Supplementary Materials for VL-PET: Vision-and-Language Parameter-Efficient Tuning via Granularity Control

Zi-Yuan Hu[1,3] Yanyang Li[1] Michael R. Lyu[1] Liwei Wang[1,2*]
[1]The Chinese University of Hong Kong [2]Centre for Perceptual and Interactive Intelligence
[3]Shanghai Artificial Intelligence Laboratory
{zyhu22,yyli21,lyu,lwwang}@cse.cuhk.edu.hk

A. Dataset Statistics

For image-text tasks, we report the performance on Karpathy test/test-dev/test-P/Karpathy test split for VQA/GQA/NLVR^2/MSCOCO. For video-text tasks, we report the performance on test-public split for TVQA, How2QA, TVC and YC2C from VALUE benchmark [15]. The detailed statistics of image-text and video-text datasets are shown in Tab. 1 and Tab. 2, respectively.

B. Implementation Details

Combining a pre-trained vision model with a generative PLM, we perform multi-task learning via unified text generation. Following [2], we use CLIP [18] as our frozen vision encoder for offline visual feature extraction, and BART-base [13] and T5-base [19] as our generative PLM backbones for parameter-efficient tuning. A trainable visual projector (i.e., a trainable linear layer) is utilized to project the visual features into the dimension space of text embeddings. The projected visual features and text embeddings are subsequently concatenated as input into our PLM backbone. For a fair experimental comparison, we follow the setting of existing PET techniques [22, 21] to adopt a shared-weight PET module for all downstream tasks and use CLIP-ResNet101 [18, 5] as our vision encoder for image-text tasks and CLIP (ViT-B/32) [18, 3] for video-text tasks. Specifically, images are resized to 224×224 at first, and then 6×6 grid features are extracted by the last convolutional layer with an adaptive maximum pooling. Video features are extracted offline at a fixed frame rate (one frame per second) with a maximum length of 64.

We train the models for 20 epochs for both image-text tasks and video-text tasks. The batch size is set as 500 for BART-base on image-text tasks, 300 for T5-base on image-text tasks, and 50 for BART-base on video-text tasks. We use AdamW [16] as our optimizer to train the models and apply a linear decay scheduler with a warmup ratio of 0.1. We evaluate the last checkpoints of models. All experiments are conducted on one A100 GPU (80G memory). The average training time is 16 hours for BART-base on image-text tasks, 18 hours for T5-base on image-text tasks, and 18 hours for BART-base on video-text tasks.

*Corresponding author.

Dataset	Statistics (#images / #QA pairs, #captions)		
	Train	Validation	Test
VQA [4]	113.2K/605.1K	5.0K/26.7K	5.0K/26.3K
GQA [8]	72.1K/943.0K	10.2K/132.1K	398/12.6K
NLVR^2 [20]	103.2K/86.4K	8.1K/7.0K	8.1K/7.0K
MSCOCO [1]	113.2K/566.8K	5.0K/5.0K	5.0K/5.0K

Table 1. The statistics of four image-text datasets.

Dataset	Statistics (#videos / #QA pairs, #captions)		
	Train	Validation	Test
TVQA [10]	17.4K/122.0K	2.2K/15.3K	2.2K/15.3K
How2QA [14]	24.5K/34.2K	3.1K/3.1K	3.1K/3.1K
TVC [11]	17.4K/86.7K	10.8K/43.6K	10.8K/43.6K
YC2C [24]	10.3K/10.3K	3.5K/3.5K	1.6K/1.6K

Table 2. The statistics of four video-text datasets.

Task	Visual Input	Text Input with Task Prompt	Text Output
VQA	image features	vqa: `[Q]`	`[A]`
GQA	image features	gqa: `[Q]`	`[A]`
NLVR	image features	nlvr: `[text]`	true/false
MSCOCO	image features	caption:	`[caption]`
TVQA	video features	tvqa: `[Q]`	`[A]`
How2QA	video features	how2qa: `[Q]`	`[A]`
TVC	video features	tvc:	`[caption]`
YC2C	video features	yc2c:	`[caption]`

Table 3. Input-output formats with task prompts from [2, 22]. `[Q]` denotes question, `[A]` denotes answer, `[text]` denotes text and `[caption]` denotes captioning results.

