Sample-Efficient Imitation Learning via Generative Adversarial Nets

—

Supplementary Material

Lionel Blondé
lionel.blonde@etu.unige.ch
University of Geneva, Switzerland

Alexandros Kalousis
alexandros.kalousis@hesge.ch
Geneva School of Business Administration, HES-SO

1 Studied environments

The environments we dealt with were provided through the OpenAI Gym (Brockman et al., 2016) API, building on the MuJoCo physics engine (Todorov et al., 2012), to model physical interactive scenarios between an agent and the environment she is thrown into. The control tasks modelled by the environments involve locomotion tasks as well as tasks in which the agent must reach and remain in a state of dynamic balance.

Environment	s DoFs	a DoFs
InvertedPendulum-v2	4	1
InvertedDoublePendulum-v2	11	1
Reacher-v2	11	2
Hopper-v2	11	3
Walker2d-v2	17	6

Figure 1: Degrees of freedom (DoF) of the considered MuJoCo simulated environments. DoFs of both continuous action and state spaces are presented, for the studied physical control tasks. Actions spaces are bounded along every dimension, while the state spaces are unbounded.

2 Reward function variants

The reward is defined as the negative of the generator loss. As for the latter, the former can be stated in two variants, the saturating version and the non-saturating version, respectively

$$r_\phi^\frown(s_t, a_t) = -\log(1 - D_\phi(s_t, a_t)) \quad (1)$$

$$r_\phi^\smile(s_t, a_t) = \log D_\phi(s_t, a_t) \quad (2)$$

The non-saturating alternative is recommended in the original GAN paper as well as in (Fedus et al., 2017) more recently, as the generator loss suffers from vanishing gradients only in areas where the generated samples are already close to the real data. GAIL relies on policy optimization to update the generator, which makes this vanishing gradient argument vacuous. Besides, in the context of simulated locomotion environments, the saturated version proved to prevail in our experiments, as our agents were unable to overcome the extremely low rewards incurred early in training when using the non-saturating rewards. With the saturated version, signals obtained in early failure cases were close to zero, which was more numerically forgiving for our agents to kick off.

3 Experimental setup

Both our algorithm and GAIL baseline model implement the MPI interface: each experiment has been launched concurrently with X parallel workers (each with its own random seed), each having its own interaction with the environment, its own replay buffer, its own optimisers and its own network updates. However, every iteration and for a given network, the gradients of the X Adam (Kingma and Ba, 2014) optimisers are pulled together, averaged, and a unique average gradient is distributed to the worker for immediate usage. In the experiments reported in this paper, we used 4 parallel workers.

Our experiments have all been conducted on a single 16-core CPU workstation (AMD Ryzen Threadripper® 1950X CPU).

4 Hyperparameters settings

In our training procedure, we adopted an alternating scheme consisting in performing 3 training iterations of the actor-critic architecture for one training iteration of the synthetic reward, in line with common practices in the GAN literature (the actor-critic acts as generator, while the synthetic reward plays the role of discriminator). This training pattern applies for both the GAIL baseline and our algorithm, Sam.

Hyperparameter	Value
# MPI workers	4
policy # layers	2
policy layer widths	$(100, 100)$
policy hidden activations	tanh
discriminator # layers	2
discriminator layer widths	$(100, 100)$
discriminator hidden activations	leaky ReLU
discount factor γ	0.995
generator training steps	3
discriminator training steps	1
non-saturating reward?	false
entropy regularization coefficient λ	0.
gradient penalty coefficient (Gulrajani et al., 2017)	10.
one-sided label smoothing	true
# interactions per iteration	1024
minibatch size	128
normalize observations?	true

Figure 2: Hyperparameters used to train GAIL agents.

