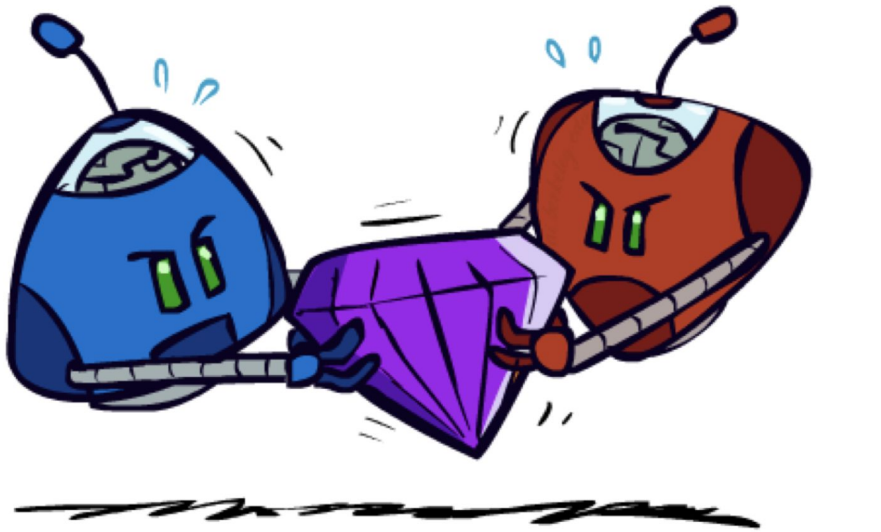
End-to-end Differentiable Adversarial Imitation Learning

by Nir Baram, Oron Anschel, Itai Caspi, Shie Mannor
NIPS 2017



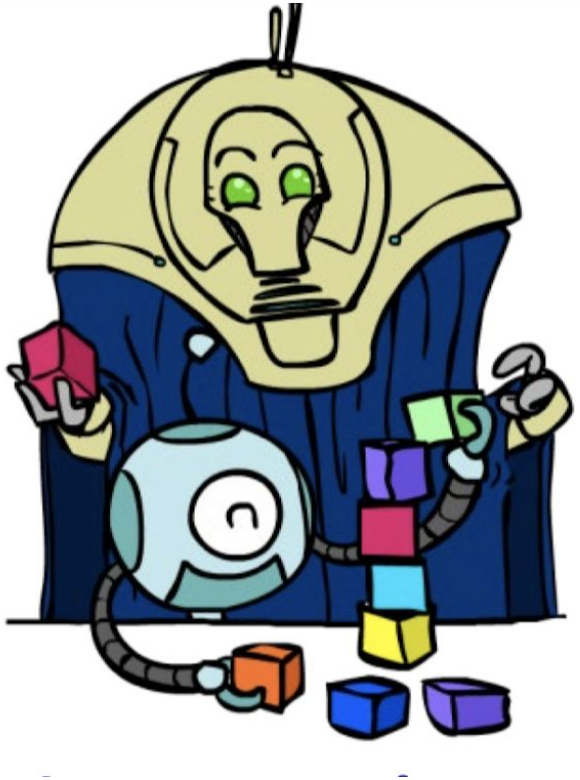
1. Warm-up
2. GAIL limitations
3. Towards full differentiability

Presented by Léonard Boussioux, Visiting researcher at Mila

Imitation learning :
two traditional approaches

Behavioral cloning

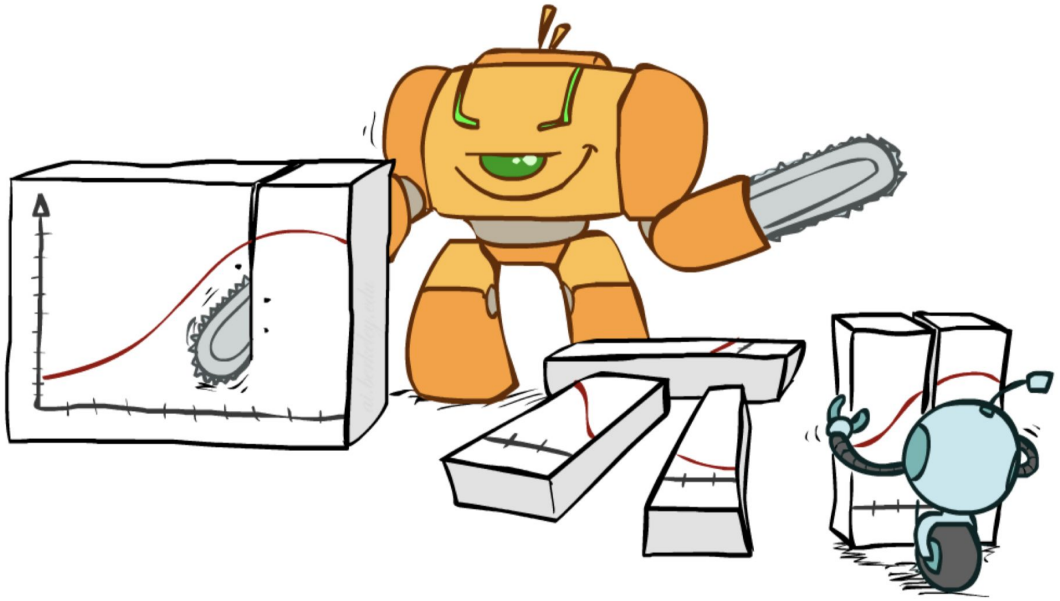
→ like Supervised Learning



Inverse Reinforcement Learning

→ Recover the expert's reward function

→ RL with this reward function



Generative Adversarial Imitation Learning

- Inspired by GANs
- The generator is the policy we train.
- The discriminator has to distinguish (state, action) pairs from the expert or the policy.

