# Supplementary Material for Hybrid Policy Optimization from Imperfect Demonstrations

**Hanlin Yang** Sun Yat-sen University
**Chao Yu*** Sun Yat-sen University
**Peng Sun** ByteDance
**Siji Chen** Sun Yat-sen University

yuchao3@mail.sysu.edu.cn

## A Pseudo-Code for HYPO

The pseudo-code for HYPO algorithm is shown in Algorithm 1

**Algorithm 1** HYbrid Policy Optimization (HYPO)

Initialize the parameters $\hat{\theta}$ of the online agent policy $\hat{\pi}$, the parameters $\theta_b$ of the offline guider policy $\pi_b$, the discriminator $d$, and the imperfect demonstrations $\mathcal{D}$ generated by suboptimal expert;
Set the the maximal iterations $I$, the replay buffer $\mathcal{B}$, and the hyperparameters $\alpha$, $\eta$;
**for** $i = 1$ to $I$ **do**
Sample suboptimal expert trajectories $\tau_E \sim \mathcal{D}$;
Sample agent trajectories $\tau_A \sim \mathcal{B}$;
Update discriminator parameters from $w_i$ to $w_{i+1}$ with the gradient:

$$\eta\mathbb{E}_{\tau_E}\Big[\nabla_w \log d_w(s,a,\log\pi_b)\Big] + \mathbb{E}_{\tau_A}\Big[\nabla_w \log\Big(1 - d_w(s,a,\log\pi_b)\Big)\Big] - \eta\mathbb{E}_{\tau_E}\Big[\nabla_w \log\Big(1 - d_w(s,a,\log\pi_b)\Big)\Big]; \tag{1}$$

Update the offline guider policy $\pi_b$ with the supervised learning method (*e.g.*, BC) using:

$$\mathbb{E}_{\tau_E}\left[\nabla_{\theta_b}\log\pi_b(a|s)\cdot\left(\alpha - \frac{\eta}{d(1-d)}\right)\right] + \mathbb{E}_{\tau_A}\left[\nabla_{\theta_b}\log\pi_b(a|s)\cdot\left(\frac{1}{1-d}\right)\right]; \tag{2}$$

Update the online agent policy $\hat{\pi}$ with the policy gradient method (*e.g.*, PPO) using:

$$\mathbb{E}_{\tau_A}\Big[\nabla_{\hat{\theta}}\min\Big(r_t(\hat{\theta})A_t, \mathrm{clip}\big(r_t(\hat{\theta}), 1-\epsilon, 1+\epsilon\big)A_t\Big) - \nabla_{\hat{\theta}} C D_{\mathrm{KL}}(\hat{\pi}||\pi_b)\Big]; \tag{3}$$

**end for**

## B Theoretical Derivation

In this section, we first list some common lemmas that are useful in our theoretical results, and then give the derivation of the BC objective weights and the complete proof of the policy improvement bound.

*Corresponding author.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

### B.1 Useful Lemmas

**Lemma 1** (Performance Difference Lemma)**.** *For any two policies $\pi$ and $\tilde{\pi}$ and any starting state distribution $\mu$, we have*

$$J_R(\pi) - J_R(\tilde{\pi}) = \frac{1}{1-\gamma}\mathbb{E}_{(s,a)\sim\pi d_{\pi,\mu}}\big[A_R^{\tilde{\pi}}(s,a)\big]. \tag{4}$$

**Lemma 2** (Achiam et al. (2017))**.** *The divergence between the discounted future state visitation distributions, $\|d^{\pi} - d^{\pi'}\|_1$, is bounded by an average divergence of the policies $\pi'$ and $\pi$:*

$$\|d^{\pi} - d^{\pi'}\|_1 \leq \frac{2\gamma}{1-\gamma} D^{\pi}_{\mathrm{TV}}(\pi, \pi'). \tag{5}$$

**Lemma 3.** *For any two policies $\pi$ and $\pi'$, we have*

$$D^{\pi}_{\mathrm{TV}}(\pi, \pi') \leq \sqrt{\frac{D^{\pi}_{\mathrm{KL}}(\pi, \pi')}{2}}. \tag{6}$$

*Proof.* From Pinsker's inequality, $D_{\mathrm{TV}}(p, q) \leq \sqrt{D_{\mathrm{KL}}(p, q)/2}$ holds for any two distributions $p, q$. Combining this with Jensen's inequality, we get Eq.(6). □

### B.2 Derivation of the BC Objective Weights

This derivation uses techniques from the derivation of Theorem 3.1 in (Xu et al., 2022), adapting them to the setting of combining the offline and online data, which is considered in this paper.

