conduct several experiments to show the effectiveness of Self-PT, including the comparative experiments with finetuning and several state-of-the-art parameter-efficient tuning methods, the comparative experiments with SOTA few-shot methods, and the comparative experiments under the setting of domain adaptation.

Comparison with Parameter-Efficient Tuning Methods. We compare our Self-PT with finetuning and several state-of-the-art parameter-efficient tuning methods. Specifically, we compare Self-PT with Houlsby [13], Compacter [19], Pfeiffer [32], AdaMix [42], MixPHM [17], LoRA [14], and BitFit [47]. Note that these methods are all re-implemented by [17] which performs hyper-parameter search on their key hyper-parameters and reports their best performance.

Table 1 shows the results with pretrained VL-T5 [4] on three datasets. Overall, our Self-PT outperforms state-of-the-art parameter-efficient tuning methods and also consistently outperforms full finetuning. Specifically, Self-PT achieves the best performance in VQA v2 and OK-VQA datasets and improves most of the evaluation metrics in the GQA dataset. The margins between Self-PT and the current SOTA method, *i.e.*, MixPHM, in the rest two metrics are not large. However, it is important to note that Self-PT markedly outperforms MixPHM when the seen sample is extremely scarce. For example, Self-PT shows more than 6% average improvements in three datasets when $\mathcal{D}$ is 16. We attribute the performance improvement to the proposed Self-PT, as it dynamically learns context-aware prompts with markedly fewer tunable parameters, making it more efficient and effective for low-resource VQA.

Comparison with SOTA Few-Shot Methods. Table 2 presents a comparison of SOTA multimodal few-shot methods in VQA. Specifically, few-shot VQA is a special case of low-resource VQA. Two in-context learning methods, Frozen [39] and PICa [46], leverage prompt-tuning to transfer large language models such as GPT-3 [2] without tuning parameters. VL-T5$_{\text{no-vqa}}$[4] and FewVLM [18] are full finetuning methods, while FewVLM additionally inserts

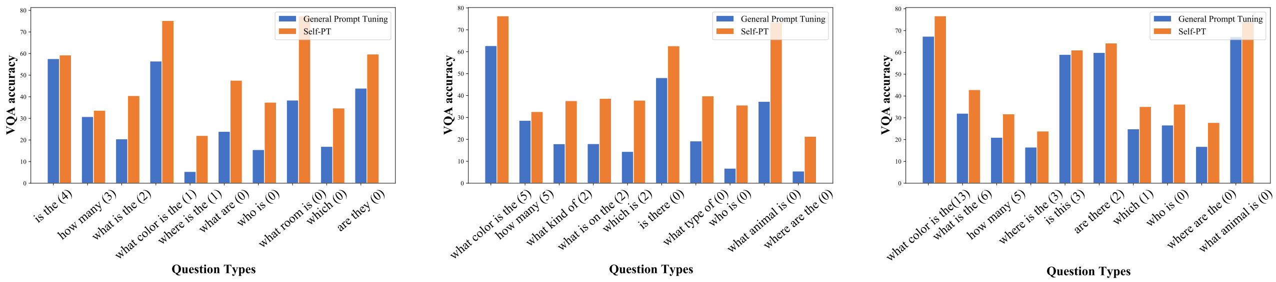

Figure 3: Visualization results on VQA v2 validation set with $N_{\mathcal{D}}$ in 16, 32, and 64, respectively. We show the performance of general prompt tuning method (red) and Self-PT (blue) in ten random types of questions. Numbers in $(\cdot)$ are the times this question type occurred in the training sets.

Method	#Tunable	VQAv2	GQA	OK-VQA
Finetuning	224.54M	41.82±1.58	28.24±2.08	11.66±2.08
BitFit [47]	0.13%	40.61±4.15	26.13±2.83	11.29±1.79
LoRA [14]	0.20%	41.60±2.27	26.89±2.74	10.26±1.53
Compacter [19]	0.15%	39.28±1.87	23.70±2.10	9.64±2.73
Houlsby [13]	2.12%	41.71±2.16	25.13±2.32	9.79±1.71
Pfeiffer [32]	1.06%	41.48±1.86	25.08±1.81	9.06±0.53
AdaMix [42]	2.64%	40.59±2.05	24.62±2.34	8.39±1.20
MixPHM [17]	0.39%	43.13±1.78	28.33±2.63	13.87±2.39
Self-PT	0.48%	**49.21±2.21**	**34.72±2.13**	**19.67±2.41**
Self-PT$_{\text{VQA}}$	0.48%	**49.21±2.21**	**31.98±2.21**	**16.14±2.32**

Table 3: Domain adaptation ability across datasets. Self-PT$_{\text{VQA}}$ **is trained and saves the best epoch on the VQA v2 training set and validation set, respectively. Then, we directly test** Self-PT$_{\text{VQA}}$ **on the GQA and OK-VQA datasets.**

hand-crafted prompts into model inputs. MixPHM [17] is the current state-of-the-art parameter-efficient tuning method.

