

---

# Sample-efficient Adversarial Imitation Learning

## Supplementary Materials

---

Dahuin Jung

Hyungyu Lee

Sungroh Yoon

Electrical and Computer Engineering,  
ASRI, INMC, and Institute of Engineering Research  
Seoul National University, Seoul 08826, South Korea  
{anna jung0625, rucy74, sryoon}@snu.ac.kr

### S1 Additional Implementation and Experimental Details

We use 5 continuous control benchmarks on Mujoco [9] (Ant-v2, HalfCheetah-v2, Hopper-v2, Swimmer-v2, and Walker2d-v2), and 2 discrete control benchmarks on Atari RAM (BeamRider-ram-v0, and SpaceInvaders-ram-v0) of OpenAI Gym [2].

Overall, we reported the mean and standard error of the performance over 3 trials. For experimental settings, we used GTX 1080 Ti for GPUs, Intel i7-6850K for CPUs, and Ubuntu 18.04 for OS. The usage of GPU memory is approximately 3000MB for training, and training time for tested benchmarks was approximately 20 hours. Our code is based on Pytorch [4] and python libraries.

We make use of the same neural net architecture and hyperparameters for all benchmarks. For the policy network, value network, discriminator, and state encoder, we use 3 hidden layers with size 100 and Tanh as activation functions. For the action encoder, we use 6 1D convolutional layers with sizes (64, 64, 64, 128, 256, and 256), 1 hidden layer with size 8 as the output, and LeakyReLU as activation functions. For the forward dynamics model, we use 1 hidden layer with size 114 and ReLU as activation functions.

For hyperparameters in all runs, the total epoch for Swimmer, and Hopper is 3,000, for BeamRider, SpaceInvaders, HalfCheetah, and Walker2d is 5,000, and for Ant is 8,000. Please refer to Table S1 for other hyperparameters. We set  $\lambda_F = 1$ ,  $\lambda_S = 100$ , and  $\lambda_A = 1$  for matching loss scale.

Table S1: Base hyperparameters used for all benchmarks.

Hyperparameters	Value
$\gamma$	0.995
Generalized advantage estimation	0.97
$N$	5,000
Learning rate (all networks except for value network)	1e-3
Learning rate (value network)	3e-4
Batch size (RERP)	256
Batch size (TRPO)	128
Batch size (GAIL)	5,000
Optimizer (all networks)	Adam
$\tau$	0.1
$\lambda_F$	1
$\lambda_S$	100
$\lambda_A$	1

For the random corruption method [12], we sampled an imputing value from  $\mathcal{N}(0, 1)$  by considering the range of state values. The mean value used for imputation [1] is the mean vector of each  $\mathcal{D}_k$  where  $k$  is the training iteration. We imputed with a mean value of the corresponding dimension of the calculated mean vector.

### S1.1 Optimality of 100% (Expert Demonstrations)

To train experts, we used a reinforcement learning algorithm, proximal policy optimization [7], uploaded in the official Github by the authors of CAIL [13]. We selected the converged policy as the expert policy. The performance of the utilized expert policy and other specifications related to the experiments of Tabs. 1, 2, 3 and 5 on the main manuscript are given in Table S2.

Table S2: Specification and the number of used demonstrations of each continuous control benchmark in the scenario of perfect expert demonstrations.

Benchmarks	$\dim(S)$	$\dim(A)$	$N_E$	Expert's Performance
HalfCheetah-v2	$\mathbb{R}^{17}$	$\mathbb{R}^6$	100	$5455.49 \pm 74.26$
Walker-v2	$\mathbb{R}^{17}$	$\mathbb{R}^6$	100	$3685.27 \pm 57.99$
Ant-v2	$\mathbb{R}^{111}$	$\mathbb{R}^8$	100	$4787.23 \pm 115.72$

### S1.2 Optimality of 25%, 50%, or 75% (A Mixture of Optimal and Non-optimal Demonstrations)

We also tested the effectiveness of the proposed method with imperfect demonstrations. We combined the proposed method with the existing method for imperfect demonstrations and checked improvement in performance. We compared our combined method against the following baselines: BC [5], GAIL [3], IC-GAIL [11], 2IWIL [11], RIL-CO [8], and CAIL [13].