As discussed in Section 4, if the offline guider policy $\pi_b$ only mimics the expert, the guider can only achieve the suboptimal performance of the expert, which leads to an excessively conservative policy of the online agent. We use $\mathcal{L}_{\mathrm{Expert}}$ to denote the objective of directly mimicking the expert behavior, which is given as follows:

$$\mathcal{L}_{\mathrm{Expert}} = \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}}\Big[ -\log \pi_b(a|s)\Big]. \tag{7}$$

In order to drive the online agent to outperform the expert, the guidance from the guider should be kept in an appropriate range. Therefore, we enable the guider to learn from both the expert and the online agent by introducing an auxiliary loss $\mathcal{L}_{\mathrm{Aux}}$:

$$\mathcal{L}_{\pi_b} = \alpha\mathcal{L}_{\mathrm{Expert}} + \mathcal{L}_{\mathrm{Aux}}, \tag{8}$$

where $\alpha > 1$ is a weight factor. The key insight of this derivation is that, challenging $d$ by maximizing $\mathcal{L}_d$ can make the learning of discriminator $d$ more robust and can get an exact form of $\mathcal{L}_{\mathrm{Aux}}$. Note that the loss of discriminator has the following form:

$$\begin{aligned}\mathcal{L}_d = \eta \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}}\Big[ -\log d(s,a,\log\pi_b)\Big] + \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{B}}\Big[ -\log\Big(1 - d(s,a,\log\pi_b)\Big)\Big] \\ - \eta \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}}\Big[ -\log\Big(1 - d(s,a,\log\pi_b)\Big)\Big].\end{aligned} \tag{9}$$

This implies that $\pi_b$ has the potential to influence $\mathcal{L}_d$, subsequently influencing the learning process of $d$. The integral form of $\mathcal{L}_d$ can be formulated as follows:

$$\begin{aligned}\mathcal{L}_d(d, \log\pi_b) = \int_{\Omega_s}\int_{\Omega_a} \Big[ & P_{\mathcal{D}}(s,a)\cdot\eta\big[-\log d(s,a,\log\pi_b(a|s))\big] \\ & + P_{\mathcal{B}}(s,a)\cdot\big[-\log\big(1 - d(s,a,\log\pi_b(a|s))\big)\big] \\ & - P_{\mathcal{D}}(s,a)\cdot\eta\big[-\log\big(1 - d(s,a,\log\pi_b(a|s))\big)\big]\Big]\mathrm{d}s\mathrm{d}a \\ \triangleq \int_{\Omega_s}\int_{\Omega_a} & F(s,a,d,\log\pi_b(a|s))\mathrm{d}s\mathrm{d}a,\end{aligned} \tag{10}$$

where $P_{\mathcal{D}}(s,a)$ and $P_{\mathcal{B}}(s,a)$ denote the probability distributions for the state-action pair $(s,a)$ within the demonstrations set $\mathcal{D}$ and the replay buffer $\mathcal{B}$, respectively. Meanwhile, $\Omega_s$ and $\Omega_a$ represent the domains for state $s$ and action $a$ along these trajectories. The functional form of $F(s,a,d,\log\pi(a|s))$ is given as:

$$\begin{aligned} F(s,a,d,\log\pi(a|s)) = & P_{\mathcal{D}}(s,a)\cdot\eta\big[-\log d(s,a,\log\pi_b(a|s))\big] \\ & + P_{\mathcal{B}}(s,a)\cdot\big[-\log\big(1-d(s,a,\log\pi_b(a|s))\big)\big] \\ & - P_{\mathcal{D}}(s,a)\cdot\eta\big[-\log\big(1-d(s,a,\log\pi_b(a|s))\big)\big]. \end{aligned} \tag{11}$$