Method	Learning Rate	Batch Size	Epoch	Other Hyper-Parameters
Full Fine-tuning	1×10^{-4}	500	20	-
BitFit [23]	1×10^{-3}	500	20	-
Prompt Tuning [12]	1×10^{-3}	500	20	prompt length $N_p = 40$, prompt dimension $d_m = 800$
Compacter [9]	1×10^{-3}	500	20	hidden dimension $d = 96$, Kronecker products $k = 2$
Hyperformer [17]	1×10^{-3}	500	20	hidden dimension $d = 96$, task dimension $d_p = 8$
LoRA [7]	1×10^{-3}	500	20	hidden dimension $d = 128$
VL-Adapter [22]	1×10^{-3}	500	20	hidden dimension $d = 96$
VL-PET$_{\text{small}}$	1×10^{-3}	500	20	encoders: $r = 96$, $s = 1.0$, $N_h = 4$; decoders: $r = 96$, $s = 1.0$, $N_h = 1$
VL-PET$_{\text{middleX}}$	1×10^{-3}	500	20	
VL-PET$_{\text{middleY}}$	1×10^{-3}	500	20	
VL-PET$_{\text{large}}$	1×10^{-3}	500	20	

Table 4. Hyper-parameters for image-text tasks with BART-base backbone. We follow the baseline settings of [22].

Method	Learning Rate	Batch size	Epoch	Other hyper-parameters
Full Fine-tuning	3×10^{-4}	300	20	-
BitFit [23]	3×10^{-3}	300	20	-
Prompt Tuning [12]	3×10^{-3}	300	20	prompt length $N_p = 40$, prompt dimension $d_m = 800$
LoRA [7]	3×10^{-4}	300	20	hidden dimension=150
VL-Adapter [22]	3×10^{-4}	300	20	hidden dimension $d = 192$
LST [21]	3×10^{-3}	300	20	r=4; all layers in side networks are kept
VL-PET$_{\text{small}}$	3×10^{-4}	300	20	encoders: $r = 192$, $s = 0.3$, $N_h = 4$; decoders: $r = 96$, $s = 1.0$, $N_h = 1$
VL-PET$_{\text{middleX}}$	3×10^{-4}	300	20	
VL-PET$_{\text{middleY}}$	3×10^{-4}	300	20	
VL-PET$_{\text{large}}$	3×10^{-4}	300	20	

Table 5. Hyper-parameters for image-text tasks with T5-base backbone. We follow the baseline settings of [21].

Method	Learning Rate	Batch size	Epoch	Other hyper-parameters
Full Fine-tuning	1×10^{-5}	50	20	-
BitFit [23]	1×10^{-4}	50	20	-
Prompt Tuning [12]	1×10^{-4}	50	20	prompt length $N_p = 40$, prompt dimension $d_m = 800$
Compacter [9]	1×10^{-4}	50	20	hidden dimension $d = 96$, Kronecker products $k = 2$
LoRA [7]	1×10^{-4}	50	20	hidden dimension $d = 128$
VL-Adapter [22]	1×10^{-4}	50	20	hidden dimension $d = 96$
VL-PET$_{\text{small}}$	7×10^{-4}	50	20	encoders: $r = 96$, $s = 1.0$, $N_h = 4$; decoders: $r = 96$, $s = 1.0$, $N_h = 1$
VL-PET$_{\text{middleX}}$	7×10^{-4}	50	20	
VL-PET$_{\text{middleY}}$	7×10^{-4}	50	20	
VL-PET$_{\text{large}}$	7×10^{-4}	50	20	

Table 6. Hyper-parameters for video-text tasks based on BART-base backbone.

C. Hyper-Parameters

For image-text tasks with the BART-base backbone, [22] conduct a detailed hyper-parameter search to find the optimal settings for the baselines. However, it only reports experimental results of one random seed. To strengthen the reliability of these experiments, we summarize statistics of three seeds with the hyper-parameters listed in Tab. 4.

For image-text tasks with the T5-base backbone, we borrow the results of three seeds from [21]. The hyper-parameters of the baselines are listed in Tab. 5.

For video-text tasks with the BART-base backbone, we show the results of only one seed due to the submission limit of the VALUE benchmark. The hyper-parameters are listed in Tab. 6.

D. Effectiveness of Visual Features and Trainable Visual Projector

In this section, we investigate the effectiveness of the visual features and a trainable visual projector on VL tasks. We denote T as text, Noise as noise features sampled from

