Supplementary Material of Once-Tuning-Multiple-Variants: Tuning Once and Expanded as Multiple Vision-Language Model Variants

Chong Yu[1] Tao Chen[2,*] Zhongxue Gan[1,*]
[1]Academy for Engineering and Technology, Fudan University, Shanghai 200433, China
[2]School for Information Science and Technology, Fudan University, Shanghai 200433, China
21110860050@m.fudan.edu.cn; {eetchen, ganzhongxue}@fudan.edu.cn

In this **Appendix**, we will provide some supplementary materials and more experimental results for the proposed Once-Tuning-Multiple-Variants (OTMV) VLM tuning method beyond the tight page limitation in manuscript.

1. Analysis of why OTMV expanded model can approximate the original VLM

In this section, we want to provide the explanation why the OTMV expanded VLM variants can approximate the original naive VLM with negligible accuracy gap.

Because most typical VLM models are based on transformer structures, without loss of generality, we use a transformer-based VLM as an example. For one naive transformer block in original VLM, we can regard its output tensor $\boldsymbol{y}$ as a nonlinear function F applied to its corresponding input tensor $\boldsymbol{x}$. By expanding the nonlinear function F, we can get the following form:

$$\begin{aligned} \boldsymbol{y} &= F\boldsymbol{x} = LN_1\left(\mathbf{W}_f\boldsymbol{z}+\boldsymbol{z}\right), \\ \boldsymbol{z} &= LN_2\left\{\mathbf{W}_o\left[Soft\left(\frac{\mathbf{W}_q\boldsymbol{x}(\mathbf{W}_k\boldsymbol{x}))^T}{\sqrt{d}}\right)\mathbf{W}_v\boldsymbol{x}\right]+\boldsymbol{x}\right\} \end{aligned} \tag{1}$$

where $\mathbf{W}_f$, $\mathbf{W}_q$, $\mathbf{W}_k$, $\mathbf{W}_v$, $\mathbf{W}_o$ are the weight parameters of feed-forward layer, query, key, value, and output linear projection layers. LN_1 and LN_2 are two separate layer normalization layers. $Soft$ and $\sqrt{d}$ refers to the softmax operation and normalization constant in attention of the transformer block. In the proposed OTMV-expanded VLM model structure, all the weight tensors in feed-forward layer, query, key, value, and linear projection layers, i.e., $\mathbf{W}_f$, $\mathbf{W}_q$, $\mathbf{W}_k$, $\mathbf{W}_v$, $\mathbf{W}_o$ are all shared among the key and high-order series weight terms. But the OTMV coefficients and layer normalization: LN_1, LN_2 are not shared.

From the whole model view[1], a naive VLM model consists with K naive transformer blocks (represented by $F_1, \cdots, F_K$ functions), $\boldsymbol{x}$ and $\boldsymbol{y}_{naive}$ are input and output tensor for this model, then it can be expressed as:

$$\boldsymbol{y}_{naive} = F_K \cdots F_3F_2F_1\boldsymbol{x} = \mathbb{F}\boldsymbol{x} \tag{2}$$

If we approximate with a OTMV-expanded VLM model consists with one shared key (i.e., first-order) transformer block and $K-1$ high-order transformer blocks, then:

$$\begin{aligned} \boldsymbol{y}_{otmv} &= c_1F_1\boldsymbol{x} + c_2F_1c_1F_1\boldsymbol{x} + c_3F_1c_2F_1c_1F_1\boldsymbol{x} + \cdots \\ &= c_1F_1\boldsymbol{x} + \tilde{c_2}F_1^2\boldsymbol{x} + \tilde{c_3}F_1^3\boldsymbol{x} + \cdots + \tilde{c_K}F_1^K\boldsymbol{x} \end{aligned} \tag{3}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}_{otmv}$ are input and output tensor for the OTMV-expanded model, $c_1, c_2, \cdots, c_K$ are the OTMV coefficients, and $\tilde{c_2}, \cdots, \tilde{c_K}$ are their combined form. By comparing the (2) and (3) expressions, the OTMV-expanded model tries to approximate the nonlinear function $\mathbb{F}$ with the first-order OTMV block's function F_1 and its series expansion form. Based on the ***mathematical series expansion theorem***, the value of an arbitrary nonlinear function can be expressed as an infinite sum of its derivative series terms.