IL methods, especially variants of BC, require a large volume of expert demonstration data for training [6]. These methods struggle with a generalization problem when using a small number of demonstrations. Empirically, we observed that BC almost fails to converge on Ant, Walker2d, and Hopper with a small number of demonstrations. For RIL-CO, they measure and optimize a classification risk with the symmetric loss. Basically, they only assumed a scenario that the majority of demonstrations are obtained using an optimal policy. IC-GAIL and 2IWIL were proposed in the same paper. Both methods are confidence-based. In the case of 2IWIL, the confidence of each state-action pair is estimated using the classification risk before training, and in the case of IC-GAIL, they implicitly utilize the confidence score in a way of matching the occupancy measure of the imitator with the expert. CAIL is the state-of-the-art work in IL algorithms for imperfect demonstrations. They jointly learn the confidence score and policy using an outer loss. Because they update two factors jointly, the training is unstable. Experimentally, there was no benchmark that CAIL, which is most recently suggested, is superior to 2IWIL or RIL-CO with a small number of imperfect demonstrations. We surmise that this is because CAIL is the method using a full trajectory rather than each state-action pair when estimating the confidence score of each pair. Consequently, we decided to combine our method with 2IWIL instead of CAIL.

Table S3: Specification and the number of used demonstrations of each continuous control benchmark in the scenario of imperfect expert demonstrations.

Benchmarks	$\dim(S)$	$\dim(A)$	$N_E$	Suboptimal 1	Suboptimal 2	Suboptimal 3	Suboptimal 4	Expert's Performance
HalfCheetah-v2	$\mathbb{R}^{17}$	$\mathbb{R}^6$	100	$1051.91 \pm 50.17$	$2280.87 \pm 651.92$	$3533.07 \pm 79.47$	$4682.89 \pm 54.33$	$5455.49 \pm 74.26$
Walker-v2	$\mathbb{R}^{17}$	$\mathbb{R}^6$	100	$691.14 \pm 96.12$	$1617.02 \pm 721.00$	$2579.41 \pm 512.19$	$2819.63 \pm 609.49$	$3685.27 \pm 57.99$
Ant-v2	$\mathbb{R}^{111}$	$\mathbb{R}^8$	100	$789.13 \pm 170.45$	$2115.17 \pm 328.20$	$2947.49 \pm 191.72$	$3739.91 \pm 96.56$	$4787.23 \pm 115.72$
Swimmer-v2	$\mathbb{R}^8$	$\mathbb{R}^2$	20	$65.56 \pm 18.93$	$148.20 \pm 8.09$	$181.27 \pm 4.23$	$228.29 \pm 5.95$	$280.5 \pm 1.24$
Hopper-v2	$\mathbb{R}^{11}$	$\mathbb{R}^3$	20	$1262.34 \pm 296.32$	$1774.65 \pm 462.52$	$2185.33 \pm 996.92$	$2802.18 \pm 489.85$	$3531.03 \pm 23.51$

For the Fig. 2 and Tabs. 4, 6, 7, and 8 on the main manuscript, we collected a mixture of optimal and non-optimal demonstrations with different optimalities. For collecting the imperfect demonstrations, we used the official Github by the authors of CAIL. Following CAIL, We selected four intermediate policies as sub-optimal policies and the converged policy as the optimal policy. The performance of sub-optimal and optimal policies and other specifications for the benchmarks are given in Table S3.

Empirically, as shown in Table S3, because the dimension of the action space of Swimmer-v2 and Hopper-v2 is very small almost like discrete control, we did not apply  $\mathcal{L}_{AC}$ .

The pseudo-code of the combined methods is given in Algorithms 2 and 3. To run the combined methods, we need additional hyperparameters and their values are summarized in Table S4.