Input	Params (%)	VQA (%)	GQA (%)	NLVR^2 (%)	COCO (CIDEr)	Avg.
VL-PET$_{\text{large}}$ (BART-base)						
T	3.10	$44.80_{0.29}$	$40.19_{0.16}$	$51.09_{0.07}$	$6.66_{0.41}$	$35.69_{0.03}$
T + Noise	4.16	$44.79_{0.10}$	$40.49_{0.16}$	$51.09_{0.22}$	$6.57_{0.98}$	$35.74_{0.16}$
T + Frozen V	3.07	$62.62_{0.50}$	$51.34_{0.67}$	$67.77_{0.39}$	$117.58_{1.90}$	$74.83_{0.72}$
T + Trainable V	4.16	$66.17_{0.27}$	$55.11_{0.17}$	$73.43_{0.35}$	$122.03_{0.46}$	$79.18_{0.14}$
VL-PET$_{\text{large}}$ (T5-base)						
T	6.70	$44.90_{0.12}$	$40.62_{0.11}$	$51.21_{0.09}$	$4.81_{0.77}$	$35.39_{0.19}$
T + Noise	7.31	$44.85_{0.18}$	$40.36_{0.02}$	$51.22_{0.09}$	$6.45_{1.81}$	$35.72_{0.42}$
T + Frozen V	6.65	$64.30_{0.20}$	$53.18_{0.55}$	$69.90_{0.68}$	$117.90_{0.37}$	$76.32_{0.44}$
T + Trainable V	7.31	$66.95_{0.21}$	$56.06_{0.21}$	$73.42_{0.46}$	$121.66_{0.06}$	$79.52_{0.21}$

Table 7. Effectiveness of visual features and visual projector. (T: text. Noise: noise features. V: visual projector.)

Method	Params (%)	VQA (%)	GQA (%)	NLVR^2 (%)	COCO (CIDEr)	Avg.
VL-PET$_{\text{large}}$ with $\Delta\mathbf{H}^{\text{vis}}$	3.29	$65.46_{0.15}$	$53.94_{0.27}$	$72.89_{0.32}$	$120.15_{0.68}$	$78.11_{0.13}$
VL-PET$_{\text{large}}$ with scaled up $\Delta\mathbf{H}^{\text{vis}}$	3.47	$65.57_{0.03}$	$53.70_{0.70}$	$73.15_{0.41}$	$120.00_{0.24}$	$78.11_{0.12}$
VL-PET$_{\text{large}}$ with $\Delta\mathbf{H}^{\text{vis}}$ and $\mathbf{G}^{\text{vis}}_{\text{large}}$	3.47	$65.50_{0.13}$	$54.03_{0.44}$	$73.35_{0.18}$	$120.71_{0.43}$	$78.40_{0.03}$
VL-PET$_{\text{large}}$	4.16	$66.17_{0.27}$	$55.11_{0.17}$	$73.43_{0.35}$	$122.03_{0.46}$	$79.18_{0.14}$

Table 8. Decomposing the visual projector on BART-base.

Method	Updating Formula	Params (%)	VQA (%)	GQA (%)	NLVR^2 (%)	COCO (CIDEr)	Avg.
VL-PET w/o granularity control	$\mathbf{H} \leftarrow \mathbf{H} + \Delta\mathbf{H}$	2.97	$65.22_{0.14}$	$53.35_{0.39}$	$72.65_{0.44}$	$120.19_{0.68}$	$77.85_{0.34}$
VL-PET$_{\text{add}}$	$\mathbf{H} \leftarrow \mathbf{H} + \Delta\mathbf{H} + \mathbf{G}_{\text{large}}$	4.16	$65.10_{0.05}$	$53.26_{0.26}$	$71.85_{0.19}$	$121.26_{1.19}$	$77.87_{0.26}$
VL-PET$_{\text{large}}$	$\mathbf{H} \leftarrow \mathbf{G}_{\text{large}} \odot (\mathbf{H} + \Delta\mathbf{H})$	4.16	$66.17_{0.27}$	$55.11_{0.17}$	$73.43_{0.35}$	$122.03_{0.46}$	$79.18_{0.14}$

Table 9. Performance improvement effect of granularity-controlled mechanism.

a uniform distribution on the interval from 0 to 1, Trainable V as visual features with a trainable visual projector and Frozen V as visual features with a frozen visual projector. We replace the standard model input (i.e., T + Trainable V) with T (text-only), T + Noise, and T + Frozen V. In Tab. 7, T + Trainable V outperforms other inputs by a large margin, demonstrating the effectiveness of visual features and the trainable visual projector, as well as the importance of VL alignment and modeling on VL tasks.

In Tab. 8, we also decompose the visual projector using multi-head modular modifications (denoted as $\Delta\mathbf{H}^{\text{vis}}$, r=96, N_h=4) and the granularity control at large level (denoted as $\mathbf{G}^{\text{vis}}_{\text{large}}$). Due to the different dimensions of visual and text features, we have to remove the residual connection for $\Delta\mathbf{H}^{\text{vis}}$. The results indicate that simply scaling up $\Delta\mathbf{H}^{\text{vis}}$ (from r=96 to r=192) dose not effectively improve the performance, whereas introducing $\mathbf{G}^{\text{vis}}_{\text{large}}$ into $\Delta\mathbf{H}^{\text{vis}}$ is more effective. Although decomposing the visual projector provides us more efficiency and effectiveness trade-offs, we do not include it in the main paper for a fair competition with the existing PET techniques (e.g., VL-Adapter).

