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 $N_{\mathcal{D}} = 64$.

Impact of Different Routing Mechanisms. In MixPHM, in addition to performance, routing mechanisms also affect 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_e, d_r, d_k, and n. More specifically, we consider the following settings: $N_e \in \{1, 2, 4, 8, 12\}$, $d_r \in \{48, 64, 96, 192\}$, $d_k \in \{1, 8, 16, 24\}$, 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 α. When tuning pretrained VLMs with MixPHM, α controls the trade-off between redundancy regularization and generative modeling loss. To investigate the impact of α on MixPHM, we perform experiments with different values of α, *i.e.*, $\alpha \in \{0.04, 0.06, 0.08, 0.1, 0.2, 0.4\}$. Figure 1 illustrates the curve of VQA-Score as α increases. We observe that varying α 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 α to 0.2.

Method	T/Itr (s)	VQA v2	GQA	OK-VQA
MixPHM-Token	0.693	47.67 ± 0.92	36.23 ± 0.89	17.77 ± 0.89
MixPHM-Sent	0.683	47.69 ± 0.99	36.13 ± 0.86	17.83 ± 1.32
MixPHM-Rep	0.675	48.00 ± 0.95	**36.77** ± 0.55	18.25 ± 1.46
MixPHM	0.668	**48.26** ± 0.56	36.75 ± 0.55	**18.58** ± 1.42

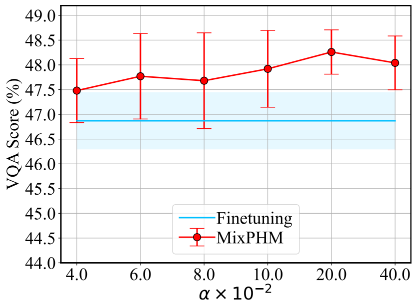
Table 1. **Ablation on different routing mechanisms with $N_{\mathcal{D}} = 64$.** T/Itr (s) is the average tuning time for each iteration.

HP		#Param	VQA v2	GQA	OK-VQA
Finetuning		224.54	46.87 ± 0.57	34.22 ± 0.59	16.65 ± 1.02
N_e	1	0.34	47.30 ± 0.97	36.30 ± 0.83	17.59 ± 0.97
	2	0.52	47.90 ± 0.65	**36.88** ± 0.75	18.08 ± 1.28
	4	0.87	**48.26** ± 0.56	36.75 ± 0.55	**18.58** ± 1.42
	8	1.59	48.09 ± 0.67	36.50 ± 0.81	18.51 ± 1.29
	12	2.30	47.80 ± 0.72	36.30 ± 0.80	18.43 ± 1.50
d_r	48	0.86	**48.36** ± 0.97	36.36 ± 0.32	18.05 ± 0.85
	64	0.87	48.26 ± 0.56	**36.75** ± 0.55	**18.58** ± 1.42
	96	0.91	48.05 ± 0.82	36.36 ± 0.51	18.39 ± 0.96
	192	1.00	47.97 ± 1.17	36.37 ± 0.82	18.26 ± 1.20
d_k	1	0.18	47.87 ± 0.73	35.74 ± 0.70	17.04 ± 0.81
	8	0.87	48.26 ± 0.56	**36.75** ± 0.55	18.58 ± 1.70
	16	1.67	**48.35** ± 1.14	36.62 ± 0.35	18.22 ± 1.36
	24	2.47	48.07 ± 1.12	36.42 ± 0.52	**18.79** ± 1.18
n	2	0.87	48.17 ± 0.93	36.53 ± 0.32	18.43 ± 0.75
	4	0.87	**48.26** ± 0.56	**36.75** ± 0.55	**18.58** ± 1.42
	8	0.87	47.97 ± 1.08	36.37 ± 0.56	17.41 ± 1.05
	16	0.88	46.65 ± 1.10	35.46 ± 0.55	17.52 ± 0.63

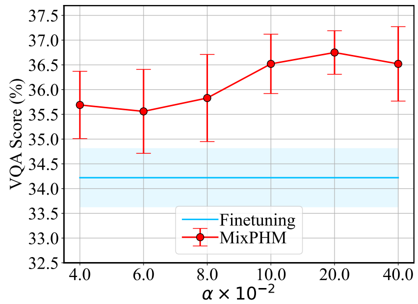
Table 2. **Impact of hyperparameters (HP) on MixPHM.** N_e: the number of PHM-experts, d_r: bottleneck dimension, d_k: rank dimension, n: the number of summations of Kronecker product.

