Supplementary Material

Yunbo Zhang Wenhao Yu Greg Turk

A. TNB Hyperparameters

We use $L = 45$ and $G = 3$ for segmenting the rollouts into training data for autoencoders. We used $G = 2$ for the Reacher and 4-Way Maze environments as exceptions because the best policy takes on average less than 45 steps. When computing the novelty reward using Equation 3 in the main text, we use $w_{novel} = 2$ across all examples.

B. Autoencoder Training

For all of our examples, we use autoencoders that are represented by fully connected neural network with 11 hidden layers consisting of {1024, 512, 256, 128, 64, 32, 64, 128, 256, 512, 1024} nodes. We use ReLUs as activation functions for the hidden layers, and linear activation for the output layer. The intuition behind using such a large neural network architecture is that we want the autoencoders to be slightly overfitted to the training data so that it will not accidentally generalize to behaviors that we may deem novel. Each of the autoencoder takes 200 epochs to train with a batch size of 1024 samples, and the training is optimized using the Adam optimizer with learning rate of 10^{-3}.

For each policy generated by TNB, we take 10 policies from the last 50 iterations of policy training to generate data for training the autoencoders. For each of these policies, we collect 100 rollouts for data parsing.

C. PPO Training

For all experiments in the paper, we use fully connected neural network policies with three 64-unit hidden layers using tanh nonlinearities. During each PPO iteration, we collect $12,000$ steps from the current policy and update the policy parameters using Adam for 3 epochs, with a minibatch size of 64. The learning rate of Adam is set to 0.003. As suggested in the OpenAI Baselines (Dhariwal et al., 2017) implementation, we use 0.2 as the clip parameter and 0 for the entropy term in the surrogate loss. For the parameters in the GAE (Schulman et al., 2015), we also use the default values in their implementation where $\gamma = 0.99$, and $\lambda = 0.95$.

Table 1. D-Maze with different weights

WEIGHTS	AVG # SUCC POLICIES	# SUCC TRIALS
100	0.4	1/5
200	0.6	3/5
500	**0.6**	**3/5**
1000	0.4	2/5

D. SAC Training

We used the OpenAI Spinning Up implementation for the Soft-Actor-Critic algorithm. For a fair comparison, we used the same number of samples per iterations as in PPO. In particular, we set the replay buffer size to 10^6, and $\alpha = 0.2$ for all tasks.

E. 4-Way Maze Reward Details

In the 4-Way Maze environment, the agent is given the task to reach one of the four red goals, thus there are four possible solutions to this task. Furthermore, we design a reward function such that different solutions yield different expected rewards. This helps test to see if an algorithm strikes the right balance between finding a novel behavior and obtaining the highest reward. Specifically, the reward signal for reaching a floor cell and a goal cell are formulated as follows:

$$r_i^{floor} = 50 * (\frac{5-i}{4})^3 \quad (1)$$

$$r_i^{goal} = 500 * (\frac{5-i}{4})^3, \quad (2)$$

where i corresponds to the path that has the i_{th} highest expected reward. The cubic order decrease in the reward value aims to create an obvious separation of rewards between paths, so that certain paths are more easily found by an algorithm. In addition to the path rewards, the environment also gives an alive penalty $r^{alive} = -1$ and a penalty of $r^{wall} = -10$ when the agent runs into the wall. An illustration of the 4-way Maze environment is shown in Figure 1, where higher reward paths are assigned brighter colors.

F. More Results for Weighted Sum Reward

For the weighted sum methods, we test multiple weights and report the best performing one. Specifically, we used 100,

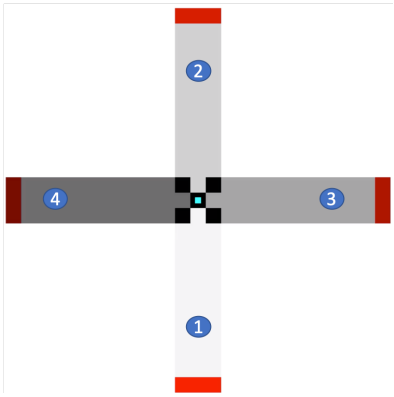

Figure 1. Paths in 4-Way Maze are labeled in the decreasing order of rewards.

Table 2. D-Reacher with different weights

WEIGHTS	AVG # SUCC POLICIES	# SUCC TRIALS
100	1.4	3/5
200	**1.2**	**4/5**
500	1.2	3/5
1000	1.2	3/5

Table 3. 4-Way Maze with different weights

WEIGHTS	AVG # PATHS EXPLORED	# TRIALS IN ORDER
100	**3.8**	**3/5**
200	4	2/5
500	4	0/5
1000	4	0/5

200, 500, and 1000 as our weights to the novelty reward term. We picked this range so that the two rewards shares comparable magnitudes. The bold weights shown in Table 1, Table 2, Table 3 are the weights used for comparison in the paper.

References

Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y. Openai baselines. GitHub, GitHub repository, 2017.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. arXiv preprint arXiv:1506.02438, 2015.