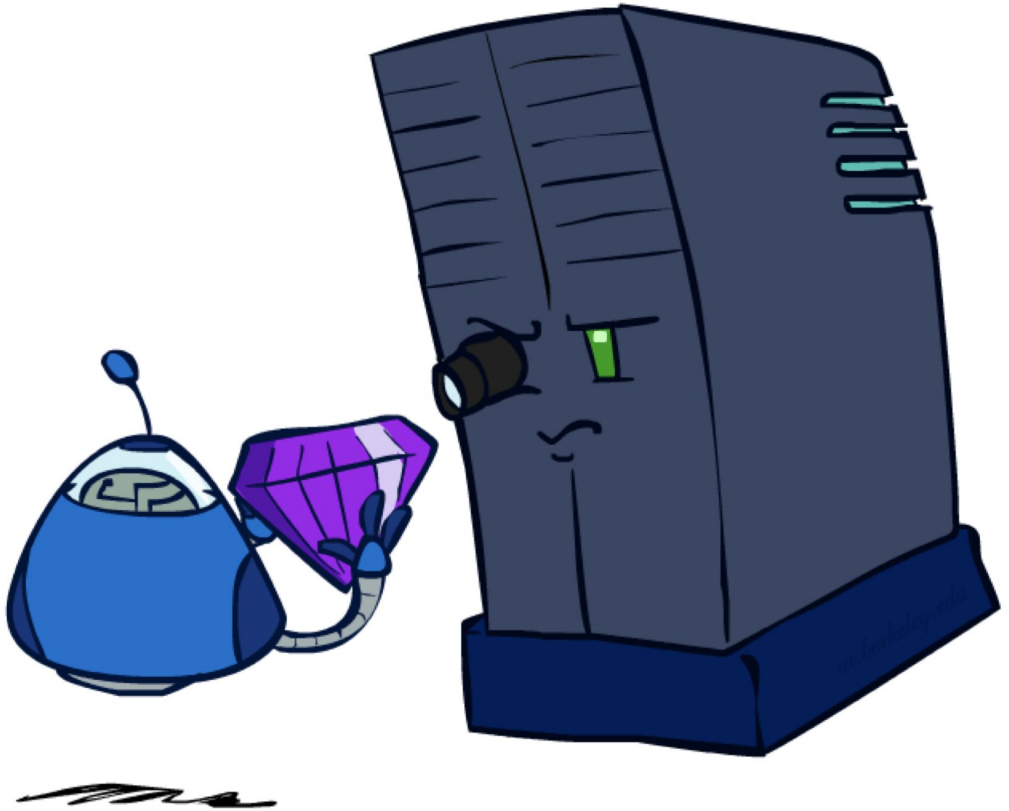
PROS

- Able to imitate complex behaviors in large, high-dimensional environments
- Generally efficient in terms of expert data.

CONS

- Not particularly sample efficient in terms of environment interactions during training

→ Needs more interactions than model-based.



Generative Adversarial Imitation Learning

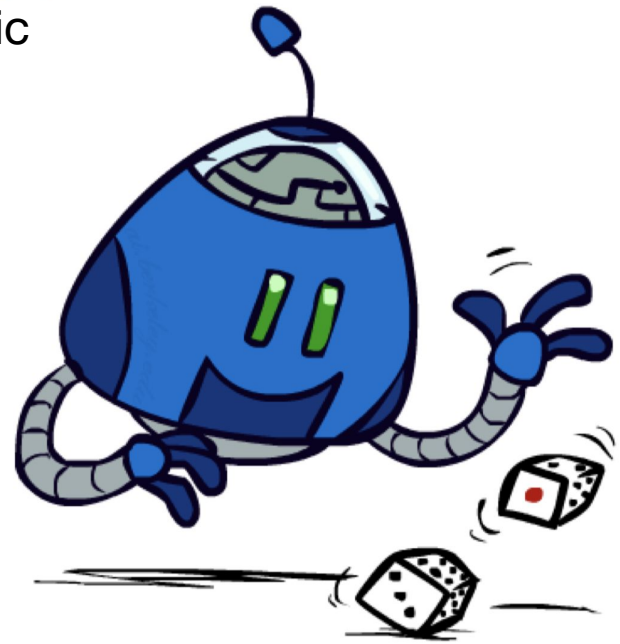
$$\min_{\pi} \max_{D} \mathbb{E}_{\pi}[\log D(s,a)] + \mathbb{E}_{\pi_E}[1 - \log D(s,a)] - \lambda H(\pi)$$

$\mathbb{E}_{s\sim\rho}[\log D(s,\pi(s))]$

Deterministic policy

$\mathbb{E}_{s\sim\rho_\pi(s)}\mathbb{E}_{a\sim\pi(\cdot|s)}[\log D(s,a)]$

Stochastic policy



$$\min_{\pi} \max_{D} \mathbb{E}_{\pi}[\log D(s,a)] + \mathbb{E}_{\pi_E}[1 - \log D(s,a)] - \lambda H(\pi)$$

Algorithm 1 Generative adversarial imitation learning

1: **Input:** Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminator parameters θ_0, w_0
2: **for** $i = 0, 1, 2, \ldots$ **do**
3: Sample trajectories $\tau_i \sim \pi_{\theta_i}$
4: Update the discriminator parameters from w_i to w_{i+1} with the gradient

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s,a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s,a))] \tag{17}$$

5: Take a policy step from θ_i to θ_{i+1}, using the TRPO rule with cost function $\log(D_{w_{i+1}}(s,a))$. Specifically, take a KL-constrained natural gradient step with

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)] - \lambda \nabla_\theta H(\pi_\theta), \quad \text{where } Q(\bar{s},\bar{a}) = \hat{\mathbb{E}}_{\tau_i}[\log(D_{w_{i+1}}(s,a)) \mid s_0 = \bar{s}, a_0 = \bar{a}] \tag{18}$$

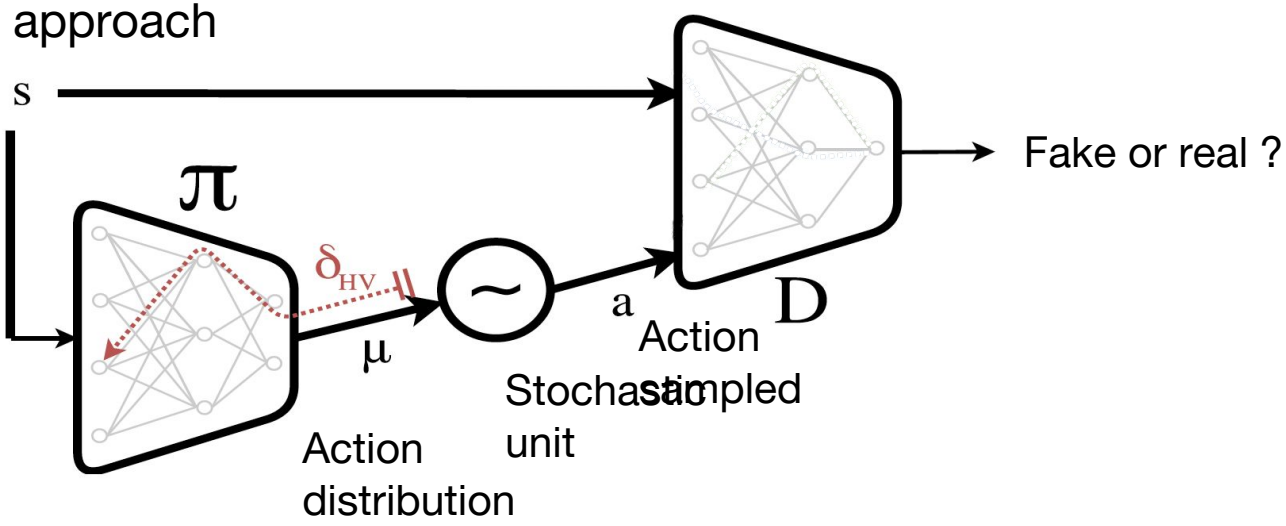
6: **end for**

REINFORCE : High variance

GAIL is a model-free approach:

some information is lost

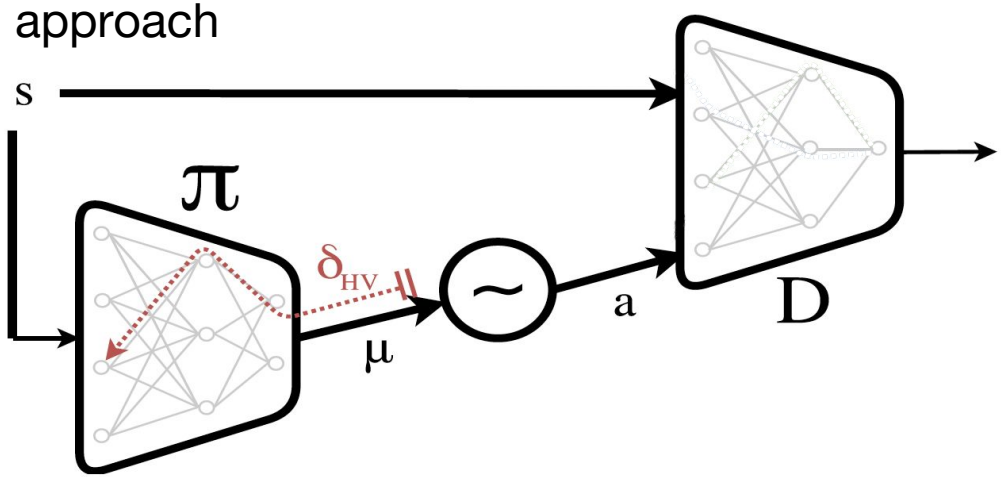
Block diagram of the model-free approach



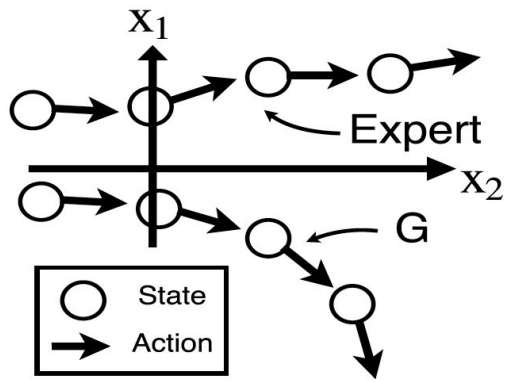
GAIL is a model-free approach:

some information is lost

Block diagram of the model-free approach



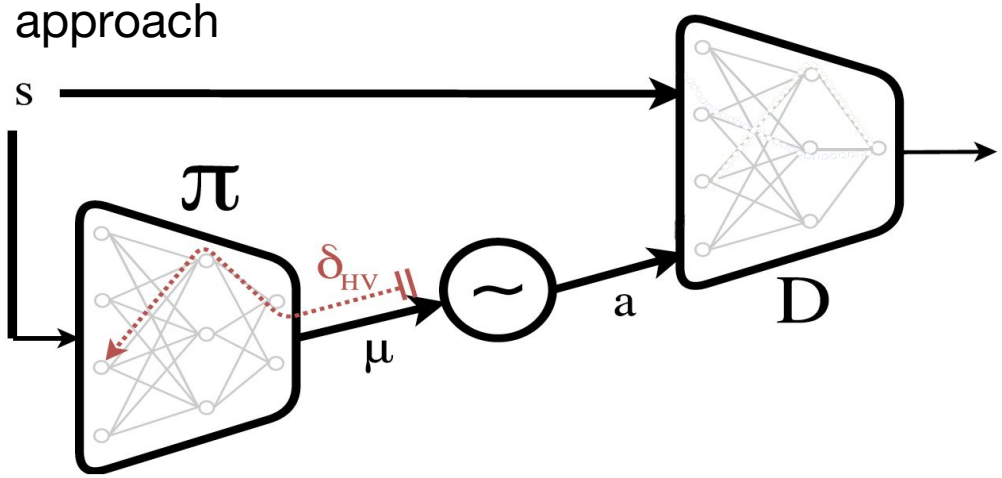
Why using $\nabla_s D$ is advantageous



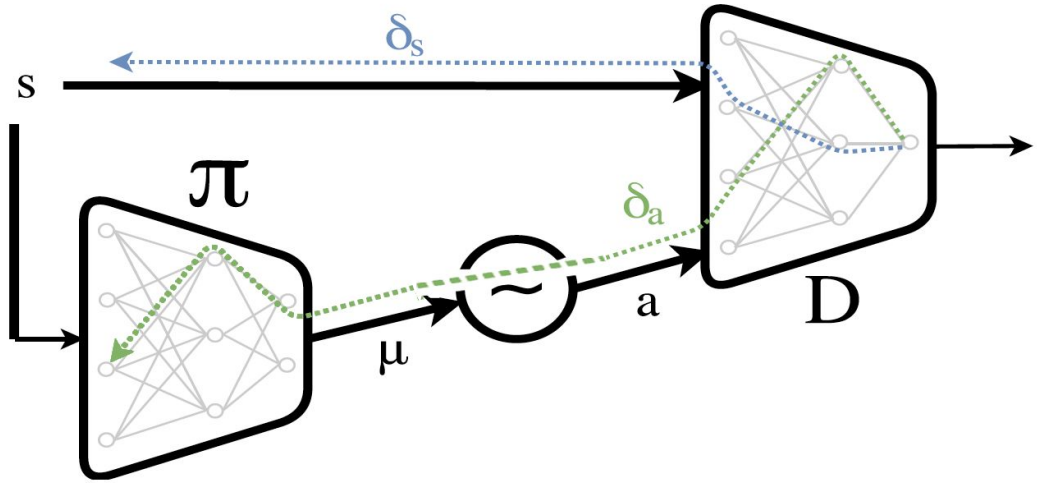
GAIL is a model-free approach:

some information is lost

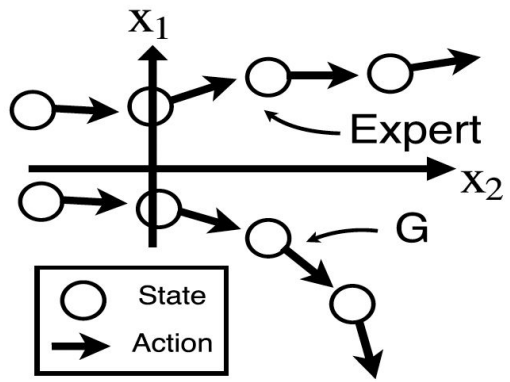
Block diagram of the model-free approach