Table S4: Additional hyperparameters for the combined methods.

Hyperparameters	Value
$\alpha$ for Mixup	4.0
Threshold for GMM	0.5
The number of non-experts	4
Ratio of labeled demonstrations	0.4

## S2 Main Pseudo-code

---

### Algorithm 1 Sample-efficient Adversarial Imitation Learning

---

```

1: input: Expert demonstrations  $\mathcal{D}_E \triangleq \{x_i\}_{i=1}^{N_E}$ , # of batches B, Training epochs T.
2: for  $k \leftarrow 1$  to T do
3:   Obtain trajectories  $\mathcal{D}_k = \{x_{k,i}\}_{i=1}^N$  using  $\pi_\theta$ 
4:    $\pi_\theta \leftarrow \text{TRPO}(\pi_\theta, V, D_\omega, \mathcal{D}_k)$ 
5:    $SE, AE \leftarrow \text{REPR}(SE, AE, F, \mathcal{D}_k)$ 
6:    $D_\omega \leftarrow \text{GAIL}(D_\omega, SE, AE, \mathcal{D}_k, \mathcal{D}_E)$ 
7: end for
8: function REPR( $SE, AE, F, \mathcal{D}_k$ )
9:   for  $b \leftarrow 1$  to B do
10:    Generate  $X'_b$  by Equation 6
11:    Obtain  $Z_b$  from  $\mathcal{D}_{k,b}$  using ( $SE, AE$ )
12:    Obtain  $Z'_b$  from  $X'_b$  using ( $SE, AE$ )
13:    Update  $SE, AE$ , and  $F$  by Equation 9
14:   end for
15:   return  $SE, AE$ 
16: end function
17: function GAIL( $D_\omega, SE, AE, \mathcal{D}_k, \mathcal{D}_E$ )
18:   for  $b \leftarrow 1$  to B do
19:    Obtain  $Z_b$  from  $\mathcal{D}_{k,b}$  using ( $SE, AE$ )
20:    Update  $SE, AE$ , and  $D_\omega$  by Equation 3
21:   end for
22:   return  $D_\omega$ 
23: end function

```

---

## S3 Pseudo-code of Combined Method

### S3.1 Combined Case #1: Ours + 2IWIL

Because labeling all state-action pairs from  $\rho_O$  or  $\rho_N$  can be expensive, following previous IL works for imperfect demonstrations, we assumed that only a few demonstrations are labeled. Then, the imperfect demonstrations are split into two demonstrations: a set of labeled demonstrations  $\mathcal{D}_L = \{(x_{l,i}, y_{l,i})\}_{i=1}^{N_L}$  and a set of unlabeled demonstrations  $\mathcal{D}_U = \{x_{u,i}\}_{i=1}^{N_U}$ , where  $N_L$  and  $N_U$  are the number of labeled demonstrations and unlabeled demonstrations, respectively.

As shown in Algorithm 2, before training, 2IWIL [11] estimates pseudo labels  $\hat{y}_{u,i}$  of  $\mathcal{D}_U$  to utilize the set of unlabeled demonstrations  $\mathcal{D}_U$  with  $\mathcal{D}_L$  using the classification risk proposed in their work as follows:

$$R_{\ell(g)} = \mathbb{E}_{x, y \sim \mathcal{D}_l} [y(\ell(g(x)) - \ell(-g(x))) + (1 - \beta)\ell(-g(x))] + \mathbb{E}_{x \sim \mathcal{D}_u} [\beta\ell(-g(x))], \quad (\text{S1})$$

