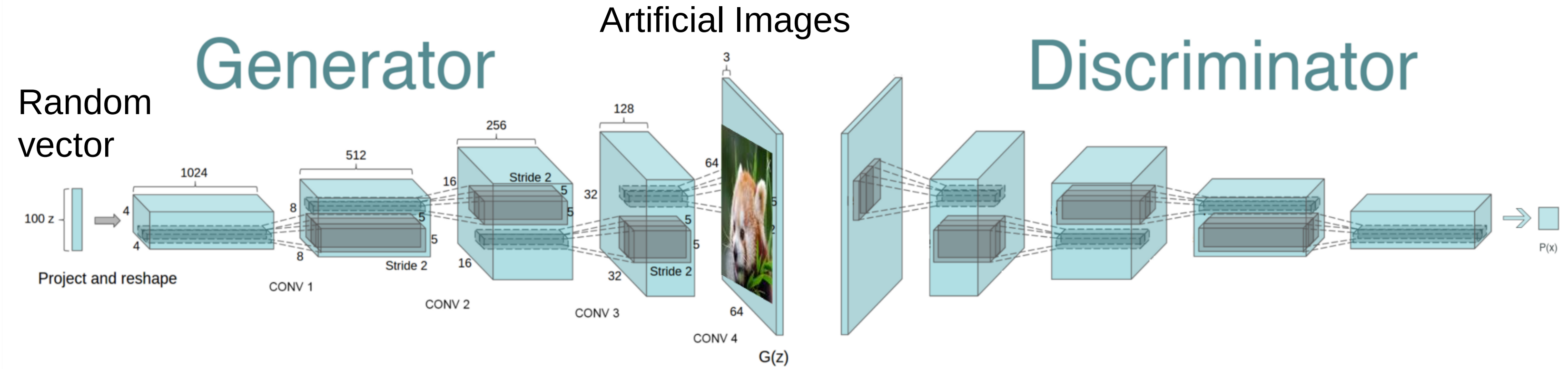
CS182/282A: Designing, Visualizing and Understanding Deep Neural Networks

John Canny

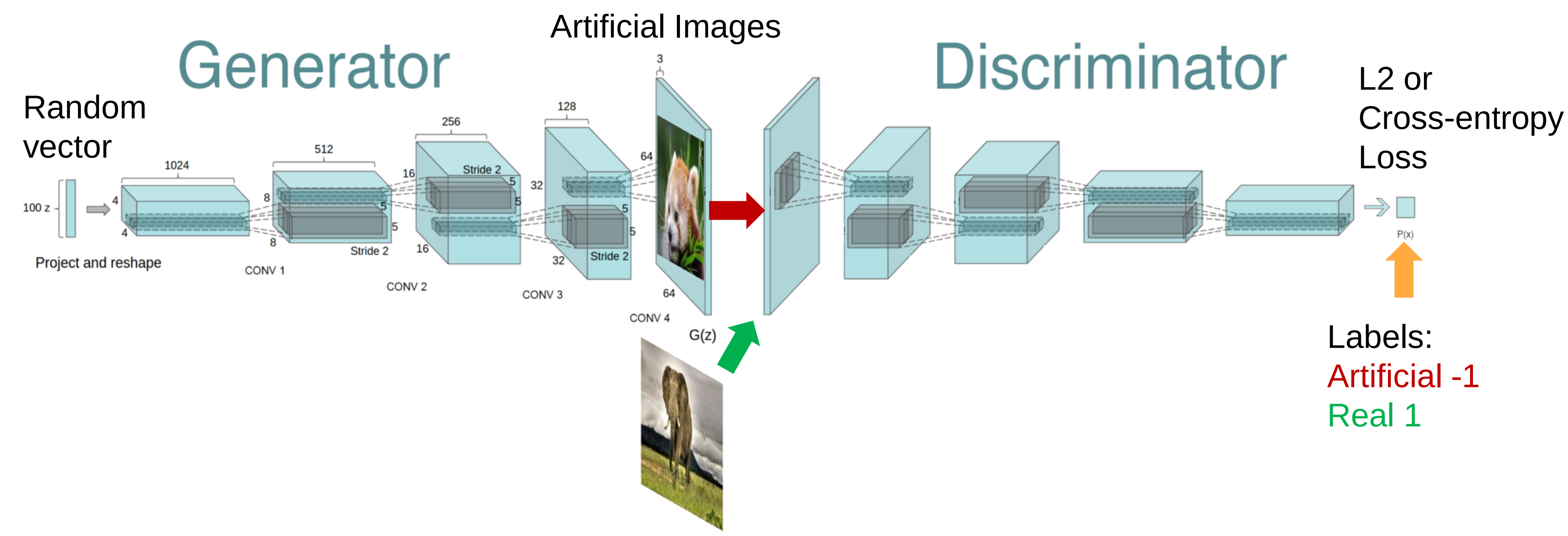
Spring 2019

Lecture 19: Imitation Learning

Last Time: Generative Adversarial Networks (GANs)

