Supplementary Material

A. Statistics of Datasets

Table 1 shows the statistical details of datasets used for the downstream VLN tasks. #Path represents the number of paths, #Instr represents the number of instructions, #Words represents the number of words, and Instr Length denotes the average lengths of instructions.

Dataset	#Path	#Instr	#Words	Instr Length
R2R [1]	7K	21.7K	625K	29
REVERIE [8]	7K	21.7K	388K	18
CVDN [10]	7K	2.1K†	167K	-
RxR [6]	16.5K	126.1K	9.8M	78

†The number of dialogues.

Table 1: Statistics of datasets for the four VLN tasks.

B. Comparison of Memory Consumption

We conduct experiments on the R2R val set to evaluate the efficiency on memory consumption of different PETL methods. We also report the results of Head-Tuning that only tunes the prediction head and Ladder Side Tuning (LST) [9] which is claimed to effectively reduce the memory consumption. We find that though some PETL methods (BitFit, Prompt Tuning, LoRA, Adapter and VLN-PETL) tune different amounts of parameters, their memory consumption is similar (about 70% compared to fine-tuning) due to the full back-propagation from outputs to inputs, in correspondence with the findings of [9]). Compared to LoRA and Adapter, our VLN-PETL improves the performance by a large margin without increasing much more memory consumption, demonstrating the effectiveness of our proposed VLN-PETL.

We also find that LST, which reduces about 65% memory, has a better performance than Head-Tuning and Prompt-Tuning while worse than BitFit. We infer that the larger pre-trained model in LST works more like a feature extractor, where the output feature of each transformer layer is fixed and used as the input of the side blocks, which could effectively reduce the memory but hurt the performance by failing in adapting the dynamic knowledge flow through the frozen pre-trained model.

Thus, how to further reduce memory consumption while keeping competitive performance will be another focus of our future work.

Methods	Memory (GB)	Updated Params(%)	Validation Seen SR ↑	Validation Seen SPL ↑	Validation Unseen SR ↑	Validation Unseen SPL ↑
Fine-Tuning	17.4	100	72.67	69.17	64.24	59.25
Head-Tuning	2.0	0.35	61.21	57.97	56.02	52.00
LST [9]	6.1	2.25	63.17	59.48	57.85	53.00
BitFit [2]	12.0	0.46	63.47	60.35	59.17	54.67
Prompt-Tuning [7]	12.3	0.37	61.02	58.59	56.49	52.32
LoRA [5]	12.7	3.02	70.13	66.00	63.60	57.59
Adapter [4]	12.7	3.08	67.38	64.42	63.01	57.42
VLN-PETL(ours)	13.2	2.82	72.28	68.50	65.47	60.01

Table 2: Comparison of memory consumption on R2R Val set.

C. Details of Language Encoder Adapter

For brevity, we omit the details of Language Encoder Adapter (LEA) in the main paper. Now, we will give a detailed description as follows.

Concretely, for the l-th transformer block in the language encoder, the input feature $\boldsymbol{f}_x^{l-1}$ is first passed through an adapter with D_{mid} bottleneck dimension to generate a new feature $\boldsymbol{f}_{\text{ad\_att}}$. Then, $\boldsymbol{f}_{\text{ad\_att}}$ is summed with the original output feature $\hat{\boldsymbol{f}}_{\text{att}}$ of the multi-head self-attention layer in the transformer block as:

$$\boldsymbol{f}_{\text{ad\_att}} = \boldsymbol{W}_{\text{up\_att}}^{\intercal}\text{ReLU}(\boldsymbol{W}_{\text{down\_att}}^{\intercal}\boldsymbol{f}_x^{l-1}), \quad (1)$$

$$\hat{\boldsymbol{f}}_{\text{att}} = \text{MSA}(\boldsymbol{f}_x^{l-1}), \quad (2)$$

$$\boldsymbol{f}_{\text{att}} = \text{LN}(\hat{\boldsymbol{f}}_{\text{att}} + \boldsymbol{f}_{\text{ad\_att}}), \quad (3)$$

where $\boldsymbol{W}_{\text{up\_att}}$ and $\boldsymbol{W}_{\text{down\_att}}$ represent projection matrices of the adapter, $\text{MSA}(\cdot)$ represents multi-head self-attention layer and $\text{LN}(\cdot)$ represents layer normalization. Note that we omit the feed-forward layer following multi-head self-attention layer for brevity.