So in theory, we can change the naive VLM model and constructing it as the OTMV-expanded structure. And the OTMV-expanded model is a perfect approximation of the naive VLM model. With the same training session, it will have a negligible accuracy gap with the naive model type.

Activation Functions. The activation functions in the naive VLM models are parameterless, and we can express them as the extra multiplication of the outputs from Multilayer Perceptron (MLP). So, the activation functions will not influence the OTMV-expanded model's approximation.

Layer Normalization Functions. The layer normalization functions in the naive VLM models play a crucial role by stabilizing the training process and improving model performance. Layer normalization computes normalization statistics across all features within a layer for each instance independently. In the OTMV-expanded model, we share the first-order transformer block's weight and bias while keeping the layer normalization in the original structure not changed. So, it will not alter the normalization statistics

*Tao Chen and Zhongxue Gan are corresponding authors.

[1]For illustration, we do not distinguish the transformer blocks from different modalities. Because we have the ablation study to discuss the transformer blocks sharing strategies within or across modalities, it will not break the analysis here.

mechanism, i.e., the layer normalization functions will not influence the OTMV-expanded model's approximation.

Inside Attention Structures. In the naive VLM models, there are two matrix multiplication operations among Q, K, and V, as well as a softmax calculation inside the attention. However, all these structures are parameterless, and the corresponding operations are calculated on activation tensors. The design of the OTMV tuning saves the memory consumption of the learnable parameters, so it will not alter the inside structures and operations for the attention part.

2. Hyper-parameters in experiments

For VLMo (*Base and Large*)[2] [1], BLIP[3] [8] and BEiT-3 (*Base and Large*)[4] [12] baseline models and the OTMV-tuned versions, we follow the hyper-parameters settings in public repositories marked by the footnotes and detailed list in Table 1. Multiple V100 [9] and A100 [10] GPUs are used for data-parallel training in each training or fine-tuning experiment.

3. Supplementary experiments

3.1. Comparison between OTMV and fine-tuning

To compare the efficacy of OTMV-tuning workflow with the traditional fine-tuning (including adaptation) paradigms, we check the accuracy of OTMV-tuned VLM models and compare them to the models tuned by typical parameter-efficient tuning and adaptation techniques, PET-Adapter[5] [5], LoRA[6] [6], VL-Adapter[7] [11], VL-PET[8] [7], Once-for-All[9] [2], MatFormer [4] and Flextron [3]. The detailed accuracy is reported in Table 2. The baseline is the naive VLM models going through the full fine-tuning in each downstream task. To make *a fair comparison*, the VLM variants tuned with other methods in Table 2 all go through the same fine-tuning stages to align with the baseline setting.

From the Table 2, we can see that the accuracy of the VLM model variants tuned by the typical parameter-efficient tuning and adaptation techniques has ***some gaps*** compared to the full fine-tuning baseline models. The accuracy of the OTMV-tuned VLM model variants has an accuracy *boost* over the counterparts, and the gaps with the full fine-tuning baseline models are *negligible*. Moreover, the VLM model variants tuned in the all-in-one fine-tuning paradigms also have an apparent accuracy gap compared to the other parameter-efficient tuning paradigms. The potential reason is that Once-for-All, MatFormer, and Flextron are designed for CNN and pure large language models rather than explicitly designed for VLM model structures.