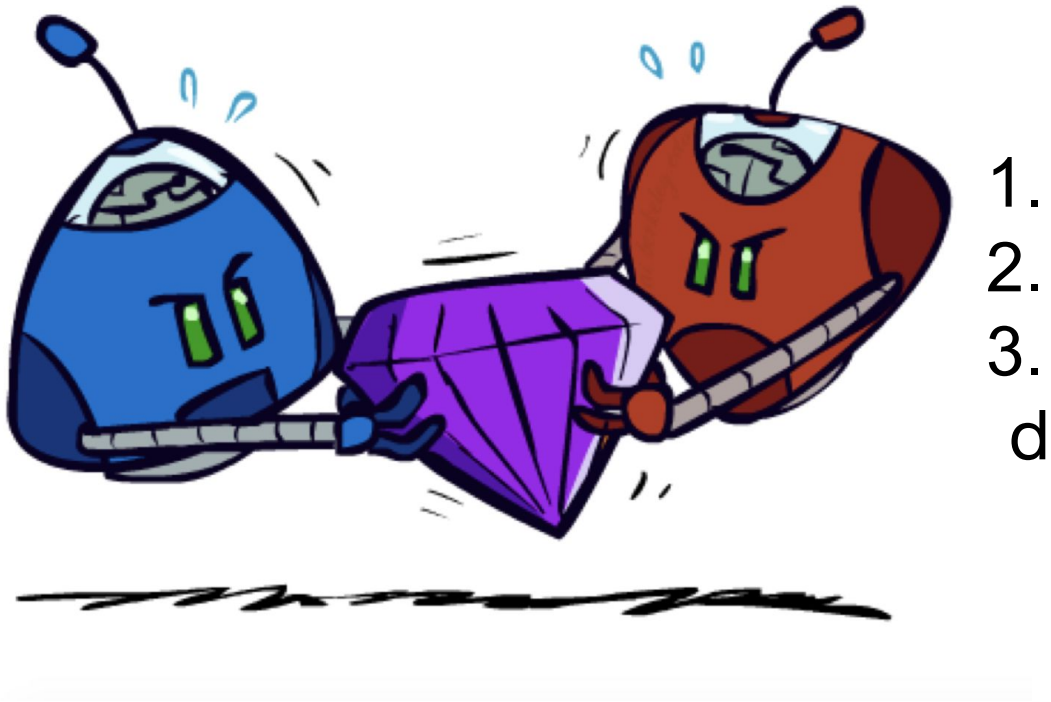
WHAT WE WANT



Why using $\nabla_s D$ is advantageous

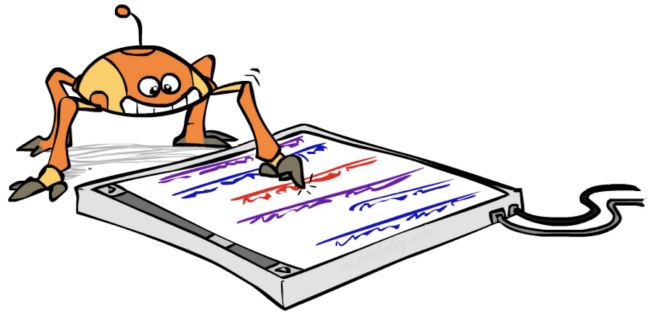


So far…



1. GAIL is model-free.
2. We use stochastic policies.
3. We need to access the discriminator's Jacobian.

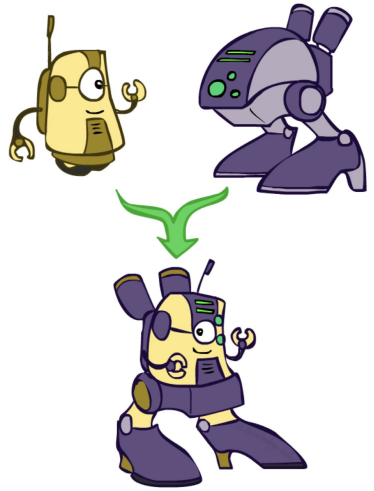
Some intuition around the discriminator



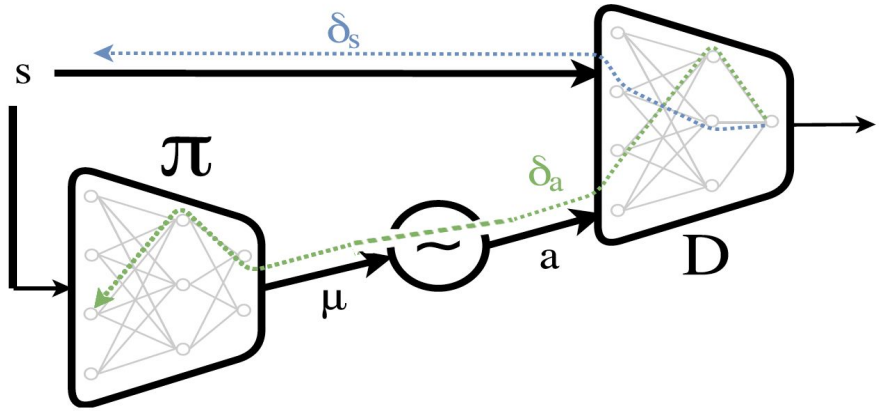
$$D(s,a) = p(\pi|s,a) = \frac{1}{1 + \frac{p(a|s,\pi_E)}{p(a|s,\pi)} \cdot \frac{p(s|\pi_E)}{p(s|\pi)}}$$

Policy likelihood ratio

State distribution likelihood



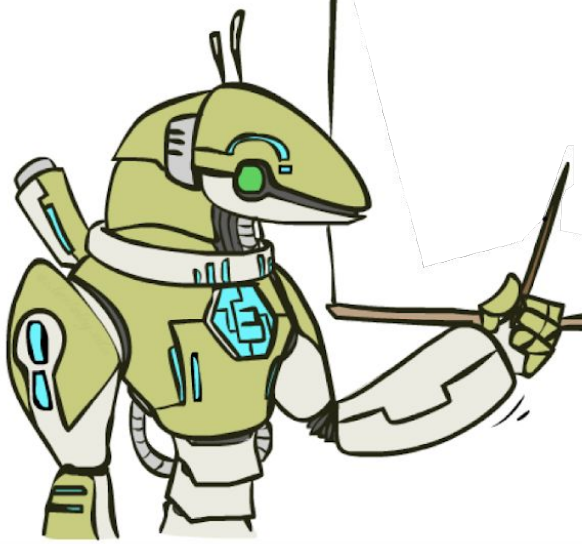
Towards full differentiability



$$\nabla_a D$$

$$\nabla_s D$$

We want to use the ~~Jacobian of D when~~ training the generator.