2. Visualization Results

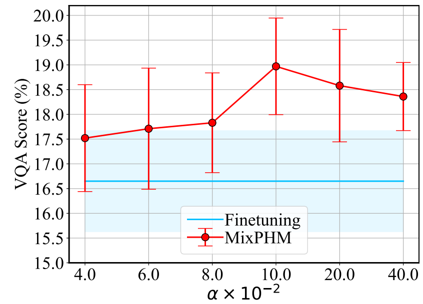
We visualize some examples of the proposed MixPHM. As depicted in Figure 2, these answers are generated by the



(a) VQA v2



(b) GQA



(c) OK-VQA

Figure 1. **The average VQA-Score with standard deviation across five seeds as α varies.**

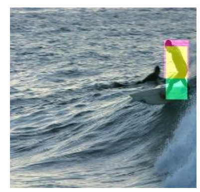
Q: What is the person doing in the air?
GT: snowboarding (1.0), jumping (0.6)
A: snowboarding

Q: Who wants to play?
GT dog (1.0)
A: dog

Q: Of the words on the signs, which ones are official?
GT: both (0.3), left (0.3)
A: both

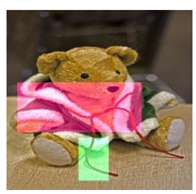
Q: Are those ducks in the water?
GT: no (1.0)
A: no

Q: How many people are in the picture?
GT: 2 (1.0)
A: 2



Q: Is the person surfing?
GT: yes (1.0), surfing (0.3)
A: yes

Q: What brand is that laptop?
GT: apple (1.0)
A: apple

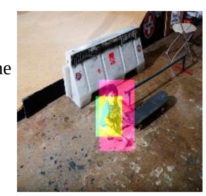


Q: What color is the bear's coat?
GT: green (1.0), green and white (0.9)
A: green

Q: How many baby teeth are visible?
GT: 4 (1.0), 3 (0.6)
A: 3

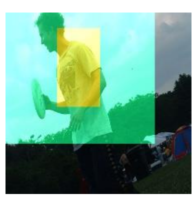
Q: What kind of material is this made of?
GT: wood (1.0)
A: wood

Q: What is the number on the player's back?
GT: 30 (1.0)
A: 30

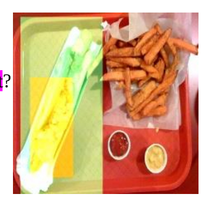


Q: Who is in the photo?
GT: child (1.0), baby (0.9), kid (0.3), boy (0.3)
A: kid

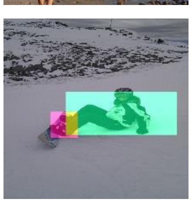
Q: What are they getting ready to do?
GT: surf (1.0)
A: surf



Q: Is the man wearing a shirt?
GT: yes (1.0)
A: yes



Q: What is on top of the hot dog?
GT: relish (0.6), pickle (0.6), onion (0.3)
A: relish



Q: Will someone be injured?
GT: no (1.0), yes (0.6)
A: yes

Q: Which color are the bananas?
GT: yellow (1.0)
A: yellow

Q: Is this woman doing something active?
GT: no (1.0)
A: no

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.