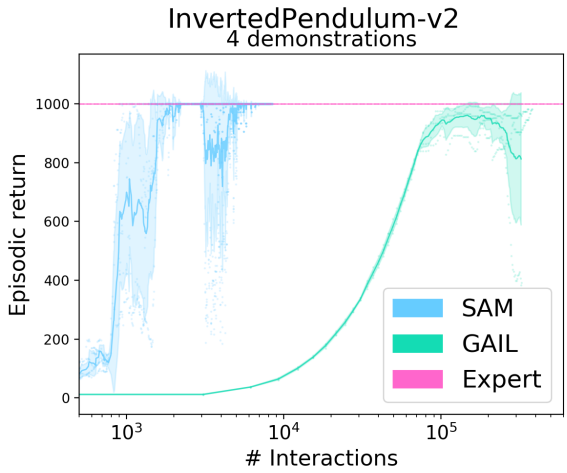
As supported by the ending discussion of the GAIL paper, performing a behavioral cloning (Pomerleau, 1989, 1990) pre-training step to warm-start GAIL can potentially yield expert-like policies in fewer number of ensuing GAIL training iterations. It is especially appealing in so far as the behavioral cloning agent does not interact with the environment at all while training. We therefore intended to precede the training of our experiments (for GAIL and SAM) with a behavioral cloning pre-training phase. However, although the previous training pipeline enables a reduction of training iterations for GAIL, we did not witness a consistent benefit for SAM in our preliminary experiments. Our proposed explanation of this phenomenon is that by pre-training both policy and critic individually as regression problems over the expert demonstrations dataset, we hinder the entanglement of the policy and critic training procedures exploited in SAM. We believe that by adopting a more elaborate pre-training procedure, we will be able to overcome this issue, and therefore leave further exploration for future work.

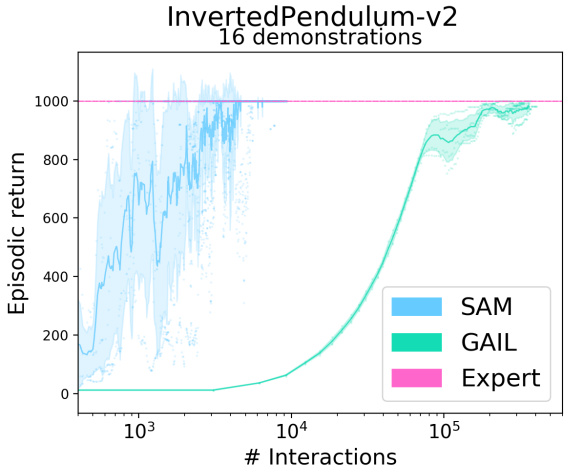
5 Enhanced plots

The following figures show the performance comparison between SAM and GAIL in terms of episodic return. The horizontal axis depicts, in logarithmic scale, the number of interactions with the environment. While there is no ambiguity for GAIL, we used the unperturbed SAM policy μ_θ (without parameter noise and additive action noise) to collect those returns during a per-iteration evaluation phase. The figures show that

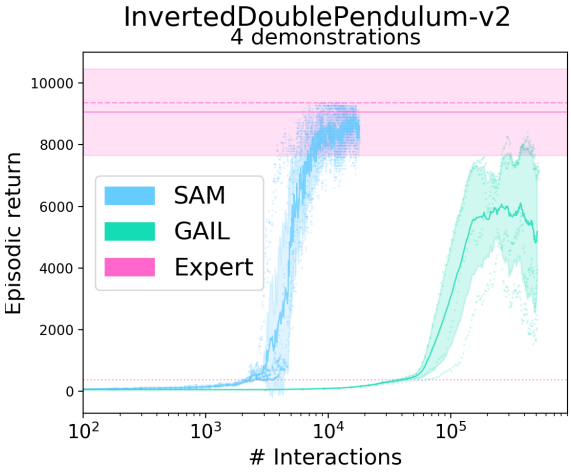
Hyperparameter	Value
# MPI workers	4
policy # layers	2
policy layer widths	$(64, 64)$
policy hidden activations	leaky ReLU
policy layer normalisation (Ba et al., 2016)	true
policy output activation	tanh
critic # layers	2
critic layer widths	$(64, 64)$
critic hidden activations	leaky ReLU
critic layer normalisation	true
discriminator # layers	2
discriminator layer widths	$(64, 64)$
discriminator hidden activations	leaky ReLU
discount factor γ	0.99
generator training steps	3
discriminator training steps	1
non-saturating reward?	false
entropy regularization coefficient	0.
gradient penalty coefficient (Gulrajani et al., 2017)	10.
one-sided label smoothing	true
# interactions per iteration	4
minibatch size	32
# training steps per iteration	20
replay buffer size	100K
normalise observations?	true
normalise returns?	true
POP-ART? (van Hasselt et al., 2016)	true
reward scaling factor	1.
critic weight decay coefficient ν	0.001
critic 1-step TD loss coefficient	1
critic n-step TD loss coefficient	1
TD lookahead length n	96
adaptive parameter noise for π_θ	0.2
Ornstein-Uhlenbeck additive noise	0.2