---

**Algorithm 2** Pseudo-code of Ours + 2IWIL [11]

---

```
1: input: Imperfect expert demonstrations  $\mathcal{D}_I = \{\mathcal{D}_L \cup \mathcal{D}_U\}$ , Labeled demonstrations  $\mathcal{D}_L \triangleq \{(x_{l,i}, y_{l,i})\}_{i=1}^{N_L}$ , Unlabeled demonstrations  $\mathcal{D}_U \triangleq \{x_{u,i}\}_{i=1}^{N_U}$ , # of batches  $B$ , Training epochs  $T$ .
2: Train a probabilistic classifier by minimizing Equation S1
3: Predict confidence scores  $\{\hat{y}_{u,i}\}_{i=1}^{N_u}$  for  $\{x_{u,i}\}_{i=1}^{N_u}$ 
4: for  $k \leftarrow 1$  to  $T$  do
5:   Obtain trajectories  $\mathcal{D}_k = \{x_{k,i}\}_{i=1}^N$  using  $\pi_\theta$ 
6:    $\pi_\theta \leftarrow \text{TRPO}(\pi_\theta, V, D_\omega, \mathcal{D}_k)$ 
7:    $SE, AE \leftarrow \text{REPR}(SE, AE, F, \mathcal{D}_k)$ 
8:    $D_\omega \leftarrow \text{GAIL}(D_\omega, SE, AE, \mathcal{D}_k, \mathcal{D}_I)$ 
9: end for
10: function  $\text{REPR}(SE, AE, F, \mathcal{D}_k)$ 
11:   for  $b \leftarrow 1$  to  $B$  do
12:     Generate  $X'_b$  by Equation 6
13:     Obtain  $Z_b$  from  $\mathcal{D}_{k,b}$  using  $(SE, AE)$ 
14:     Obtain  $Z'_b$  from  $X'_b$  using  $(SE, AE)$ 
15:     Update  $SE, AE$ , and  $F$  by Equation 9
16:   end for
17:   return  $SE, AE$ 
18: end function
19: function  $\text{GAIL}(D_\omega, SE, AE, \mathcal{D}_k, \mathcal{D}_I)$ 
20:   for  $b \leftarrow 1$  to  $B$  do
21:     Obtain  $Z_b$  from  $\mathcal{D}_{k,b}$  using  $(SE, AE)$ 
22:     Update  $SE, AE$ , and  $D_\omega$  by Equation S2
23:   end for
24:   return  $D_\omega$ 
25: end function
```

---

where  $\beta = \frac{N_U}{N_L + N_U}$ ,  $g(\cdot)$  is a neural network classifier, and  $\ell$  is a strictly proper composite loss. Then, the predicted confidence score  $\hat{y}_u$  represents the probability that a given state-action pair is optimal. The estimated confidence of each state-action pair is utilized as a sample weight in their discriminator loss. As a result, the combined discriminator loss with our state and action encoders is defined as follows:

$$\max_{\omega} \mathbb{E}_{x \sim \mathcal{D}_\pi} [\log D_\omega(z)] + \mathbb{E}_{(x,y) \sim (\tilde{\mathcal{D}}_O \cup \tilde{\mathcal{D}}_N)} \left[ \frac{y}{\epsilon} \log(1 - D_\omega(z)) \right], \quad (\text{S2})$$

where  $\epsilon = \frac{1}{N_L} \sum_{i=1}^{N_L} y_i$ ,  $z = z^s \oplus z^a$ .  $z^s$  is a state representation embedded by  $SE(s)$ , and  $z^a$  is an action representation embedded by  $AE(a)$ .

### S3.2 Combined Case #2: Ours + 2IWIL + Manifold Mixup

For utilizing Manifold mixup (MM), we modeled the per-sample confidence score distribution of  $(y_l, \hat{y}_u)$  with a two-component Gaussian mixture model to divide the imperfect demonstrations  $\mathcal{D}_I$  into optimal demonstrations and non-optimal demonstrations,  $\tilde{\mathcal{D}}_O$  and  $\tilde{\mathcal{D}}_N$ . The feature representation  $z_o \sim \tilde{\mathcal{D}}_O$  is interpolated with the feature representation  $z_n \sim \tilde{\mathcal{D}}_N$ . More formally, for a batch of features  $(z_o, z_n)$  and corresponding confidence scores  $(y_o, y_n)$ , the mixed  $(\bar{z}, \bar{y})$  can be computed by:

$$\begin{aligned} \lambda &\sim \text{Beta}(\alpha, \alpha), \quad \lambda' = \max(\lambda, 1 - \lambda), \\ \bar{z} &= \lambda' \cdot z_o + (1 - \lambda') \cdot z_n, \\ \bar{y} &= \lambda' \cdot y_o + (1 - \lambda') \cdot y_n, \end{aligned} \quad (\text{S3})$$

where  $z = z^s \oplus z^a$ .  $z^s$  is a state representation embedded by  $SE(s)$ ,  $z^a$  is an action representation embedded by  $AE(a)$ . Equation S3 can ensure that  $\bar{z}$  are closer to optimal demonstrations than non-optimal demonstrations.

---

**Algorithm 3** Pseudo-code of Ours + 2IWIL + Manifold Mixup [10]

---

```
1: input: Imperfect expert demonstrations  $\mathcal{D}_I = \{\mathcal{D}_L \cup \mathcal{D}_U\}$ , Labeled demonstrations  $\mathcal{D}_L \triangleq \{(x_{l,i}, y_{l,i})\}_{i=1}^{N_L}$ , Unlabeled demonstrations  $\mathcal{D}_U \triangleq \{x_{u,i}\}_{i=1}^{N_U}$ , # of batches  $B$ , Training epochs  $T$ .
2: Train a probabilistic classifier by minimizing Equation S1
3: Predict confidence scores  $\{\hat{y}_{u,i}\}_{i=1}^{N_U}$  for  $\{x_{u,i}\}_{i=1}^{N_U}$ 
4:  $\tilde{\mathcal{D}}_O, \tilde{\mathcal{D}}_N \leftarrow GMM(\mathcal{D}_L, (\mathcal{D}_U, \{\hat{y}_{u,i}\}_{i=1}^{N_U}))$ 
5: for  $k \leftarrow 1$  to  $T$  do
6:   Obtain trajectories  $\mathcal{D}_k = \{x_{k,i}\}_{i=1}^N$  using  $\pi_\theta$ 
7:    $\pi_\theta \leftarrow \text{TRPO}(\pi_\theta, V, D_\omega, \mathcal{D}_k)$ 
8:    $SE, AE \leftarrow \text{REPR}(SE, AE, F, \mathcal{D}_k)$ 
9:    $D_\omega \leftarrow \text{GAIL}(D_\omega, SE, AE, \mathcal{D}_k, \tilde{\mathcal{D}}_O, \tilde{\mathcal{D}}_N)$ 
10: end for
11: function  $\text{REPR}(SE, AE, F, \mathcal{D}_k)$ 
12:   for  $b \leftarrow 1$  to  $B$  do
13:     Generate  $X'_b$  by Equation 6
14:     Obtain  $Z_b$  from  $\mathcal{D}_{k,b}$  using  $(SE, AE)$ 
15:     Obtain  $Z'_b$  from  $X'_b$  using  $(SE, AE)$ 
16:     Update  $SE, AE$ , and  $F$  by Equation 9
17:   end for
18:   return  $SE, AE$ 
19: end function
20: function  $\text{GAIL}(D_\omega, SE, AE, \mathcal{D}_k, \tilde{\mathcal{D}}_O, \tilde{\mathcal{D}}_N)$ 
21:   for  $b \leftarrow 1$  to  $B$  do
22:     Obtain  $Z_b$  from  $\mathcal{D}_{k,b}$  using  $(SE, AE)$ 
23:     Obtain  $(Z_o, Z_n)$  from  $(\tilde{\mathcal{D}}_O, \tilde{\mathcal{D}}_N)$  using  $(SE, AE)$ 
24:     Compute  $(\bar{Z}, \bar{Y})$  by Equation S3
25:     Update  $SE, AE$ , and  $D_\omega$  by Equation S4
26:   end for
27:   return  $D_\omega$ 
28: end function
```