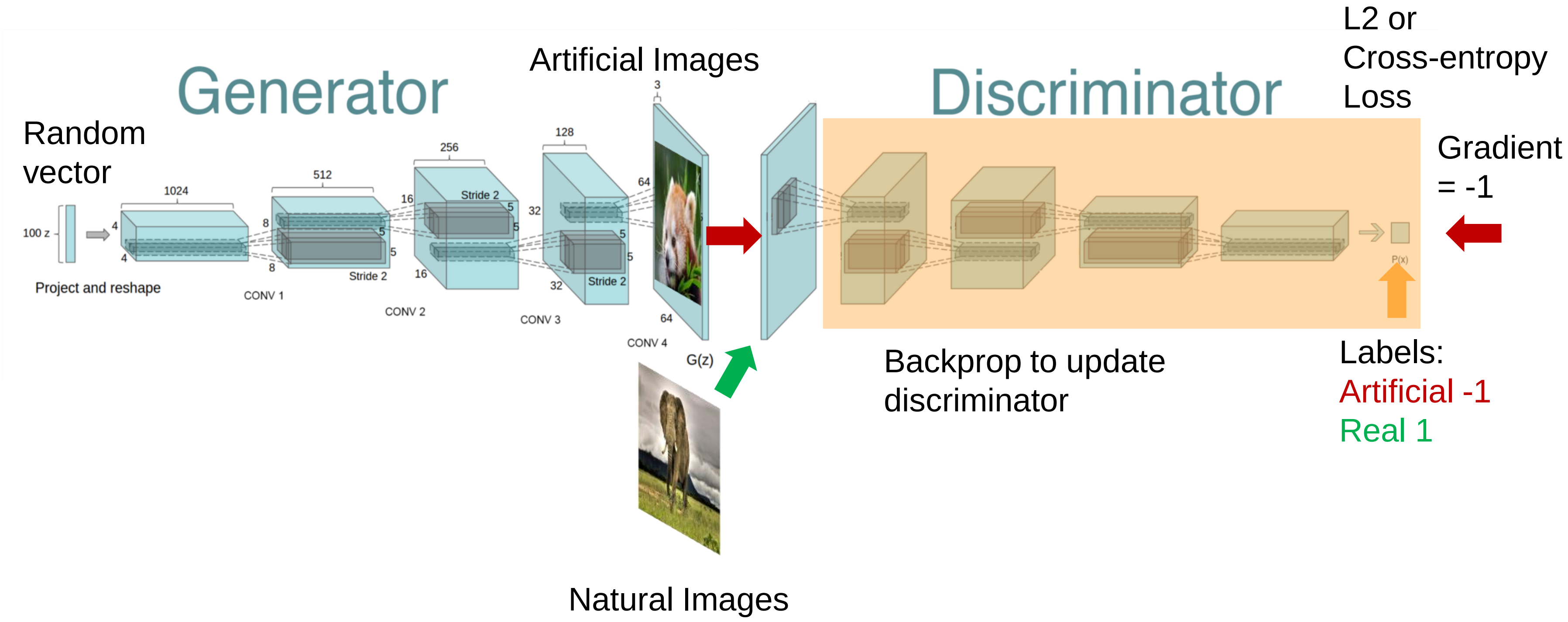


Last Time: GAN Discriminator Training

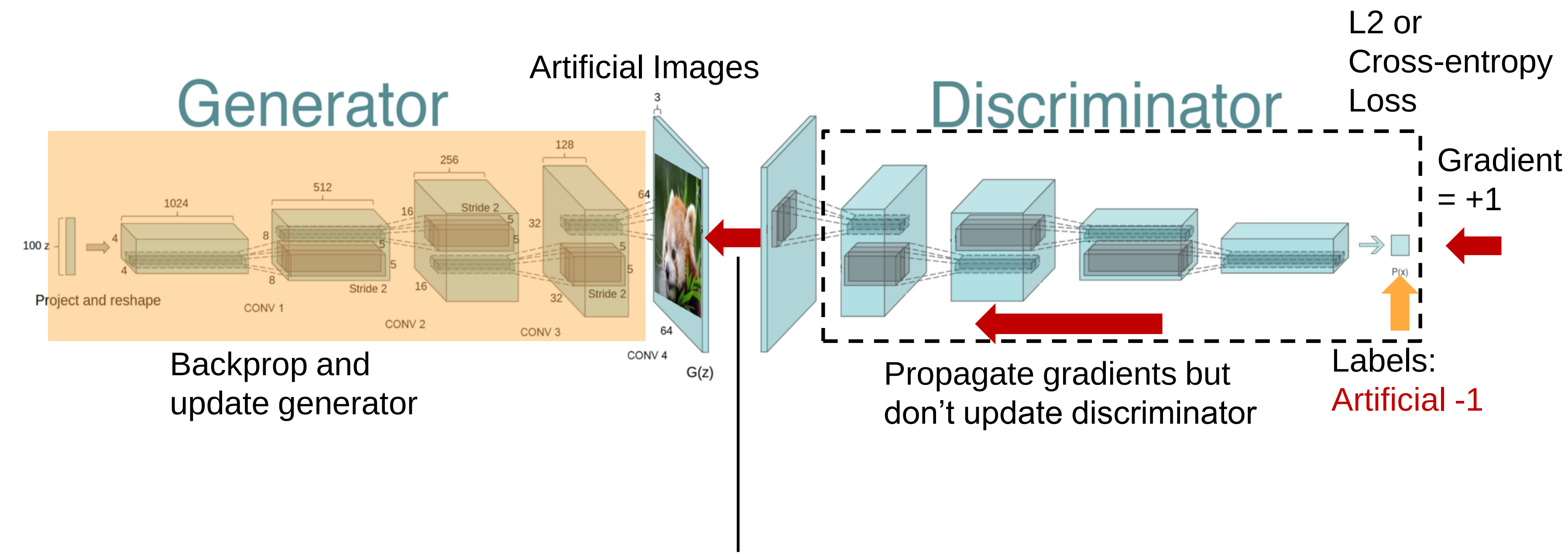


Natural Images

GAN Training: Minimize Discriminator classification loss



GAN Training: Train Generator to Fool the Discriminator

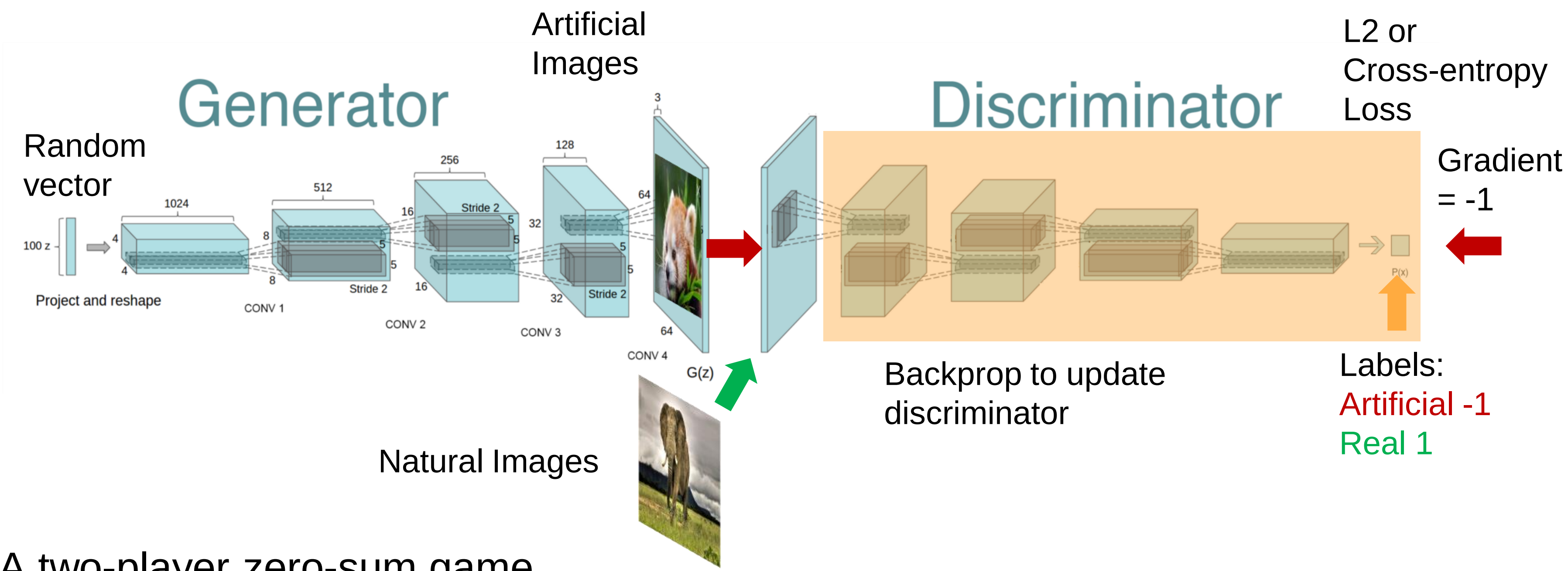


Backprop and
update generator

Propagate gradients but
don’t update discriminator

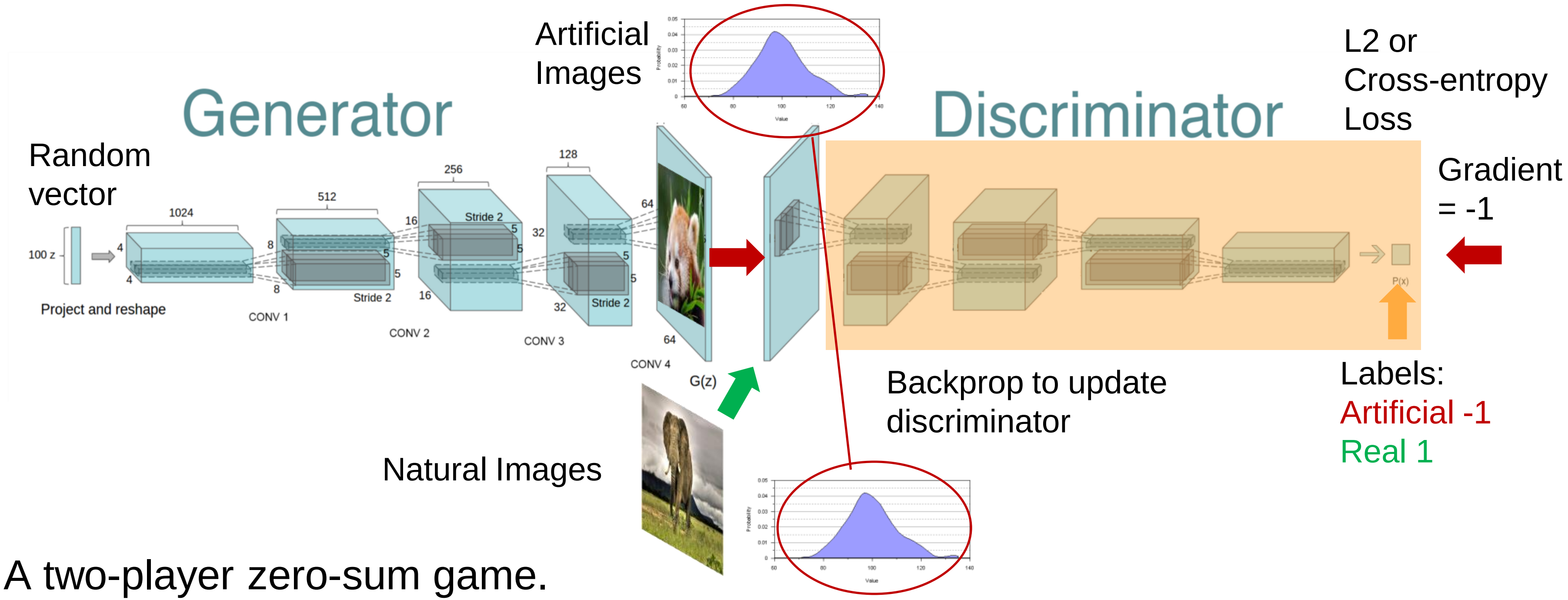
This gradient nudges the image
from “artificial” toward “natural”

GAN Training: Alternate Discriminator/Generator training



A two-player zero-sum game.

GAN Training: Alternate Discriminator/Generator training



A two-player zero-sum game.

Optimizing with minimax (alternating optimization) minimizes the difference (Jensen-Shannon divergence) between generator and natural image probability distributions.

Last Time: Evaluating GANs

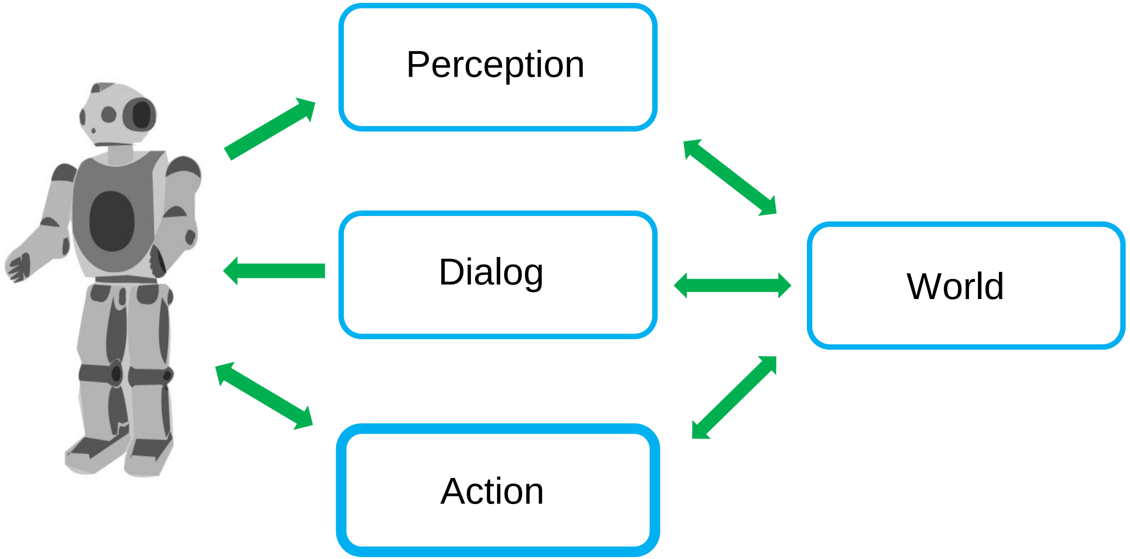
- Inception Score:

$$\mathrm{IS}(G) = \exp\left(\mathbb{E}_{\mathbf{x}\sim p_g}\, D_{KL}(\, p(y|\mathbf{x}) \parallel p(y)\,)\right),$$

  - Works mostly for natural images
  - Large dataset of images + dense labels for a well-designed network to pre-train on.
  - Correlates well to human judgment: Inception Model

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium.

This Time: Imitation Learning for Robot Control



Deep Control: First Idea: Imitate Human Actions

Supervised training of deep networks (with image category labels, captions, translations,…) from human data has worked well so far…

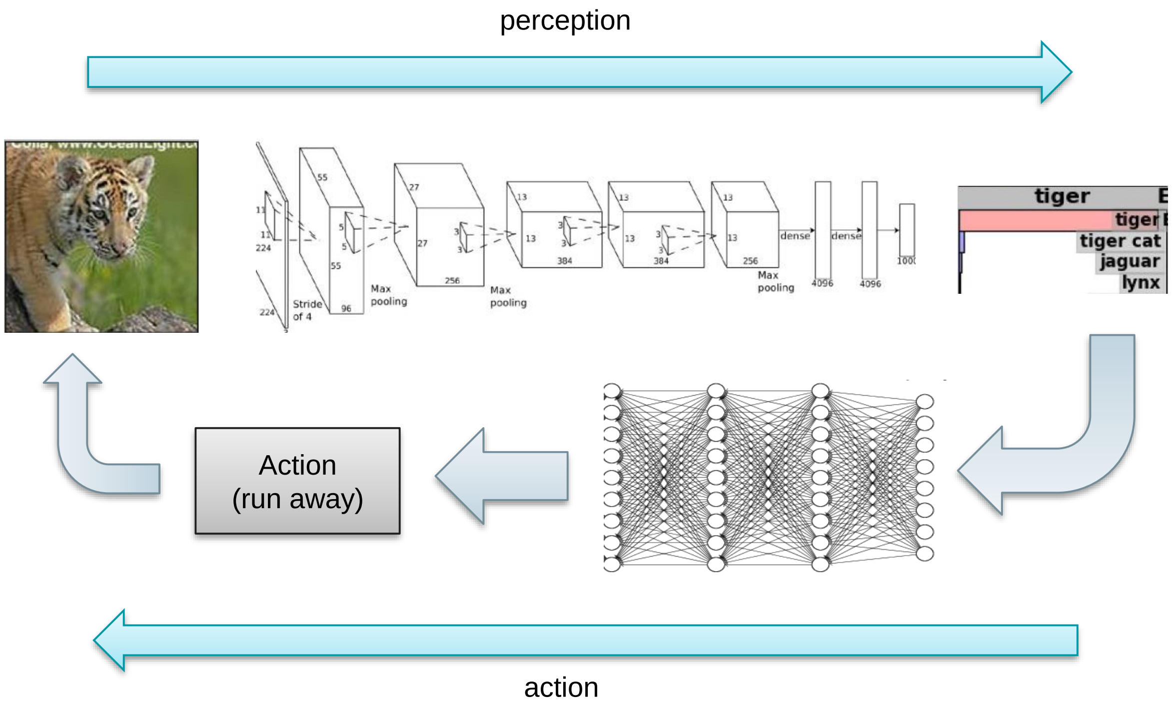
What about mimicking human control actions?

Imitation Learning via Behavior Cloning

This approach is called behavior cloning. Note that its not enough to record human actions, because humans are constantly adapting to the world.

