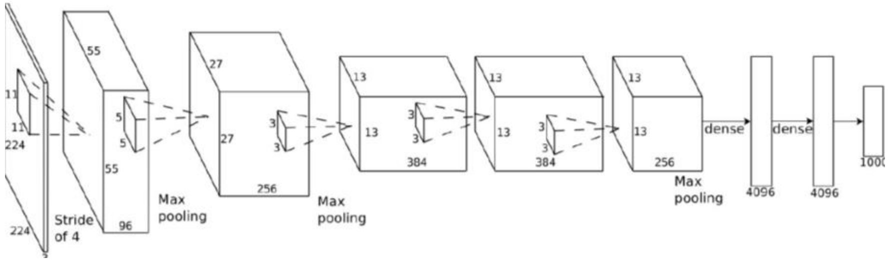
Supervised Learning of Behaviors

CS 285

Instructor: Sergey Levine
UC Berkeley

Terminology & notation

$\mathbf{o}_t$

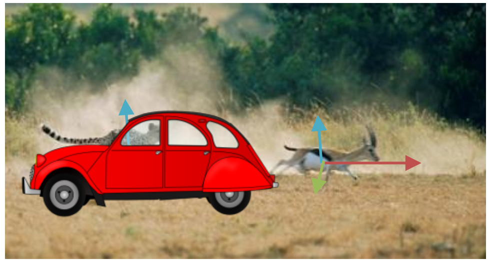


$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

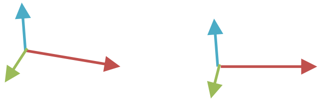
$\mathbf{a}_t$

$\mathbf{s}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{a}_t$ – action

$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ – policy
$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ – policy (fully observed)



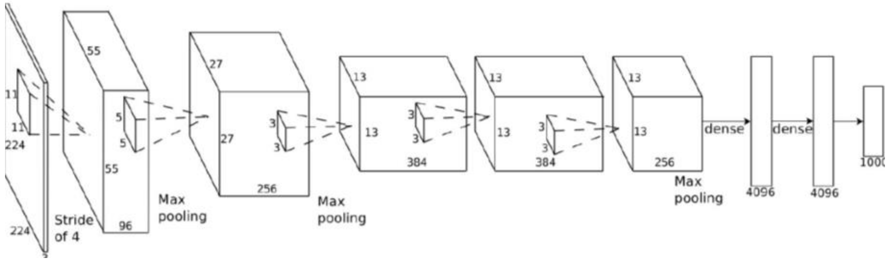
$\mathbf{o}_t$ – observation



$\mathbf{s}_t$ – state

a policy maps states/ observations to actions

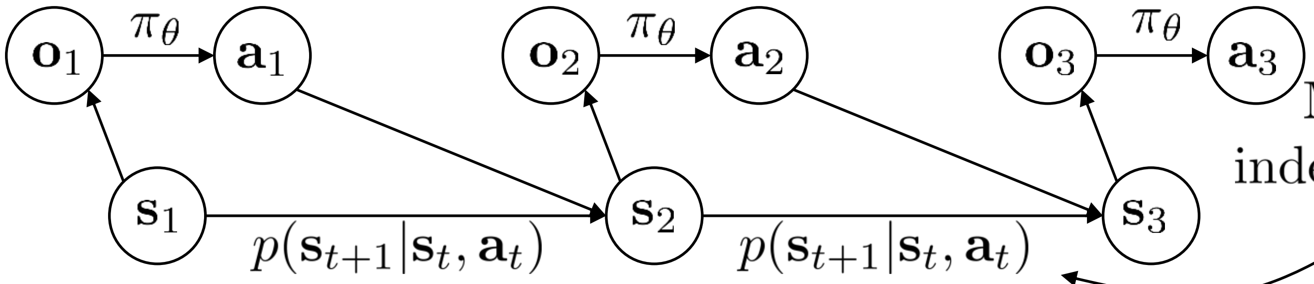
Terminology & notation



$\mathbf{o}_t$ $\quad$ $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ $\quad$ $\mathbf{a}_t$

$\mathbf{s}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{a}_t$ – action

$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ – policy
$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ – policy (fully observed)