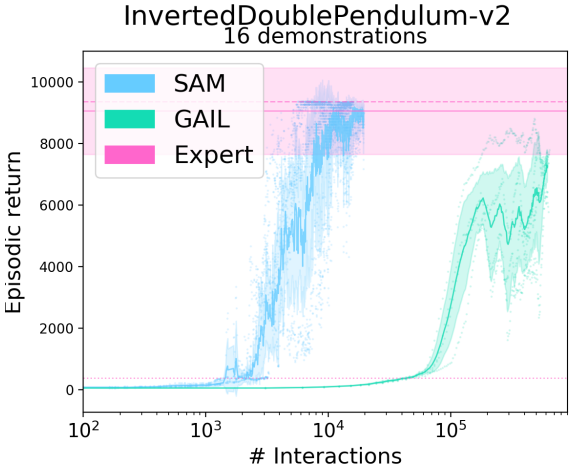
Figure 3: Hyperparameters used to train SAM agents.





our method has a considerably better sample-efficiency than GAIL in various continuous control tasks, often by several orders of magnitude. We use scatter plots to visualise every episodic return, for every random seed. Solid blue and green lines represent the mean episodic return across the random seeds for the given number of interactions. The filled areas are confidence intervals around the solid lines, corresponding to a fixed fraction of the standard deviation around the mean for the given number of interactions. Every item coloured in red relates to the expert performance, for a given environment. The solid red line corresponds to the mean episodic return of the demonstrations present in the expert dataset associated with the given environment. The filled red region is a trust region whose width is equal to the standard deviation of returns in the expert dataset. The dotted line depicts the minimum return in the demonstration dataset while the dashed line represents the maximum. Having statistics about the demonstration datasets is particularly insightful when evaluating the results of experiments dealing with few demonstrations.





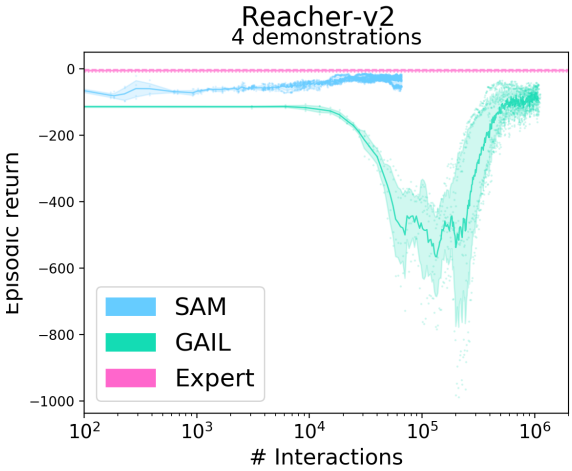
References

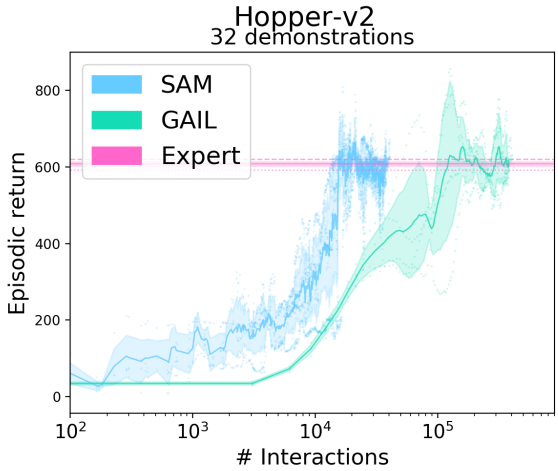
Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer Normalization.

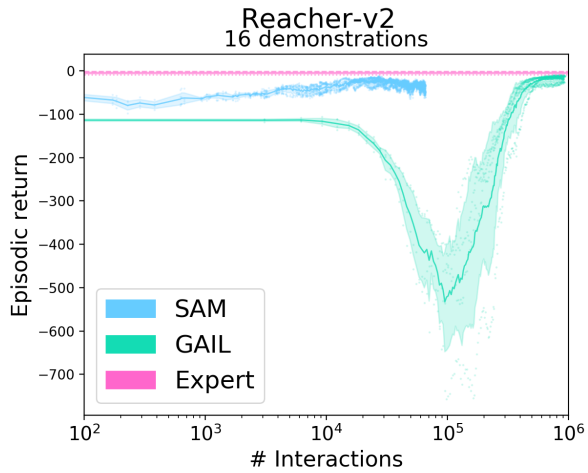
Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). OpenAI Gym.