The typical parameter-efficient tuning and adaptation techniques freeze most of the pre-trained parameters and only tune the limited learnable parameters in the adapters so that the tuning cost can be *significantly reduced*, such as PET-Adapter, LoRA, VL-Adapter, and VL-PET tuning paradigms. The focus of all-in-one fine-tuning paradigms is unifying the tuning of the elastic model structures and parameters in one process. So, the tuning cost is reduced from the number of tasks multiplied by the single full fine-tuning cost in each task. However, this unifying is ***a non-trivial effort to maintain the accuracy*** for multiple elastic model structures; *a longer tuning process* than the single full fine-tuning cost in a single task is expected for accuracy compensation. That explains why the tuning costs from Once-for-All, MatFormer, and Flextron are more significant than the baseline cost. The proposed OTMV tuning method does not change the naive full fine-tuning process. The only overhead comes from the extra learnable OTMV coefficients, which is ***tiny*** as the coefficients are all scaler variables. So, the tuning cost from OTMV is in the 1.01-1.02$\times$ range against the baseline cost.

3.2. Ablation of OTMV skip probabilities

To measure the accuracy influence of different OTMV gate skip probabilities and the dynamic expansion capabilities, we make an ablation study in this section. We choose the OTMV-tuned BEiT-3 (Large) as the target model and evaluate the accuracy of OTMV dynamic expanded variants with different OTMV gate skip probabilities. Figure 1 shows the ablation comparison for the visual question answering task on VQA v2.0, and Figure 2 shows the ablation comparison for the image caption task on COCO.

From the results shown in Figures 1 and 2, if the OTMV gate skip probability has a minimal value (e.g., 0.01), the dynamic expansion capabilities of the OTMV-tuned VLM will be apparently injured. The accuracy drops significantly, especially for the expanded variants with small model parameter sizes. If the OTMV gate skip probability has a relatively large value (e.g., 0.2), the dynamic expansion capabilities of the OTMV-tuned VLM will not be harmed. However, the accuracy of the OTMV-expanded VLM variants with larger model parameter sizes has a slight drop. The skip probability between 0.05 and 0.1 provides ***a better balance between the acceptable accuracy degradation and the dynamic expansion capability*** for the OTMV-expanded VLM variants.

[2] https://github.com/microsoft/unilm/tree/master/vlmo
[3] https://github.com/salesforce/BLIP
[4] https://github.com/microsoft/unilm/tree/master/beit3
[5] https://github.com/google-research/adapter-bert
[6] https://github.com/microsoft/LoRA
[7] https://github.com/ylsung/VL_adapter
[8] https://github.com/HenryHZY/VL-PET
[9] https://github.com/mit-han-lab/once-for-all

Models	Task	Datasets	Optimizer	Initial LR	Weight Decay	Epochs	Batch Size	GPU Num
VLMo (Base)	**Visual Question Answering**	**VQA v2.0**	AdamW	3e-5	0.01	10	128	8
	Visual Reasoning	**NLVR2**	AdamW	5e-5	0.01	10	128	8
	Image-Text Retrieval	**COCO**	AdamW	3e-5	0.01	50	3072	8
	Image-Text Retrieval	**Flickr30k**	AdamW	3e-5	0.01	50	3072	8
VLMo (Large)	**Visual Question Answering**	**VQA v2.0**	AdamW	1.5e-5	0.01	10	128	8
	Visual Reasoning	**NLVR2**	AdamW	3e-5	0.01	10	128	8
	Image-Text Retrieval	**COCO**	AdamW	2e-5	0.01	50	3072	8
	Image-Text Retrieval	**Flickr30k**	AdamW	2e-5	0.01	50	3072	8
BLIP	**Visual Question Answering**	**VQA v2.0**	AdamW	2e-5	0.05	10	256	16
	Visual Reasoning	**NLVR2**	AdamW	3e-5	0.05	10	256	16
	Image-Text Retrieval	**COCO**	AdamW	1e-5	0.05	6	256	8
	Image-Text Retrieval	**Flickr30k**	AdamW	1e-5	0.05	6	256	8
	Image Captioning	**COCO**	AdamW	1e-5	0.05	5	256	8
BEiT-3 (Base)	**Visual Question Answering**	**VQA v2.0**	AdamW	3e-5	0.01	10	128	8
	Visual Reasoning	**NLVR2**	AdamW	5e-4	0.2	20	256	8
	Image-Text Retrieval	**COCO**	AdamW	2e-4	0.05	15	3072	16
	Image-Text Retrieval	**Flickr30k**	AdamW	1e-4	0.05	20	3072	16
	Image Captioning	**COCO**	AdamW	4e-5	0.05	10	256	8
BEiT-3 (Large)	**Visual Question Answering**	**VQA v2.0**	AdamW	2e-5	0.01	10	128	8
	Visual Reasoning	**NLVR2**	AdamW	1e-4	0.2	20	256	8
	Image-Text Retrieval	**COCO**	AdamW	5e-5	0.05	15	3072	32
	Image-Text Retrieval	**Flickr30k**	AdamW	5e-5	0.05	20	3072	32
	Image Captioning	**COCO**	AdamW	8e-6	0.05	10	256	8