Results in Table 2 demonstrate that except Frozen and PICa which employ additional pretrained prompt encoders and such large-size pretrained models, Self-PT achieves better results than both prompt-based finetuning and parameter-efficient tuning methods. Specifically, Self-PT shows 2.72% average improvements in three datasets compared with the prompt-based finetuning method, and 6.09% average improvements compared with SOTA parameter efficient tuning method. This demonstrates the superiority of Self-PT in terms of performance and parameter efficiency.

Comparison Under Domain Adaptation Setting. We test the domain adaptation ability of Self-PT across datasets with $N_{\mathcal{D}} = 16$. The results of Self-PT$_{\text{VQA}}$ in Table 3 show that Self-PT can generalize well across datasets. Specifically, Self-PT trained on the VQA dataset outperforms finetuning and the SOTA parameter-efficient tuning method, *i.e.*, MixPHM on GQA and OK-VQA by 3.65% and 2.27%, respectively. This demonstrates the strong generalization ability of Self-PT.

4.3 Ablation Studies

If not specifically mentioned, We conduct ablated experiments with pretrained VL-T5 on VQA v2, GQA, and OK-VQA with $\mathcal{D}_{\text{train}} = \mathcal{D}_{\text{dev}} = 16$.

Effectiveness of Each Component. We ablate the three key components in Self-PT: adaptive prompt encoder ($\mathcal{A}$), hyper-network ($\mathcal{F}_{\mathcal{H}}$), and low-rank parameter factorization ($\mathcal{F}_{\mathcal{L}}$). The results are shown in Table 4. We implement the general prompt tuning method (the first row) using an embedding layer as well as an MLP with the

$\mathcal{A}$	$\mathcal{F}_{\mathcal{H}}$	$\mathcal{F}_{\mathcal{L}}$	#Tunable Param	Dataset		
				VQAv2	GQA	OK-VQA
Finetuning			224.54M	41.82±1.58	28.24±2.08	11.66±2.08
			2.11%	39.69±2.78	24.71±1.81	10.73±1.57
✓			6.10%	46.44±2.17	31.54±2.05	15.54±2.52
✓	✓		4.19%	47.79±2.27	34.24±2.10	17.84±1.82
✓		✓	0.64%	47.64±2.32	33.11±2.41	17.25±2.73
✓	✓	✓	0.48%	**49.21±2.21**	**34.72±2.13**	**19.67±2.41**

Table 4: Ablation studies on each component. $\mathcal{A}$: adaptive prompt encoder, $\mathcal{F}_{\mathcal{H}}$: hyper-networks, $\mathcal{F}_{\mathcal{L}}$: low-rank parameter factorization.

same middle dimension as that in adaptive self-prompt embedding module. Results in the first row and the second row demonstrate that constructing instance-level context-aware prompts shows great improvements compared with the general prompt tuning method, *i.e.*, 6.16% average improvements in three datasets. Results in rows 2-4 show that hyper-network and low-rank parameter factorization achieve about 1.5x and 9x reduction in the number of parameters respectively and both get better performance. This demonstrates that hyper-network and low-rank parameter factorization highly reduce parameters while maintaining model capacity.

Generalization Analysis. To evaluate the generalization ability of Self-PT to different question types, we visualize the performance of various types of unseen questions in Fig. 3, which usually require unseen answers. Specifically, for the question types that occur more than 3 times, Self-PT consistently outperforms the general prompt tuning method. For the question types that occur less than 2 times and those unseen types of questions, Self-PT outperforms the general prompt tuning method by a large margin, especially in lower-resource settings, *i.e.*, $N_{\mathcal{D}}$ in 16 and 32. This shows that its generalization ability is stronger than the general prompt tuning method.

Prompt Length Analysis. To study the effects of the prompt length on low-resource VQA, we evaluate Self-PT performance with a prompt length selected from $\{2, 5, 10, 15, 20, 30, 60, 100\}$. As shown in Fig 4, when the prompt lengths are less than 10, increasing prompt lengths can usually bring performance improvements in Self-PT (expect $N_{\mathcal{D}} = 16$ in VQA v2 dataset). This phenomenon of improvements cannot extend to all scenarios, the increase might meet saturation when the prompt length is more than 10. We advise that the length of 5 tokens can achieve better performance on average in different datasets and different settings.