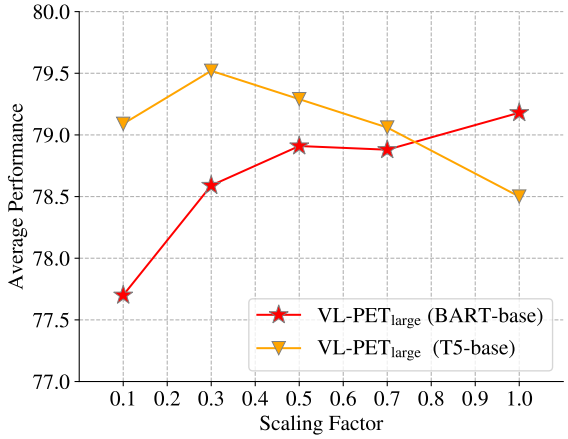

Figure 1. Effectiveness of scaling factor for the encoders. Experiments are conducted on image-text tasks based on different PLM backbones with one seed.

Method	Trainable Params (%)	VQA Acc. (%)	GQA Acc. (%)	NLVR^2 Acc. (%)	COCO Cap. (CIDEr)	Avg.
Backbone: BART-base						
Without task prompts						
VL-PET$_{\text{small}}$	2.98	$64.83_{0.09}$	$54.23_{0.26}$	$72.27_{0.06}$	$121.03_{0.28}$	$78.09_{0.10}$
VL-PET$_{\text{middleX}}$	2.98	$65.14_{0.16}$	$54.55_{0.09}$	$72.77_{0.27}$	$120.91_{0.33}$	$78.34_{0.07}$
VL-PET$_{\text{middleY}}$	2.98	$64.69_{0.14}$	$53.40_{0.35}$	$73.04_{0.16}$	$120.14_{0.58}$	$77.82_{0.12}$
VL-PET$_{\text{large}}$	4.16	$65.78_{0.08}$	$54.45_{0.32}$	$72.90_{0.36}$	$121.46_{0.36}$	$78.65_{0.22}$
With task prompts						
VL-PET$_{\text{small}}$	2.98	$65.43_{0.06}$	$54.03_{0.14}$	$72.43_{0.22}$	$120.68_{0.35}$	$78.14_{0.11}$
VL-PET$_{\text{middleX}}$	2.98	$65.54_{0.09}$	$54.53_{0.15}$	$72.66_{0.17}$	$120.72_{0.51}$	$78.37_{0.14}$
VL-PET$_{\text{middleY}}$	2.98	$65.36_{0.15}$	$53.83_{0.39}$	$73.43_{0.78}$	$120.31_{0.09}$	$78.23_{0.19}$
VL-PET$_{\text{large}}$	4.16	$66.17_{0.27}$	$55.11_{0.17}$	$73.43_{0.35}$	$122.03_{0.46}$	$79.18_{0.14}$
Backbone: T5-base						
Without task prompts						
VL-PET$_{\text{small}}$	4.51	$65.67_{0.31}$	$56.09_{0.29}$	$73.54_{0.63}$	$120.51_{1.11}$	$78.96_{0.57}$
VL-PET$_{\text{middleX}}$	4.50	$65.68_{0.17}$	$56.57_{0.10}$	$74.05_{0.25}$	$119.92_{0.65}$	$79.06_{0.15}$
VL-PET$_{\text{middleY}}$	4.50	$65.75_{0.34}$	$56.35_{0.41}$	$73.93_{0.60}$	$119.84_{1.08}$	$78.97_{0.28}$
VL-PET$_{\text{large}}$	7.31	$66.31_{0.06}$	$56.56_{0.27}$	$73.61_{0.22}$	$121.95_{0.09}$	$79.61_{0.08}$
With task prompts						
VL-PET$_{\text{small}}$	4.51	$65.88_{0.31}$	$54.96_{1.01}$	$72.64_{0.09}$	$120.05_{0.41}$	$78.38_{0.37}$
VL-PET$_{\text{middleX}}$	4.50	$66.63_{0.14}$	$55.87_{0.25}$	$74.11_{0.37}$	$120.41_{0.31}$	$79.26_{0.26}$
VL-PET$_{\text{middleY}}$	4.50	$66.62_{0.20}$	$55.87_{0.13}$	$73.91_{0.45}$	$120.26_{0.40}$	$79.17_{0.08}$
VL-PET$_{\text{large}}$	7.31	$66.95_{0.21}$	$56.06_{0.21}$	$73.42_{0.46}$	$121.66_{0.06}$	$79.52_{0.21}$

Table 10. Effectiveness of task prompts.