Given that the online data in the replay buffer undergoes continuous evolution, enhancing the robustness of the discriminator becomes imperative. This can be attained through the maximization of $\mathcal{L}_d(d,\log\pi_b)$. Essentially, this leads to the subsequent formulation of the min-max optimization problem for $\mathcal{L}_d(d,\log\pi_b)$:

$$\min_d \max_\pi \mathcal{L}_d(d,\log\pi_b). \tag{12}$$

A tractable form of $\mathcal{L}_{\text{Aux}}$ can be derived by solving the inner maximization problem for $\pi_b$ in Eq. (12). According to the calculus of variations (Gelfand et al., 2000), the extrema of functional $\mathcal{L}_d(d,\log\pi_b)$ with respect to $\pi_b$ ($d$ is a given function and considered as fixed) can be obtained by solving the associate Euler-Lagrangian equation as follows:

$$\frac{\partial F}{\partial \pi_b} - \frac{\partial}{\partial s}F_{\frac{\partial\pi_b}{\partial s}} - \frac{\partial}{\partial a}F_{\frac{\partial\pi_b}{\partial a}} = 0. \tag{13}$$

The later two terms are zero since $\frac{\partial\pi_b}{\partial s}$ and $\frac{\partial\pi_b}{\partial a}$ do not appear in $F$. So we get:

$$\frac{\partial F}{\partial \pi_b} = \frac{\partial F}{\partial d}\cdot\frac{\partial d}{\partial\log\pi_b}\cdot\frac{\partial\log\pi_b}{\partial\pi_b} = 0. \tag{14}$$

Consider $\theta_b$ as the parameters of $\pi_b$, above equation also suggests that:

$$\frac{\partial F}{\partial d}\cdot\frac{\partial d}{\partial\log\pi_b}\cdot\frac{\partial\log\pi_b}{\partial\pi_b}\cdot\frac{\partial\pi_b}{\partial\theta_b} = \frac{\partial F}{\partial d}\cdot\frac{\partial d}{\partial\log\pi_b}\cdot\nabla_{\theta_b}\log\pi_b = 0. \tag{15}$$

As both $d$ and $F$ are real-valued functions, hence the same with their derivatives $\partial F/\partial d$ and $\partial d/\partial\log\pi$. If the continuity of $\partial F/\partial d$ and $\partial d/\partial\log\pi$ is satisfied, their order in Eq. (15) can be swapped since the set of real-valued continuous function is a commutative ring (Hewitt, 1948). So we get

$$\frac{\partial d}{\partial\log\pi}\cdot\frac{\partial F}{\partial d}\cdot\nabla_{\theta_b}\log\pi_b = 0, \tag{16}$$

where $d$ is determined by the outer minimization problem in Eq. (12), obtaining $\partial d/\partial\log\pi$ solely from the inner maximization problem proves to be infeasible. Thus, we consider a solution where $\frac{\partial F}{\partial d}\cdot\nabla_{\theta_b}\log\pi_b = 0$. In our specific framework, both $\mathcal{D}$ and $\mathcal{B}$ are finite, while $\Omega_s$ and $\Omega_a$ are closed and bounded. Consequently, we can establish a relaxed and tractable condition, given that the integration of $\frac{\partial F}{\partial d}\cdot\nabla_{\theta_b}\log\pi_b$ yields 0:

$$\int_{\Omega_s}\int_{\Omega_a}\frac{\partial F}{\partial d}\cdot\nabla_{\theta_b}\log\pi_b \mathrm{d}s\mathrm{d}a = 0. \tag{17}$$

Substituting Eq. (11) into Eq. (17), we obtain:

$$
\begin{aligned}
0 &= \int_{\Omega_s}\int_{\Omega_a} \frac{\partial F(s,a,d,\log\pi_b(a|s))}{\partial d(s,a,d,\log\pi_b(a|s))} \cdot \nabla_{\theta_b}\log\pi_b(a|s)\mathrm{d}s\mathrm{d}a \\
&= \int_{\Omega_s}\int_{\Omega_a} \Big[ - P_{\mathcal{D}}(s,a)\cdot\frac{\eta}{d(s,a,\log\pi_b(a|s))} \\
&\quad + P_{\mathcal{B}}(s,a)\frac{1}{1-d(s,a,\log\pi_b(a|s))} \\
&\quad - P_{\mathcal{D}}(s,a)\cdot\frac{\eta}{1-d(s,a,\log\pi_b(a|s))}\Big]\cdot\nabla_{\theta_b}\log\pi_b(a|s)\mathrm{d}s\mathrm{d}a \\
&= \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}}\Big[\frac{\eta}{d}\cdot\nabla_{\theta_b}\log\pi_b(a|s)\Big] \\
&\quad + \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{B}}\Big[\frac{1}{1-d}\cdot\nabla_{\theta_b}\log\pi_b(a|s)\Big] \\
&\quad - \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}}\Big[\frac{\eta}{1-d}\cdot\nabla_{\theta_b}\log\pi_b(a|s)\Big],
\end{aligned}
\tag{18}
$$