Enc	Dec	VQAv2	GQA	OK-VQA
✓		**49.58±1.81**	34.68±2.52	18.58±2.19
	✓	48.52±2.76	33.11±2.08	17.64±1.87
✓	✓	49.21±2.21	**34.72±2.13**	**19.67±2.41**

Table 5: Evaluation of different prompt insertion methods. We specifically evaluate Self-PT inserted to the encoder, to the decoder only, or both the encoder and decoder.

Condition	VQAv2	GQA	OK-VQA
w/ mean	**49.21±1.71**	34.69±2.08	19.55±2.19
w/ max	49.08±2.43	34.50±2.34	19.41±2.24
w/ [cls]	**49.21±2.21**	**34.72±2.13**	**19.67±2.41**

Table 6: Evaluation of different input conditions for Self-PT. We evaluate three variants of conditions: average pooling (mean), max pooling (max), and directly using the [cls] token.

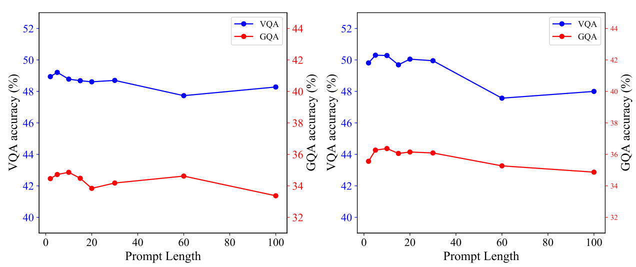

Figure 4: Analysis of prompt lengths on VQA v2 and GQA dataset when $N_{\mathcal{D}}$ is 16 (left) and 64 (right).

Prompt Depth Analysis. We evaluate the performance of inserting prompts to the encoder, to the decoder only, or to both the encoder and decoder. Experimental results are demonstrated in Table 5. We find that it is better to insert prompts into every layer of the whole VLM. In the comparison between insertion to the encoder only and to the decoder only, we observe that the former solution leads to better results. This is because the prompts instruct multimodal information fusion indirectly in the decoder.

Prompt Condition Analysis. We analyze the input conditions of Self-PT to generate context-aware prompts. Since the decoder acts in an auto-regressive manner and is hard to change the conditions for Self-PT, We evaluate three variants of conditions for Self-PT in the encoder layer: average or max pooling of all input tokens and directly using the [cls] token. Experimental results in Table 6 demonstrate that Self-PT can leverage various forms of global multimodal representations to achieve stable performance. It is mainly because of the strong prompt embedding capacity of Self-PT to generate proper prompts.

Hyper-Parameter Analysis. To investigate the impact of different hyper-parameters on Self-PT, we conduct experiments by varying ℓ_{in}, d_{mid}, r, and n. More specifically, we consider the following settings: $\ell_{\text{in}} \in \{1, 2, 5, 10\}$, $d_{\text{mid}} \in \{96, 128, 192, 384\}$, $r \in \{2, 4, 8, 16\}$, and $n \in \{2, 4, 8\}$. The results in Table 7 show that changing these hyper-parameters has a slight impact on the performance of Self-PT. This suggests that Self-PT does not significantly depend on the hyper-parameter selection. We finally choose $\ell_{\text{in}} = 2$, $d_{\text{mid}} = 128$, $r = 8$, and $n = 4$ for better performance and parameter efficiency.

Hyperparam		#Tunable	VQAv2	GQA	OK-VQA
Finetuning		224.54M	41.82±1.58	28.24±2.08	11.66±2.08
ℓ_B	1	0.4%	**49.68±2.07**	34.24±2.42	18.55±2.65
	2	0.48%	49.21±2.21	**34.72±2.13**	**19.67±2.41**
	5	0.74%	48.62±1.93	33.53±1.74	17.68±2.11
	10	1.16%	48.40±2.23	32.33±2.16	18.90±2.24
d_{mid}	384	0.57%	49.09±2.28	34.24±2.17	17.87±2.36
	192	0.50%	49.04±2.15	32.49±2.22	19.59±2.55
	128	0.48%	**49.21±2.21**	**34.72±2.13**	**19.67±2.41**
	96	0.47%	48.96±2.20	33.50±2.19	19.58±2.32
r	2	0.25%	49.17±2.35	34.59±2.40	17.46±2.43
	4	0.33%	**49.25±2.18**	**34.96±2.43**	17.28±2.38
	8	0.48%	49.21±2.21	34.72±2.13	19.67±2.41
	16	0.78%	49.16±2.02	34.54±2.23	**19.70±2.21**
n	2	0.47%	48.52±2.17	32.95±2.02	16.38±2.73
	4	0.48%	49.21±2.21	**34.72±2.13**	**19.67±2.41**
	8	0.50%	**49.46±2.05**	33.94±2.32	18.76±2.40

Table 7: Hyper-Parameter Analysis. ℓ_B: width of weight bank, d_{mid}: bottleneck dimension, r: rank of parameter factorization and n: the number of summations of Kronecker product.

