Supplementary for Paper: Improving Vision-and-Language Navigation by Generating Future-View Image Semantics

Jialu Li Mohit Bansal
UNC Chapel Hill
{jialuli, mbansal}@cs.unc.edu

1. Overview

In this supplementary, we provide the following:

- Detailed description of the architecture of the baseline method we use in Sec. 2.
- Detailed description of the pre-training tasks used in the baseline method in Sec. 3, and more implementation details in Sec. 4.
- Performance of our method on R4R and RxR validation unseen set in Sec. 5.
- Proof that demonstrates weighted patch probability could learn a better optimal value than mean patch probability in Sec. 6.
- More examples of the future views generated by our APIG head in Sec. 7, and quantitative analysis of the semantics underlying our generated tokens in Sec. 8.

2. HAMT Model Architecture

HAMT [1] utilizes a transformer-based architecture to encode the instructions, navigation history, and current step observation. Specifically, the instructions are encoded with a BERT architecture.

As the navigation history is a sequence of panorama observations, HAMT encodes the navigation history with a hierarchical architecture. It first uses a panorama encoder to encode panorama into view representation, and uses multiple transformer layers as the temporal encoder to encode the observations on the trajectory. Formally, given the encoded history observation v_i, which is the output of the panorama encoder, the output of the temporal encoder is $h_i = LN(W_t v_i) + LN(W_a a_i) + E_i^S + E_2^T$, where a_i is the action embedding at step i, E_i^S is the step encoding, and E_2^T is the token type encoding which indicates the input is history views.

The current step observation is represented as 36 discretized views. Each view is passed through the transformer encoder to learn the view representation: $o_i = LN(W_o v_i^o) + LN(W_a^o a_i^o) + E_i^O + E_1^T$, where v_i^o is the encoding of view i, a_i^o is the action embedding for view i, E_i^O is the embedding indicating whether the current view is navigable, and E_1^T is the token type encoding which indicates the input is current step observation.

The agent predicts the next step by comparing the similarity between the observation encoding o_i and the <CLS> token which contains instruction-trajectory information.

More implementation details can be found in [1].

3. Pre-training Tasks in HAMT

In this section, we describe the six pre-training tasks we adopted from [1]. Specifically, the six tasks are Masked Language Modeling (MLM), Masked Region Modeling (MRM), Instruction Trajectory Matching (ITM), Single-step Action Prediction/Regression (SAP/SAR), and Spatial Relationship Prediction (SPREL).

In Masked Language Modeling, we randomly masked out 15% of the words in the instructions, and predict the masked words given surrounding words and the full trajectory, which improves agents' language understanding. We optimize the negative log-likelihood of the original words: $L_{MLM} = -logp(w_i|I_{lang\setminus i}, I_{visual})$, where $I_{lang\setminus i}$ is the language input without masked words w_i and I_{visual} is the visual input.

In Masked Region Modeling, the agent learns to predict the objects in the masked views in the trajectory. The target is the object detection probability p_i predicted by an image classification model pre-trained on ImageNet. We optimize a KL-divergence between the target probability and predicted probability: $L_{MRM} = -p_i log\hat{p}_i$, where $\hat{p}_i$ is the predicted probability.

In Instruction Trajectory Matching, the agent learns the alignment between the language instructions and the environment by picking the correct instruction-trajectory pairs from one positive pair and four negative pairs. The four negative pairs are created by randomly sampling two trajectories from the same batch, and shuffling the order of views in the correct trajectory.

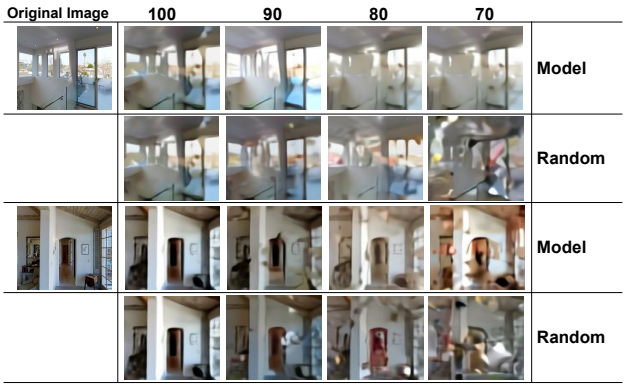

Figure 1. Qualitative analysis of images generated with our agent. Given 90%/ 80%/ 70% ground truth tokens, the rest tokens predicted by our model contain closer semantic information compared with randomly filled tokens.

The agent optimizes a noisy contrastive loss: $L_{ITM} = -log\frac{exp(f_{ITM}(h_{lang} * h_{visual}))}{exp(f_{ITM}(h_{lang} * h_{visual})) + \sum^{4} exp(f_{ITM}(h_{lang} * h^{neg}_{visual}))}$, where h_{lang} and h_{visual} are the outputs of the <CLS> token of the instructions and the trajectories separately.

In Single-step Action Prediction and Single-step Action Regression, the agent needs to select the next step from a set of candidates. Specifically, the agent optimizes a negative log probability of the target view action in Single-step Action Prediction, and predicts the heading and elevation of the target view action by optimizing the L2 loss.