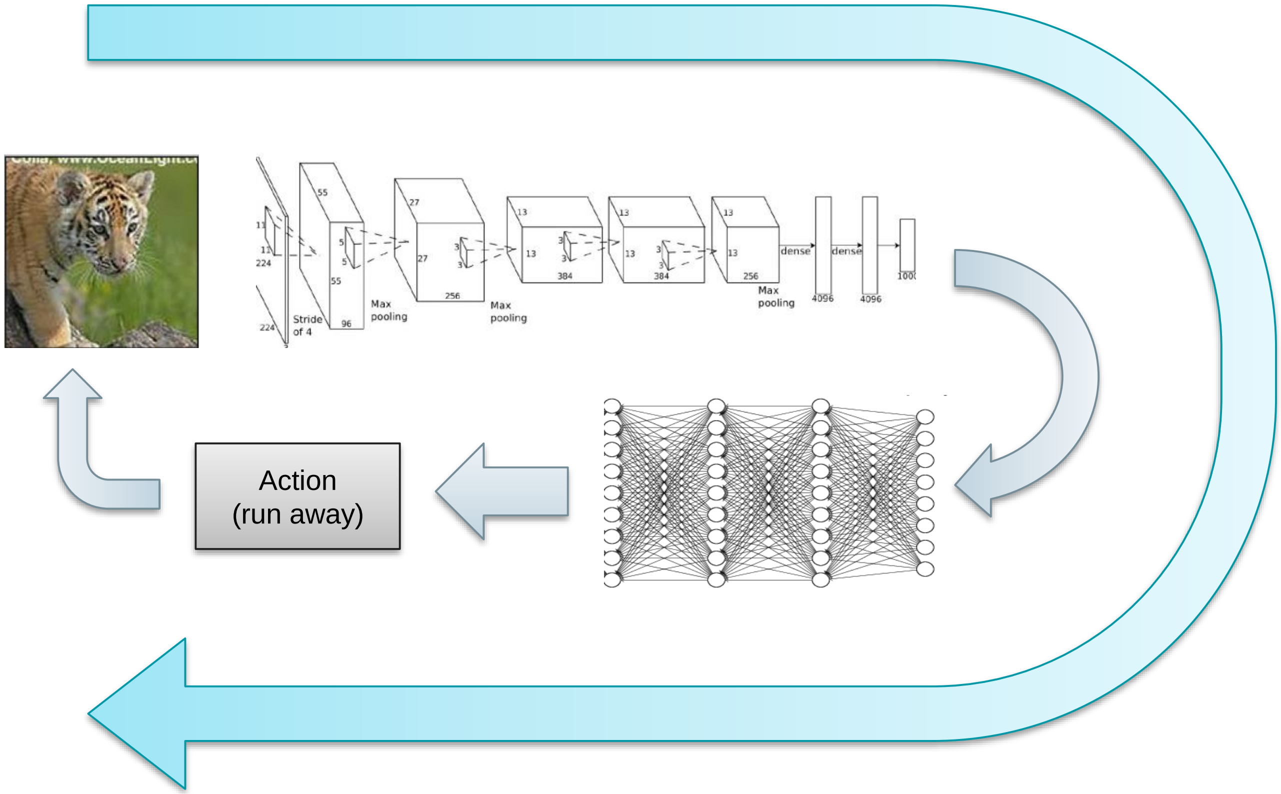
We need to learn a control loop from sensors to actuators.

Sensorimotor Learning



Slide from “Deep Reinforcement Learning” CS285, Levine and Finn

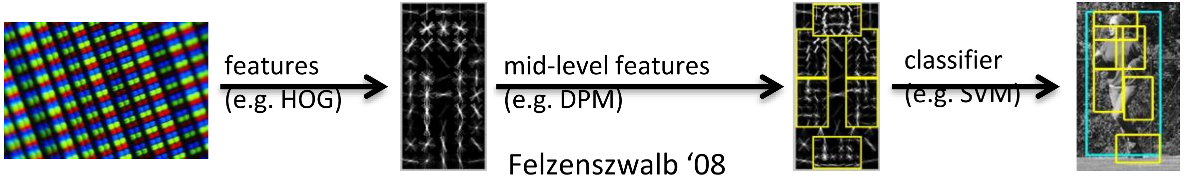
sensorimotor loop



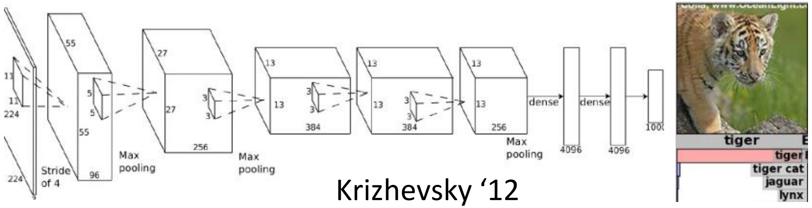
Slide from “Deep Reinforcement Learning” CS285, Levine and Finn

End-to-end vision

standard computer vision

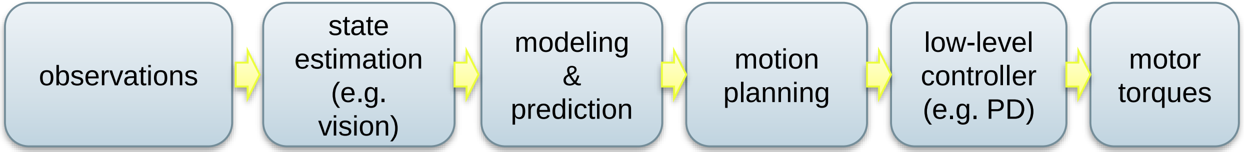


deep learning

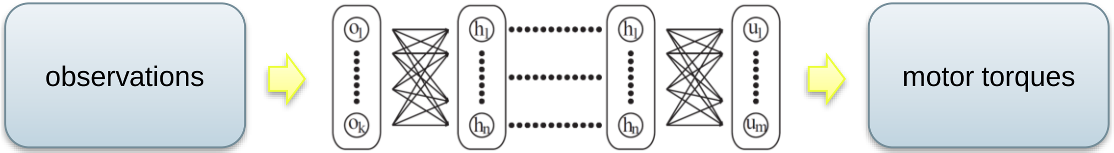


End-to-end control

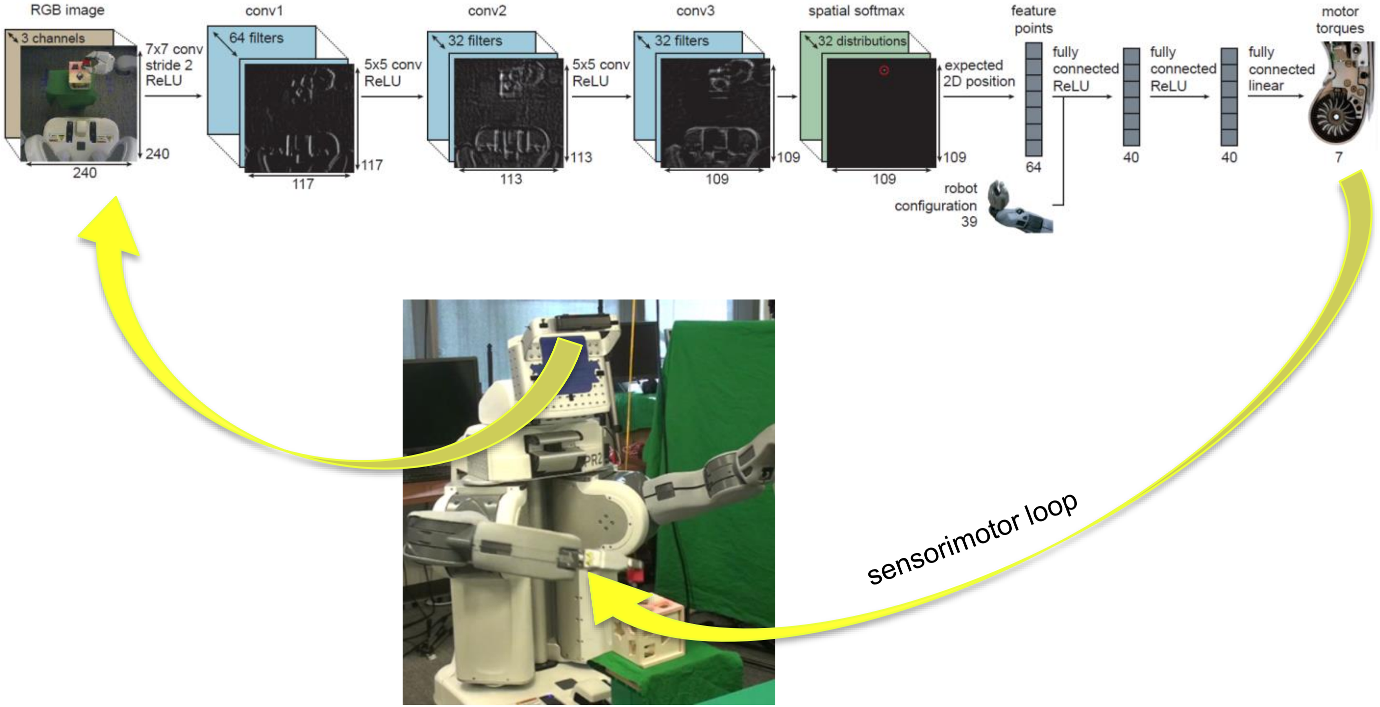
standard robotic control



deep sensorimotor learning



Slide from "Deep Reinforcement Learning" CS285, Levine and Finn



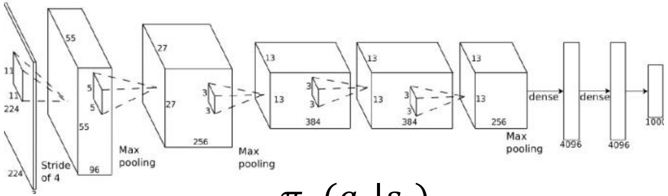
indirect supervision

actions have consequences

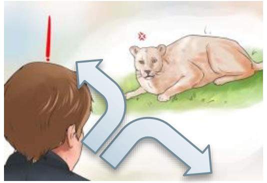
Slide from “Deep Reinforcement Learning” CS285, Levine and Finn