where $d$ denotes $d(s,a,\log\pi(a|s))$. This implies that condition (17) can be met by minimizing the auxiliary loss $\mathcal{L}_{\mathrm{Aux}}$ in the following form with respect to $\theta_b$:

$$
\mathcal{L}_{\mathrm{Aux}} = \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}}\Big[\frac{\eta}{d}\cdot\log\pi_b(a|s)\Big] + \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{B}}\Big[\frac{1}{1-d}\cdot\log\pi_b(a|s)\Big] - \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}}\Big[\frac{\eta}{1-d}\cdot\log\pi_b(a|s)\Big]. \tag{19}
$$

Hence, minimizing $\mathcal{L}_{\mathrm{Aux}}$ with respect to $\pi_b$ (make $\partial\mathcal{L}_{\mathrm{Aux}}/\partial\theta_b = 0$) satisfies the condition (17). Therefore, by adding the auxiliary loss $\mathcal{L}_{\mathrm{Aux}}$ to $\mathcal{L}_{\mathrm{Expert}}$, we obtain the final loss objective of $\pi_b$ for the offline guider policy:

$$
\mathcal{L}_{\pi_b} = \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}}\Big[-\log\pi_b(a|s)\cdot\Big(\alpha - \frac{\eta}{d(1-d)}\Big)\Big] + \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{B}}\Big[-\log\pi_b(a|s)\cdot\Big(\frac{1}{1-d}\Big)\Big]. \tag{20}
$$

The weights of $\mathcal{F}_{\mathrm{Expert}}$ and $\mathcal{G}_{\mathrm{Agent}}$ can be extracted from Eq.(20), that is:

$$
\mathcal{F}_{\mathrm{Expert}}(d) = \alpha - \frac{\eta}{d(1-d)}, \qquad \mathcal{G}_{\mathrm{Agent}}(d) = \frac{1}{1-d}, \tag{21}
$$

### B.3 Proof of Policy Improvement Bound

**Proposition 1.** *Let $\tilde{\pi}$ be a policy that satisfies Assumption 1. Then, for policy $\hat{\pi}$, we have*

$$
J_R(\hat{\pi}) - J_R(\tilde{\pi}) \geq (1-\gamma)^{-1}\xi - (1-\gamma)^{-1}\epsilon_{R,\tilde{\pi}}\sqrt{2D_{\mathrm{KL}}^{\hat{\pi}}(\hat{\pi},\pi_b)}, \tag{22}
$$

*where* $\epsilon_{R,\tilde{\pi}} = \max_{s,a}|A_R^{\tilde{\pi}}(s,a)|$.

*Proof.* Starting from Lemma 1:

$$\begin{aligned}
J_R(\hat{\pi}) - J_R(\tilde{\pi}) &= \frac{1}{1-\gamma}\mathbb{E}_{(s,a)\sim\pi d_{\pi,\mu}}\big[A_R^{\tilde{\pi}}(s,a)\big] \\
&= \frac{1}{1-\gamma}\sum_s d^{\hat{\pi}}(s)\sum_a \pi_b(s,a)A_R^{\tilde{\pi}}(s,a) \\
&\quad + \frac{1}{1-\gamma}\sum_s d^{\hat{\pi}}(s)\sum_a \big(\hat{\pi}(s,a)-\pi_b(s,a)\big)A_R^{\tilde{\pi}}(s,a) \\
&\overset{(a)}{\geq} \frac{1}{1-\gamma}\xi + \frac{1}{1-\gamma}\sum_s d^{\hat{\pi}}(s)\sum_a \big(\hat{\pi}(s,a)-\pi_b(s,a)\big)A_R^{\tilde{\pi}}(s,a) \\
&\overset{(b)}{\geq} \frac{1}{1-\gamma}\xi - \frac{1}{1-\gamma}\epsilon_{R,\tilde{\pi}} 2D_{\mathrm{TV}}^{\hat{\pi}}(\hat{\pi},\pi_b) \\
&\overset{(c)}{\geq} \frac{1}{1-\gamma}\xi - \frac{1}{1-\gamma}\epsilon_{R,\tilde{\pi}}\sqrt{2D_{\mathrm{KL}}^{\hat{\pi}}(\hat{\pi},\pi_b)},
\end{aligned} \tag{23}$$