E. Performance Improvement Effect of Granularity-controlled Mechanism

Granularity control can dynamically assign importance weights to the intermediate hidden states. To fully investigate where the performance improvement is coming from, we replace the element-wise product with addition for $\mathbf{G}_{\text{large}}$ on BART-base, denoted as VL-PET$_{\text{add}}$. The results (VL-PET$_{\text{large}}$ $>$ VL-PET$_{\text{add}}$) in Tab. 9 verify that the importance assignment is more effective than the added parameters in performance improvement. Moreover, we observe that VL-PET$_{\text{add}}$ introduces more trainable parameters but perform on par with VL-PET without granularity control, as their updating formulas are equivalent to conventional PET (Eq. (1)). These results further demonstrate the effectiveness of the proposed granularity control.

F. Effectiveness of Scaling Factor

As mentioned in Sec. 3.2, the scaling factor is specialized for different PLMs. Our experimental results in Fig. 1 show that the optimal scaling factors for BART-base encoders and T5-base encoders are 1.0 and 0.3, respectively.

G. Unified Text Generation with Task Prompts

Inspired by [22, 21], we prepend a task-specific prompt to the input sentence for each downstream task. The detailed task prompts are listed in Tab. 3. We conduct experiments to test the validity of these task prompts over three seeds in Tab. 10. For the BART-base backbone, all VL-PET modules with task prompts outperform those without task prompts. However, for T5-base, some VL-PET modules without task prompts exhibit superior performance compared to those with task prompts. For a fair comparison with state-of-the-art PET techniques, we still adopt task prompts to facilitate multi-task learning via unified text generation for our generative PLM backbones.

H. Scalability of VL-PET Modules

We scale up the trainable parameters of the VL-PET modules by increasing the projected hidden dimensions r of the Encoder VL-PET modules in BART-base. All scaled

Method	Params (%)	VQA (%)	GQA (%)	NLVR^2 (%)	COCO (CIDEr)	Avg.
VL-PET$_{\text{small}}$ ($r = 96$)	2.98	$65.43_{0.06}$	$54.03_{0.14}$	$72.43_{0.22}$	$120.68_{0.35}$	$78.14_{0.11}$
$r = 144$	3.58	$65.87_{0.12}$	$54.05_{0.21}$	$72.76_{0.09}$	$121.22_{0.40}$	$78.48_{0.07}$
$r = 192$	4.16	$66.14_{0.16}$	$54.73_{0.18}$	$72.63_{0.23}$	$121.46_{1.38}$	$78.74_{0.34}$
VL-PET$_{\text{middleX}}$ ($r = 96$)	2.98	$65.54_{0.09}$	$54.53_{0.15}$	$72.66_{0.17}$	$120.72_{0.51}$	$78.37_{0.14}$
$r = 144$	3.57	$66.08_{0.10}$	$54.82_{0.29}$	$72.82_{0.23}$	$121.05_{0.11}$	$78.70_{0.11}$
$r = 192$	4.16	$66.23_{0.06}$	$54.55_{0.33}$	$73.57_{0.34}$	$122.01_{0.65}$	$79.09_{0.23}$
VL-PET$_{\text{middleY}}$ ($r = 96$)	2.98	$65.36_{0.15}$	$53.83_{0.39}$	$73.43_{0.78}$	$120.31_{0.09}$	$78.23_{0.19}$
$r = 144$	3.57	$65.58_{0.18}$	$54.06_{0.32}$	$73.02_{0.52}$	$120.59_{0.26}$	$78.31_{0.30}$
$r = 192$	4.16	$65.94_{0.06}$	$54.29_{0.40}$	$73.53_{0.25}$	$120.89_{0.30}$	$78.66_{0.18}$
VL-PET$_{\text{large}}$ ($r = 96$)	4.16	$66.17_{0.27}$	$55.11_{0.17}$	$73.43_{0.35}$	$122.03_{0.46}$	$79.18_{0.14}$
$r = 144$	5.31	$66.47_{0.10}$	$55.07_{0.25}$	$73.21_{0.15}$	$122.05_{0.58}$	$79.20_{0.13}$
$r = 192$	6.43	$66.72_{0.06}$	$54.61_{0.31}$	$73.55_{0.88}$	$122.16_{0.18}$	$79.26_{0.12}$

Table 11. Scalability of VL-PET Modules. (r: the projected hidden dimension of the encoder VL-PET modules.)

