MixPHM: Redundancy-Aware Parameter-Efficient Tuning for Low-Resource Visual Question Answering
Supplemental Material

Jingjing Jiang  Nanning Zheng
Institute of Artificial Intelligence and Robotics, Xi’an Jiaotong University

The document provides some supplementary materials for our experiments. Specifically, in Sec. 1, we explore the impact of different routing mechanisms and hyperparameters on MixPHM performance. Sec. 2 presents some visualization results of our method. Sec. 4 describes more implementation details. Sec. 3 provides additional results using pretrained X-VLM on VQA v2.

1. Ablation Study and Parameter Analysis

In this section, using pretrained VL-T5 [2] as the underlying pretrained VLMs, we conduct additional ablation experiments on routing mechanisms and hyperparameter analysis on VQA v2, GQA, and OK-VQA with \(N_D = 64\).

Impact of Different Routing Mechanisms. In MixPHM, in addition to the training speed, i.e., T/Itr (s). To analyze the impact of different routing strategies on performance and speed, we first introduce two random routing methods, i.e., token-level and sentence-level routing [3]. In addition, we develop a simple representation-based rounding by averaging the outputs of all PHM-experts in each MixPHM. Table 1 shows that random routing mechanism is the fastest and has the best performance on both VQA v2 and OK-VQA.

Impact of Hyperparameters. To investigate the impact of different hyperparameters on MixPHM, we conduct experiments by varying \(N_c\), \(d_r\), \(d_k\), and \(n\). More specifically, we consider the following settings: \(N_c \in \{1, 2, 4, 8, 12\}\), \(d_r \in \{48, 64, 96, 192\}\), \(d_k \in \{1, 8, 16, 24\}\), and \(n \in \{2, 4, 8, 16\}\). The results in Table 2 show that changing these hyperparameters has only a slight impact on the performance of MixPHM. In addition, the performance of MixPHM with different hyperparameters always outperforms full finetuning. This suggests that the performance improvement brought by MixPHM does not significantly depend on the hyperparameter selection.

Impact of \(\alpha\). When tuning pretrained VLMs with MixPHM, \(\alpha\) controls the trade-off between redundancy regularization and generative modeling loss. To investigate the impact of \(\alpha\) on MixPHM, we perform experiments with different values of \(\alpha\), i.e., \(\alpha \in \{0.04, 0.06, 0.08, 0.1, 0.2, 0.4\}\). Figure 1 illustrates the curve of \(\text{QVA-Score}\) as \(\alpha\) increases. We observe that varying \(\alpha\) within a certain range \([0.04, 0.4]\) does not hinder the advantage of MixPHM over full finetuning. In addition, according to the results on three datasets, we empirically set \(\alpha\) to 0.2.

2. Visualization Results

We visualize some examples of the proposed MixPHM. As depicted in Figure 2, these answers are generated by the
The average VQA-Score with standard deviation across five seeds as $\alpha$ varies.

Figure 2. Qualitative results on VQA v2 validation set. The answer (A) is generated by the VL-T5 tuned with the proposed MixPHM. GT is the annotated answer and the corresponding score. We visualize the top-down attention [1] of images and mark the task-relevant tokens of questions for the first and second highest attention scores.
3. Results with Pretrained X-VLM

As a supplement to the results in Table 5 of the main paper, we utilize pretrained X-VLM [13] as a representative and compare our methods with state-of-the-art parameter-efficient tuning methods on VQA v2 validation set. The key hyperparameter settings for these parameter-efficient methods are the same as those in Table 4. The conclusions that we observe in Table 3 are consistent with Table 5, i.e., our method consistently outperforms existing parameter-efficient tuning methods when using other pretrained VLMs, which further demonstrates the generalization capability of MixPHM.

4. Implementation Details

For parameter-efficient tuning methods, we search the bottleneck dimension $d_e$ from $\{48, 64, 96, 192\}$ for all adapter-based methods (i.e., MixPHM, AdaMix, Pfeiffer, Houlsby and Compacter), the number of experts $N_e$ from $\{1, 2, 4, 8, 12\}$ for MixPHM and AdaMix, the rank dimension $d_r$ (for MixPHM and Compacter), $r$ (for LoRA) from $\{1, 8, 16, 24\}$, as well as the number of summations of Kronecker product $n$ from $\{2, 4, 8, 16\}$ for MixPHM and Compacter. Table 4 presents the final configuration of the hyperparameters used in our experiments. For MixPHM, we set the trade-off factor $\alpha$ to 0.2.

All methods are implemented using Pytorch [9] on an NVIDIA GeForce RTX 3090Ti GPU. In addition, we also perform a grid search to select the best learning rate from $\{5 \times 10^{-5}, 1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}, 5 \times 10^{-3}\}$. The batch size and the number of epochs are set to 16 and 1000, respectively. We utilize AdamW optimizer [8] and the early stopping strategy with a patience of 200 non-increasing epochs, where the stopping metric is the VQA-Score on the development set $D_{dev}$ of datasets.

References


<table>
<thead>
<tr>
<th>Method</th>
<th>Learning rate</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finetuning</td>
<td>$5 \times 10^{-5}$</td>
<td>—</td>
</tr>
<tr>
<td>BitFit</td>
<td>$5 \times 10^{-5}$</td>
<td>—</td>
</tr>
<tr>
<td>LoRA</td>
<td>$5 \times 10^{-5}$</td>
<td>$r = 4$</td>
</tr>
<tr>
<td>Compacter</td>
<td>$5 \times 10^{-3}$</td>
<td>$d_e = 64$, $d_k = 8$, $n = 4$</td>
</tr>
<tr>
<td>Houlsby</td>
<td>$5 \times 10^{-5}$</td>
<td>$d_e = 64$</td>
</tr>
<tr>
<td>Pfeiffer</td>
<td>$5 \times 10^{-5}$</td>
<td>$d_e = 64$</td>
</tr>
<tr>
<td>AdaMix</td>
<td>$5 \times 10^{-4}$</td>
<td>$N_e = 4$, $d_e = 64$</td>
</tr>
<tr>
<td>MixPHM</td>
<td>$5 \times 10^{-3}$</td>
<td>$N_e = 4$, $d_e = 64$, $d_k = 8$, $n = 4$</td>
</tr>
</tbody>
</table>

Table 4. Hyperparameter settings of all parameter-efficient tuning methods. $N_e$: the number of experts, $d_e$: bottleneck dimension, $d_k$ and $r$: rank dimension, $n$: the number of summations of Kronecker product.