A. Training details

For the training we used 14 parallel environments and we compute the gradients using the Adam optimizer (Kingma & Ba, 2014) with fixed learning rate of 10^{-5}. In the Animal AI Olympics environment the agent perceives the environment through a 84 by 84 RGB pixels observations in a stack of 4. At each time-step the agent is allowed to take one of nine actions. We use the network architecture proposed in (Kostrikov, 2018) which includes a gated recurrent unit (GRU) (Cho et al., 2014) with a hidden layer of size 256. We ran the experiments on machines with 32 CPUs and 3 GPUs, model GeForce RTX 2080 Ti. The experiments for the Animal-AI Olympics environment where carried out with the following hyperparameters.

Table 1. Model and PPO Hyperparameters

Parameter	Value
clip-param	0.15
gamma	0.998
frame-skip	2
frame-stack	4
num-steps	1000
num-mini-batch	6
entropy-coef	0.02
value-loss-coef	0.1
num-processes	14
learning rate	1e-5
eps (RMSprop optimizer epsilon)	1e-5
alpha (RMSprop optimizer apha)	0.99
gae-lambda	0.95
max-grad-norm	0.5
ppo-epoch	4

Different hyperparameters were used for the experiments in the ReacherPyBulletEnv-v0 and LunarLander-v2 environments. Table 2 shows the hyperparameters used for ReacherPyBulletEnv-v0. When training PPO+D, PPO+BC and vanilla PPO for LunarLander-v2, only slight changes were made in comparison to the ReacherPyBulletEnv-v0 hyperparameters: the number of processes (num-processes) changed from 64 to 32 and the entropy coefficient (entropy-coef) changed from 0.02 to 0.01. For both these tasks the policy is not recurrent as for the Animal AI Olympics tasks.

Table 2. Model and PPO Hyperparameters (ReacherPyBulletEnv-v0)

Parameter	Value
clip-param	0.2
gamma	0.99
frame-skip	1
frame-stack	1
num-steps	2048
num-mini-batch	32
entropy-coef	0.02
value-loss-coef	0.3
num-processes	64
learning rate	2e-4
eps (RMSprop optimizer epsilon)	1e-5
alpha (RMSprop optimizer apha)	0.99
gae-lambda	0.95
max-grad-norm	0.5
ppo-epoch	10

When training PPO+BC the loss combines the PPO loss and the behavior cloning loss in the following way:

$$L = \begin{cases} L_{PPO}, & \text{if } \tau_i \sim \mathcal{D} \\ L_{BC}, & \text{if } \tau_i \sim \text{Env} \end{cases}$$

where L_{BC} is defined as the cross-entropy loss

$$L_{CE} = -\sum_{t=1}^{n} a_t \log(\pi(a_t|s_t))$$

when the action space is discrete, and as the mean squared error:

$$L_{MSE} = \frac{1}{n}\sum_{t=1}^{n}(\pi(a_t|s_t) - a_t)^2$$

when the action space is continuous. Note that in the above equations a_t refers to the action taken in the demonstration.

For the Animal AI Olympics tasks we performed no hyperparameter search over the replay ratios ϕ and ρ but set them to a reasonable number. We found other configurations of these parameters to be sometimes more efficient in the training, such for example setting $\rho = 0.5$ and $\phi = 0.0$ in the task "One box easy". The parameters we ran all the experiments with have been chosen because they allow to solve all of the experiments with one demonstration. We did run a hyperparameter search for the parameter ρ for the LunarLander-v2 and ReacherPyBulletEnv-v0 task for both PPO+D and PPO+BC. The rest of the hyperparamters were adopted from Schulman et al. (2017).