4.4 Discussions and Limitations

Self-PT shows superiority in performance and generalization ability with few tunable parameters, although there exist limitations in sharing information across layers and generalizing it to more vision-language tasks. Some recent works [12, 17, 52] have shown that sharing parameters between layers can improve performance and parameter efficiency. We suppose that the prompt index embedding and the adaptive prompt encoder can benefit from sharing parameters for each layer. Besides, Self-PT currently serves for visual question answering tasks only, which is not explored in a multi-tasking learning scenario. In the future, we plan to expand our exploration to 1) knowledge sharing between layers and 2) adapt Self-PT to more V&L tasks.

5 CONCLUSION

In this paper, we propose a prompt tuning method for low-resource VQA named Adaptive Self-Prompt Tuning (Self-PT). Specifically, Self-PT utilizes instance-level multimodal representations as conditions to obtain context-aware prompts, avoiding implicit correlations between static prompts and seen answers. Moreover, we use hyper-networks and low-rank parameter factorization to reduce the trainable parameters of Self-PT while maintaining the prompt embedding capacity. Experiments conducted on VQA v2, GQA, and OK-VQA with different low-resource settings show that our Self-PT outperforms the state-of-the-art parameter-efficient methods, especially in lower-shot settings, *e.g.*, 6% average improvements cross three datasets in 16-shot.

ACKNOWLEDGMENTS

This work was supported by the National Key Research and Development Project (No.2020AAA0106200), the National Nature Science Foundation of China under Grants (No.61936005, No.62325206), the Opening Foundation of Key Laboratory of Computer Vision and System, Ministry of Education, Tianjin University of Technology, China, and the Graduate Research and Innovation Projects in Jiangsu Province (KYCX23_1022).

REFERENCES

[1] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*. 2425–2433.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[3] Guanzheng Chen, Fangyu Liu, Zaiqiao Meng, and Shangsong Liang. 2022. Revisiting Parameter-Efficient Tuning: Are We Really There Yet?. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 2612–2626.

[4] Jaemin Cho, Jie Lei, Hao Tan, and Mohit Bansal. 2021. Unifying vision-and-language tasks via text generation. In *International Conference on Machine Learning*. PMLR, 1931–1942.

[5] Zi-Yi Dou, Yichong Xu, Zhe Gan, Jianfeng Wang, Shuohang Wang, Lijuan Wang, Chenguang Zhu, Pengchuan Zhang, Lu Yuan, Nanyun Peng, et al. 2022. An empirical study of training end-to-end vision-and-language transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 18166–18176.

[6] Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL-IJCNLP 2021*. Association for Computational Linguistics (ACL), 3816–3830.

[7] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer Feed-Forward Layers Are Key-Value Memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 5484–5495.

[8] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 6904–6913.

[9] 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)*. 8410–8423.

[10] David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106* (2016).

[11] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a Unified View of Parameter-Efficient Transfer Learning. In *International Conference on Learning Representations*.

[12] Yun He, Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, et al. 2022. Hyperprompt: Prompt-based task-conditioning of transformers. In *International Conference on Machine Learning*. PMLR, 8678–8690.

[13] 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*. PMLR, 2790–2799.

[14] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

[15] Drew A Hudson and Christopher D Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 6700–6709.

[16] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. 2022. Visual prompt tuning. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII*. Springer, 709–727.

[17] Jingjing Jiang and Nanning Zheng. 2023. MixPHM: Redundancy-Aware Parameter-Efficient Tuning for Low-Resource Visual Question Answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 24203–24213.

[18] 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 *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2763–2775.

[19] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems* 34 (2021), 1022–1035.

[20] Wonjae Kim, Bokyung Son, and Ildoo Kim. 2021. Vilt: Vision-and-language transformer without convolution or region supervision. In *International Conference on Machine Learning*. PMLR, 5583–5594.

[21] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597* (2023).

[22] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning*. PMLR, 12888–12900.

