Cross-domain Imitation from Observations – Supplementary materials

1. Implementation details

Baselines. We use a total of 200 expert trajectories for each proxy task, in the both expert and self domains, to learn the state map. For IF, we use Dynamic Time Warping (DTW) (Müller, 2007) to obtain state correspondences. First, we randomly pair trajectories (due to lack of pairing) and simply pair the states that are visited in the same time step in the two proxy domains and use it to learn a common feature space. This feature space serves as a metric space for DTW to re-estimate correspondences across domains. The new correspondences are then used as pairs for learning a better feature space, and so on. For CycleGAN, we follow an adversarial learning scheme similar to our framework, with consistency applied both on the state and latent spaces. To visualize and evaluate the state maps learned in prior work, we use the encoder and decoder for IF and the Moore-Penrose pseudo inverse of the embedding matrix for CCA.

Architecture. The state maps $\{\psi, \phi\}$ are neural networks, with hidden layers of size $[128, 64]$ (both encoder and decoder), on the Reacher experiments and $[512, 256]$ for the others. The state space discriminators $\{D_A^j, D_E^j\}_{j=1}^M$ and latent space discriminators $\{q^j\}_{j=1}^M$ comprises hidden layers of size $[128, 128]$ for the Reacher experiments and $[512, 256, 128]$ for the rest. All discriminators use spectral normalization (Miyato et al., 2018) and additionally, replace the negative log likelihood objective in $\mathcal{L}_{adv}$ by a least-squares loss (Mao et al., 2017). This loss has been shown to be more stable during training. Temporal position estimators $\{P_A^j, P_E^j\}_{j=1}^M$ consist of hidden layers of size $[200, 128]$. Latent space position estimator V_z, for the inference task adaptation, contain hidden layers of size $[64, 64]$. The fitted policy $\pi_A^{\mathcal{T}}$ and the inverse dynamics model $\mathcal{I}_A$ have hidden layers of size $[64, 64]$ and $[100, 100]$ respectively. For CycleGAN, we use the same architecture as the state map in our framework. For IF, we use hidden layers with $[128, 64]$ units and leaky ReLU non-linearities to parameterize the encoders and decoders. We use Adam optimizer with default decay rates and learning rate $1e$-4 for training. With regards to the hyperparameters in Eqn. 10, we set them as For our experiments, $\lambda_1 = 2, \lambda_2 = \lambda_3 = \lambda_4 = \lambda_5 = 1$. Finally, for CCA, the embedding dimension is the minimum state dimension between the expert and self domains. We train all our models on a single Titan Xp GPU using PyTorch.

2. Environment details

The various reacher environments used in the tasks are extended from the "Reacher-v2" OpenAI Gym (Brockman et al., 2016) environment. A k link reacher has a state vector of the form $(\omega_1, \ldots, \omega_k, \dot{\omega}_1, \ldots, \dot{\omega}_k, x_g, y_g)$, where ω_i and $\dot{\omega}_i$ are the joint angle and angular velocity of the ith joint, and (x_g, y_g) is the position of the goal. The action vector has the form $(\tau_1, \ldots, \tau_k)$, where τ_i is the torque applied to the ith joint. The state map acts only on the non-goal dimensions. Following (Kim et al., 2020), proxy goals are placed near the wall of the arena and the target tasks are reaching for 4 new goals near the corner of the arena. The new goals are placed as far as possible from the proxy goals within the bounds of the arena. Figure 1 depicts the location of the goals.

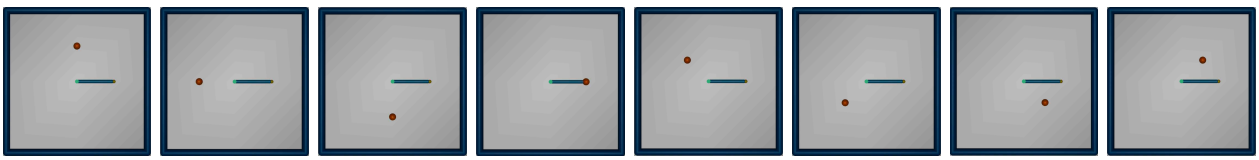

Figure 1. **Reacher task visualization.** The goal locations used in our reacher experiments. First four goals constitute the set of proxy tasks, the next four serve as inference goals.