Terminology & notation

s_t



$\pi_\theta(a_t|s_t)$

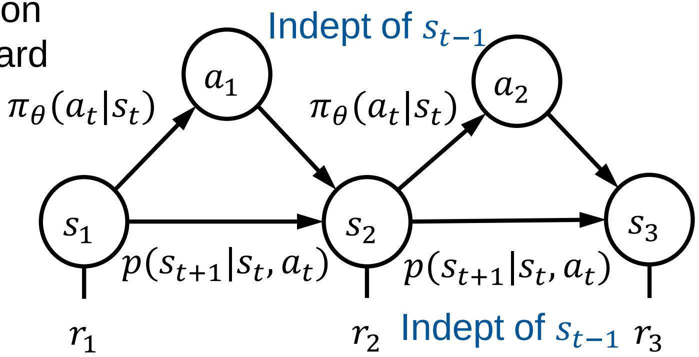


a_t

Environment

s_t - state
a_t - action
r_t - reward

$\pi_\theta(a_t|s_t)$ = policy = probability of doing action a_t in state s_t

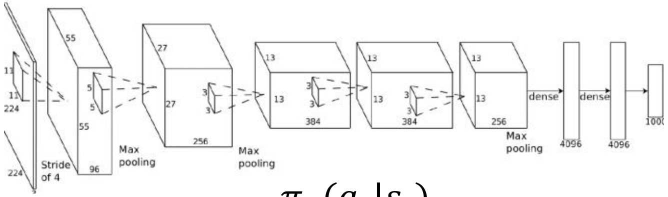


A Markov Decision Process

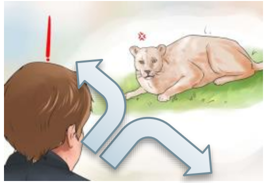
Markov property

Terminology & notation

s_t



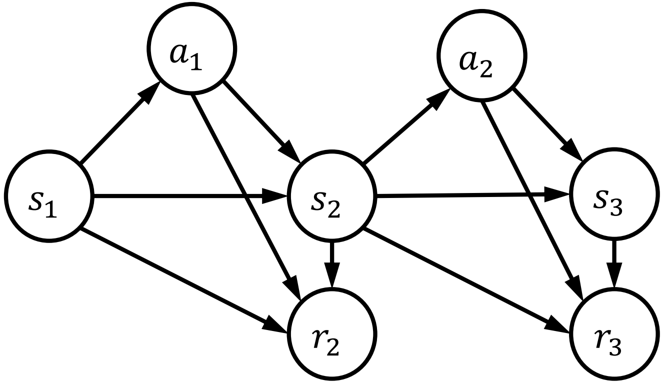
$\pi_\theta(a_t|s_t)$



a_t

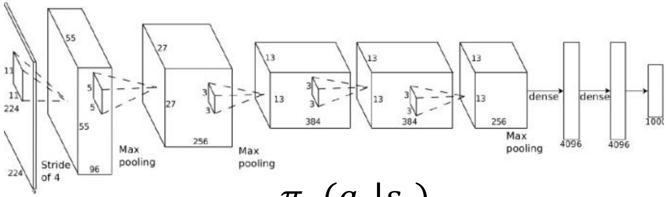
Environment

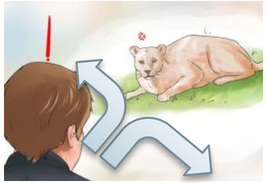
s_t - state
a_t - action
r_t - reward



Reward is a deterministic function of s_t, s_{t+1} and a_t.

Terminology & notation



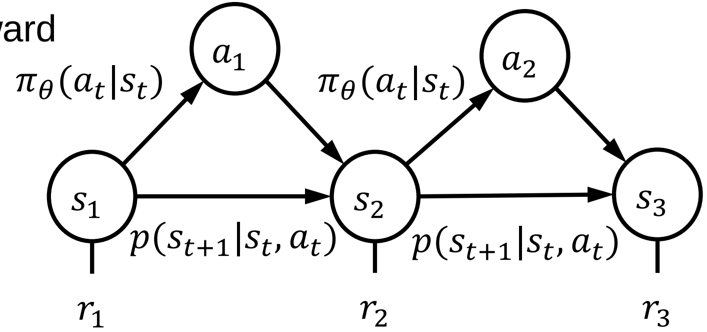


$\pi_\theta(a_t|s_t)$

s_t a_t

Environment

s_t - state
a_t - action
r_t - reward

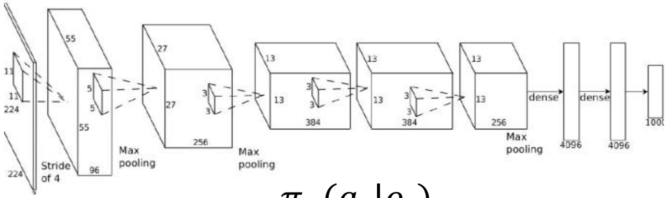


Reward is a deterministic function of s_t, s_{t+1} and a_t.

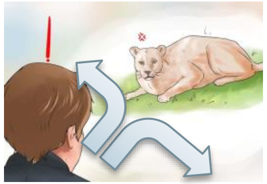
Usually omitted from the state diagram, e.g. shown as at left.

Terminology & notation

o_t



$\pi_\theta(a_t|o_t)$

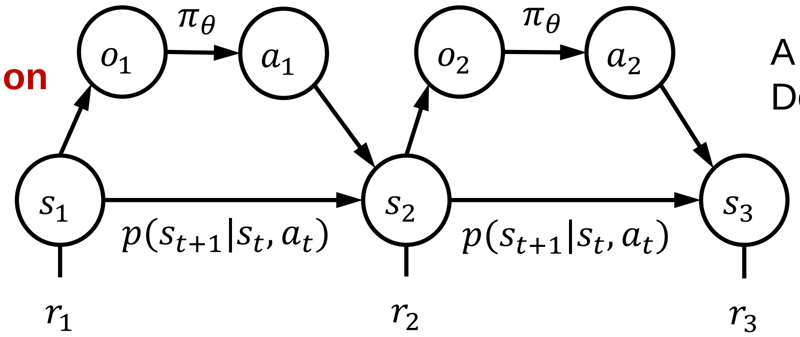


a_t

Environment

s_t - state
a_t - action
r_t - reward
o_t - **observation**

$\pi_\theta(a_t|o_t)$ = policy = probability of doing action a_t given **observation** o_t

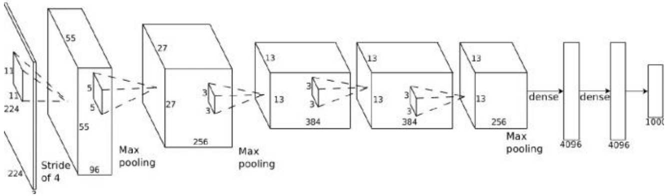


A Partially-Observable Markov Decision Process (POMDP)

e.g. o_t = image of tiger
s_t includes image of tiger
+ tiger is hungry

Imitation Learning (assume environment is an MDP)

s_t



$\pi_\theta(a_t|s_t)$

a_t

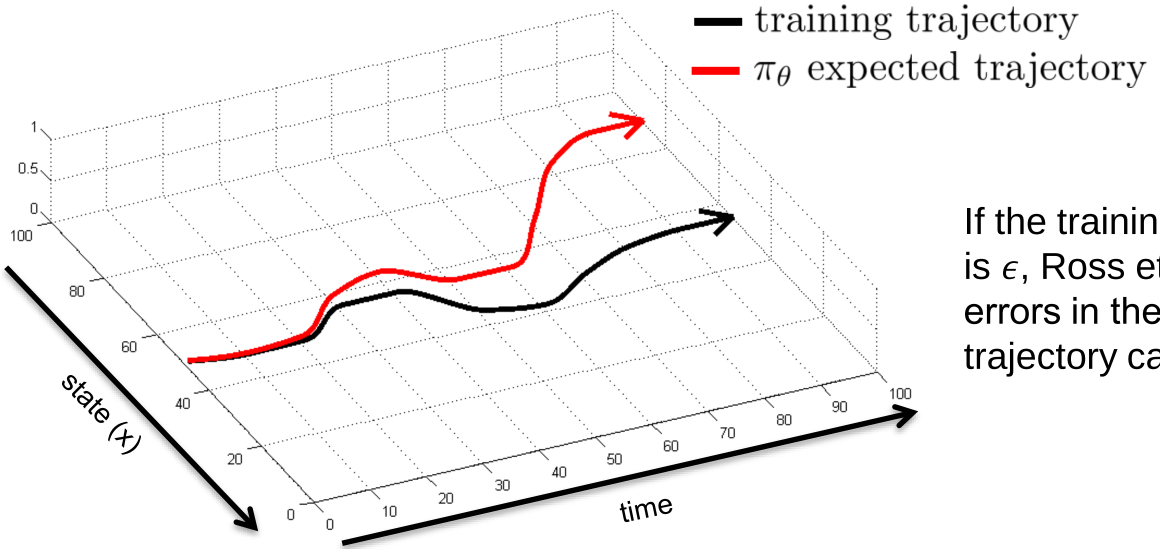
s_t
a_t

→ training data → supervised learning $\pi_\theta(a_t|s_t)$

Images: Bojarski et al. '16, NVIDIA

Does it work?

No!



If the training trajectory error is ϵ, Ross et al. show that errors in the learned model's trajectory can be order $O(T^2\epsilon)$

Slide from "Deep Reinforcement Learning" CS285, Levine and Finn

Does it work?

No!

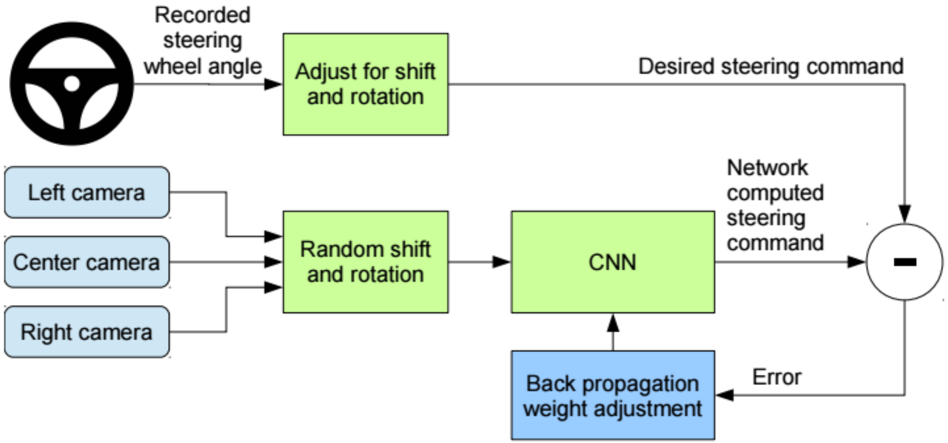
Yes!

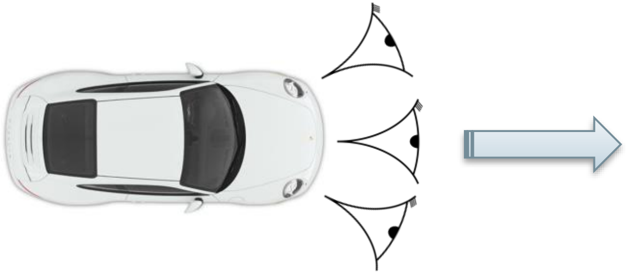
Video: Bojarski et al. ‘16, NVIDIA

Why did that work?

At training time:

3 camera views are used to simulate deviations from the human trajectory.

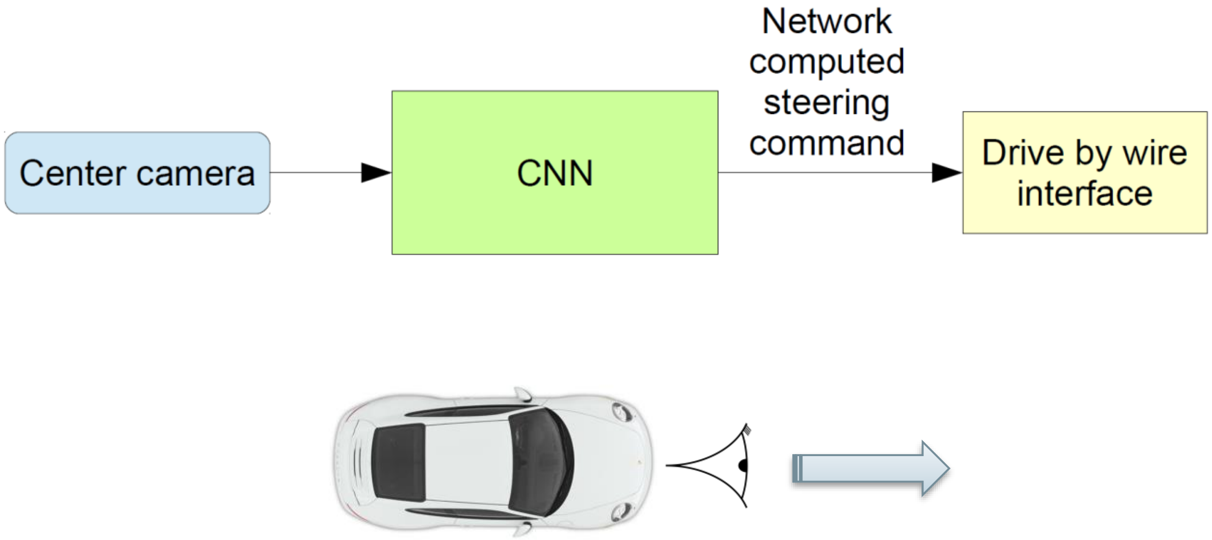




Bojarski et al. ‘16, NVIDIA

Why did that work?