In computing the probability of a trajectory to be replayed $P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$, $\alpha = 10$. The total buffer size is $|\mathcal{D}| =$

51 with $|\mathcal{D}_V|_0 = 50$ plus the human generated trajectory. $|\mathcal{D}_R|_0 = 51$ meaning once the agent collects 50 successful trajectories, new successful trajectories overwrite old ones, following a FIFO strategy and no trajectory is replayed from the value-buffer. The implementation used is based on the repository (Kostrikov, 2018). On our infrastructure we could train at approximately a speed of 1.3 millions frames per hour. The code, pre-trained models, data-set of arenas used for training are available at https://doi.org/10.6084/m9.figshare.13853039.v1

B. Hyperparameters ablation

In this section we present the results of four different experiments on a variation of the "One box easy task" where the agent position does not change across episodes and it is shared with the demonstration. We test on this variation of the task because it is one of the simplest problems we can use to test PPO+D performance. We only perform the ablation study on ρ because ϕ is harder to test: it is indispensable for solving difficult tasks but it can slow down the performance on easy tasks. This being an easy task, the results obtained, do not provide any insights on the effect of ϕ in harder problems (as shown in Figure 5). The following figure shows the performance of the PPO+D algorithm where the ρ parameter is changed and $\phi = 0$. Interestingly we observe that, among the values chosen, the performance peaks for $\rho = 0.3$. We hypothesize that lower ρ values have worse performance because the interval between demo replays is so large that allows the network to forget the optimal policy learned with the demonstrations. On the other side, higher values of ρ are even more counterproductive as they prevent the agent from learning from its own experience, most critically learning what not to do.

Our implementation of GAIL based on (Li et al., 2017) was trained with the following hyperparameters besides the PPO parameters in Table 1.

C. GAIL test

To verify the correctness of our GAIL implementation we use for the experiments in Figure 5 we test it on a simple task in the Animal-AI environment. The task is shown in Figure 6, it consists in collecting green food of random size and position.

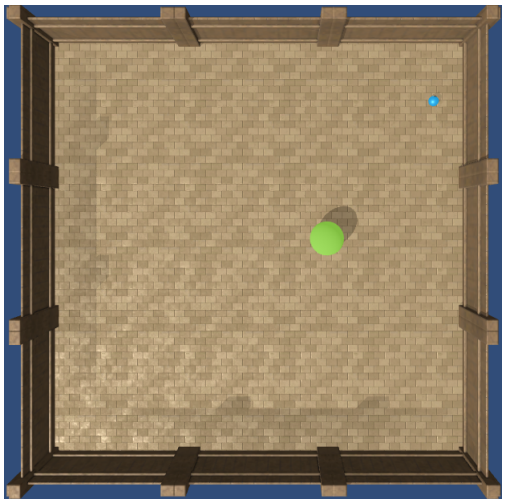


Figure 6. **Food collection task.** In this task the the agent is spawned into the arena with one green ball. The green food size and position are set randomly at the beginning of each episode. The episode ends when the green food is collected.

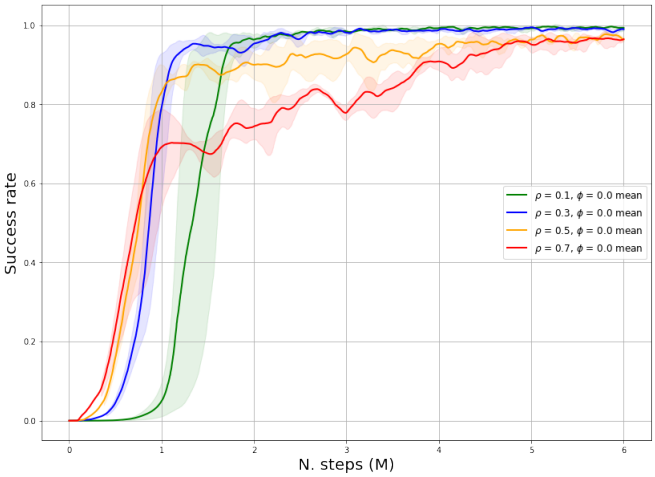


Figure 7. **Ablation study** Performance for PPO+D with $\rho = 0.1, \phi = 0.0$, $\rho = 0.3, \phi = 0.0$, $\rho = 0.5, \phi = 0.0$ and $\rho = 0.5, \phi = 0.0$ and PPO+D with $\rho = 0.7, \phi = 0.0$, on a variation of the "One box easy" task were the initial position of the agent is fixed. The curves represent the mean, min and max performance for each of the baselines across 3 different seeds.