Markov property
independent of $\mathbf{s}_{t-1}$

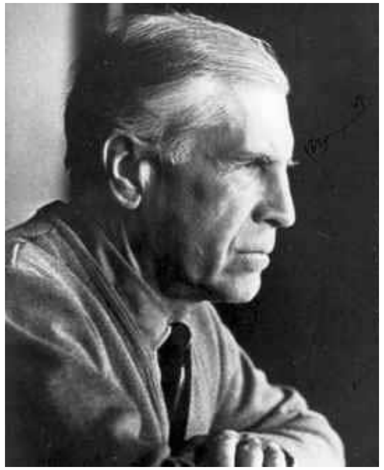
observations are NOT markov!

Aside: notation

$\mathbf{s}_t$ – state
$\mathbf{a}_t$ – action

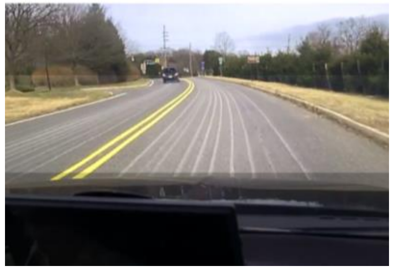
Richard Bellman

$\mathbf{x}_t$ – state
$\mathbf{u}_t$ – action   управление

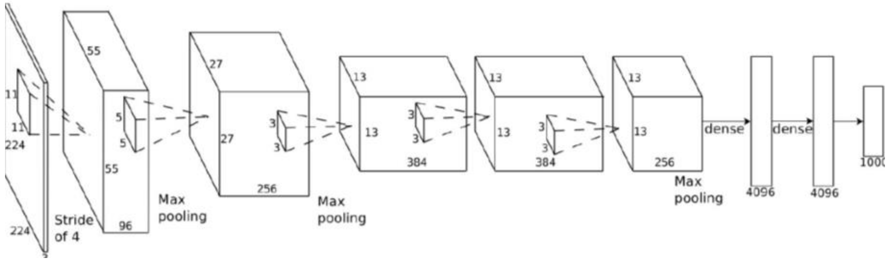


Lev Pontryagin

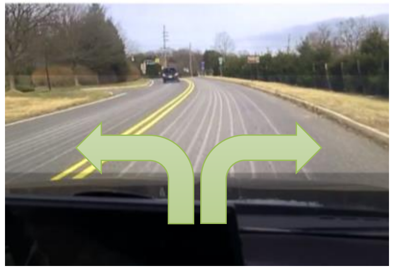
Imitation Learning



$\mathbf{o}_t$



$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

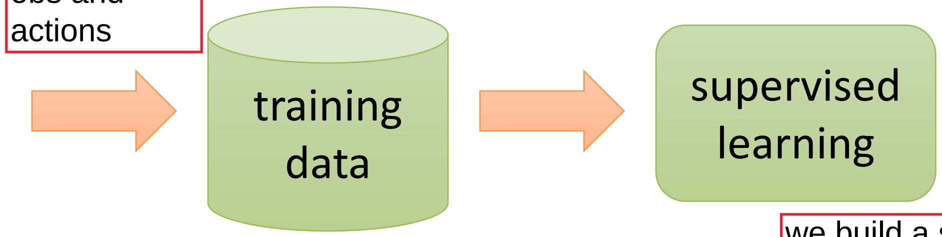