At test time:
Control by center camera only.

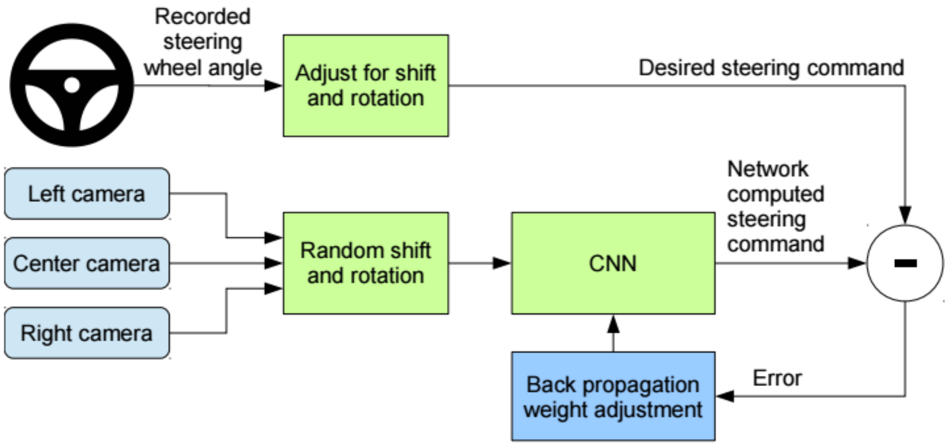


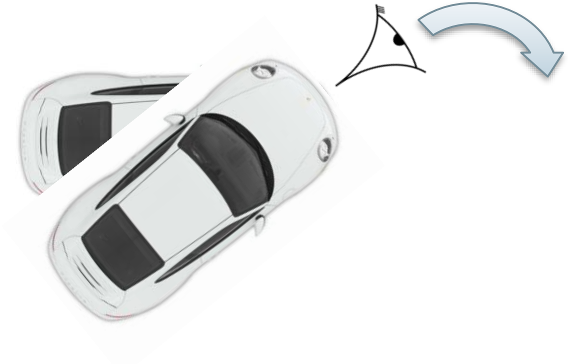
Bojarski et al. ‘16, NVIDIA

Why did that work?

At training time:
Left camera view is similar to what center camera would see if the vehicle were heading to the left.

It should steer more to the right.



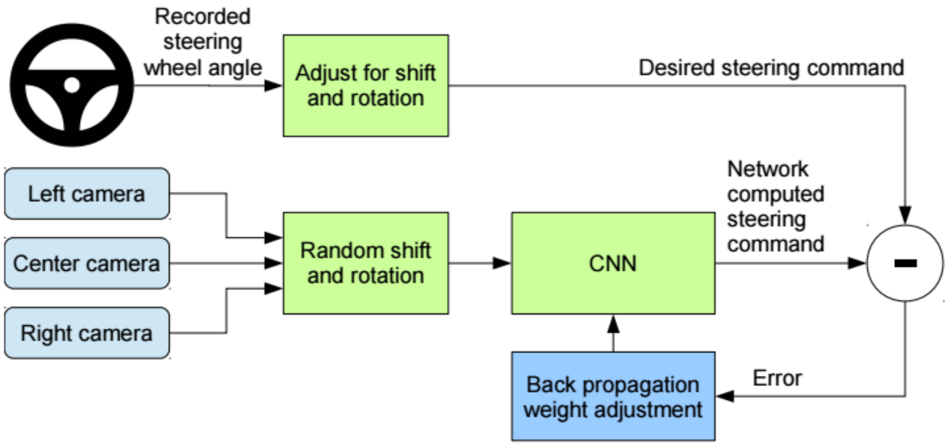


Bojarski et al. ‘16, NVIDIA

Why did that work?

At training time:
Right camera view is similar to what center camera would see if the vehicle were heading to the right.

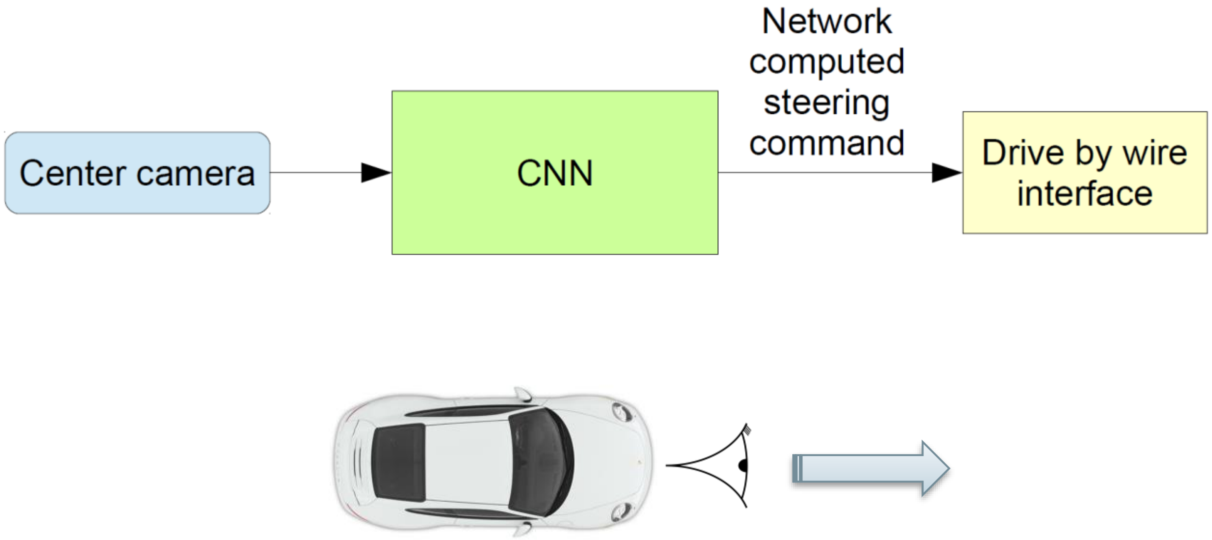
It should steer more to the left.



Bojarski et al. ‘16, NVIDIA

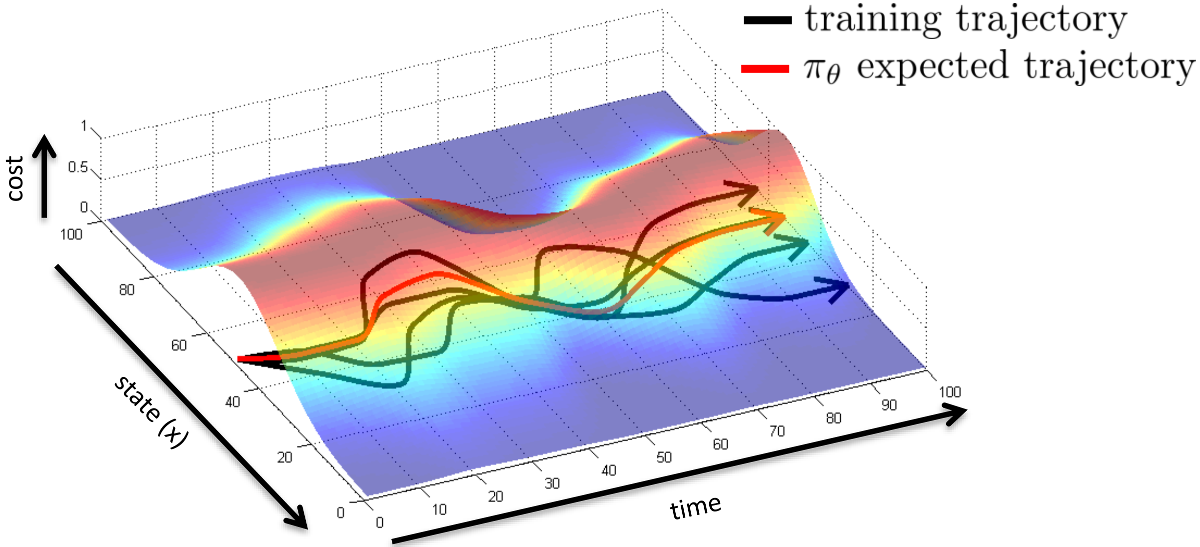
At Test Time

At test time:
Control by center camera only.



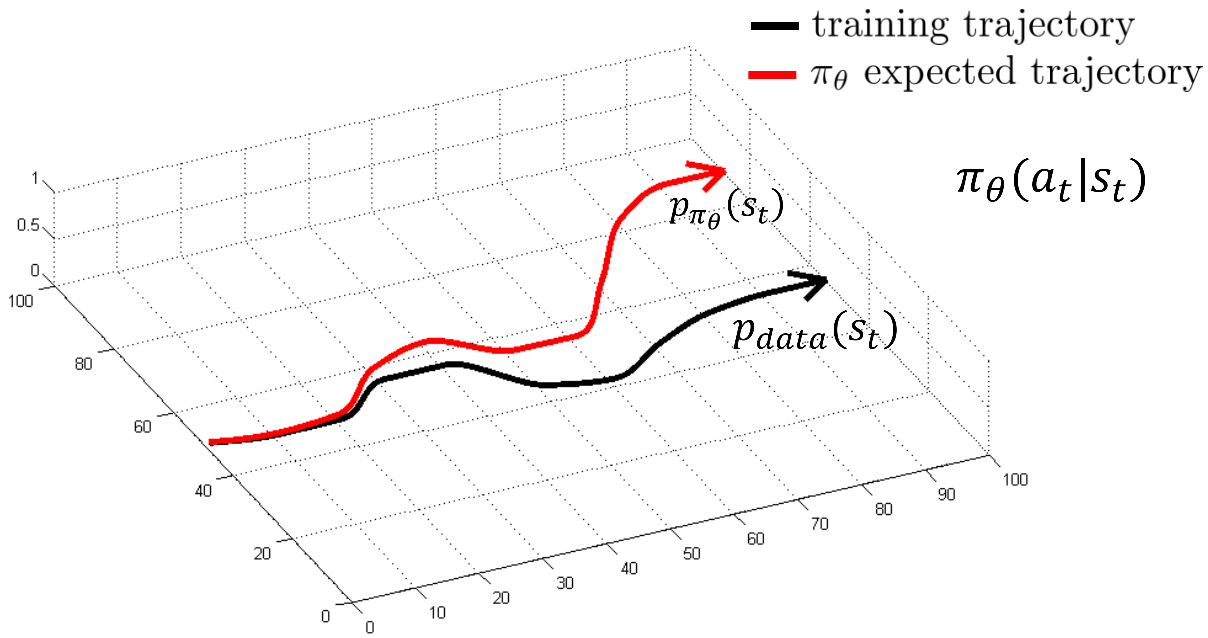
Bojarski et al. ‘16, NVIDIA

Can we make it work more often?



stability

Can we make it work more often?