where $(a)$ follows from the premise that $\tilde{\pi}$ satisfies Assumption 1, $(b)$ is obtained by denoting $\epsilon_{R,\tilde{\pi}} = \max_{s,a}|A_R^{\tilde{\pi}}(s,a)|$, and $(c)$ follows Lemma 3. □

**Proposition 2.** *For policy $\hat{\pi}$ and any policy $\tilde{\pi}$, we have*

$$J_R(\hat{\pi}) - J_R(\tilde{\pi}) \geq -\frac{3R_{\max}}{2(1-\gamma)^2}\sqrt{2D_{\mathrm{KL}}^{\max}(\hat{\pi},\tilde{\pi})}, \tag{24}$$

*where $R_{\max} = \max_{s,a}|R(s,a)|$.*

*Proof.*

$$\begin{aligned}
J_R(\hat{\pi}) - J_R(\tilde{\pi}) &= \frac{1}{1-\gamma}\left(\sum_{s,a} d^{\hat{\pi}}(s)\hat{\pi}(s,a)R(s,a) - \sum_{s,a} d^{\tilde{\pi}}(s)\tilde{\pi}(s,a)R(s,a)\right) \\
&= \frac{1}{1-\gamma}\left(\sum_{s,a} d^{\hat{\pi}}(s)\hat{\pi}(s,a)R(s,a) - \sum_{s,a} d^{\tilde{\pi}}(s)\hat{\pi}(s,a)R(s,a)\right) \\
&\quad + \frac{1}{1-\gamma}\left(\sum_{s,a} d^{\tilde{\pi}}(s)\hat{\pi}(s,a)R(s,a) - \sum_{s,a} d^{\tilde{\pi}}(s)\tilde{\pi}(s,a)R(s,a)\right) \\
&\geq -\frac{1}{1-\gamma}R_{\max}\|d^{\tilde{\pi}} - d^{\hat{\pi}}\|_1 - \frac{1}{1-\gamma}R_{\max}D_{\mathrm{TV}}^{\max}(\tilde{\pi},\hat{\pi}) \\
&\overset{(a)}{\geq} -\frac{1}{1-\gamma}R_{\max}\frac{2\gamma}{1-\gamma}D_{\mathrm{TV}}^{\max}(\hat{\pi},\tilde{\pi}) - \frac{1}{1-\gamma}R_{\max}D_{\mathrm{TV}}^{\max}(\hat{\pi},\tilde{\pi}) \\
&\geq -\frac{3R_{\max}}{(1-\gamma)^2}D_{\mathrm{TV}}^{\max}(\hat{\pi},\tilde{\pi}) \\
&\overset{(b)}{\geq} -\frac{3R_{\max}}{2(1-\gamma)^2}\sqrt{2D_{\mathrm{KL}}^{\max}(\hat{\pi},\tilde{\pi})},
\end{aligned} \tag{25}$$

where $(a)$ follows Lemma 2, and $(b)$ follows Lemma 3. □

# C Implementation Details

All experiments in this paper are implemented with PyTorch (Paszke et al., 2019) and executed on NVIDIA A30 GPUs. All the runs in experiments use 5 random seed.

## C.1 MuJoCo Simulation Experiment

In MuJoCo tasks, we use a two layered $(64 \times 64)$ fully connected neural network with the *tanh* activation function to parameterize our policy and value function. The policies of the guider and the

agent have the same structure. The common parameters of all algorithms in the MuJoCo environment are shown in Table 1.