Table 1. Experiments hyper-parameters for the vision-language models fine-tuned and tested in this paper.

Models	Tuning Models	Tuning Cost	Tasks	Visual Question Answering VQA v2.0		Visual Reasoning NLVR2		Image Captioning COCO	
			Datasets Metrics	test-dev (%)	test-std (%)	dev (%)	test-P (%)	BLEU@4	CIDEr
BLIP	*Full Fine-tuning (Baseline)*	1×		78.25	78.32	82.15	82.24	39.70	133.30
	PET-Adapter tuning	0.30×		77.65 (-0.60)	77.69 (-0.63)	79.95 (-2.20)	80.01 (-2.13)	39.11 (-0.59)	131.93 (-1.37)
	LoRA tuning	0.18×		77.61 (-0.64)	77.66 (-0.66)	79.89 (-2.26)	79.98 (-2.16)	39.07 (-0.63)	131.81 (-1.49)
	VL-Adapter tuning	0.20×		78.17 (-0.08)	78.21 (-0.11)	80.27 (-1.88)	80.34 (-1.90)	39.32 (-0.38)	132.12 (-1.18)
	VL-PET tuning	0.19×		78.02 (-0.23)	78.10 (-0.22)	81.06 (-1.09)	81.17 (-1.07)	39.46 (-0.24)	132.85 (-0.45)
	Once-for-All tuning	1.95×		77.13 (-1.12)	77.20 (-1.12)	79.79 (-2.36)	79.89 (-2.24)	39.03 (-0.67)	131.75 (-1.55)
	MatFormer tuning	2.56×		77.32 (-0.93)	77.31 (-1.01)	79.87 (-2.28)	79.96 (-2.17)	39.06 (-0.64)	131.82 (-1.48)
	Flextron tuning	1.76×		77.34 (-0.91)	77.33 (-0.99)	79.93 (-2.22)	79.99 (-2.14)	39.08 (-0.62)	131.86 (-1.44)
	OTMV tuning	1.01×		78.19 (-0.06)	78.27 (-0.05)	82.03 (-0.12)	82.17 (-0.07)	39.61 (-0.09)	133.32 (+0.02)
BEiT-3 (Base)	*Full Fine-tuning (Baseline)*	1×		78.45	78.52	84.61	85.28	39.35	133.60
	PET-Adapter tuning	0.26×		76.79 (-1.66)	77.00 (-1.52)	83.59 (-1.02)	84.29 (-0.99)	38.86 (-0.49)	132.88 (-0.72)
	LoRA tuning	0.16×		76.78 (-1.67)	77.01 (-1.51)	83.61 (-1.00)	84.30 (-0.98)	38.88 (-0.47)	132.86 (-0.74)
	VL-Adapter tuning	0.18×		77.14 (-1.31)	77.20 (-1.32)	84.06 (-0.55)	84.74 (-0.54)	39.18 (-0.17)	133.11 (-0.49)
	VL-PET tuning	0.17×		77.60 (-0.85)	77.50 (-1.02)	84.32 (-0.29)	84.94 (-0.34)	39.27 (-0.08)	133.30 (-0.30)
	Once-for-All tuning	1.88×		76.45 (-2.00)	76.83 (-1.69)	83.22 (-1.39)	84.04 (-1.24)	38.52 (-0.83)	132.57 (-1.03)
	MatFormer tuning	2.59×		76.57 (-1.88)	76.90 (-1.62)	83.33 (-1.28)	84.17 (-1.11)	38.68 (-0.67)	132.77 (-0.83)
	Flextron tuning	1.68×		76.69 (-1.76)	76.99 (-1.53)	83.45 (-1.16)	84.26 (-1.02)	38.73 (-0.62)	132.82 (-0.78)
	OTMV tuning	1.02×		78.32 (-0.13)	78.41 (-0.11)	84.53 (-0.08)	85.22 (-0.06)	39.31 (-0.04)	133.49 (-0.11)
BEiT-3 (Large)	*Fine-tuning (Baseline)*	1×		82.53	82.58	89.22	89.95	41.46	143.29
	PET-Adapter tuning	0.29×		80.99 (-1.54)	81.04 (-1.54)	88.19 (-1.03)	88.85 (-1.10)	40.86 (-0.60)	142.02 (-1.27)
	LoRA tuning	0.17×		80.97 (-1.56)	81.03 (-1.55)	88.21 (-1.01)	88.91 (-1.04)	40.83 (-0.63)	141.99 (-1.30)
	VL-Adapter tuning	0.20×		81.35 (-1.18)	81.41 (-1.17)	88.64 (-0.58)	89.38 (-0.57)	41.24 (-0.22)	142.79 (-0.50)
	VL-PET tuning	0.19×		81.64 (-0.89)	81.52 (-1.06)	88.91 (-0.31)	89.58 (-0.37)	41.37 (-0.09)	142.95 (-0.34)
	Once-for-All tuning	1.92×		80.61 (-1.92)	80.75 (-1.83)	88.01 (-1.21)	88.66 (-1.29)	40.71 (-0.75)	141.82 (-1.47)
	MatFormer tuning	2.63×		80.84 (-1.69)	80.92 (-1.66)	88.11 (-1.11)	88.77 (-1.18)	40.77 (-0.69)	141.95 (-1.34)
	Flextron tuning	1.72×		80.90 (-1.63)	80.96 (-1.62)	88.15 (-1.07)	88.80 (-1.15)	40.80 (-0.66)	141.97 (-1.32)
	OTMV tuning	1.02×		82.50 (-0.03)	82.59 (+0.01)	89.19 (-0.03)	89.96 (+0.01)	41.49 (+0.03)	143.30 (+0.01)

