Bayesian Relational Memory for Semantic Visual Navigation Supplementary Materials

Yi Wu[†¶] Yuxin Wu[‡] Aviv Tamar[§] Stuart Russell[†] Georgia Gkioxari[‡] Yuandong Tian[‡]
[†]UC Berkeley [‡]Facebook AI Research [§]Technion [¶]OpenAI
[†]{jxwuyi,russell}@cs.berkeley.edu [‡]{yuxinwu,gkioxari,yuandong}@fb.com [§]avivt@technion.ac.il

A. Video Demo

A video demo visualizing a successful navigation trajectory by BRM can be found at the following url: `https://drive.google.com/file/d/1vCFQZfFK1X6WJacrQID2kMQVzRD4WeTs/view?usp=sharing`.

B. Environment Details

In RoomNav the 8 targets are: kitchen, living room, dining room, bedroom, bathroom, office, garage and outdoor. We inherit the success measure of "see" from [2]: the agent needs to see some corresponding object for at least 450 pixels in the input frame and stay in the target area for at least 3 time steps.

Originally the House3D environment supports a set of 13 discrete actions. Here we reduce it to 9 actions: large forward, forward, left-forward, right-forward, large left rotate, large right rotate, left rotate, right rotate and stay still. More environment details can be found in the appendix of Wu et al. [2]. We also implemented a faster and customized variant of the House3D environment, which is available at `https://github.com/jxwuyi/House3D/tree/C++`.

C. Evaluation Details

We measure the success rate on $\mathcal{E}_{\text{test}}$ over 5689 test episodes, which consists of 5000 randomly generated configurations and 689 specialized for faraway targets to increase the confidence of measured success rates. These 689 episodes are generated such that for each plan-distance, there are at least 500 evaluation episodes. Each test episode has a fixed configuration for a fair comparison between different approaches, i.e., the agent will always start from the same location with the same target in that episode. Note that we always ensure that (1) the target is connected to the birthplace of the agent, and (2) the the birthplace of the agent is never within the target room. In addition to the detailed numbers in Table 1,we visualize the success rates with *confidence intervals* for BRM and baseline methods in Figure 8. The confidence interval is obtained by fitting a binomial distribution.

D. Ablation Study: the Semantic Detector

In BRM, we use a CNN detector to extract the semantic signals at test time. Here we also evaluate the performances of all the approaches using the oracle signals from the House3D environment. The results are in Table 6, where we also include the BRM agent using CNN detector as a reference. Generally, using both the ground truth signal and using the CNN detector yield comparable overall performances in both metrics of success rate and SPL. They all consistently outperform all the baseline methods, which indicates that the probabilistic relational graph is robust over the noise on semantic signals (the robustness if controlled by ψ^{obs}). One interesting observation is that there are many cases, using CNN detector produces better results than using the ground truth signals. We hypothesis that this is because the semantic labels in House3D is noisy and therefore a well-trained CNN detector will not be influenced by the noisy labels at test time.

E. Additional Results on Episode Length

We illustrate the ground truth shortest distance information as well as the average episode length of success episodes for all the approaches. The results are shown in Table 7. The average ground truth shortest path is around 46.86 steps. Note that the agent has 9 actions per step and suffers from strong partial observability, which indicates the difficulty of the task.

F. Additional Implementation Details

The source code is available at `https://github.com/jxwuyi/HouseNavAgent`.

F.1. Learning the LSTM Locomotion

Policy Architecture: We utilize the same policy architecture and settings as [2]: we have 4 convolution layers of 64, 64, 128, 128 channels each and with kernel size 5 and stride 2, an MLP layer of 256 units, an LSTM cell of 256 units, two MLP layers of 126 and 64 units for policy head and another 2 MLP layers of 64 and 32 units for value head. Batch normalization is applied to all the layers before LSTM. Activation is ReLU. The a only difference is that the original policy uses a gated attention mechanism for target conditioning while we use a behavior approach by training a separate sub-policy for each semantic target.

For the semantic augmented policy, we feed the semantic information to the MLP layer before LSTM.