Method	Trainable Params (%)	VQA Acc. (%)	GQA Acc. (%)	NLVR^2 Acc. (%)	COCO Cap. (CIDEr)	Avg.
Backbone: BART-base						
Random Gaussian Initialization (Default)						
VL-PET$_{\text{small}}$	2.98	$65.43_{0.06}$	$54.03_{0.14}$	$72.43_{0.22}$	$120.68_{0.35}$	$78.14_{0.11}$
VL-PET$_{\text{middleX}}$	2.98	$65.54_{0.09}$	$54.53_{0.15}$	$72.66_{0.17}$	$120.72_{0.51}$	$78.37_{0.14}$
VL-PET$_{\text{middleY}}$	2.98	$65.36_{0.15}$	$53.83_{0.39}$	$73.43_{0.78}$	$120.31_{0.09}$	$78.23_{0.19}$
VL-PET$_{\text{large}}$	4.16	$66.17_{0.27}$	$55.11_{0.17}$	$73.43_{0.35}$	$122.03_{0.46}$	$79.18_{0.14}$
Zero Initialization						
VL-PET$_{\text{small}}$	2.98	$65.33_{0.21}$	$53.96_{0.15}$	$72.48_{0.10}$	$121.05_{0.28}$	$78.21_{0.13}$
VL-PET$_{\text{middleX}}$	2.98	$65.49_{0.06}$	$54.28_{0.20}$	$73.01_{0.47}$	$120.36_{0.26}$	$78.29_{0.15}$
VL-PET$_{\text{middleY}}$	2.98	$65.21_{0.20}$	$53.86_{0.53}$	$72.84_{1.21}$	$120.38_{0.65}$	$78.07_{0.30}$
VL-PET$_{\text{large}}$	4.16	$66.18_{0.13}$	$54.44_{0.58}$	$72.99_{0.69}$	$121.91_{0.52}$	$78.88_{0.35}$
Backbone: T5-base						
Random Gaussian Initialization						
VL-PET$_{\text{small}}$	4.51	$66.80_{0.22}$	$56.15_{0.22}$	$74.25_{0.59}$	$120.57_{0.81}$	$79.44_{0.45}$
VL-PET$_{\text{middleX}}$	4.50	$66.33_{0.47}$	$55.89_{0.05}$	$74.23_{0.21}$	$119.79_{0.59}$	$79.06_{0.31}$
VL-PET$_{\text{middleY}}$	4.50	$66.52_{0.12}$	$55.52_{0.65}$	$73.96_{0.45}$	$120.99_{0.21}$	$79.25_{0.05}$
VL-PET$_{\text{large}}$	7.31	$66.78_{0.10}$	$55.62_{0.18}$	$73.21_{0.15}$	$121.12_{0.36}$	$79.19_{0.08}$
Zero Initialization (Default)						
VL-PET$_{\text{small}}$	4.51	$65.88_{0.31}$	$54.96_{1.01}$	$72.64_{0.09}$	$120.05_{0.41}$	$78.38_{0.37}$
VL-PET$_{\text{middleX}}$	4.50	$66.63_{0.14}$	$55.87_{0.25}$	$74.11_{0.37}$	$120.41_{0.31}$	$79.26_{0.26}$
VL-PET$_{\text{middleY}}$	4.50	$66.62_{0.20}$	$55.87_{0.13}$	$73.91_{0.45}$	$120.26_{0.40}$	$79.17_{0.08}$
VL-PET$_{\text{large}}$	7.31	$66.95_{0.21}$	$56.06_{0.21}$	$73.42_{0.46}$	$121.66_{0.06}$	$79.52_{0.21}$

Table 12. Experimental results of different weight initialization strategies.

VL-PET modules outperform the unscaled ones in Tab. 11, indicating the scalability of our VL-PET framework.

I. Weight Initialization

In Tab. 12, we show the experimental results over three seeds with two popular weight initialization strategies, i.e., random Gaussian initialization and zero initialization, for both BART-base and T5-base backbones. Random Gaussian initialization is widely used in PET techniques to initialize the weight of PET modules from a Gaussian distribution, while some PET techniques [7] utilize zero initialization to set the up projection layers of the PET modules as zero. Based on Tab. 12, it can be inferred that random Gaussian initialization is more appropriate for BART-base and zero initialization for T5-base. Therefore, we employ