Similarly, another adapter is inserted into the feed-forward layer which takes $\boldsymbol{f}_{\text{att}}$ as input:

$$\boldsymbol{f}_{\text{ad\_ffn}} = \boldsymbol{W}_{\text{up\_ffn}}^{\intercal}\text{ReLU}(\boldsymbol{W}_{\text{down\_ffn}}^{\intercal}\boldsymbol{f}_{\text{att}}), \quad (4)$$

$$\hat{\boldsymbol{f}}_{\text{ffn}} = \text{FFN}(\boldsymbol{f}_{\text{att}}), \quad (5)$$

$$\boldsymbol{f}_{\text{ffn}} = \text{LN}(\hat{\boldsymbol{f}}_{\text{ffn}} + \boldsymbol{f}_{\text{ad\_ffn}}), \quad (6)$$

where $\text{FFN}(\cdot)$ represents feed-forward layer. Finally, $\boldsymbol{f}_{\text{ffn}}$ is used as the output feature $\boldsymbol{f}_x^l$ of the l-th transformer block:

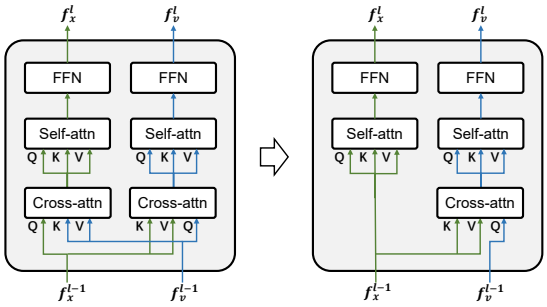


Figure 1: Variant of VLN baseline for RxR.

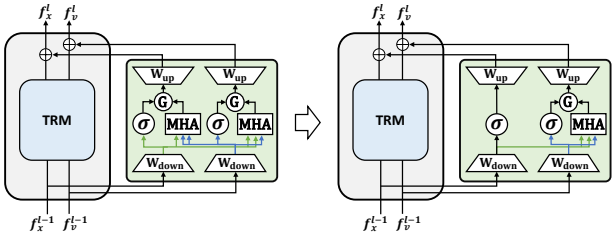


Figure 2: Variant of Cross-modal Interaction Booster for RxR.

$$\boldsymbol{f}_x^l = \boldsymbol{f}_{\text{ffn}}. \tag{7}$$

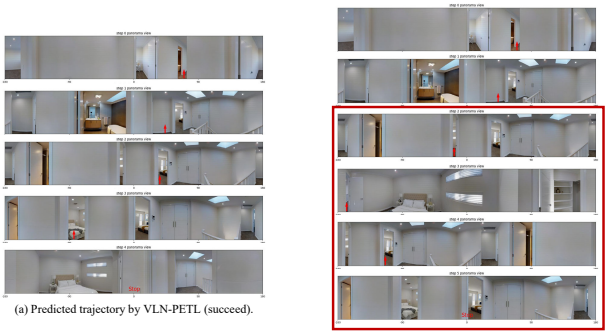
D. Variant of VLN-PETL for RxR

To improve the efficiency of HAMT for VLN tasks with much longer instructions such as RxR, [3] adopts a structure variant for cross-modal encoder as shown in Figure 1. We follow this practice for our VLN baseline in the RxR task and correspondingly our proposed cross-modal Interaction Booster (CIB) also has a modification for the language sub-branch as shown in Figure 2. Specifically, for the language input feature $\boldsymbol{f}_x^{l-1}$, only a bottleneck layer with an activation layer works as a block-level adapter, excluding the multi-head cross-attention mechanism and gating mechanism. The forward pass of this variant can be formulated as follows:

$$\boldsymbol{f}_x^l = \text{LN}\big(\text{TRM}(\boldsymbol{f}_x^{l-1}) + \text{ADAPTER}(\boldsymbol{f}_x^{l-1})\big). \tag{8}$$

E. Qualitative Examples

The visualized trajectories depicted in Figure 3 show that upon removing the CIB module, the agent failed to reach the target location.



(a) Predicted trajectory by VLN-PETL (succeed).

(b) Predicted trajectory by VLN-PETL w/o CIB (failed).

Figure 3: Comparisons of predicted trajectory with VLN-PETL and VLN-PETL w/o CIB module. The navigation steps inside the red box are incorrect. Instruction:" Turn right to head down the hallway. Go to the end of the hallway and turn into the last bedroom on the left. Stop once in the door. "

References

[1] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian D. Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, pages 3674–3683, 2018. 1

[2] Elad Ben-Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *ArXiv*, abs/2106.10199, 2022. 1

[3] Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev. History aware multimodal transformer for vision-and-language navigation. In *NeurIPS*, pages 5834–5847, 2021. 2

[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. 1

[5] Edward Hu, Yelong Shen, Phil Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022. 1

[6] Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *EMNLP*, pages 4392–4412, 2020. 1

[7] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, pages 3045–3059, 2021. 1

[8] Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. REVERIE: remote embodied visual referring expression in real indoor environments. In *CVPR*, pages 9979–9988, 2020. 1

[9] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. LST: ladder side-tuning for parameter and memory efficient transfer learning. *CoRR*, abs/2206.06522, 2022. 1

[10] Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. Vision-and-dialog navigation. In *CoRL*, pages 394–406, 2019. 1