Hyperparameters: We run a parallel version of A2C [1] with 1 optimizer and 200 parallel rollout workers, each of which simulates a particular training house. We collect a training batch of 64

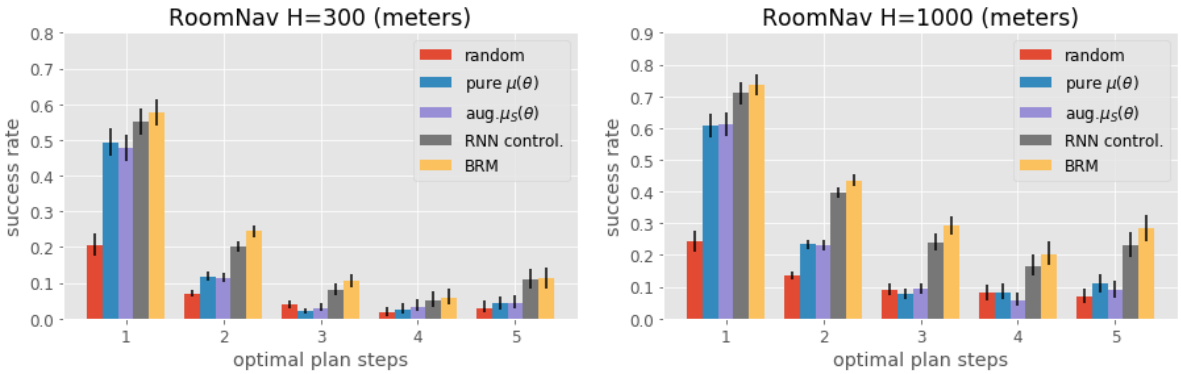

Figure 8. Comparing BRM with baselines in success rate with confidence interval. Approaches of interest include random policy (red), pure LSTM policy (blue), the semantic-aware LSTM policy (purple), the hierarchical policy (grey) and BRM (yellow). In all plots, the y-axis is success rate while the x-axis is the optimal planning distance. BRM outperforms all baselines and the gap becomes more significant when horizon increase, namely, more planning computations.

plan-dist	1	2	3	4	5	overall
Horizon $H = 300$						
plan-dist	1	2	3	4	5	avg.
random	20.5 / 15.9	6.9 / 16.7	3.8 / 10.7	1.6 / 4.2	3.0 / 8.8	7.2 / 13.6
pure $\mu(\theta)$	49.4 / 47.6	11.8 / 27.6	2.0 / 4.8	2.6 / 10.8	4.2 / 13.2	13.1 / 22.9
aug.$\mu_S(\theta)$ (true)	51.9 / **66.4**	11.1 / 24.2	3.3 / 7.8	2.4 / 6.0	3.0 / 8.7	13.2 / 23.3
RNN control. (true)	54.9 / 48.1	20.2 / 37.7	8.2 / 22.5	5.6 / 13.8	9.8 / 22.7	20.0 / 32.6
BRM (true)	**58.8** / 60.7	**25.3 / 55.6**	10.4 / 26.9	**7.6 / 22.2**	9.2 / 23.4	**23.6** / 44.9
BRM (CNN)	57.8 / 65.4	24.4 / 54.3	**10.5 / 28.3**	5.8 / 18.6	**11.2 / 29.8**	23.1 / **45.3**
Horizon $H = 1000$						
plan-dist	1	2	3	4	5	avg.
random	24.3 / 17.6	13.5 / 20.3	9.1 / 14.3	8.0 / 9.3	7.0 / 11.5	13.0 / 17.0
pure $\mu(\theta)$	60.8 / 47.6	23.3 / 27.6	7.6 / 4.8	8.2 / 10.8	11.0 / 13.2	22.5 / 22.9
aug.$\mu_S(\theta)$ (true)	62.4 / 61.3	22.9 / 30.7	8.9 / 14.3	7.2 / 12.8	9.0 / 11.4	22.5 / 28.1
RNN control. (true)	70.2 / 51.3	40.8 / 48.6	22.8 / 32.2	16.4 / 23.4	24.2 / 41.0	37.4 / 42.9
BRM (true)	70.3 / 61.8	**44.9 / 70.5**	**31.7 / 50.8**	19.0 / **33.3**	28.0 / 42.2	**41.7 / 59.8**
BRM (CNN)	**73.7 / 74.9**	43.6 / 66.0	29.2 / 44.9	**20.4** / 27.1	**28.4 / 42.5**	41.1 / 57.5

Table 6. Metrics of ***Success Rate(%) / SPL(‰)*** evaluating the performances of BRM and baselines agents using the ground truth oracle semantic signals provided by the environments. We also include the performance of the original BRM agent using CNN detector as a reference. The performance of BRM-CNN agents is comparable to BRM-true agents and sometimes even better. More discussions are in Sec. D.

trajectories with 30 continuous time steps in each iteration. We set $\gamma = 0.97$, batch size 64, learning rate 0.001 with Adam, weight decay 10^{-5} and entropy bonus 0.1. We also add the squared l_2 norm of policy logits to the total loss with a coefficient of 0.01. We normalize the advantage to mean 0 and standard deviation 1. We totally run 60000 training iterations and use the final model as our learned policy.