Table 1: Common hyperparameters used in all algorithms in MuJoCo

| Hyperparameter | Hopper | Walker2d | HalfCheetah | Ant |
|---|---|---|---|---|
| learning rate | 0.0003 | 0.0003 | 0.0003 | 0.0003 |
| gamma | 0.995 | 0.995 | 0.995 | 0.995 |
| clip | 0.2 | 0.2 | 0.2 | 0.2 |
| optimizer | Adam | Adam | Adam | Adam |
| lambda | 0.97 | 0.97 | 0.97 | 0.97 |
| epoch | 10 | 10 | 10 | 10 |
| batch size | 2048 | 4096 | 4096 | 4096 |
| mini batch size | 256 | 256 | 256 | 256 |
| gradient clip norm | 10.0 | 10.0 | 10.0 | 10.0 |

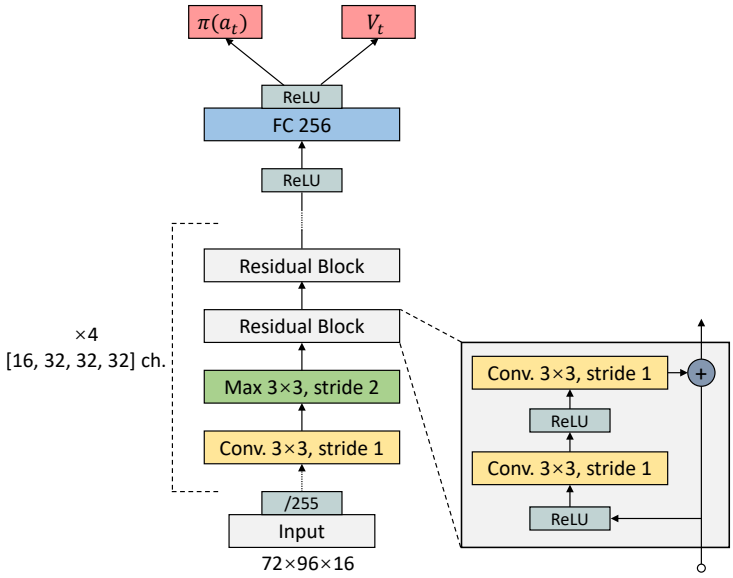


Figure 1: Architecture of the policy and value function networks used in the GRF environment.

### C.2 Google Research Football Experiment

**Problem Setting** In the football experiments, the playing field is restricted to the frontcourt of the left team, which is also the backcourt of the right team. We use the Football Academy, which includes a diverse set of scenarios of varying difficulty.

**Reward Setting** In the sparse reward setting, there is only a +1 reward when the players scores a goal. In the dense reward setting, there are non-zero rewards when the players reach the specific region.

**Observation Representation** We use the default Super Mini Map (SMM) representation, which consists of four $72 \times 96$ matrices encoding information about the home team, the away team, the ball, and the active player, respectively. The encoding is binary, representing whether there is a player or ball in the corresponding coordinate. We use a sequence of 4 consecutive observations as input, thus the channels of input is 16. The architecture of network is shown in Figure 1.

### C.3 AirSim Simulation Experiment

**Problem Setting** This environment is built on Unreal Engine 4, which is a professional game engine. In the AirSim task, the drone needs to avoid obstacles and reach the designated endpoint. If a collision occurs, the task fails. Moreover, there are several checkpoint levels on the way to the endpoint.

**Reward Setting** In the sparse reward setting, the agent only receives a reward when it reaches the checkpoint levels. In the dense reward setting, there will be a penalty if the speed of the drone is too slow, or the collision occurs. The drone also can receive an additional reward which is related to the forward speed in the dense reward setting.

**Observation Representation** As shown in Figure 2, the drone uses the depth images from its front camera as the observation, which has the shape of $(72 \times 128)$. As the same in the GRF tasks, we use a sequence of 4 consecutive observations as input. The architecture of network is similar to the one in GRF tasks, but a GRU layer is added after flattening.

Figure 2: Demo of the AirSim task. The image in left-bottom is the observation of the drone.

## D Additional Results

### D.1 Comparison of Run Time

We evaluate the run time of training HYPO and other baselines for 6M training steps in Hopper environment, while keeping all the external factors the same (e.g., the evaluation, batch size). All the run time experiments are executed on NVIDIA A30 GPUs. For a fair comparison, we use the same policy network size and the same discriminator network size. The results are shown in Figure 3. As expected, the run time of HYPO is only slightly more than GAIL due to the additional cost to train an offline guider.