$\mathbf{a}_t$

collect dataset of obs and actions

$\mathbf{o}_t$

$\mathbf{a}_t$



$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

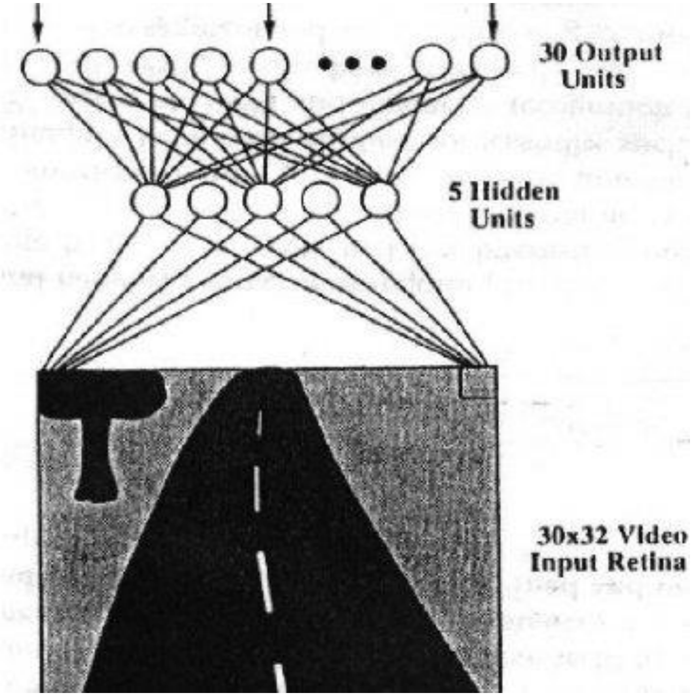
we build a supervised learning model to predict actions based on observations.

behavioral cloning

Images: Bojarski et al. '16, NVIDIA

The original deep imitation learning system

ALVINN: **A**utonomous **L**and **V**ehicle **I**n a **N**eural **N**etwork
1989



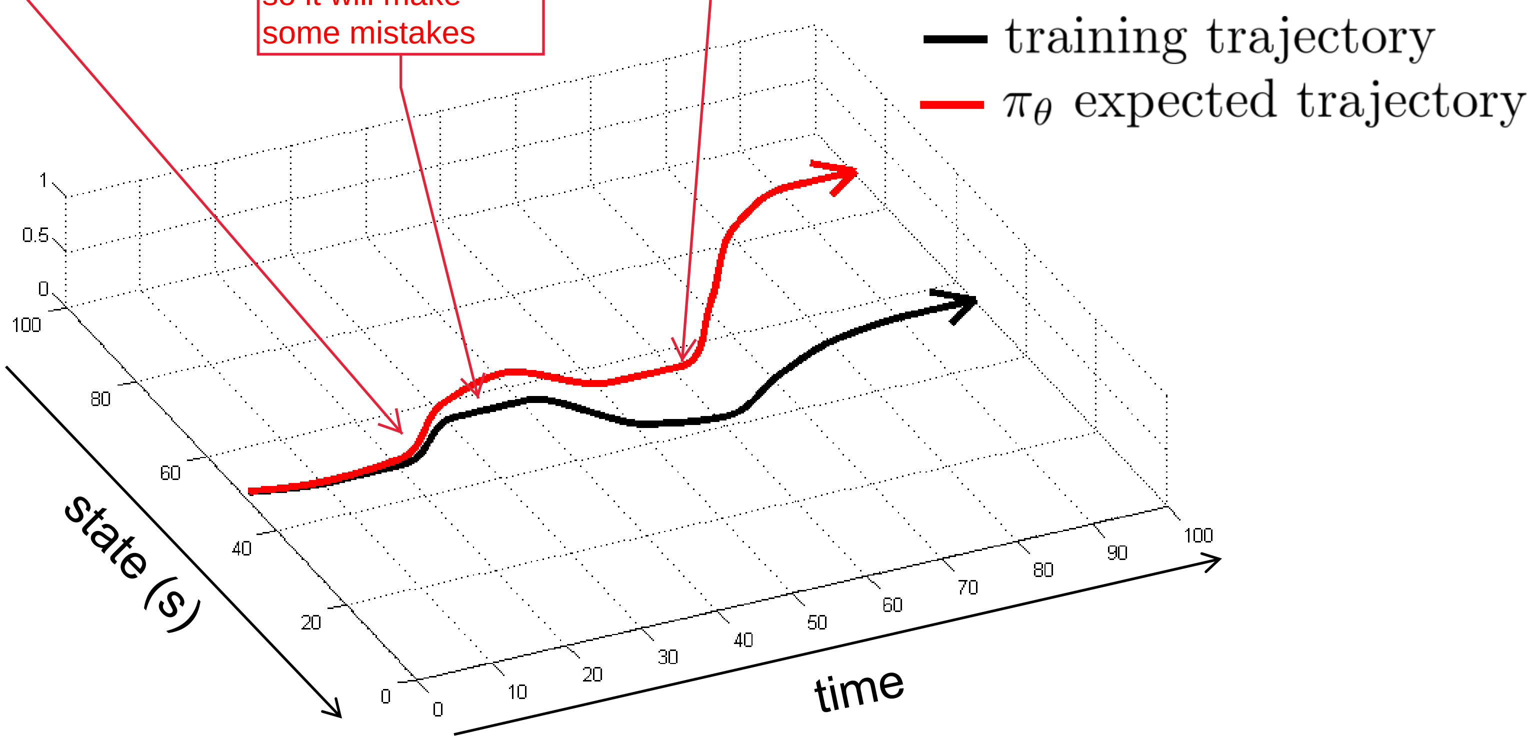
Does it work?