Reward shaping: The reward at each time step is computed by the difference of shortest paths in meters from the agent's location to the goal after taking a action. We also add a time penalty of 0.1 and a collision penalty of 0.3. When the agent reaches the goal, the success reward is 10.

Curriculum learning: We run a curriculum learning by increasing the maximum of distance between agent's birth meters and target by 3 meters every 10000 iterations. We totally run 60000 training iterations and use the final model as our learned policy $\mu(\theta)$.

F.2. Building the Relational Graph

We run random exploration for 300 steps to collect a sample of z. For a particular environment, we collect totally 50 samples for each $z_{i,j}$. For all $i \neq j$, we set $\psi^{\text{obs}}_{i,j,0} = 0.001$ and $\psi^{\text{obs}}_{i,j,1} = 0.15$.

F.3. Training the CNN Semantic Extractor

We take the panoramic view as input, which consists of 4 images, $s^1_o, \ldots, s^4_o$ with different first person view angles. The only exception is that for target "outdoor", we notice that instead of using a panoramic view, simply keeping the recent 4 frames in the

Average Ground Truth Shortest Path Length						
plan-dist	1	2	3	4	5	overall
Oracle	12.27	42.53	61.09	72.47	63.74	46.86
Average Successful Episode Length						
plan-dist	1	2	3	4	5	overall
Horizon $H = 300$						
random	34.0	112.7	143.8	148.0	149.7	89.8
pure $\mu(\theta)$	55.2	107.0	127.9	140.8	139.4	84.7
aug.$\mu_S(\theta)$	49.7	112.5	159.9	179.1	176.8	89.2
RNN control.	65.0	132.3	157.2	142.7	144.1	111.8
BRM	56.5	124.4	167.8	150.7	127.6	107.7
Horizon $H = 1000$						
random	121.7	354.7	426.6	532.8	409.5	322.1
pure $\mu(\theta)$	55.2	107.0	127.9	140.8	139.4	84.7
aug.$\mu_S(\theta)$	163.1	360.9	471.9	460.7	432.5	307.1
RNN control.	174.0	368.4	465.3	466.6	397.6	339.5
BRM	172.9	350.5	460.0	512.3	418.1	337.0

Table 7. Averaged successful episode length for different approaches. The length of shortest path reflects the strong difficulty of this task.

trajectory leads to the best prediction accuracy. We use an CNN feature extractor to extract features $f(s_o^i)$ by applying CNN layers with kernel size 3, strides $[1, 1, 1, 2, 1, 2, 1, 2, 1, 2]$ and channels $[4, 8, 16, 16, 32, 32, 64, 64, 128, 256]$. We also use relu activation and batch norm. Then we compute the attention weights over these 4 visual features by $l_i = f(s_o^i)W_1^T W_2 \left[f(s_o^1), \ldots, f(s_o^4)\right]$ and $a_i = \text{softmax}(l_i)$. Then we compute the weighted average of these four frames $g = \sum_i a_i f(s_o^i)$ and feed it to a single layer perceptron with 32 hidden units. For each semantic signal, we generate 15k positive and 15k negative training data from $\mathcal{E}_{\text{train}}$ and use Adam optimizer with learning rate $5e$-4, weight decay $1e$-5, batch size 256 and gradient clip of 5. We keep the model that has the best prediction accuracy on $\mathcal{E}_{\text{valid}}$.

For a smooth prediction during testing, we also have a hard threshold and filtering process on the CNN outputs: $s_s(T_i)$ will be 1 only if the output of CNN remains a confidence for T_i over 0.9 for consecutively 3 steps.

References

[1] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[2] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3D environment. *arXiv preprint arXiv:1801.02209*, 2018.