[23] Junnan Li, Ramprasaath Selvaraju, Akhilesh Gotmare, Shafiq Joty, Caiming Xiong, and Steven Chu Hong Hoi. 2021. Align before fuse: Vision and language representation learning with momentum distillation. *Advances in neural information processing systems* 34 (2021), 9694–9705.

[24] 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)*. 4582–4597.

[25] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 61–68.

[26] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT understands, too. *arXiv preprint arXiv:2103.10385* (2021).

[27] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

[28] Haoyu Lu, Mingyu Ding, Yuqi Huo, Guoxing Yang, Zhiwu Lu, Masayoshi Tomizuka, and Wei Zhan. 2023. UniAdapter: Unified Parameter-Efficient Transfer Learning for Cross-modal Modeling. *arXiv preprint arXiv:2302.06605* (2023).

[29] Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021. Parameter-efficient Multi-task Fine-tuning for Transformers via Shared Hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 565–576.

[30] Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. 2019. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*. 3195–3204.

[31] Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. *Advances in neural information processing systems* 34 (2021), 11054–11070.

[32] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-Destructive Task Composition for Transfer Learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 487–503.

[33] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.

[34] Haoyu Song, Li Dong, Weinan Zhang, Ting Liu, and Furu Wei. 2022. CLIP Models are Few-Shot Learners: Empirical Studies on VQA and Visual Entailment. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 6088–6100.

[35] Shangchao Su, Mingzhao Yang, Bin Li, and Xiangyang Xue. 2022. Cross-domain Federated Adaptive Prompt Tuning for CLIP. *arXiv preprint arXiv:2211.07864* (2022).

[36] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5227–5237.

[37] Yi-Lin Sung, Varun Nair, and Colin A Raffel. 2021. Training neural networks with fixed sparse masks. *Advances in Neural Information Processing Systems* 34 (2021), 24193–24205.

[38] Shengeng Tang, Richang Hong, Dan Guo, and Meng Wang. 2022. Gloss semantic-enhanced network with online back-translation for sign language production. In *Proceedings of the 30th ACM International Conference on Multimedia*. 5630–5638.

[39] Maria Tsimpoukelli, Jacob L Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. 2021. Multimodal few-shot learning with frozen language models. *Advances in Neural Information Processing Systems* 34 (2021), 200–212.

[40] Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. 2022. Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. In *International Conference on Machine Learning*. PMLR, 23318–23340.

[41] Wenhui Wang, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, and Furu Wei. 2023. Image as a Foreign Language: BEiT Pretraining for Vision and Vision-Language Tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 19175–19186.

[42] Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan, and Jianfeng Gao. 2022. AdaMix: Mixture-of-Adaptations for Parameter-efficient Model Tuning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 5744–5760.

[43] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. 2022. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022,*

Proceedings, Part XXVI. Springer, 631–648.

[44] 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*. 139–149.

[45] Hao Yang, Junyang Lin, An Yang, Peng Wang, Chang Zhou, and Hongxia Yang. 2022. Prompt Tuning for Generative Multimodal Pretrained Models. *arXiv preprint arXiv:2208.02532* (2022).

[46] Zhengyuan Yang, Zhe Gan, Jianfeng Wang, Xiaowei Hu, Yumao Lu, Zicheng Liu, and Lijuan Wang. 2022. An empirical study of gpt-3 for few-shot knowledge-based vqa. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 3081–3089.

[47] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 1–9.

[48] Yan Zeng, Xinsong Zhang, and Hang Li. 2022. Multi-Grained Vision Language Pre-Training: Aligning Texts with Visual Concepts. In *International Conference on Machine Learning*. PMLR, 25994–26009.

[49] Aston Zhang, Yi Tay, SHUAI Zhang, Alvin Chan, Anh Tuan Luu, Siu Hui, and Jie Fu. 2020. Beyond Fully-Connected Layers with Quaternions: Parameterization of Hypercomplex Multiplications with $1/n$ Parameters. In *International Conference on Learning Representations*.

[50] Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. 2021. Vinvl: Revisiting visual representations in vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5579–5588.

[51] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2022. Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. In *The Eleventh International Conference on Learning Representations*.

[52] Zhengkun Zhang, Wenya Guo, Xiaojun Meng, Yasheng Wang, Yadao Wang, Xin Jiang, Qun Liu, and Zhenglu Yang. 2022. Hyperpelt: Unified parameter-efficient language model tuning for both language and vision-and-language tasks. *arXiv preprint arXiv:2203.03878* (2022).

[53] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. 2022. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16816–16825.

[54] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. 2022. Learning to prompt for vision-language models. *International Journal of Computer Vision* 130, 9 (2022), 2337–2348.