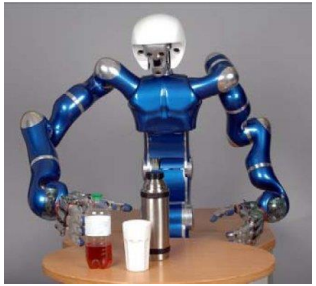


$\pi_\theta(a_t|s_t)$

Can we make $p_{data}(s_t) = p_{\pi_\theta}(s_t)$?

More Terminology

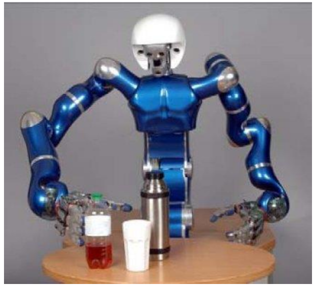
Behavior Policy: The policy $\pi_\theta(a|s)$ that the agent uses to act in the world.



Target Policy: A policy $\pi^t_{\theta^t}(a|s)$ the agent is learning.

More Terminology

On Policy: Agent learns from its own experience, so target policy = behavior policy.



Off Policy: Target policy ≠ behavior policy. More general. Can use experience from other agents

Training

On Policy training data is much easier to learn from.

The human operator visits states with distribution $p_{data}(s_t)$.

The policy visits states with distribution $p_{\pi_\theta}(s_t)$.

If $p_{data}(s_t) \neq p_{\pi_\theta}(s_t)$, then the agent is visiting states at different frequency from the human. The experience of the human is less useful – this if off-policy learning.

Can we make it work more often?

Can we make $p_{data}(s_t) = p_{\pi_\theta}(s_t)$?

Idea: instead of being clever about $p_{\pi_\theta}(s_t)$, be clever about $p_{data}(s_t)$!

DAGGer: Dataset AGGregation

Goal: collect training data from $p_{\pi_\theta}(s_t)$ instead of $p_{data}(s_t)$

How? Just run $\pi_\theta(a_t|s_t)$

But we need "labels" a_t

1. Train $\pi_\theta(a_t|s_t)$ from human data $\mathcal{D} = \{s_1, a_1, \dots, s_N, a_N\}$
2. Run $\pi_\theta(a_t|s_t)$ to get a dataset $\mathcal{D}_\pi = \{s_1, \dots, s_N\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions a_t
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

Current policy
$\pi_\theta(a_t|s_t)$

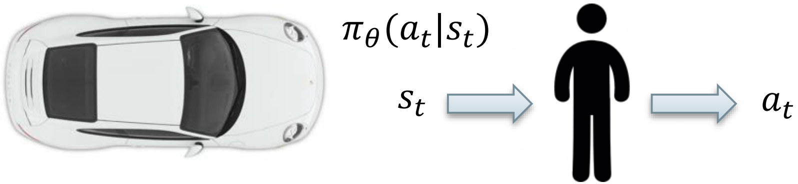
Ross et al. '11

DAgger Example

Ross et al. ‘11

What’s the problem?

1. Train $\pi_\theta(a_t|s_t)$ from human data $\mathcal{D} = \{s_1, a_1, \ldots, s_N, a_N\}$
2. Run $\pi_\theta(a_t|s_t)$ to get a dataset $\mathcal{D}_\pi = \{s_1, \ldots, s_N\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions a_t
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

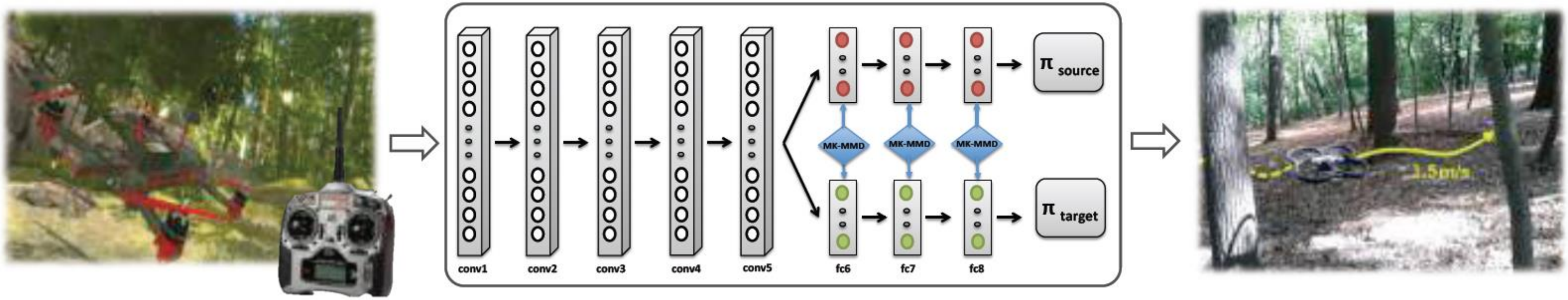


Ross et al. ‘11

Domain Adaptation: Learning Reactive Controls for an MAV

Challenge: It's often much easier to get human training data in an environment different from the target environment (e.g. in simulation).

Developing a controller for the target domain after training in a different domain is a domain adaptation challenge.



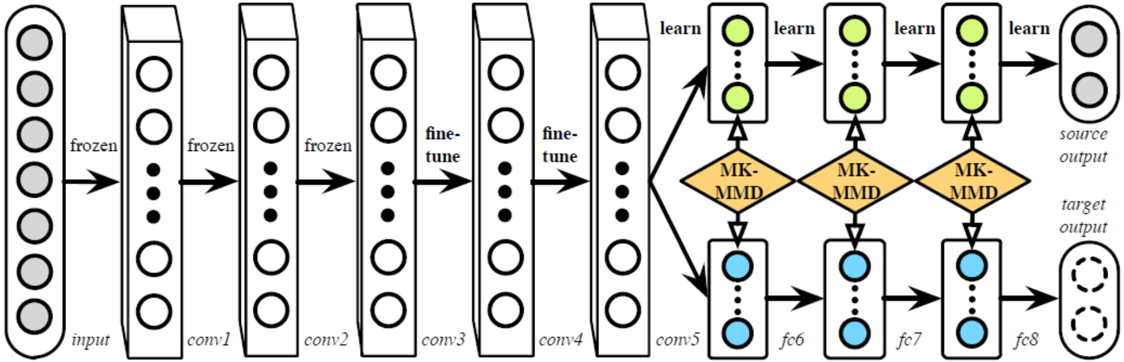
Training Domain

Target Domain

Shreyansh Daftry, J. Andrew Bagnell, and Martial Hebert, "Learning Transferable Policies for Monocular Reactive MAV Control" 2016

Domain Adaptation: Learning Reactive Controls for an MAV

The domain adaptation network shares early layers, fine-tunes last CNN layers, and replicates FC layers:



It minimizes a composite loss:

$$\min_{\Theta} \underbrace{\frac{1}{n_s}\sum_{i=1}^{n_s} J(\theta(x_i^s), y_i^s)}_{\text{Label loss}} + \lambda \underbrace{\sum_{l=l_1}^{l_2} d_k^2(\mathcal{D}_s^l, \mathcal{D}_t^l)}_{\text{Distribution loss}}$$

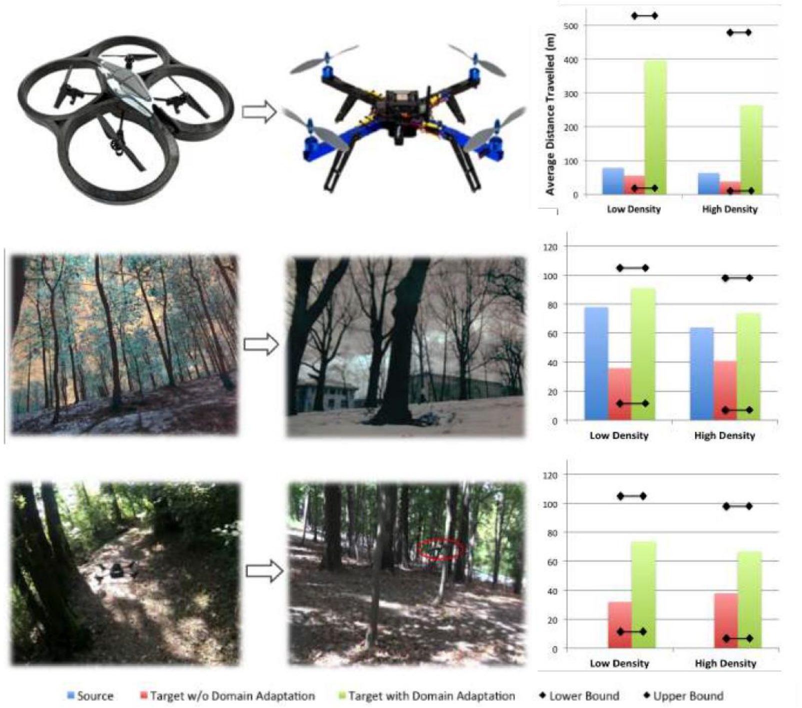
Hidden layer activations (layers 6 – 8)

Train on labeled source data, and unlabeled target data.

Daftry et al. 2016

Domain Adaptation: Learning Reactive Controls for an MAV

Examples:



Experiments and Results for (Row-1) Transfer across physical systems from ARDrone to ArduCopter, (Row-2) Transfer across weather conditions from summer to winter and (Row-3) Transfer across environments from Univ. of Zurich to CMU.

Daftry et al. 2016

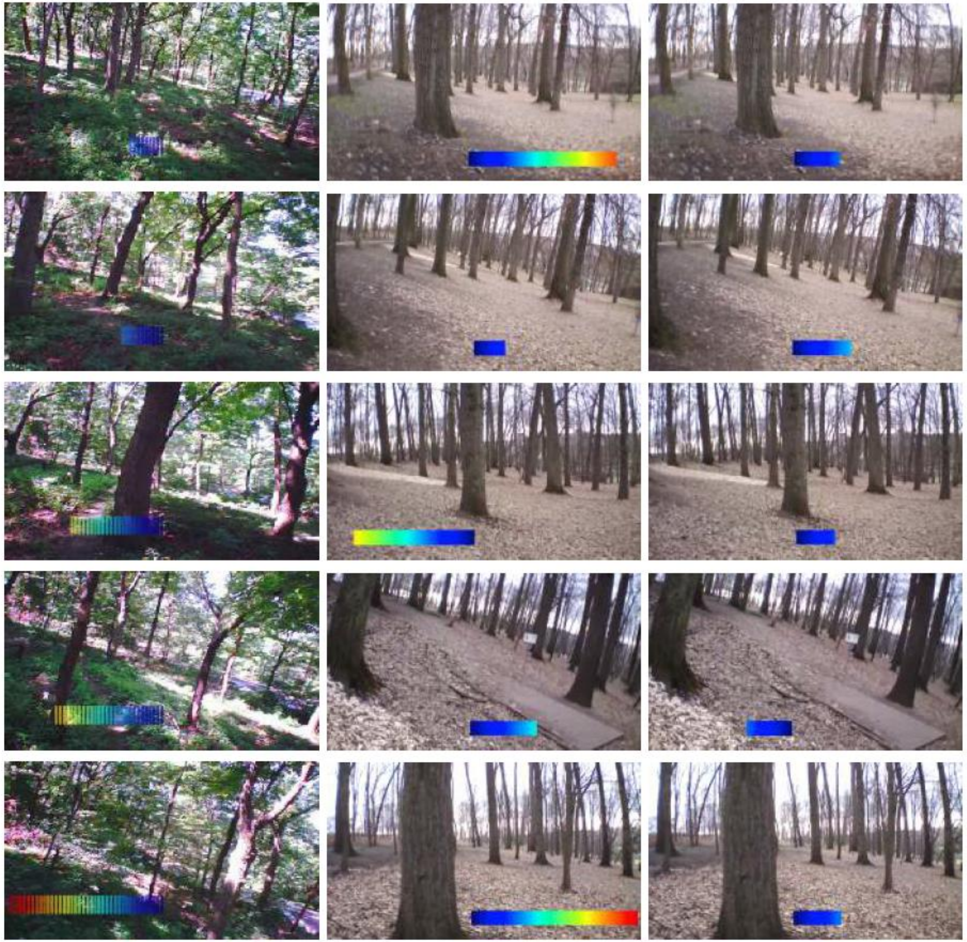
Reactive MAV Controls

Qualitative visualization of an example flight in dense forest.