Method	#Param (M)	#Param (%)	#Sample $N_{\mathcal{D}}$=16	$N_{\mathcal{D}}$=32	$N_{\mathcal{D}}$=64	$N_{\mathcal{D}}$=100	$N_{\mathcal{D}}$=500	$N_{\mathcal{D}}$=1,000
Finetuning	293.48	100%	26.63 $\pm$0.98	29.33 $\pm$1.68	30.45 $\pm$1.80	31.48 $\pm$1.57	38.96 $\pm$1.56	43.92 $\pm$1.22
BitFit [12]	0.29	0.13%	25.48 $\pm$3.81	28.90 $\pm$1.14	30.73 $\pm$1.18	31.92 $\pm$1.14	36.77 $\pm$1.32	40.77 $\pm$0.79
LoRA [5]	0.37	0.13%	25.31 $\pm$1.50	26.91 $\pm$3.09	30.52 $\pm$1.67	31.97 $\pm$1.11	36.13 $\pm$1.12	40.49 $\pm$0.87
Compacter [7]	0.25	0.09%	25.69 $\pm$2.34	28.04 $\pm$1.63	28.10 $\pm$2.06	31.35 $\pm$0.34	35.91 $\pm$0.65	40.44 $\pm$0.77
Houlsby [4]	3.57	1.20%	26.54 $\pm$2.57	29.34 $\pm$2.25	30.74 $\pm$1.20	31.71 $\pm$1.43	38.48 $\pm$0.91	41.96 $\pm$0.72
Pfeiffer [10]	1.78	0.60%	26.57 $\pm$2.00	28.46 $\pm$1.74	29.22 $\pm$2.56	31.95 $\pm$1.34	37.39 $\pm$0.73	40.96 $\pm$1.09
AdaMix [11]	4.44	1.49%	26.11 $\pm$1.58	28.91 $\pm$1.36	30.71 $\pm$2.05	31.15 $\pm$1.26	38.48 $\pm$1.53	43.26 $\pm$0.85
MixPHM	0.66	0.22%	27.54 $\pm$1.52	30.65 $\pm$1.09	31.80 $\pm$1.61	32.58 $\pm$1.09	41.05 $\pm$1.22	48.06 $\pm$0.64

Table 3. **Experimental results with pretrained X-VLM.** The average VQA-Score with standard deviation across 5 different seeds are evaluated on VQA v2 validation set. The best and second best parameter-efficient tuning methods are highlighted. The number of tuned parameters and the percentage of tuned parameters relative to X-VLM (*i.e*., 293.48M) are reported.

Method	Learning rate	Configuration
Finetuning	5×10^{-5}	—
BitFit	5×10^{-5}	—
LoRA	5×10^{-5}	$r = 4$
Compacter	5×10^{-3}	$d_r = 64, d_k = 8, n = 4$
Houlsby	5×10^{-5}	$d_r = 64$
Pfeiffer	5×10^{-5}	$d_r = 64$
AdaMix	5×10^{-4}	$N_e = 4, d_r = 64$
MixPHM	5×10^{-3}	$N_e = 4, d_r = 64, d_k = 8, n = 4$

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

VL-T5 tuned via MixPHM on VQA v2 with $N_{\mathcal{D}} = 64$. In addition, we visualize the top-down attention [1] of images and mark the top two task-relevant tokens of questions. Specifically, we follow a recent work [6] to compute an attention score between task-relevant representations and visual input features obtained using bottom-up Faster R-CNN [1] and visualize the top-down attention for the first and second highest scores. Analogously, we compute the score between task-relevant representations and linguistic embeddings of questions and mark the tokens for the first and second highest scores. Figure 2 qualitatively shows that our MixPHM can generate the consistent and question-relevant visual and textual attention.

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_r 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 α 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 $\mathcal{D}_{\text{dev}}$ of datasets.

References

[1] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*, pages 6077–6086, 2018. 2, 3

[2] Jaemin Cho, Jie Lei, Hao Tan, and Mohit Bansal. Unifying vision-and-language tasks via text generation. In *ICML*, pages 1931–1942, 2021. 1

[3] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021. 1

[4] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona

Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *ICML*, pages 2790–2799, 2019. 3

[5] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022. 3

[6] Jingjing Jiang, Ziyi Liu, and Nanning Zheng. Correlation information bottleneck: Towards adapting pretrained multi-modal models for robust visual question answering. *arXiv preprint arXiv:2209.06954*, 2022. 3

[7] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. In *NeurIPS*, pages 1022–1035, 2021. 3

[8] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 3

[9] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8024–8035, 2019. 3

[10] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*, 2020. 3

[11] Yaqing Wang, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. Adamix: Mixture-of-adapter for parameter-efficient tuning of large language models. In *EMNLP*, pages 5744–5760, 2022. 3

[12] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *ACL*, pages 1–9, 2022. 3

[13] Yan Zeng, Xinsong Zhang, and Hang Li. Multi-grained vision language pre-training: Aligning texts with visual concepts. In *ICML*, pages 25994–26009, 2022. 3