Table 2. Comparison among the naive VLM models with variants tuned by traditional fine-tuning, adaptation, and proposed OTMV methods to check their corresponding efficacy and tunning costs. (All values are averaged with 3 runs. Variances: 0.07)

Models	OTMV Coefficients	Tasks	Visual Question Answering		Visual Reasoning		Image-Text Retrieval			
		Datasets	VQA v2.0		NLVR2		COCO		Flickr30k	
		Metrics	test-dev (%)	test-std (%)	dev (%)	test-P (%)	I-TR (%)	T-IR (%)	I-TR (%)	T-IR (%)
VLMo (Base)	*Fixed as 1*		76.26 (-0.27)	76.50 (-0.31)	82.48 (-0.24)	83.03 (-0.26)	74.6 (-0.2)	56.6 (-0.4)	92.2 (-0.1)	79.0 (-0.3)
	Learnable		76.53	76.81	82.72	83.29	74.8	57.0	92.3	79.2
VLMo (Large)	*Fixed as 1*		79.33 (-0.59)	79.38 (-0.61)	85.11 (-0.55)	86.25 (-0.58)	77.9 (-0.4)	59.9 (-0.7)	95.1 (-0.3)	84.1 (-0.5)
	Learnable		79.92	79.99	85.66	86.83	78.3	60.6	95.4	84.6
BLIP	*Fixed as 1*		77.87 (-0.32)	77.92 (-0.35)	81.77 (-0.26)	81.88 (-0.29)	80.9 (-0.2)	63.7 (-0.3)	97.0 (-0.2)	87.2 (-0.2)
	Learnable		78.19	78.27	82.03	82.17	81.1	64.0	97.2	87.4
BEiT-3 (Base)	*Fixed as 1*		78.02 (-0.30)	78.08 (-0.33)	84.30 (-0.23)	84.95 (-0.27)	78.8 (-0.2)	60.8 (-0.4)	96.0 (-0.2)	85.9 (-0.2)
	Learnable		78.32	78.41	84.53	85.22	79.0	61.2	96.2	86.1
BEiT-3 (Large)	*Fixed as 1*		81.87 (-0.63)	81.93 (-0.66)	88.62 (-0.57)	89.38 (-0.58)	81.6 (-0.5)	62.8 (-0.6)	96.9 (-0.4)	87.8 (-0.3)
	Learnable		82.50	82.59	89.19	89.96	82.1	63.4	97.3	88.1