No!

these mistakes compound because when the model makes mistakes, it finds itself in states that are different than those it trained on. Now it doesn't know what to do, so it makes a bigger mistakes

initially the policy stays same to the training data

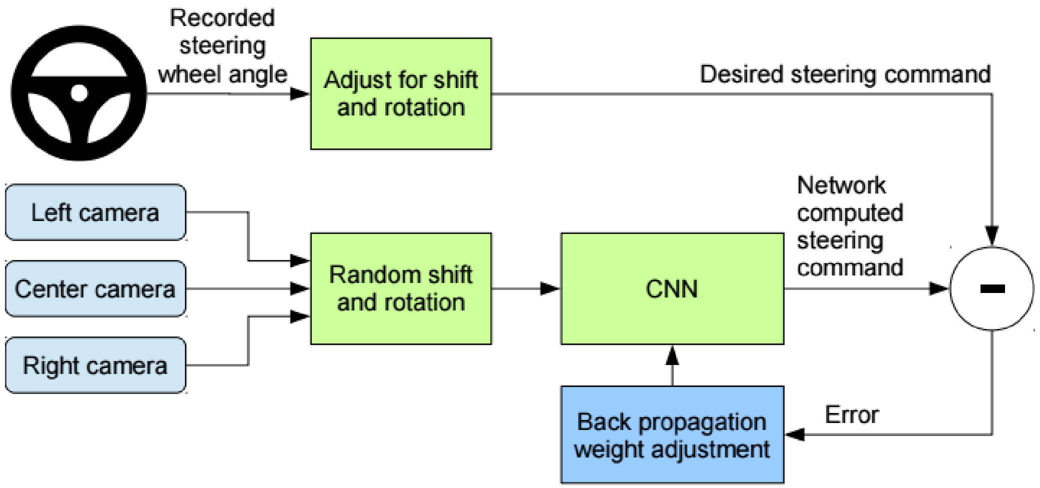
the model will never be perfect, so it will make some mistakes



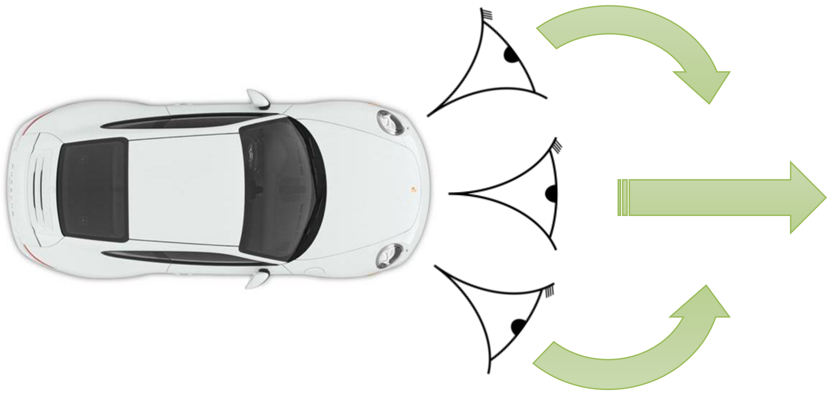
Does it work? Yes!

Video: Bojarski et al. ‘16, NVIDIA

Why did that work?