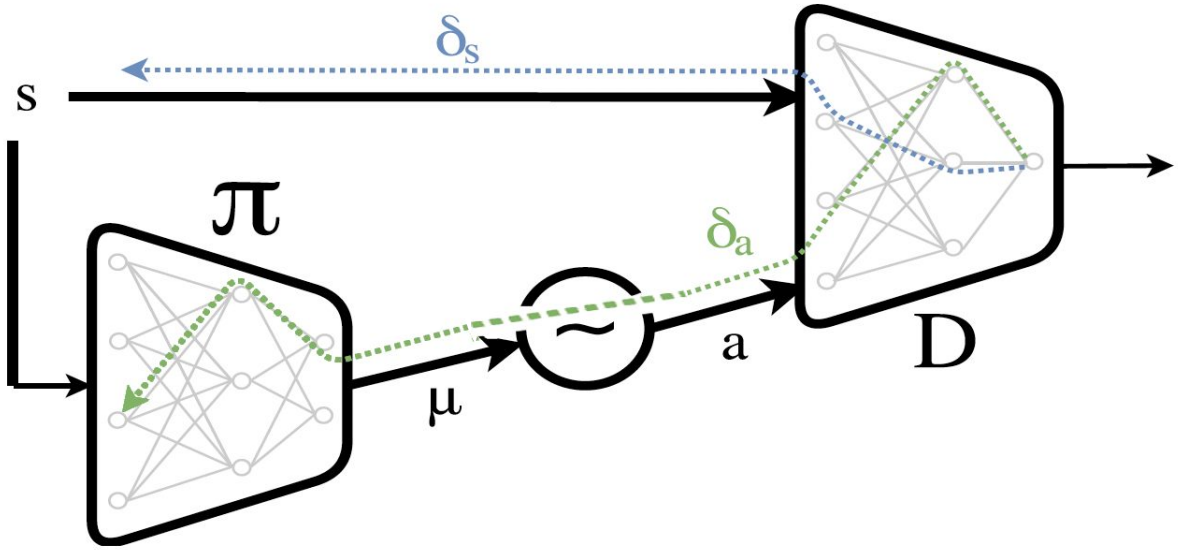


1. Backpropagation through stochastic units
2. Backpropagation through a forward model
3. Combine this in a Model-based GAIL algorithm

1. Backpropagation through stochastic units

For **continuous** action policies
→ Reparametrization

For **categorical** action distributions
→ Categorical reparametrization using Gumbel-Softmax trick

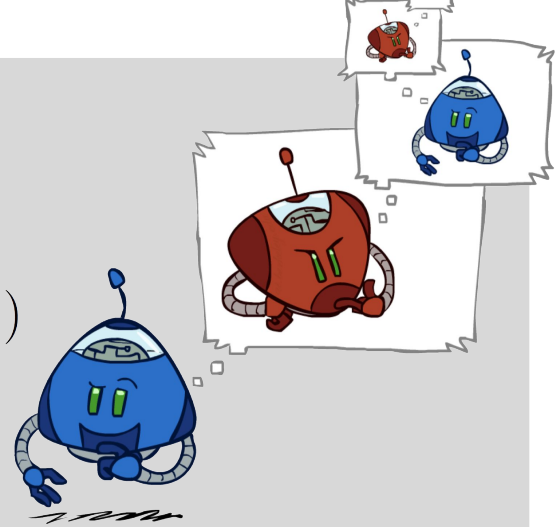


1. Backpropagation through stochastic units

For **continuous** action policies
→ Reparametrization

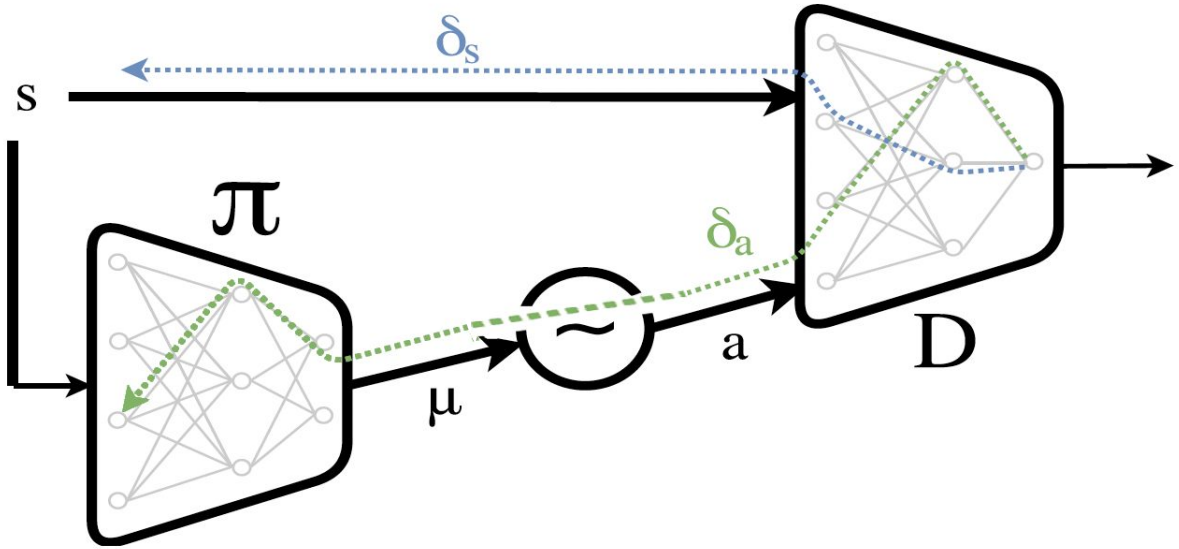
For **categorical** action distributions
→ Categorical reparametrization using Gumbel-Softmax trick

2. Backpropagation through a forward model



We want to compute $s_t = f(s_{t-1}, a_{t-1})$ where *f* is the forward model.

→ During the backward pass, the error message regarding deviations of future states propagates back in time and influences the actions of the policy in earlier times.



1. Backpropagation through stochastic units

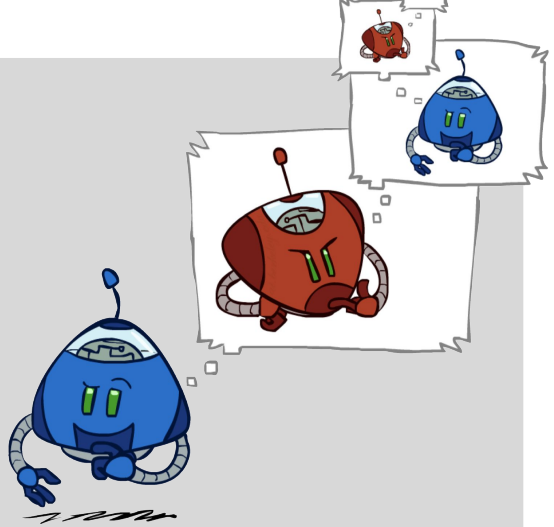
For **continuous** action policies
→ Reparametrization

For **categorical** action distributions
→ Categorical reparametrization using Gumbel-Softmax trick