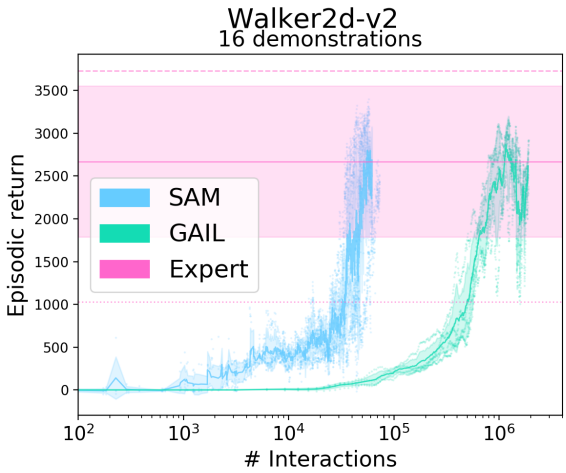
Fedus, W., Rosca, M., Lakshminarayanan, B., Dai, A. M., Mohamed, S., and Goodfellow, I. (2017). Many Paths to Equilibrium: GANs Do Not Need to Decrease a Divergence At Every Step.

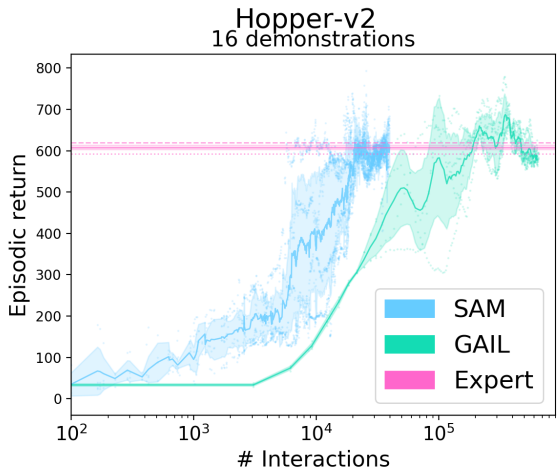
Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017). Improved Training of Wasserstein GANs. In *Neural Information Processing Systems (NIPS)*.

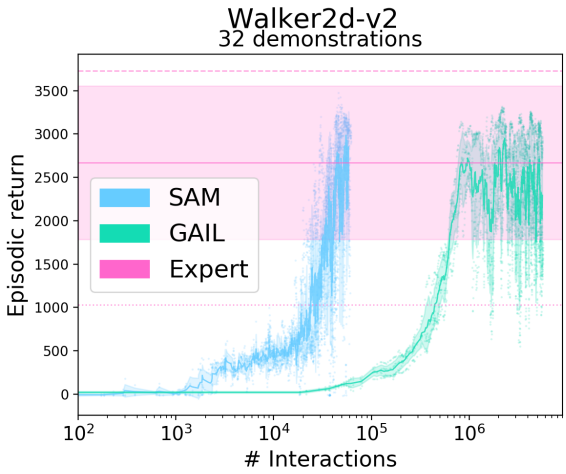

Reacher-v2
4 demonstrations
Episodic return
Interactions
SAM
GAIL
Expert


Hopper-v2
32 demonstrations
Episodic return
Interactions
SAM
GAIL
Expert


Reacher-v2
16 demonstrations
Episodic return
Interactions
SAM
GAIL
Expert


Walker2d-v2
16 demonstrations
Episodic return
Interactions
SAM
GAIL
Expert


Hopper-v2
16 demonstrations
Episodic return
Interactions
SAM
GAIL
Expert


Walker2d-v2
32 demonstrations
Episodic return
Interactions
SAM
GAIL
Expert

Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization.

Pomerleau, D. (1989). ALVINN: An Autonomous Land Vehicle in a Neural Network. In *Neural Information Processing Systems (NIPS)*, pages 305–313.

Pomerleau, D. (1990). Rapidly Adapting Artificial Neural Networks for Autonomous Navigation. In *Neural Information Processing Systems (NIPS)*, pages 429–435.

Todorov, E., Erez, T., and Tassa, Y. (2012). MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5026–5033.

van Hasselt, H., Guez, A., Hessel, M., Mnih, V., and Silver, D. (2016). Learning values across many orders of magnitude.