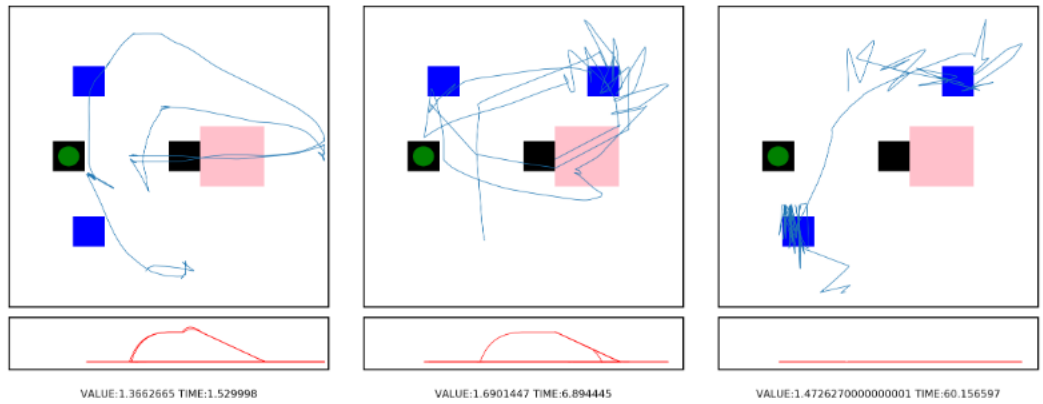
Table 3. GAIL Hyperparameters

Parameter	Value
scaling-factor	0.001
gail-epoch	0.4
gail-batch-size	200

Table 4. Performance on the "Food collection task"

Method	Avg. Success rate	Std.
GAIL	0.997	0.045
BC	0.617	0.487

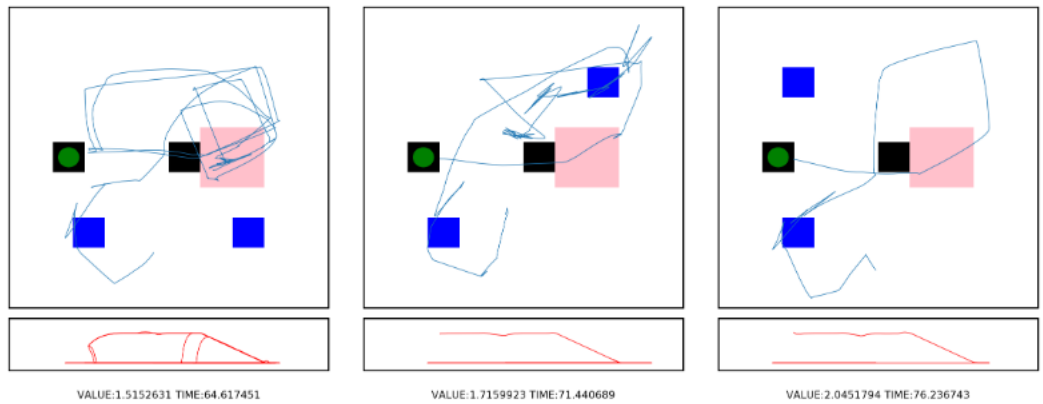
D. Analysis of the effect of the value-buffer



(a) Learning about climbing up the ramp.

(b) Learning about pushing boxes.

(c) Focusing on boxes.



(d) Learning about jumping on boxes.

(e) Learning how to solve the task pushing two boxes.

(f) Learning how to solve the task with only one box.

Figure 8. **Sub-behaviors.** Trajectories the agent played on the task "Two boxes easy". In each of the figure the upper part shows the movements of the agent on the X-Y plane while the lower part shows the movement on the X-Z plane. The images are ordered by the time they were executed in the training in millions of frames.

The value-buffer experience replay creates an incremental curriculum for the agent to learn, keeping different trajectories that achieved an high maximum value in the buffer incentives the agent to combine these different sub-behaviors e.g. pushing the blocks and going up the ramp.