2. Backpropagation through a forward model

Forward model: $(s_t, a_t) \rightarrow s_{t+1}$

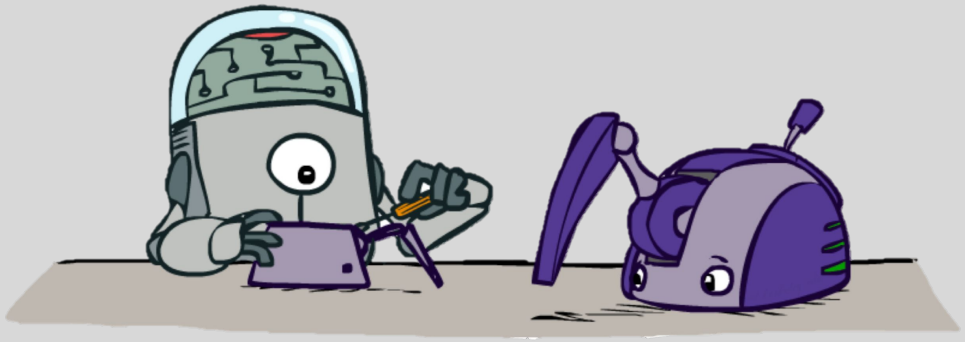
→ During the backward pass, the error message regarding deviations of future states propagates back in time and influences the actions of the policy in earlier times.