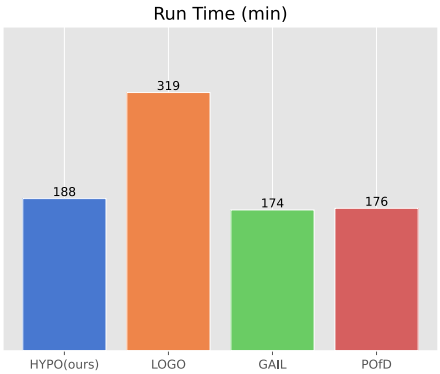


Figure 3: Run time comparison of the main baselines.

### D.2 Complete Results in MuJoCo

In order to present the detailed experimental results and a digital comparison in MuJoCo environments, we give the the original un-scaled results in Table 2.

Table 2: Detailed results in MuJoCo environments

| Task | HYPO(ours) | LOGO | GAIL | POfD | PPO |
|---|---|---|---|---|---|
| Hopper | **3163.6±95.7** | 627.0±908.8 | 1245.4±1058.2 | 804.2±962.2 | 8.4±9.2 |
| Walker2d | **4200.0±118.7** | 1778.1±839.7 | 2157.2±554.3 | 2296.0±343.0 | -6.2±6.4 |
| HalfCheetah | **5105.4±162.7** | 2746.2±1093.3 | 1359.7±161.8 | 2178.9±696.5 | 247.74±120.2 |
| Ant | **5096.7±228.5** | 3194.0±722.0 | 1982.4±403.8 | 2479.8±1636.0 | -862.1±690.1 |

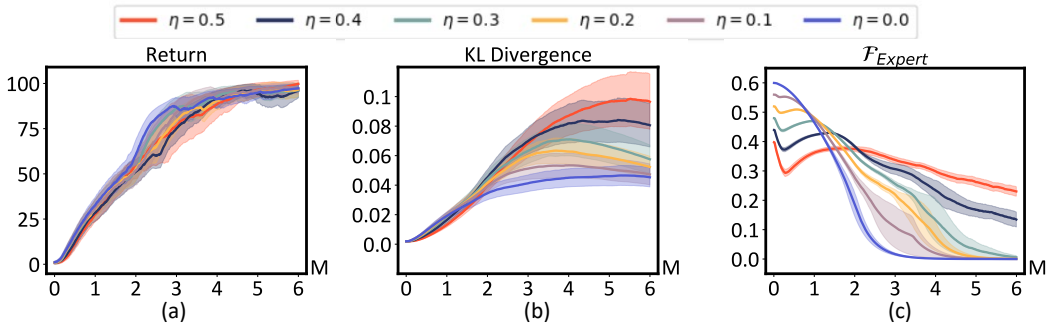


Figure 4: The sensitivity of $\eta$ in Hopper environment.

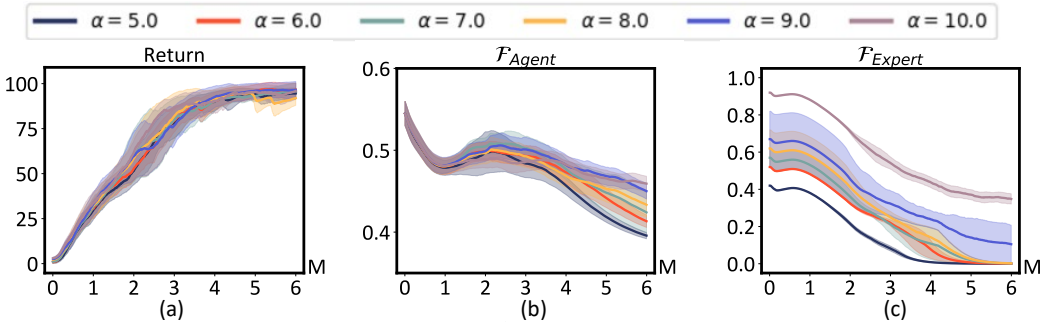


Figure 5: The sensitivity of $\alpha$ in Hopper environment.