Method	Trainable Params (%)	VQA Acc. (%)	GQA Acc. (%)	NLVR^2 Acc. (%)	COCO Cap. (CIDEr)	Avg.
Backbone: BART-base **Granularity-controlled Mechanism:$\mathbf{G}_{\text{large}}$**						
Down Multi-head Modular Modifications (used in the main paper)						
$N_h = 2$	4.16	$66.20_{0.12}$	$54.74_{0.43}$	$72.88_{0.43}$	$121.53_{0.64}$	$78.84_{0.26}$
$N_h = 4$	4.16	$66.17_{0.27}$	$55.11_{0.17}$	$73.43_{0.35}$	$122.03_{0.46}$	$79.18_{0.14}$
Up Multi-head Modular Modifications						
$N_h = 2$	4.16	$66.16_{0.12}$	$54.88_{0.21}$	$73.19_{0.13}$	$121.69_{0.62}$	$78.98_{0.14}$
$N_h = 4$	4.16	$66.15_{0.13}$	$55.00_{0.36}$	$73.19_{0.27}$	$121.44_{0.73}$	$78.95_{0.23}$
Down-Up Multi-head Modular Modifications						
$N_h = 2$	4.16	$66.24_{0.10}$	$54.73_{0.24}$	$72.81_{0.21}$	$121.56_{0.15}$	$78.84_{0.07}$
$N_h = 4$	4.16	$66.18_{0.16}$	$54.60_{0.27}$	$73.17_{0.42}$	$122.05_{0.20}$	$79.00_{0.10}$
Down-Up-Pair Multi-head Modular Modifications						
$N_h = 2$	3.86	$65.86_{0.04}$	$54.51_{0.36}$	$72.98_{0.19}$	$121.65_{0.21}$	$78.75_{0.07}$
$N_h = 4$	3.72	$65.70_{0.09}$	$54.24_{0.12}$	$72.43_{0.35}$	$122.76_{0.33}$	$78.28_{0.20}$

Table 13. Experimental results of multi-head modular modifications with various numbers of heads.

random Gaussian initialization for BART-base and zero initialization for T5-base by default in this work.

J. More Designs for Multi-head Modular Modification

We propose a multi-head modular modification $\Delta\mathbf{H}' \in \mathbb{R}^{N\times d}$ of length N and dimension d in the main paper, which is formulated as follows:

$$\Delta\mathbf{H}' = \phi(\text{Concat}(\mathbf{X}'\mathbf{W}_{\text{down}}^{(1)}, \cdots, \mathbf{X}'\mathbf{W}_{\text{down}}^{(N_h)}))\mathbf{W}_{\text{up}}, \quad (1)$$

where N_h is the number of heads, $\mathbf{X}' \in \mathbb{R}^{N\times d}$ is the input, $\mathbf{W}_{\text{down}}^{(i)} \in \mathbb{R}^{d\times \frac{r}{N_h}}$ is a down projection layer for the i-th head, ϕ is the GELU function [6], $\mathbf{W}_{\text{up}} \in \mathbb{R}^{r\times d}$ is a up projection layer and r is the projected hidden dimension.

Since the multi-head idea is applied to the down projection layer in the main paper, we can call this type of multi-head modular modification as down multi-head modular modification. As the multi-head idea can be applied to any linear projection within a modular modification, it yields a series of variant designs. In this section, we present additional designs of multi-head modular modification, such as up multi-head modular modification, down-up multi-head modular modification and down-up-pair multi-head modular modification. The primary difference among these four designs lies in where the multi-head idea is employed.

For up multi-head modular modification, we apply the multi-head idea to the up projection layer and describe it as follows:

$$\begin{aligned}\Delta\mathbf{H}' = \text{Concat}(&\phi((\mathbf{X}'\mathbf{W}_{\text{down}}))\mathbf{W}_{\text{up}}^{(1)}, \cdots, \\ &\phi((\mathbf{X}'\mathbf{W}_{\text{down}}))\mathbf{W}_{\text{up}}^{(N_h)}),\end{aligned} \quad (2)$$

where $\mathbf{W}_{\text{down}} \in \mathbb{R}^{d\times r}$ is a down projection layer and $\mathbf{W}_{\text{up}}^{(i)} \in \mathbb{R}^{r\times \frac{d}{N_h}}$ is a down projection layer for head$_i$.

For down-up multi-head modular modification, we apply the multi-head idea to the down projection layer and up projection layer, respectively. The formulation is described as follows:

$$\begin{aligned}\Delta\mathbf{H}'_{\text{down}} &= \phi(\text{Concat}(\mathbf{X}'\mathbf{W}_{\text{down}}^{(1)}, \cdots, \mathbf{X}'\mathbf{W}_{\text{down}}^{(N_h)})), \\ \Delta\mathbf{H}' &= \text{Concat}(\Delta\mathbf{H}'_{\text{down}}\mathbf{W}_{\text{up}}^{(1)}, \cdots, \Delta\mathbf{H}'_{\text{down}}\mathbf{W}_{\text{up}}^{(N_h)}),\end{aligned} \quad (3)$$

where $\Delta\mathbf{H}'_{\text{down}} \in \mathbb{R}^{N\times r}$ is an intermediate output.