Table 3. Comparison between using the learnable OTMV coefficients and fixing OTMV coefficients as 1.

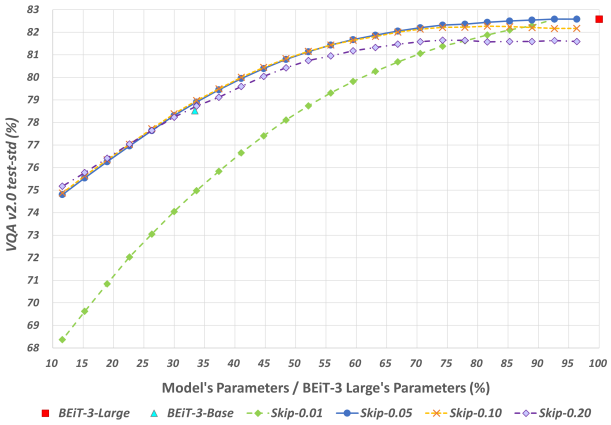

Figure 1. Ablation comparison of the OTMV gate skip probabilities with OTMV tuned BEiT-3 Large variants on visual question answering task (VQA v2.0 dataset).

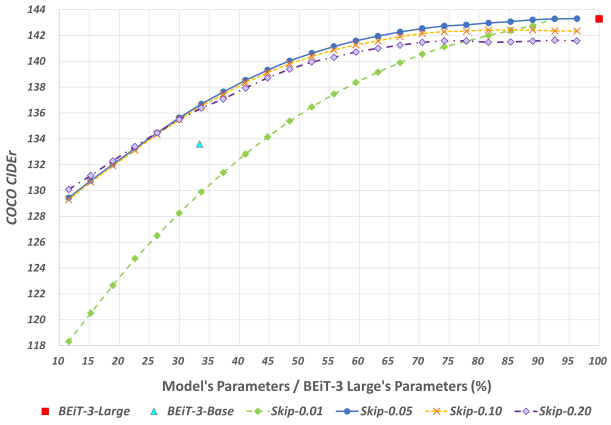

Figure 2. Ablation comparison of the OTMV gate skip probabilities with OTMV tuned BEiT-3 Large variants on image captioning task (COCO dataset).

3.3. Ablation of the influence from learnable or fixed OTMV coefficients

To measure the influence of learnable OTMV coefficients on accuracy, we conducted an ablation study to determine whether to use the learnable OTMV coefficients or fix the value of OTMV coefficients as one. The comparison results are shown in Table 3.

From the accuracy results in Table 3, the learnable OTMV coefficients show an ***obvious advantage*** against the fixed coefficients in various visual-language tasks, proving its accuracy contribution to the overall OTMV tuning workload.