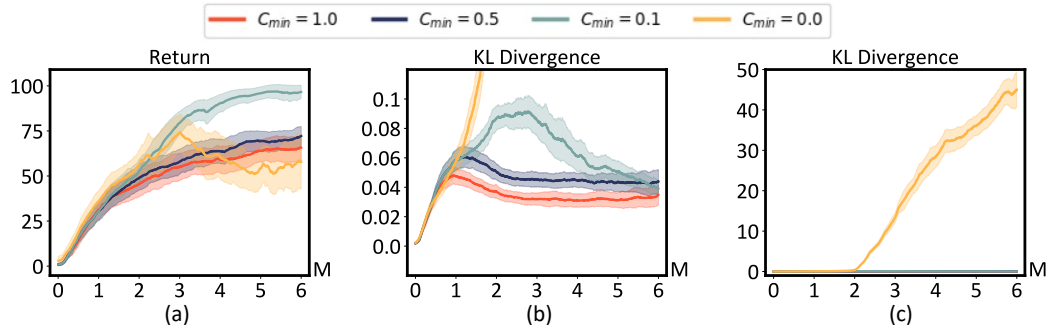


Figure 6: The sensitivity of $C$ in Hopper environment.

## E Sensitivity Analysis

There are three main hyperparameters in our method HYPO: the positive class prior $\eta$, the weight factor $\alpha$, and the regularization coefficient $C$. During the development of our algorithm, we did not tune these hyperparameters much, as they are easy to estimate. For example, $\eta$ is set to 0.5 in most of the PU-Learning tasks, while we set it to [0.2, 0.8] due to the performance improving of the online agent policy. As for $\alpha$, we need to ensure $\mathcal{F}_{\text{Expert}} > 0$ whith the domains of $\eta$ and $d$. Therefore, the value of $\alpha$ must be greater than 5.0. Finally $C$, which can be set like the entropy coefficient, needs to ensure that the losses are within the similar order of magnitude.

First, Figure 4 illustrates the impact of parameter $\eta$ on our method. In this figure $\eta$ indicates the initial value, and linearly increases to the end value of $1 - \eta$. $\eta$ primarily affects the updates of the guider, since it determines the weight of the positive samples that come from the online agent. But the online agent performance is rarely affected by the value of $\eta$ (Figure 4a). Second, Figure 5 illustrates the impact of parameter $\alpha$ on HYPO. Similar to $\eta$, $\alpha$ also primarily affects the updates of the guider. However, the offline imitation policy can dynamically adjust these two weights of $\mathcal{F}$ and $\mathcal{G}$, which means that bigger $\mathcal{F}$ can lead to bigger $\mathcal{G}$ (Figure 5b, 5c). In this way, the weight factor $\alpha$ rarely affect the learning of the guider. Therefore, the agent's performance is not sensitive to $\alpha$ (Figure 5a). Finally, Figure 6 illustrates the impact of parameter $C$ on HYPO. In this figure the $C_{min}$ indicates the end value of $C$ (all decay from 1.0). If $C$ has a large value, such as 0.5 or 1.0 (not decaying), the performance of the online agent can be affected due to the overly restrictions (Figure 6a). However, $C$ also can not be too small, such as 0. In this case, the KL between the offline guider and the online agent would be very large, leading to collapse in the agent's performance.

## F Performance of the LOGO Baseline

The experimental results of the LOGO algorithm presented in our paper were directly conducted using the publicly available LOGO source code, without any modifications to its components. Our reproduced results align with the outcomes reported in their paper. The reason for the comparatively weaker performance of LOGO in the figures of our paper is attributed to the limitations imposed by the training samples. In the original LOGO paper, the x-axis reaches 1e7, which consumes an enormous amount of samples. However, in our experiments, such as the Hopper environment, we only considered a maximum of 6M (x-axis is 1e6) samples.

Furthermore, although both LOGO and HYPO use suboptimal samples, LOGO's suboptimal samples reach over 60% of the performance of the optimal expert (e.g., in the Walker and HalfCheetah environments). In contrast, HYPO requires suboptimal samples that achieve just 20% or 10% of the optimal expert's performance, indicating that HYPO imposes less stringent requirements on expert trajectories.

## References


Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In *International conference on machine learning*, pp. 22–31. PMLR, 2017.

Gelfand, I. M., Silverman, R. A., et al. *Calculus of variations*. Courier Corporation, 2000.

Hewitt, E. Rings of real-valued continuous functions. i. *Transactions of the American Mathematical Society*, 64(1):45–99, 1948.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Xu, H., Zhan, X., Yin, H., and Qin, H. Discriminator-weighted offline imitation learning from suboptimal demonstrations. In *International Conference on Machine Learning*, pp. 24725–24742. PMLR, 2022.