---

As shown in Algorithm 3, by additionally including synthetic data through MM, the discriminator loss is expressed as follows:

$$\begin{aligned} \max_{\omega} \mathbb{E}_{x \sim \mathcal{D}_\pi} [\log D_\omega(z)] + & \mathbb{E}_{(x,y) \sim (\tilde{\mathcal{D}}_O \cup \tilde{\mathcal{D}}_N)} \left[ \frac{y}{\epsilon} \log(1 - D_\omega(z)) \right] + \\ & \mathbb{E}_{(\bar{z}, \bar{y}) \sim (\tilde{\mathcal{D}}_O \cup \tilde{\mathcal{D}}_N)} [\log((1 - \bar{y}) \cdot D_\omega(\bar{z}) + \bar{y} \cdot (1 - D_\omega(\bar{z})))] \end{aligned} \quad (\text{S4})$$

## S4 Expert Data Size

We assessed our method with varying expert data sizes. As shown in the table, there is a relatively small or no decrease in the performance up to  $N_E = 20$ . When  $N_E$  is reduced to 20 for Ant and Walker2d and 10 for HalfCheetah, the performance is comparable to the baseline AIL.

Table S5: Ablation studies using 10, 20, or 50 expert state-action pairs on Ant-v2, HalfCheetah-v2, and Walker2d-v2 of MuJoCo.

	GAIL	Ours			
	$N_E = 100$	$N_E = 10$	$N_E = 20$	$N_E = 50$	$N_E = 100$
Ant	4198.2 $\pm$ 72.6	2855.9 $\pm$ 1139.7	4286.4 $\pm$ 125.6	4432.4 $\pm$ 41.1	4554.8 $\pm$ 162.6
HalfCheetah	2034.6 $\pm$ 2384.6	2787.6 $\pm$ 3454.4	5381.6 $\pm$ 81.3	5410.6 $\pm$ 44.2	5416.0 $\pm$ 203.8
Walker2d	3513.4 $\pm$ 172.9	3023.1 $\pm$ 439.6	3412.6 $\pm$ 195.6	3520.9 $\pm$ 108.5	3527.6 $\pm$ 131.4

## S5 Role of Gaussian Noise

Appending noise dimensions increases the performance on all three tasks as shown in Table S6. -71.09, -81.29, and -87.90 represent the decrease without appending the noise dimensions.

Table S6: Ablation studies on appending Gaussian noise using 100 expert state-action pairs on Ant-v2, HalfCheetah-v2, and Walker2d-v2 of MuJoCo.

Ours	Ant	HalfCheetah	Walker2d
w/o noise	4483.7±159.6 (-71.09)	5334.7±43.0 (-81.29)	3439.7±122.2 (-87.90)
w noise	4554.8±162.6	5416.0±203.8	3527.6±131.4

## S6 Loss function of $\mathcal{L}_{SC}$ and $\mathcal{L}_{AC}$

As shown in Table S7, we assessed varying SSL loss functions for both  $\mathcal{L}_{SC}$  and  $\mathcal{L}_{AC}$ . Notably, for  $\mathcal{L}_{SC}$ , the MSE loss that is exposed to the collapsing problem shows the highest performance on average. This is because  $\mathcal{L}_F$  cannot be minimized if the state representation is only the same constant. Rather, the MSE loss that can flow the gradients to both the input pair shows a higher performance than the loss functions using a stop-gradient or different discrepancy measures. For  $\mathcal{L}_{AC}$ , the Barlow twins and SimSiam losses showed the first- and second-best performance on average, respectively. Because the role of the state is greater than that of the action when predicting the next state, the action representation  $z^a$  is not completely free from the collapsing problem. Therefore, unlike the trend of  $\mathcal{L}_{SC}$ , clearly, the loss functions that have a certain technique to prevent it showed stable performance.

Table S7: Ablation studies using 50 optimal and 50 non-optimal state-action pairs on Ant-v2 to test the role of a loss function of  $\mathcal{L}_{SC}$  and  $\mathcal{L}_{AC}$ . BT = Barlow twins.