For the V-R2W task, the proxy tasks are the same as the ones discussed previously, while the target task is tracing the letter C (shown in Figure 2) as fast as possible. The goal location in the writing task represents the next vertex of the letter to trace. Once the first vertex is reached, the goal coordinates are updated to be the next

Figure 2. **V-R2W inference task.** The sequence of goals need to be reached as quick as possible.

vertex coordinates. The reward function is defined as follows:

$$R_{write}(s) = \begin{cases} 100, & \text{if state } s \text{ corresponds to reaching a vertex} \\ -1, & \text{otherwise} \end{cases}$$

Thus the agent must perform a sequential reaching task and accomplish it as fast as possible. The key difference with a normal reaching task is that the reacher must not slow down at each vertex and plan its path accordingly in order to minimize drastic direction changes.

The two ant environments and the cheetah environment are derived derived from the "Ant-v2" and "HalfCheetah-v2" environments respectively. A k-legged Ant has a state vector of the form $(c_x, c_y, c_z, q_0, \ldots, q_3, \omega_1, \ldots, \omega_{2k}, \dot{c}_x, \dot{c}_y, \dot{c}_z, \dot{q}_1, \dot{q}_2, \dot{q}_3, \dot{\omega}_1, \ldots, \dot{\omega}_{2k}, x_g, y_g)$, where (c_x, c_y, c_z) denotes the torso 3D co-ordinates, $(q_0, \ldots, q_3)$ denotes the torso orientation quarternion, $(\dot{c}_x, \dot{c}_y, \dot{c}_z)$ denotes the torso 3D velocity and $(\dot{q}_1, \dot{q}_2, \dot{q}_3)$ denotes the torso angular velocity. The rest are the same as the reacher, with 2 hinge joints per leg. The action vector has the form $(\tau_1, \ldots, \tau_{2k})$, where τ_i is the torque applied to the ith joint. For the cheetah, the state vector is of the form $(r_x, r_y, r_z, \omega_1, \ldots, \omega_6, \dot{r}_x, \dot{r}_y, \dot{r}_z, \dot{\omega}_1, \ldots, \dot{\omega}_6, x_g, y_g)$ where (r_x, r_y, r_z) denotes the root 3D co-ordinates and $(\dot{r}_x, \dot{r}_y, \dot{r}_z)$ are the corresponding velocities; rest are the same as the reacher for the 6 hinge joints (3 for each leg). The action vector has the form $(\tau_1, \ldots, \tau_6)$, where τ_i is the torque applied to the ith joint. For all these environments, the task is to reach the center of a circle of radius 5m with the agent being initialized on a 2° arc of the circle. Different initializations define the different tasks as shown in Figure 3.

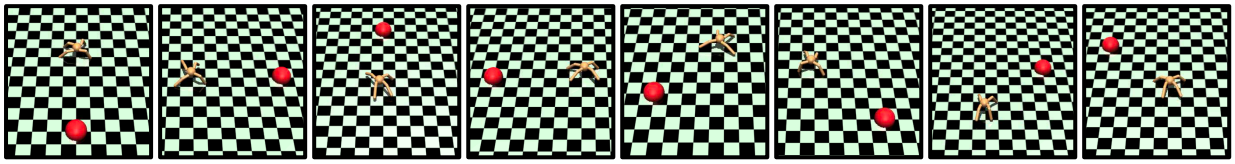

Figure 3. **Ant task visualization.** The tasks used in our ant/cheetah experiments. First four constitute the set of proxy tasks, the next four serve as inference tasks.

References

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Kim, K., Gu, Y., Song, J., Zhao, S., and Ermon, S. Domain adaptive imitation learning. In *International Conference on Machine Learning*, pp. 5286–5295. PMLR, 2020.

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Paul Smolley, S. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2794–2802, 2017.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Müller, M. Dynamic time warping. *Information retrieval for music and motion*, pp. 69–84, 2007.