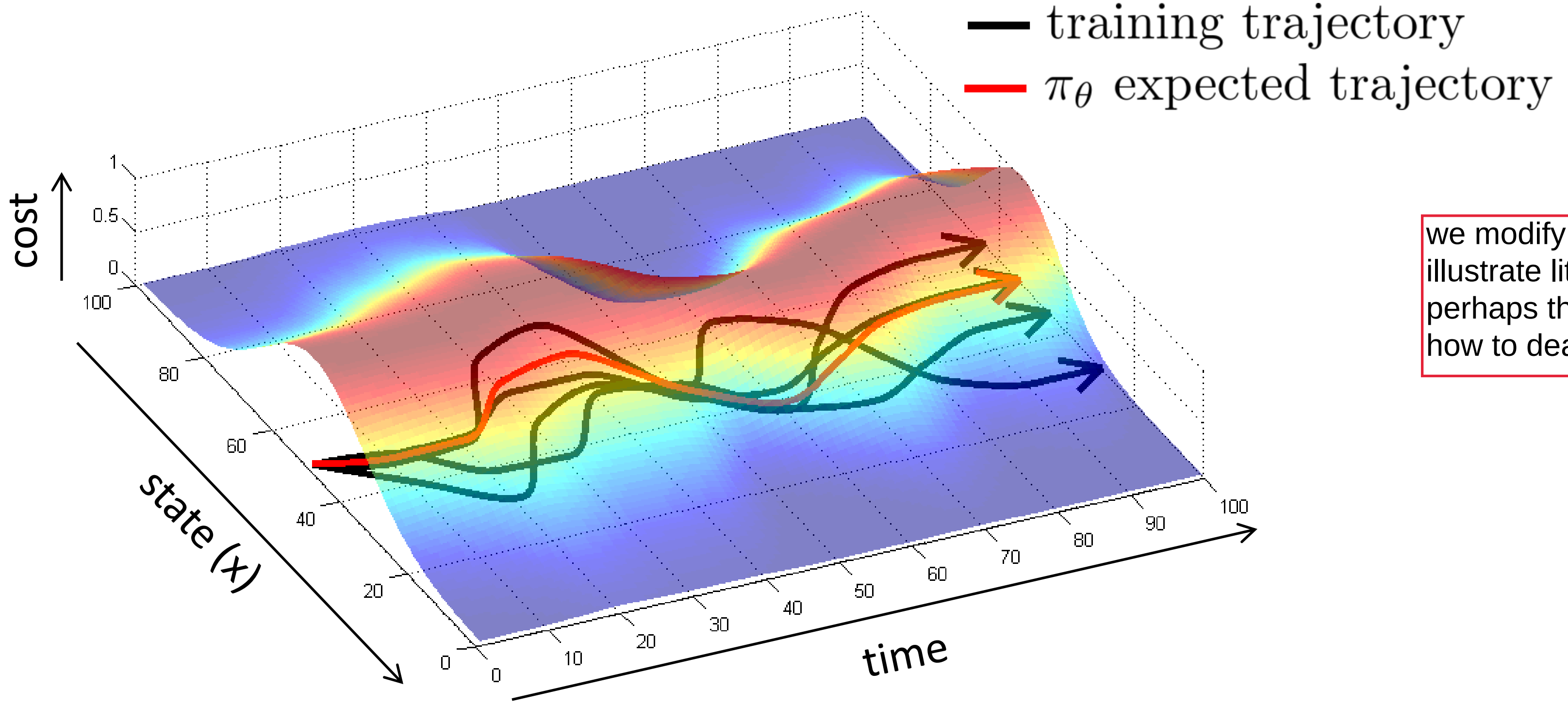


record three cameras at the same time. The camera to the left is supervised to turn to the right. So the right and left cameras are teaching the policy to correct for mistakes



Bojarski et al. '16, NVIDIA

Can we make it work more often?



we modify our training data to illustrate little mistakes, then perhaps the policy can learn how to deal with those mistakes

stability

(more on this later)

Can we make it work more often?

