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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Path</th>
<th>#Instr</th>
<th>#Words</th>
<th>Instr Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>R2R [1]</td>
<td>7K</td>
<td>21.7K</td>
<td>625K</td>
<td>29</td>
</tr>
<tr>
<td>REVERIE [8]</td>
<td>7K</td>
<td>21.7K</td>
<td>388K</td>
<td>18</td>
</tr>
<tr>
<td>CVDN [10]</td>
<td>7K</td>
<td>2.1K</td>
<td>167K</td>
<td>-</td>
</tr>
<tr>
<td>RxR [6]</td>
<td>16.5K</td>
<td>126.1K</td>
<td>9.8M</td>
<td>78</td>
</tr>
</tbody>
</table>

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.

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 $f^{l-1}_x$ is first passed through an adapter with $D_{mid}$ bottleneck dimension to generate a new feature $f_{ad, att}$. Then, $f_{ad, att}$ is summed with the original output feature $f_{att}$ of the multi-head self-attention layer in the transformer block as:

$$f_{ad, att} = W^\top_{up, att} \text{ReLU}(W^\top_{down, att} f^{l-1}_x), \quad (1)$$

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

$$f_{att} = \text{LN}(f_{att} + f_{ad, att}), \quad (3)$$

where $W_{up, att}$ and $W_{down, att}$ represent projection matrices of the adapter, MSA($\cdot$) represents multi-head self-attention layer and 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 $f_{att}$ as input:

$$f_{ad, ffn} = W^\top_{up, ffn} \text{ReLU}(W^\top_{down, ffn} f_{att}), \quad (4)$$

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

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

where FFN($\cdot$) represents feed-forward layer. Finally, $f_{ffn}$ is used as the output feature $f^l_x$ of the l-th transformer block.
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 $f_{l}^{i-1}$, 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:

$$f_{x}^{i} = f_{\text{fnn}}.$$  \hspace{1cm} (7)

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.

References