3.4. Key weights merge strategies for feed forward

In the manuscript, we apply the elementwise mean to merge the key weight tensors of *Feed Forward* layers in the first transformer block in the vision modality backcone: $W_{FF(1)}^{V-Mod}$ and the counterpart in the language modality backbone: $W_{FF(1)}^{L-Mod}$ into a unified cross-vision-language key tensor, refer as $W_{mg\,FF(1)}^{VL-Mod}$.

$$W_{mg\,FF(1)}^{VL-Mod} = Mean_E\left(W_{FF(1)}^{V-Mod}, W_{FF(1)}^{L-Mod}\right) \quad (4)$$

We can also merge these two key tensors in other strategies. For example, we can merge them with weighted sum form, refer as the *weighted sum merging strategy*:

$$W_{mg\,FF(1)}^{VL-Mod} = \alpha \cdot W_{FF(1)}^{V-Mod} + \beta \cdot W_{FF(1)}^{L-Mod} \quad (5)$$

where α, β are the adjustment factors.

Another example is merging the two key tensors with least mean square error (MSE), refer as *MSE merging strategy*:

$$\min \sqrt{\frac{\left(W_{FF(1)}^{V-Mod} - W_{mg\,FF(1)}^{VL-Mod}\right)^2 + \left(W_{FF(1)}^{L-Mod} - W_{mg\,FF(1)}^{VL-Mod}\right)^2}{2}} \quad (6)$$

We also make a quick ablation to compare with these three merging strategies[10]. For the feed-forward layers, the

[10]For the *weighted sum merging strategy*, we choose several different group values for the adjustment factors α and β.

Methods	Tasks Datasets	Visual Question Answering VQA v2.0	Visual Reasoning NLVR2	Image Captioning COCO
Full Fine-tuning (Baseline)		1x	1x	1x
PET-Adapter tuning		0.64x	0.66x	0.63x
LoRA tuning		1x	1x	1x
VL-Adapter tuning		0.69x	0.72x	0.67x
VL-PET tuning		0.60x	0.61x	0.59x
Once-for-All tuning		1.27x	1.30x	1.24x
MatFormer tuning		1.14x	1.17x	1.13x
Flextron tuning		1.25x	1.26x	1.22x
EfficientVLM Compression		1.47x	1.49x	1.45x
DistillVLM Compression		1.16x	1.18x	1.14x
UPop Compression		1.21x	1.23x	1.20x
OTMV tuning		1.63x	1.65x	1.60x

Table 4. Comparison among the naive BEiT-3 Large model with variants tuned by traditional fine-tuning, adaptation, and proposed OTMV methods to check deployment performance on NVIDIA A100 GPU. For *visual question answering* task, the performance is measured with the batch size fixed to 16. For *visual reasoning* and *image captioning* tasks, the performance is measured with the batch size fixed to 32.

unified cross-vision-language key tensor shared with three merging strategies ***make marginal differences*** in the final accuracy metrics on various benchmarks. For example, the gap of the VQA scores on the VQA v2.0 dataset from these three merging strategies is only *0.15*. This phenomenon also support the hypothesis in the manuscript. While the ***Feed Forward*** layers are responsible for the nonlinear transformations of the attention outputs, and are ***not directly related*** to the alignment of the different modalities. So they will not cause apparent accuracy differences when sharing across modalities with different merging strategies.

With this experiment result, ***from the accuracy perspective***, we can also share the key weight tensors for the *Feed Forward* layers within the vision and language modalities, respectively. However, sharing across modalities leads to the minor physical memory cost for saving the key weight tensors of the *Feed Forward* layers. So, ***from the performance perspective***, sharing across modalities is preferred to sharing within the same modality.

3.5. Deployment of OTMV-tuned VLM variants

Because the OTMV-tuned VLM variants use the OTMV gates to skip the weight layers in the inference process, there is ***no reliance on the specific software stack support for actual deployment***. With the real saving of the VLM models' memory cost on hardware, it can bring a considerable performance boost. For example, the OTMV-tuned BEiT-3 Large variant with 60% parameters saving can have a 63% performance improvement on the A100 GPU.

In contrast, deploying compact variants obtained by traditional model compression techniques may rely on special support from the software stack. For example, the compressed sparse VLM variants require the sparse calculation kernels to run efficiently. The quantized VLM variants require specific quantized kernels and the extra handling of quantized scale factors to deploy the VLM model with the correct outputs.