		$\mathcal{L}_{SC}$					
		MSE	BYOL	SimSiam	BT	VICReg	Avg.
$5*\mathcal{L}_{AC}$	MSE	3658.9	-627.0	1516.0	2891.0	701.5	1628.1
	BYOL	2668.3	1240.8	1315.7	265.1	902.0	1278.4
	SimSiam	2717.8	4017.7	3854.5	3656.7	2943.9	3438.1
	BT	<b>4384.7</b>	2871.3	2590.4	3474.2	4269.2	<b>3517.9</b>
	VICReg	1927.1	2363.0	4027.8	3768.1	3458.1	3108.8
Avg.		<b>3071.4</b>	1973.1	2660.9	2811.0	2454.9	

## S7 Sensitivity to Corruption Rate

We tested the sensitivity of state and action to corruption rate and report the results in Table. S8. The results showed that the action is very vulnerable to a high corruption rate. For the state, the performance is maintained to some extent below 0.4. Additionally, we confirmed that fixing the corruption rate is better than providing a range.

Table S8: Ablation studies using 50 optimal and 50 non-optimal state-action pairs to test sensitivity to corruption rate of state and action.

$c^a$	0.2			0.5	<= 0.5
$c^s$	0.1	0.2	0.3	0.4	<= 0.5
Ant	<b>4384.7±49.2</b>	4282.8±170.5	4206.7±502.7	4127.3±451.8	3546.2±487.7
					2425.9±65.6
					560.6±326.9

## S8 Discrete control Benchmarks

To test the scalability of the proposed method, we evaluated the performance of the proposed method on 2 discrete control benchmarks: BeamRider-ram-v0, and SpaceInvaders-ram-v0 of OpenAI Gym.

Table S9: Final performance using 20 expert state-action pairs on BeamRider-ram-v0, and SpaceInvaders-ram-v0 of OpenAI Gym. Best results are in **bold**.

BeamRider		SpaceInvaders	
GAIL	Ours	GAIL	Ours
$399.43 \pm 55.63$	<b><math>433.04 \pm 76.15</math></b>	$166.39 \pm 107.88$	<b><math>289.87 \pm 5.37</math></b>

For comparison, we chose GAIL, which showed the second-best performance in the experiments of continuous control benchmarks. To apply the proposed method on discrete control benchmarks, we ignored the action encoder  $AE$  and its corresponding loss  $\mathcal{L}_{AC}$  of the proposed model. Nevertheless, as shown in Table S9, the proposed method still showed better performance compared to GAIL on discrete control benchmarks.

## References

- [1] D. Bahri, H. Jiang, Y. Tay, and D. Metzler. Scarf: Self-supervised contrastive learning using random feature corruption. *arXiv preprint arXiv:2106.15147*, 2021.
- [2] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.
- [3] J. Ho and S. Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29:4565–4573, 2016.
- [4] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.
- [5] D. A. Pomerleau. Alvin: An autonomous land vehicle in a neural network. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA ARTIFICIAL INTELLIGENCE AND PSYCHOLOGY ..., 1989.
- [6] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.
- [7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [8] V. Tangkaratt, N. Charoenphakdee, and M. Sugiyama. Robust imitation learning from noisy demonstrations. In *International Conference on Artificial Intelligence and Statistics*, pages 298–306. PMLR, 2021.
- [9] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.
- [10] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, D. Lopez-Paz, and Y. Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pages 6438–6447. PMLR, 2019.
- [11] Y.-H. Wu, N. Charoenphakdee, H. Bao, V. Tangkaratt, and M. Sugiyama. Imitation learning from imperfect demonstration. In *International Conference on Machine Learning*, pages 6818–6827. PMLR, 2019.
- [12] J. Yoon, Y. Zhang, J. Jordon, and M. van der Schaar. Vime: Extending the success of self- and semi-supervised learning to tabular domain. *Advances in Neural Information Processing Systems*, 33, 2020.
- [13] S. Zhang, Z. Cao, D. Sadigh, and Y. Sui. Confidence-aware imitation learning from demonstrations with varying optimality. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.