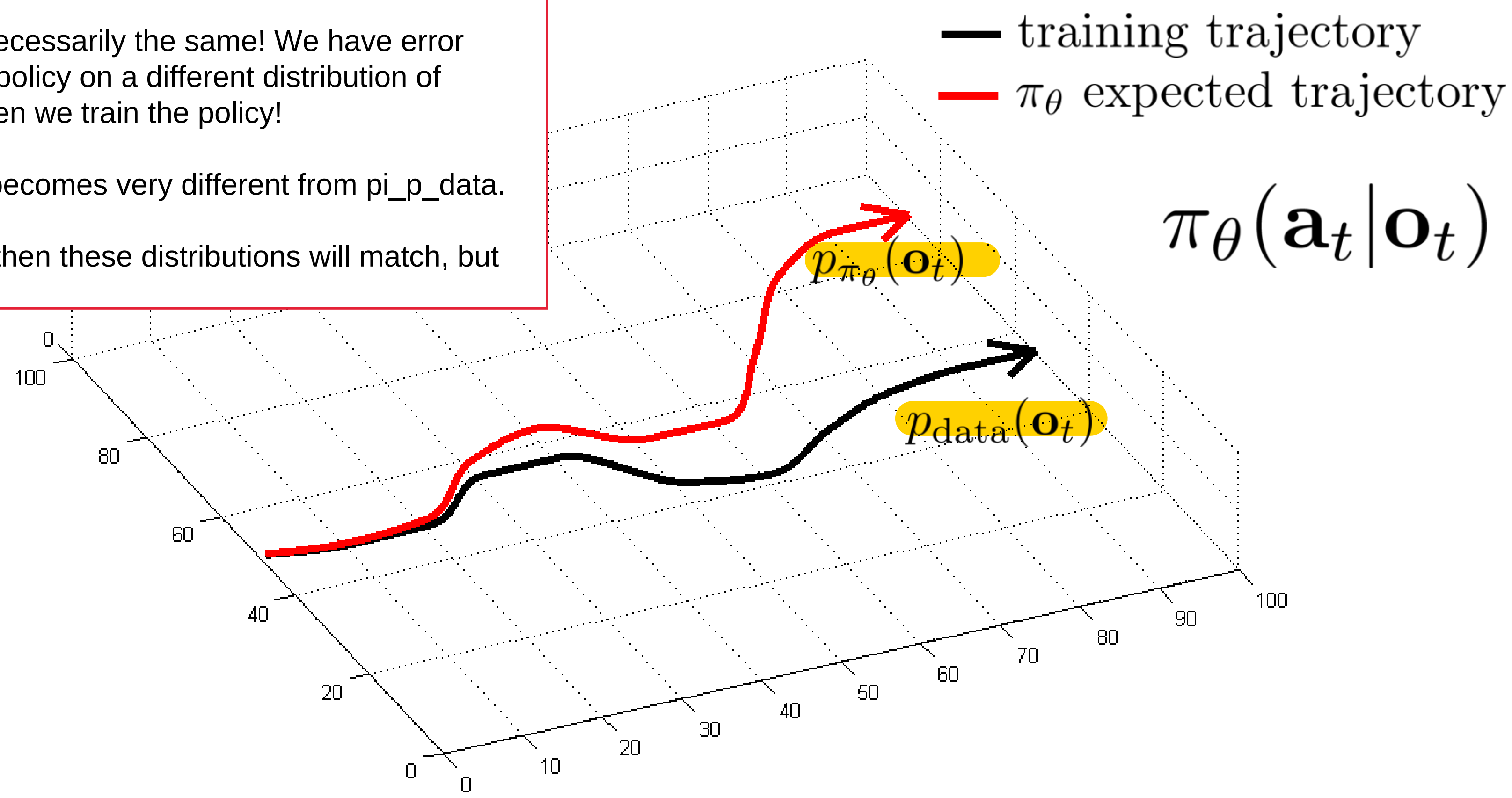
when we run the policy, we sample from p_pi_theta(o_t).

when we train the policy we sample from p_pi_data(o_t).

Those distributions are not necessarily the same! We have error because we are running our policy on a different distribution of observations that we see when we train the policy!

So after a while, p_pi_theta becomes very different from pi_p_data.

If we made pi_theta perfect, then these distributions will match, but that's not realistic.



$$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$$

can we make $p_{\text{data}}(\mathbf{o}_t) = p_{\pi_\theta}(\mathbf{o}_t)$?

Can we make it work more often?

can we make $p_{\text{data}}(\mathbf{o}_t) = p_{\pi_\theta}(\mathbf{o}_t)$?

idea: instead of being clever about $p_{\pi_\theta}(\mathbf{o}_t)$, be clever about $p_{\text{data}}(\mathbf{o}_t)$!