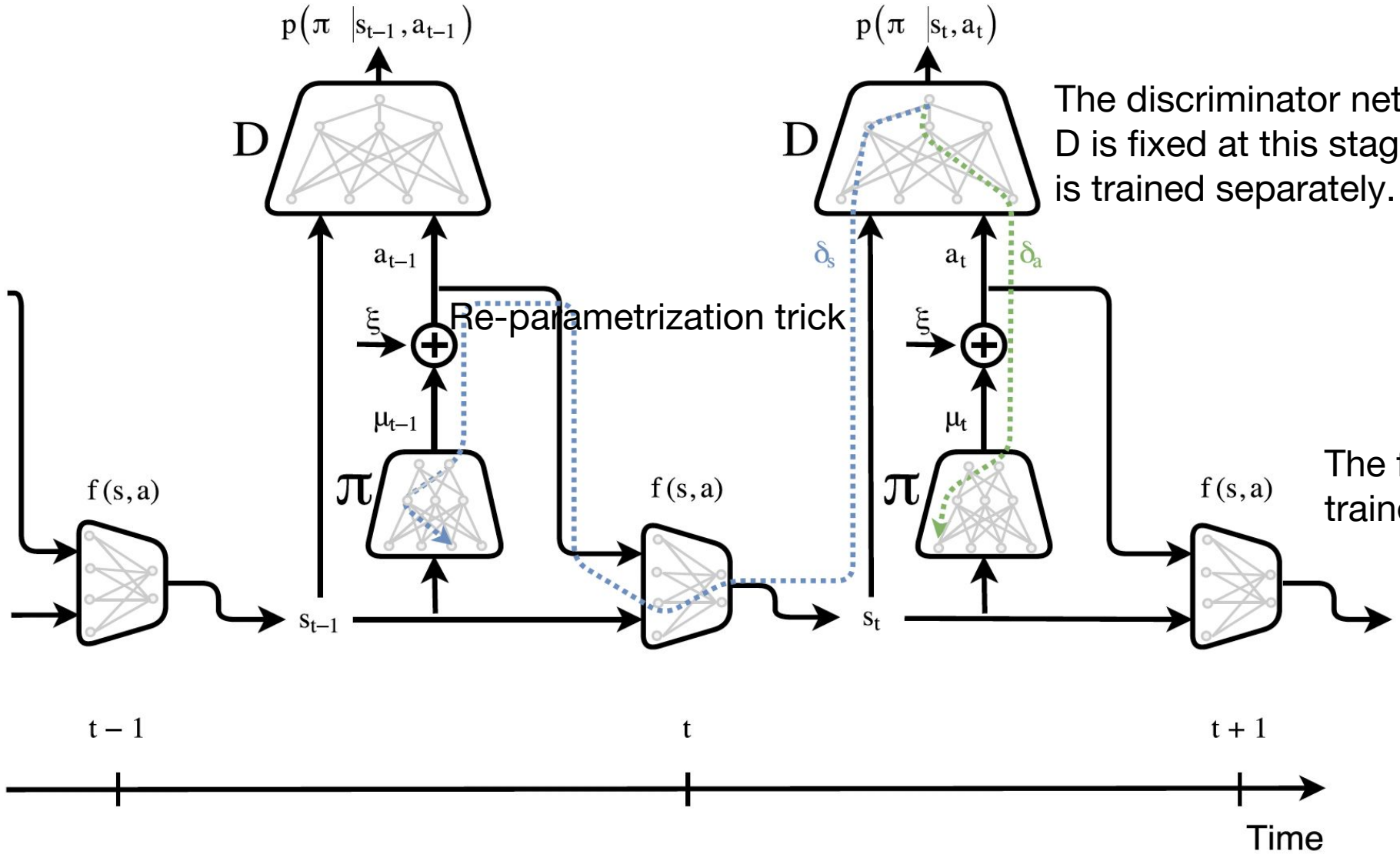
$\nabla_a D$

$\nabla_s D$

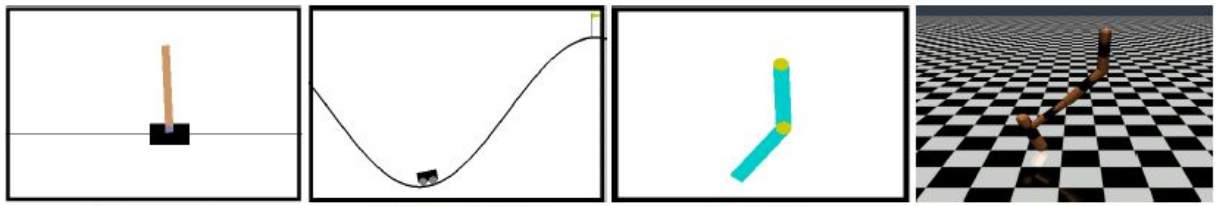
3. Model-based GAIL



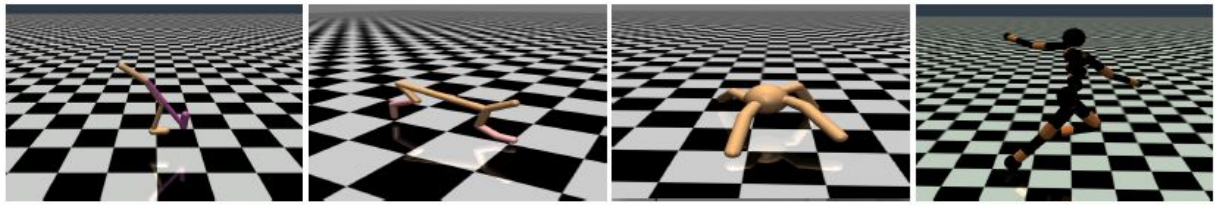
The block diagram of model-based GAIL



The experiments



(a) Cartpole (b) MountainCar (c) Acrobot (d) Hopper



(e) Walker (f) Half-Cheetah (g) Ant (h) Humanoid

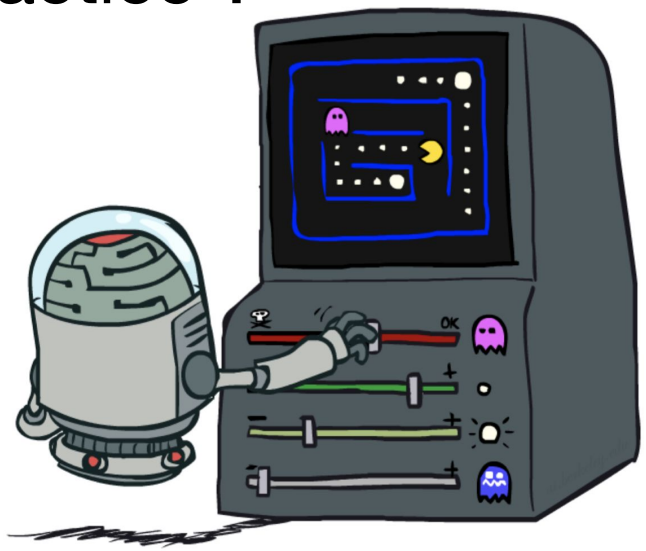
Task	Dataset size	Behavioral cloning	GAIL	MGAIL
Cartpole	1	72.02 ± 35.82	200.00 ± 0.00	200.00 ± 0.00
	4	169.18 ± 59.18	200.00 ± 0.00	200.00 ± 0.00
	7	188.60 ± 29.61	200.00 ± 0.00	200.00 ± 0.00
	10	177.19 ± 52.83	200.00 ± 0.00	200.00 ± 0.00
Mountain Car	1	-136.76 ± 34.44	-101.55 ± 10.32	-107.4 ± 10.89
	4	-133.25 ± 29.97	-101.35 ± 10.63	-100.23 ± 11.52
	7	-127.34 ± 29.15	-99.90 ± 7.97	-104.23 ± 14.31
	10	-123.14 ± 28.26	-100.83 ± 11.40	-99.25 ± 8.74
Acrobot	1	-130.60 ± 55.08	-77.26 ± 18.03	-85.65 ± 23.74
	4	-93.20 ± 35.58	-83.12 ± 23.31	-81.91 ± 17.41
	7	-96.92 ± 34.51	-82.56 ± 20.95	-80.74 ± 14.02
	10	-95.09 ± 33.33	-78.91 ± 15.76	-77.93 ± 14.78
Hopper	4	50.57 ± 0.95	3614.22 ± 7.17	3669.53 ± 6.09
	11	1025.84 ± 266.86	3615.00 ± 4.32	3649.98 ± 12.36
	18	1949.09 ± 500.61	3600.70 ± 4.24	3661.78 ± 11.52
	25	3383.96 ± 657.61	3560.85 ± 3.09	3673.41 ± 7.73
Walker	4	32.18 ± 1.25	4877.98 ± 2848.37	6916.34 ± 115.20
	11	5946.81 ± 1733.73	6850.27 ± 91.48	7197.63 ± 38.34
	18	1263.82 ± 1347.74	6964.68 ± 46.30	7128.87 ± 141.98
	25	1599.36 ± 1456.59	6832.01 ± 254.64	7070.45 ± 30.68
Half-Cheetah	4	-493.62 ± 246.58	4515.70 ± 549.49	4891.56 ± 654.43
	11	637.57 ± 1708.10	4280.65 ± 1119.93	4844.61 ± 138.78
	18	2705.01 ± 2273.00	4749.43 ± 149.04	4876.34 ± 85.74
	25	3718.58 ± 1856.22	4840.07 ± 95.36	4989.95 ± 351.14
Ant	4	1611.75 ± 359.54	3186.80 ± 903.57	4645.12 ± 179.29
	11	3065.59 ± 635.19	3306.67 ± 988.39	4657.92 ± 94.27
	18	2579.22 ± 1366.57	3033.87 ± 1460.96	4664.44 ± 183.11
	25	3235.73 ± 1186.38	4132.90 ± 878.67	4637.52 ± 45.66
Humanoid	80	1397.06 ± 1057.84	10200.73 ± 1324.47	10312.34 ± 388.54
	160	3655.14 ± 3714.28	10119.80 ± 1254.73	10428.39 ± 46.12
	240	5660.53 ± 3600.70	10361.94 ± 61.28	10470.94 ± 54.35

Conclusion

- The forward model enables to train policies using the exact gradient of the discriminator network.

→ We need to learn a forward model separately.

- Appears tricky, in practice ?



References

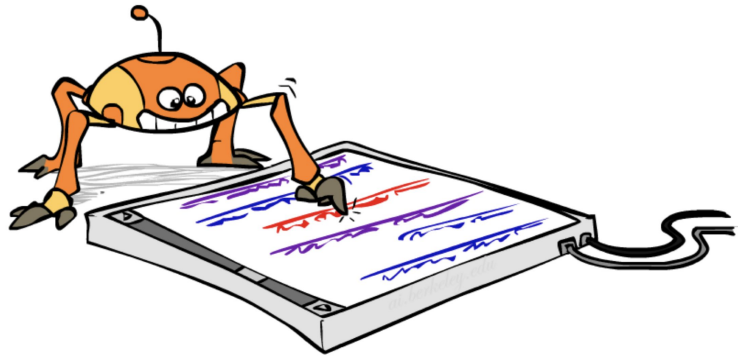
Generative Adversarial Imitation Learning
2016, Jonathan Ho, Stefano Ermon
https://arxiv.org/abs/1606.03476

End-to-End Differentiable Adversarial Imitation Learning
NIPS 2017, Nir Baram, Oron Anschel, Itai Caspi, Shie Mannor
http://proceedings.mlr.press/v70/baram17a/baram17a.pdf

Credits to **Ketrina Yim** for the drawings (some photoshopped by me to adapt to the presentation)

Appendix

Analyse the discriminator

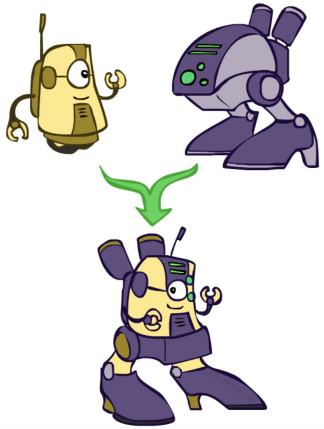


$$D(s,a) = p(\pi|s,a) = \frac{p(s,a|\pi)p(\pi)}{p(s,a)} = \frac{p(s,a|\pi)}{p(s,a|\pi) + p(s,a|\pi_E)}$$

$$D(s,a) = \frac{1}{1 + \frac{p(s,a|\pi_E)}{p(s,a|\pi)}} = \frac{1}{1 + \frac{p(a|s,\pi_E)}{p(a|s,\pi)} \cdot \frac{p(s|\pi_E)}{p(s|\pi)}}$$