For down-up-pair multi-head modular modification, we formulate it as follows:

$$\begin{aligned}\Delta\mathbf{H}'^{(i)}_{\text{pair}} &= \phi(\mathbf{X}'\mathbf{W}_{\text{down-pair}}^{(i)})\mathbf{W}_{\text{up-pair}}^{(i)}, \\ \Delta\mathbf{H}' &= \text{Concat}(\Delta\mathbf{H}'^{(1)}_{\text{pair}}, \cdots, \Delta\mathbf{H}'^{(N_h)}_{\text{pair}}),\end{aligned} \quad (4)$$

where $\mathbf{W}_{\text{down-pair}}^{(i)} \in \mathbb{R}^{d\times \frac{r}{N_h}}$ is a down projection layer, $\mathbf{W}_{\text{up-pair}}^{(i)} \in \mathbb{R}^{\frac{r}{N_h}\times \frac{d}{N_h}}$ is a up projection layer and $\Delta\mathbf{H}'^{(i)}_{\text{pair}} \in \mathbb{R}^{N\times \frac{d}{N_h}}$ is an intermediate output for head$_i$.

Similar to the usage of multi-head modular modifications in the main paper, we perform experiments on image-text tasks with BART-base backbone for these four multi-head modular modifications with our proposed granularity-controlled mechanism at a large level.

As shown in Tab. 13, the down multi-head modular modification with $N_h = 4$ achieves the best performance in the experiment, which is adopted in the main paper. We observe that even the worst down-up-pair multi-head modular modification still outperforms the state-of-the-art PET techniques (e.g., VL-Adapter). Moreover, down-up-pair multi-head modular modification significantly reduces the number of trainable parameters as the number of heads increases, indicating a promising direction for further parameter reduction with a minimal impact on performance.

K. Limitations

In this section, we discuss some limitations of our work.

- We perform extensive experiments and analysis on image-text tasks and video-text tasks. However, the video-text experiments are conducted with only one seed due to the submission limit of the VALUE benchmark, which may affect the reliability of the video-text experimental results.

- Our VL-PET framework focuses on challenging VL downstream tasks, including some discriminative and generative tasks (e.g., question-answering tasks and captioning tasks). But there are many other downstream tasks in the real life, and we do not include all types of tasks in the multi-task learning experiments. Therefore, our designs and experimental results are not always guaranteed to be generalized to other VL tasks (e.g., image-text retrieval and video-text retrieval) and other domains (e.g., NLP and CV).

- We validate the enhanced effect of employing our VL-PET designs (e.g., granularity-controlled mechanism and lightweight PET module designs) on existing PET techniques (i.e., Compacter and VL-Adapter). But our VL-PET designs may not be applicable to enhance all PET techniques.

- PET techniques are proposed to reduce the number of trainable parameters and save model storage space. Our work utilizes multi-task learning to achieve further reduction. However, similar to most PET techniques, our work integrates newly-designed modules into large PLM backbones, which still evoke a lot of memory and time consumption during training and inference.

References

[1] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO captions: Data collection and evaluation server. *CoRR*, abs/1504.00325, 2015.

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

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[4] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *CVPR*, 2017.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[6] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[7] 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.

[8] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *CVPR*, 2019.

[9] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. In *NeurIPS*, 2021.

[10] Jie Lei, Licheng Yu, Mohit Bansal, and Tamara L. Berg. Tvqa: Localized, compositional video question answering. In *EMNLP*, 2018.

[11] Jie Lei, Licheng Yu, Tamara L. Berg, and Mohit Bansal. Tvr: A large-scale dataset for video-subtitle moment retrieval. In *ECCV*, 2020.

[12] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, 2021.

[13] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, 2020.

[14] Linjie Li, Yen-Chun Chen, Yu Cheng, Zhe Gan, Licheng Yu, and Jingjing Liu. Hero: Hierarchical encoder for video+language omni-representation pre-training. In *EMNLP*, 2020.

[15] Linjie Li, Jie Lei, Zhe Gan, Licheng Yu, Yen-Chun Chen, Rohit Pillai, Yu Cheng, Luowei Zhou, Xin Eric Wang, William Yang Wang, et al. Value: A multi-task benchmark for video-and-language understanding evaluation. In *NeurIPS*, 2021.

[16] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2017.

[17] Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *ACL*, 2021.

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

[19] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[20] Alane Suhr, Stephanie Zhou, Ally Zhang, Iris Zhang, Huajun Bai, and Yoav Artzi. A corpus for reasoning about natural language grounded in photographs. In *ACL*, 2019.

[21] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. LST: ladder side-tuning for parameter and memory efficient transfer learning. *CoRR*, abs/2206.06522, 2022.

[22] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *CVPR*, 2022.

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

[24] Luowei Zhou, Chenliang Xu, and Jason J. Corso. Towards automatic learning of procedures from web instructional videos. In *AAAI*, 2018.