In Spatial Relationship Prediction, the agent learns to predict the relative spatial position of two views in a panorama. Specifically, it optimizes a L2 loss between the predicted heading and elevation difference and ground truth heading and elevation difference between two views.

More implementation details can be found in [1].

4. Implementation Details

We adopt the model architecture from [1]. For the image tokenizer, the input image size is 224, and the patch size is 16. We set 1.0 for γ and 0.5 for λ in dynamic codebook selection, and $|S|$ to be 1000 for both codebook selection methods. In pre-training, the ratio to select tasks is set to be 3 for MTM and 1 for others. The mask ratio r is 0.5 for MTM and u is 0.3 for MPM. In fine-tuning, L_{AT} is added to the IL loss with ratio 1. The ratio to combine IL and RL is 0.15 when adding L_{AT} and 0.2 otherwise. We use weighted patch probability with block-wise sampling for Room-to-Room dataset and base weighted patch probability for CVDN dataset. Other hyperparameters are the same as in [1] for fair comparison. For training time, our model takes 30 hours to converge on 2 NIVIDIA A6000 GPU, while HAMT full model takes 20 hours training on 20 NVIDIA V100 GPUs in addition to initial pre-training for 1 day on 4 NVIDIA Tesla. For model parameters updated during pre-training, our model updates 191M parameters while HAMT full model updates 260M, saving 27% parameters.

Model	R4R				RxR		
	Val Unseen				Val Unseen		
	SR↑	nDTW↑	sDTW↑	CLS↑	SR↑	nDTW↑	sDTW↑
HAMT [1]	44.6	50.3	31.8	57.7	56.5	63.1	48.3
Ours	**45.8**	**52.9**	**33.6**	**59.1**	**60.0**	**65.3**	**51.4**

Table 1. Comparison with state-of-the-art agents on R4R and RxR validation unseen set.

5. Performance on R4R and RxR dataset

In this section, we show our agents' performance on R4R and RxR dataset. As shown in Table 1, our model surpasses the HAMT full model by 2.6% in nDTW and 1.8% in sDTW on R4R validation unseen set. Besides, we show that our model achieves significantly better performance on RxR unseen set, improving the HAMT method by 2.1% in nDTW and 3.1% in sDTW.

6. Weighted Patch Probability Proof

In weighted patch probability calculation, we represent image semantics generation probability $p^{o_w}_{ik}$ as:

$$p^{o_w}_{ik} = \sum_{j=1}^{N} w_j p_{ijk} \tag{1}$$

where w_j is a randomly sampled weight for patch j, and $\sum_{j=1}^{N} w_j = 1$.

During training, we randomly sample $\{w_j\}_{j=1}^{N}$ for each example, and minimize the differences between $\sum_{j=1}^{N} w_j p_{ijk}$ and predicted $\hat{p}^{o_w}_{ik}$. We hypothesize that the agent learns to predict $\hat{p}^{o_w}_{ik}$ by predicting $\sum_{j=1}^{N} w_j \hat{p}_{ijk}$ as we conditioned the prediction based on sampled weights $\{w_j\}_{j=1}^{N}$.

We want to reach the optimal where:

$$p^{o_w}_{ik} = \hat{p}^{o_w}_{ik} \tag{2}$$

$$\sum_{j=1}^{N} w_j p_{ijk} = \sum_{j=1}^{N} w_j \hat{p}_{ijk} \tag{3}$$

Since we randomly sample $\{w_j\}_{j=1}^{N}$ during training, this guarantees that for every patch j in the image v_i, $p_{ijk} = \hat{p}_{ijk}$, otherwise $\exists \{w_j\}_{j=1}^{N}$ that makes $\sum_{j=1}^{N} w_j p_{ijk} \neq \sum_{j=1}^{N} w_j \hat{p}_{ijk}$.

7. Future View Generation Examples

We demonstrate that our model could reasonably generate future semantics and reconstruct future images with more examples. As shown in Figure 1, our generated image could almost reconstruct the doors and the overall layout of the room when given 70% of the ground truth tokens. In comparison, filling the 30% patches with random tokens will produce distorted images which are hard to infer how does the original images look like.

8. Analysis of Generated Semantics

In this section, we compare the generated semantics with the ground truth semantics quantitatively to demonstrate that the semantic information underlying them is similar. Specifically, we represent the semantics of each visual token as the output of the first embedding layer in the dVAE decoder (which maps each token to a 128 dimension representation space). We calculate the distance between generated semantics and ground truth semantics, and compare it with the distance between the ground truth semantics and all other tokens in the vocabulary (i.e., the distance between the ground truth token and other 8191 tokens for each patch). We normalize each semantic representation and use l2-norm as the distance. Our method has a distance of 0.95, while the baseline is 1.31. This shows that the distance between our generated semantics and ground truth semantics is closer.

References

[1] Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev. History aware multimodal transformer for vision-and-language navigation. *Advances in Neural Information Processing Systems*, 34, 2021. 1, 2