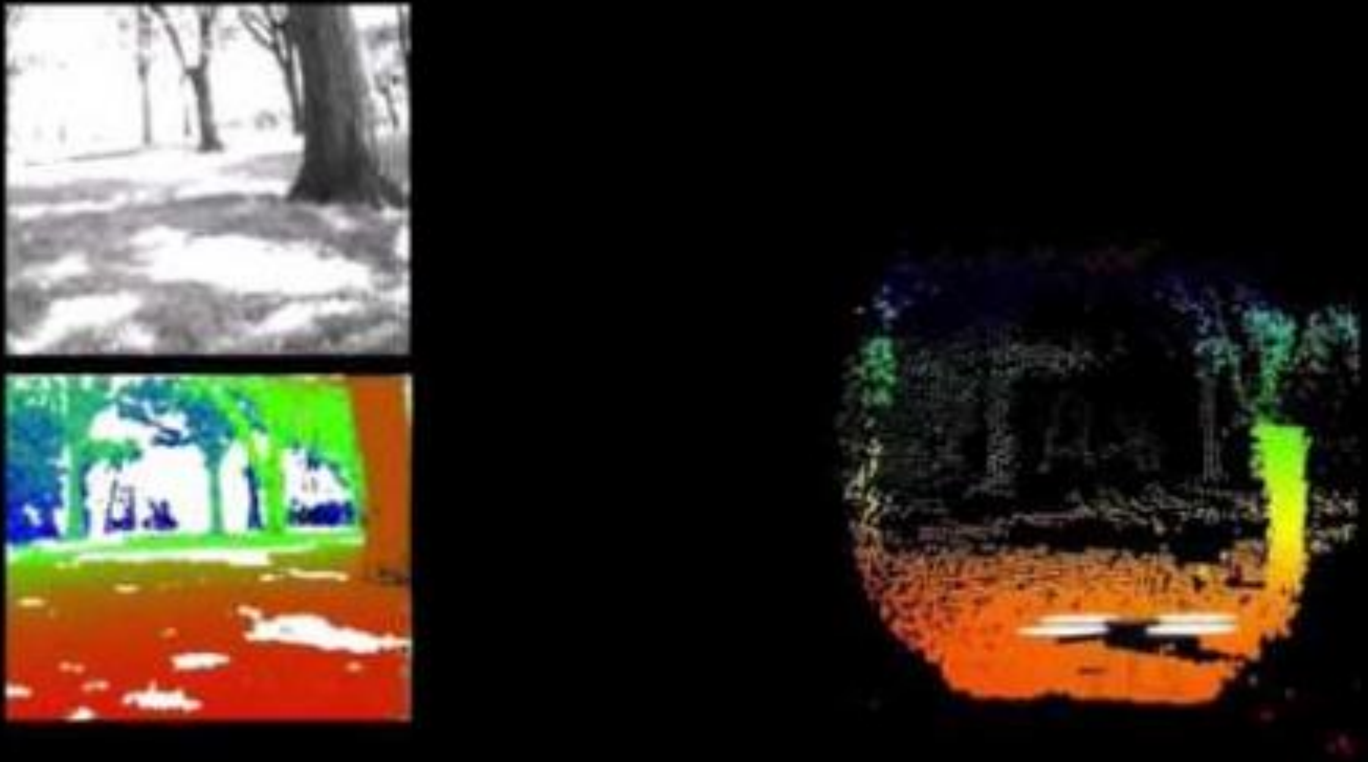
The training data was collected from the same environment during summer season (Col-1) and tested during the winter season (Col-2).

The image sequence of MAVs on-board view is chronologically ordered from top to bottom and overlaid with color-coded commands issued by the policy learned using our proposed approach.

Additionally, we also compute the commands that would have been generated by the policy without domain adaptation (Col-3), for qualitative comparison.



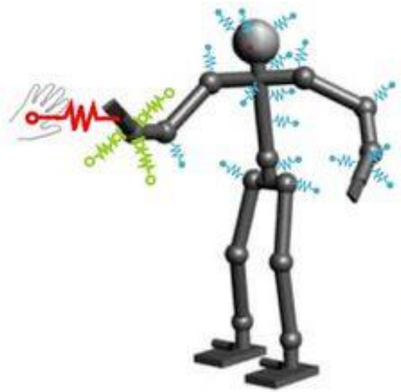
Daftry et al. 2016



GAIL: Generative Adversarial Imitation Learning

Rather than trying to mimic the user blindly, try to solve the same control problem that the user is solving. i.e. estimate the user's cost function, and then optimize the cost by training.

This is called Inverse Reinforcement Learning (IRL).



Jonathan Ho and Stefano Ermon, "Generative Adversarial Imitation Learning"

GAIL: Generative Adversarial Imitation Learning

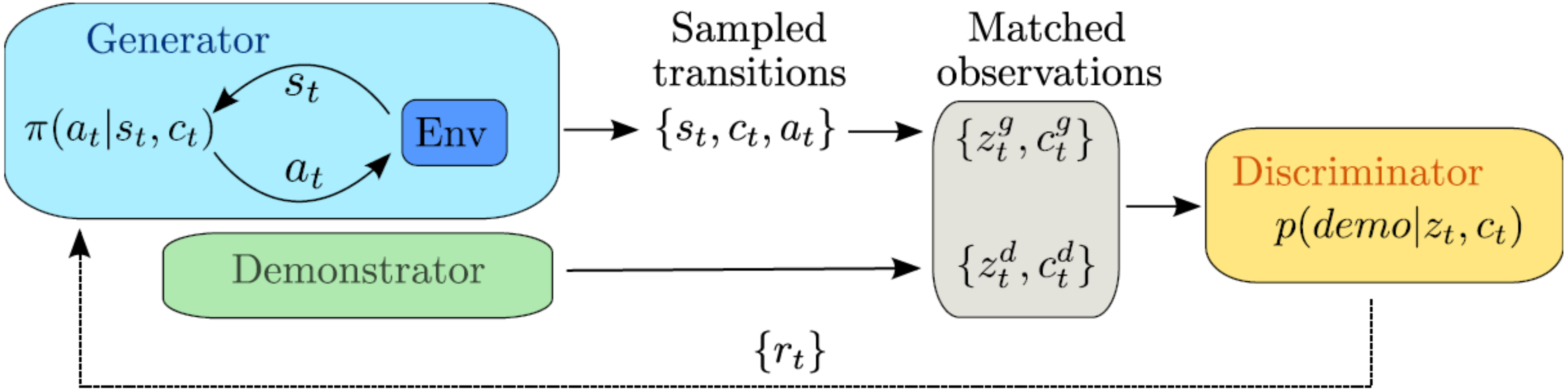


Figure 2: GAIL framework with the addition of context variable, c, for multi-behavior policies. A stochastic policy π interacting with an environment produces trajectories of states and actions (analogous to generator in GAN framework). The state-action pairs are transformed into features, z, which we show may exclude actions. The demonstration data are assumed to be in the same feature space. Either demonstration data or generated data are evaluated by the discriminator to yield a probability of the data being demonstration data. The discriminator provides a reward function for the policy.

GAIL: Generative Adversarial Imitation Learning

Inverse Reinforcement Learning (IRL) is under-constrained, and often uses regularization heuristics.

Entropy regularization: define $H(\pi) \triangleq \mathbb{E}_{\pi}[-\log \pi(a|s)]$

Estimating the cost function is: $\underset{c \in \mathcal{C}}{\text{maximize}} \left(\min_{\pi \in \Pi} -H(\pi) + \mathbb{E}_{\pi}[c(s,a)] \right) - \mathbb{E}_{\pi_E}[c(s,a)]$

Then the imitation learning problem is: $\mathrm{RL}(c) = \underset{\pi \in \Pi}{\arg\min} -H(\pi) + \mathbb{E}_{\pi}[c(s,a)]$

Jonathan Ho and Stefano Ermon, “Generative Adversarial Imitation Learning”

GAIL: Generative Adversarial Imitation Learning

Inverse Reinforcement Learning (IRL) is under-constrained, and often uses regularization heuristics.

Entropy regularization: define $H(\pi) \triangleq \mathbb{E}_{\pi}[-\log \pi(a|s)]$

Entropy is highest for a random policy (random actions at every step).

Entropy is lowest (0) for a deterministic policy that takes a single action at each step.

Jonathan Ho and Stefano Ermon, “Generative Adversarial Imitation Learning”

GAIL: Generative Adversarial Imitation Learning

Inverse Reinforcement Learning (IRL) is under-constrained, and often uses regularization heuristics.

Entropy regularization: define $H(\pi) \triangleq \mathbb{E}_{\pi}[-\log \pi(a|s)]$

Expert trajectory cost

Estimating the cost function is: $\underset{c\in\mathcal{C}}{\text{maximize}} \left(\min_{\pi\in\Pi} -H(\pi) + \mathbb{E}_{\pi}[c(s,a)] \right) - \mathbb{E}_{\pi_E}[c(s,a)]$

Then the imitation learning problem is: $\mathrm{RL}(c) = \underset{\pi\in\Pi}{\arg\min} -H(\pi) + \mathbb{E}_{\pi}[c(s,a)]$

Jonathan Ho and Stefano Ermon, “Generative Adversarial Imitation Learning”

GAIL: Generative Adversarial Imitation Learning

Inverse Reinforcement Learning (IRL) is under-constrained, and often uses regularization heuristics.

Entropy regularization: define $H(\pi) \triangleq \mathbb{E}_{\pi}[-\log \pi(a|s)]$

Estimating the cost function is: $\underset{c \in \mathcal{C}}{\text{maximize}} \left(\min_{\pi \in \Pi} -H(\pi) + \mathbb{E}_{\pi}[c(s,a)] \right) - \mathbb{E}_{\pi_E}[c(s,a)]$

Learned policy cost

Then the imitation learning problem is: $\mathrm{RL}(c) = \underset{\pi \in \Pi}{\arg\min} -H(\pi) + \mathbb{E}_{\pi}[c(s,a)]$

Jonathan Ho and Stefano Ermon, "Generative Adversarial Imitation Learning"

GAIL: Generative Adversarial Imitation Learning

MaxEnt IRL looks for a cost function which assigns low cost to the expert policy, and high cost to other policies.