We measure the deployment performance among the naive BEiT-3 Large model with variants tuned by traditional fine-tuning, adaptation, and proposed OTMV methods on NVIDIA A100 GPU [10]. The comparison results on *visual question answering*, *visual reasoning*, and *image captioning* tasks are shown in Table 4.

From the Table 4, we can see that the deployment performance of the BEiT-3 Large model variants tuned by the typical parameter-efficient tuning and adaptation techniques is usually *worse* than the full fine-tuning baseline model. The performance regression comes from the *overhead of inserting the additional adapters* in the original model structures. LoRA fuses the low-rank adapter parameters with the original pre-trained parameters, so there is no extra deployment performance overhead. The BEiT-3 Large model variants tuned in the all-in-one fine-tuning paradigms have an apparent performance boost compared to the other parameter-efficient tuning paradigms as well as the full fine-tuning baseline model. The performance boost mainly comes from *the benefit of elastic structures* designed with all-in-one fine-tuning paradigms. Compared to the full fine-tuning baseline model, the BEiT-3 Large model variants tuned in the compression techniques also have an apparent performance boost. The performance boost mainly comes from *the benefit of reduced parameters or FLOPs saving* by model compression paradigms. The OTMV-tuned BEiT-3 Large model has an apparent deployment performance boost, which is ***superior to prior arts***.

4. Limitations

We acknowledge the potential limitations of our study and the need for continuous improvement. While the dynamic expansion capacity of OTMV-tuned models is promising, it may require further validation when meeting the new-emerging tasks and datasets. We encourage the research community to extend our work, contributing to developing more efficient VLMs in various industrial applications.

5. Ethical considerations and statement

Our research was conducted with a commitment to ethical standards and considerations, ensuring that all aspects of the study were performed responsibly and with respect for the broader impacts on society and the environment.

Data Privacy and Security: All data used in this study were sourced from publicly available datasets, ensuring compliance with data privacy regulations and ethical guidelines. No personally identifiable information was used, and all data handling processes adhered to strict confidentiality and security protocols to prevent unauthorized access or misuse.

Environmental Impact: Training large-scale VLMs can be resource-intensive and have a significant ecological footprint. Efforts were made to optimize computational resources and reduce energy consumption wherever possible. The proposed Once-Tuning-Multiple-Variants (OTMV) paradigm is designed to be more efficient, reducing the need for multiple training runs and thereby lowering the overall carbon footprint.

References

[1] Hangbo Bao, Wenhui Wang, Li Dong, Qiang Liu, Owais Khan Mohammed, Kriti Aggarwal, Subhojit Som, Songhao Piao, and Furu Wei. Vlmo: Unified vision-language pre-training with mixture-of-modality-experts. *Advances in Neural Information Processing Systems*, 35:32897–32912, 2022. 2

[2] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*, 2020. 2

[3] Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. Flextron: Many-in-one flexible large language model. *arXiv preprint arXiv:2406.10260*, 2024. 2

[4] Fnu Devvrit, Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit S Dhillon, Yulia Tsvetkov, Hannaneh Hajishirzi, Sham M Kakade, Ali Farhadi, et al. Matformer: Nested transformer for elastic inference. In *Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ NeurIPS 2023)*, 2023. 2

[5] 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 *International conference on machine learning*, pages 2790–2799. PMLR, 2019. 2

[6] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. 2

[7] Zi-Yuan Hu, Yanyang Li, Michael R Lyu, and Liwei Wang. Vl-pet: Vision-and-language parameter-efficient tuning via granularity control. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3010–3020, 2023. 2

[8] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR, 2022. 2

[9] NVIDIA. NVIDIA Tesla V100 GPU Architecture, 2017. 2

[10] NVIDIA. NVIDIA A100 Tensor Core GPU Architecture, 2020. 2, 5

[11] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5227–5237, 2022. 2

[12] Wenhui Wang, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, et al. 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*, pages 19175–19186, 2023. 2