Policy likelihood ratio

State distribution likelihood)



3.3. Backpropagating through a Forward model

So far we showed the changes necessary to use the exact partial derivative $\nabla_a D$. Incorporating the use of $\nabla_s D$ as well is a more involved and constitutes the **crux** of this work. To understand why, we can look at the block diagram of the model-free approach in Figure 2. There, s is treated as fixed (it is given as an input), therefore $\nabla_s D$ is discarded. On the contrary, in the model-based approach, s_t can be written as a function of the previous state and action: $s_t = f(s_{t-1}, a_{t-1})$, where f is the forward model. This way, using the law of total derivatives, we get that:

$$\nabla_\theta D(s_t, a_t)\Big|_{s=s_t, a=a_t} = \frac{\partial D}{\partial a}\frac{\partial a}{\partial \theta}\Big|_{a=a_t} + \frac{\partial D}{\partial s}\frac{\partial s}{\partial \theta}\Big|_{s=s_t} = \frac{\partial D}{\partial a}\frac{\partial a}{\partial \theta}\Big|_{a=a_t} + \frac{\partial D}{\partial s}\left(\frac{\partial f}{\partial s}\frac{\partial s}{\partial \theta}\Big|_{s=s_{t-1}} + \frac{\partial f}{\partial a}\frac{\partial a}{\partial \theta}\Big|_{a=a_{t-1}}\right). \tag{11}$$

Considering that a_{t-1} is a function of θ, we understand that by creating a computation graph that spans over more than a single time-step, we can *link* between $\nabla_s D$ and the policy. Put in words, during the backward pass, the error message regarding deviations of future states (ψ_s), propagates back in time and influences the actions of the policy in earlier times. Figure 3 summarizes this idea.

Model-based GAIL

We showed that a good approach for imitation requires: *(a)* to use a model, and *(b)* to process multi-step transitions. This setup was previously suggested by Shalev-Shwartz et al. (2016) and Heess et al. (2015), who built a multi-step computation graph for describing the familiar policy gradient objective, which in our case is given by: $J(\theta) = \mathbb{E}\Big[\sum_{t=0} \gamma^t D(s_t, a_t) \Big| \theta\Big]$. To show how to differentiate $J(\theta)$ over a trajectory of (s, a, s') transitions, we rely on the results of Heess et al. (2015):

$$J_s = \mathbb{E}_{p(a|s)} \mathbb{E}_{p(s'|s,a)} \Big[D_s + D_a \pi_s + \gamma J'_{s'} (f_s + f_a \pi_s) \Big], \tag{12}$$

$$J_\theta = \mathbb{E}_{p(a|s)} \mathbb{E}_{p(s'|s,a)} \Big[D_a \pi_\theta + \gamma (J'_{s'} f_a \pi_\theta + J'_\theta) \Big]. \tag{13}$$

The final policy gradient $\nabla_\theta J$ is calculated by applying Eq. 12 and 13 recursively, starting from $t = T$ all the way down to $t = 0$. The full algorithm is presented in Algorithm 1.

Algorithm 1 Model-based Generative Adversarial Imitation Learning

Input: Expert trajectories τ_E, experience buffer $\mathcal{B}$, initial policy and discriminator parameters θ_g, θ_d
for $trajectory = 0$ **to** ∞ **do**
  for $t = 0$ **to** T **do**
    Act on environment: $a = \pi(s, \xi; \theta_g)$
    Push (s, a, s') into $\mathcal{B}$
  end for
  train forward model f using $\mathcal{B}$
  train discriminator model D_{θ_d} using $\mathcal{B}$
  set: $j'_s = 0, j'_{\theta_g} = 0$
  for $t = T$ **down to** 0 **do**
    $j_{\theta_g} = [D_a \pi_{\theta_g} + \gamma(j'_{s'} f_a \pi_{\theta_g} + j'_{\theta_g})]\big|_\xi$
    $j_s = [D_s + D_a \pi_s + \gamma j'_{s'}(f_s + f_a \pi_{\theta_g})]\big|_\xi$
  end for
  Apply gradient update using $j^0_{\theta_g}$
end for

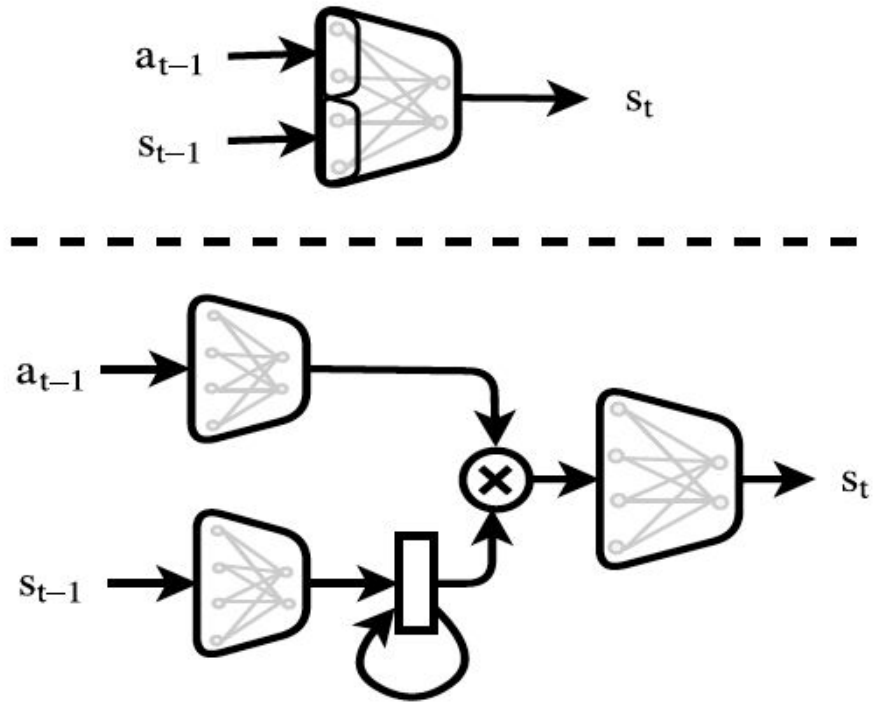


Figure 4. **Comparison between a vanilla forward model (Top) and an advanced forward model (Bottom).** In the vanilla forward model, the state and action vectors are concatenated and then passed through a two-layer neural network. In the advanced forward model, the action and state are passed independently through two neural networks and are then combined using Hadamard product. The result is then passed through another neural network to form the output.