Estimating the cost function is: $\underset{c \in \mathcal{C}}{\text{maximize}} \left(\min_{\pi \in \Pi} -H(\pi) + \mathbb{E}_{\pi}[c(s,a)] \right) - \mathbb{E}_{\pi_E}[c(s,a)]$

Jonathan Ho and Stefano Ermon, “Generative Adversarial Imitation Learning”

GAIL: Generative Adversarial Imitation Learning

GAIL then uses an adversary to discriminate the expert and learned policies by their state occupancy functions ρ_π and ρ_{π^E}. Don't worry about TRPO for now…

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

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

6: **end for**

Jonathan Ho and Stefano Ermon, “Generative Adversarial Imitation Learning”


Target
Trajectory
Desired Force
Output Velocity
Overtaking
x2 speed
Third person view
Depth image input

GAIL for High-Level Motion Synthesis

Train low-level behavioral policy to match motion capture feature distribution

Train high-level task policy by RL using lower-level behaviors

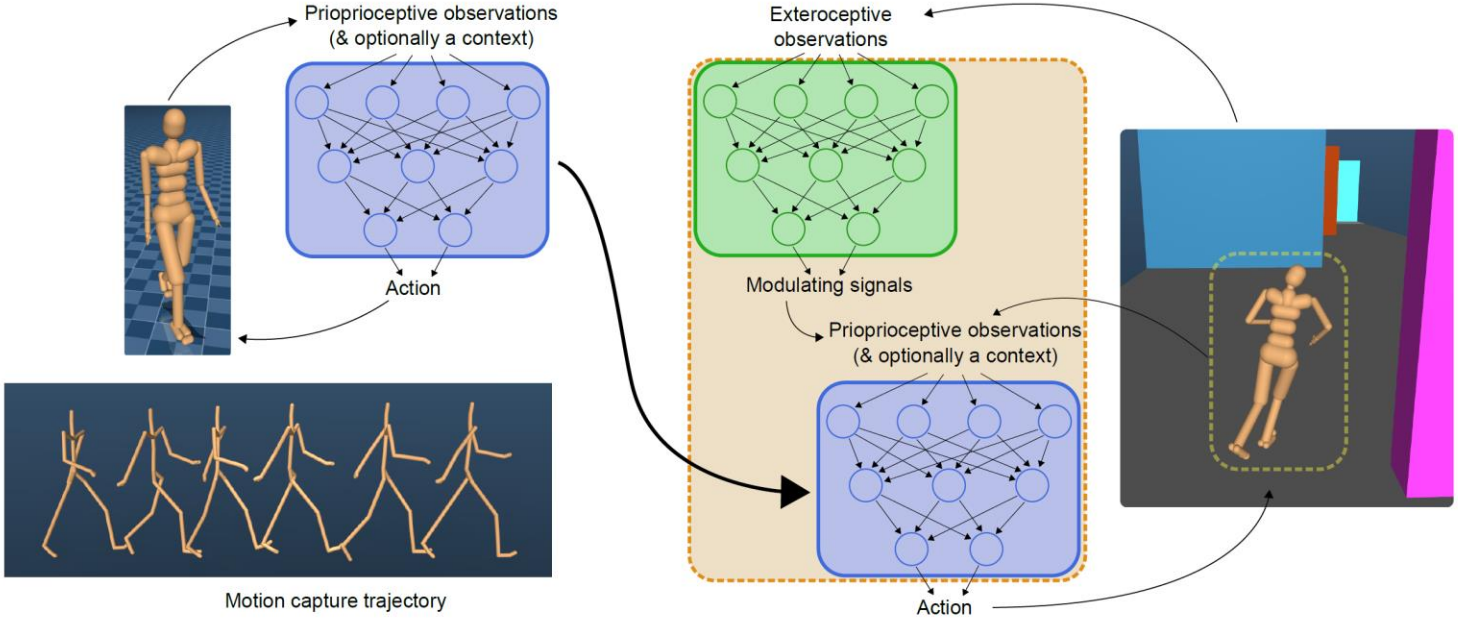
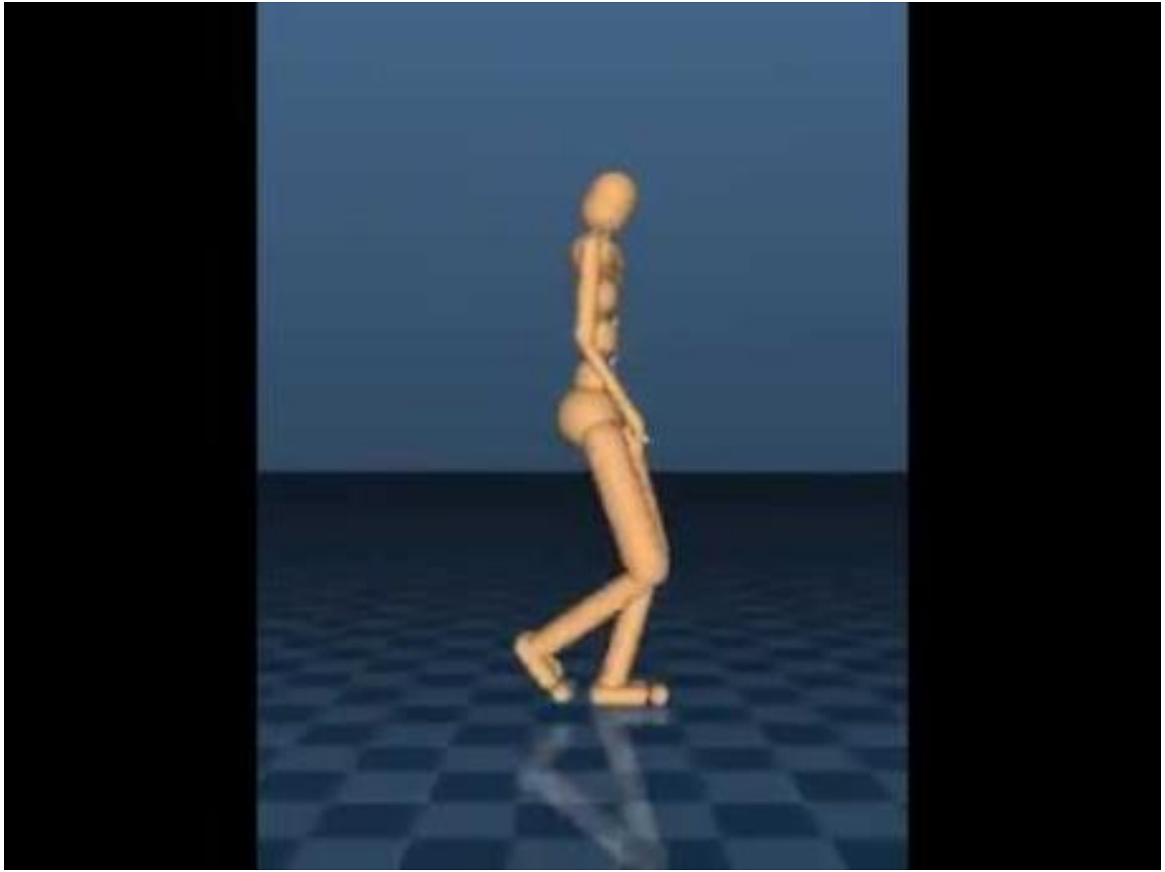


Figure 1: Overview of our approach: (Left) First train specific skills into low-level controller (LLC) policies by imitation learning from motion capture data. (Right) Train a high-level controller (HLC) by RL to reuse pre-trained LLCs.

Merel et al. "Learning human behaviors from motion capture by adversarial imitation"

GAIL for High-Level Motion Synthesis

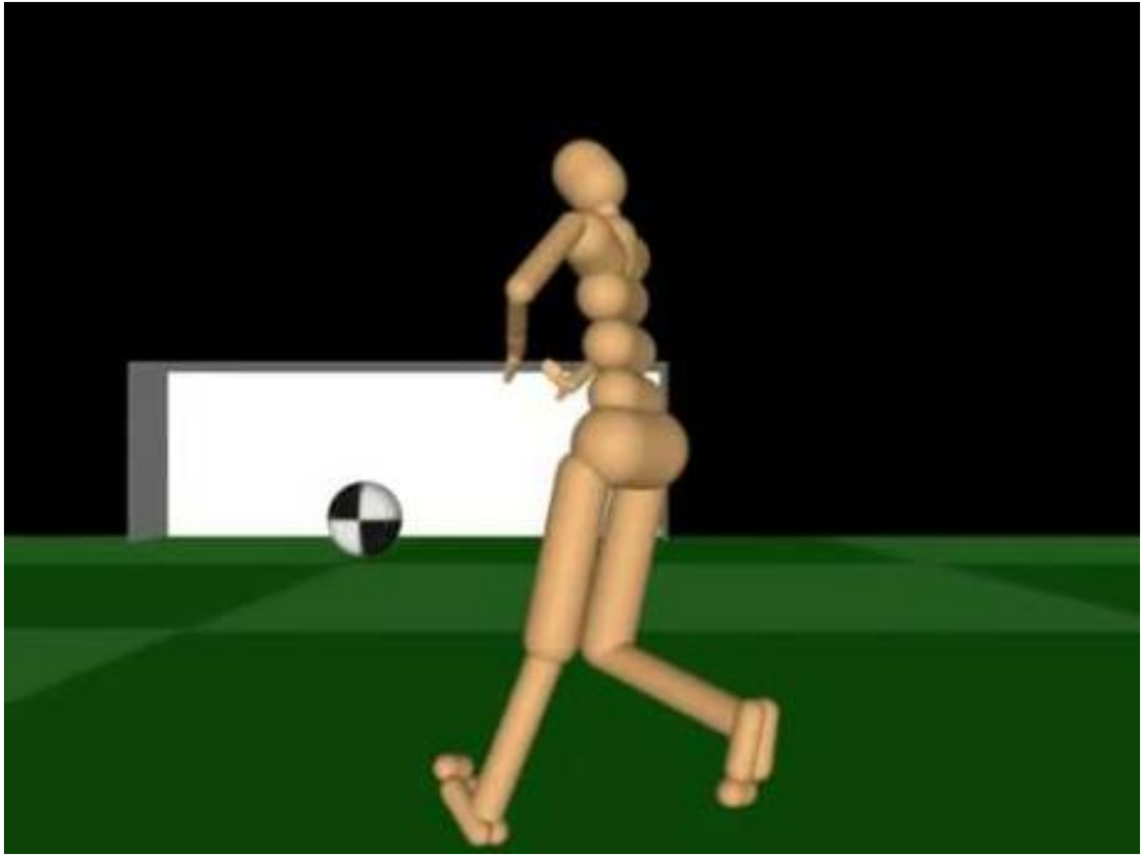


Merel et al. “Learning human behaviors from motion capture by adversarial imitation”

GAIL for High-Level Motion Synthesis

Merel et al. “Learning human behaviors from motion capture by adversarial imitation”

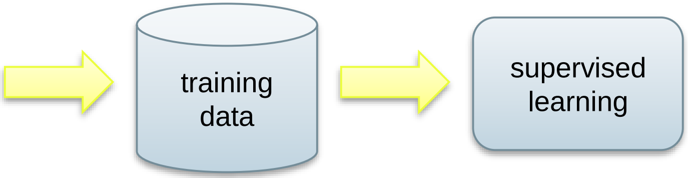
GAIL for High-Level Motion Synthesis



Merel et al. “Learning human behaviors from motion capture by adversarial imitation”

Imitation learning: recap

s_t
a_t



$\pi_\theta(a_t|s_t)$

Usually (but not always) insufficient by itself

- Distribution mismatch problem between human and agent

Sometimes works well

- Ad-hoc data augmentation
- Add more **on-policy** data, e.g. using Dagger
- Domain adaptation and error recovery

GAIL: Define an adversarial reward for learner, then use RL.