DA**gg**er: Dataset **Agg**regation

goal: collect training data from $p_{\pi_\theta}(\mathbf{o}_t)$ instead of $p_{\text{data}}(\mathbf{o}_t)$

how? just run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

but need labels $\mathbf{a}_t$!

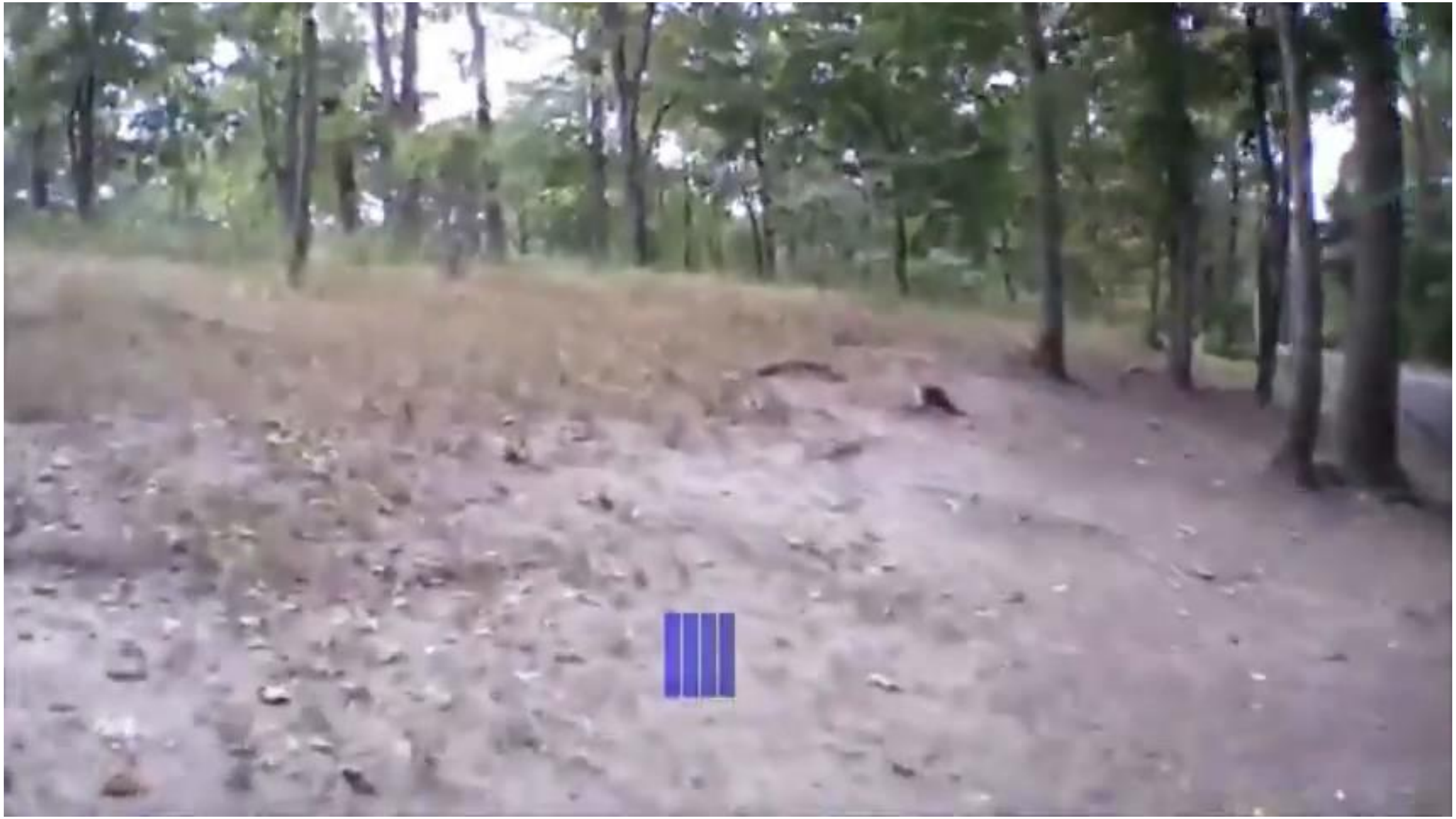
1. train $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{a}_1, \dots, \mathbf{o}_N, \mathbf{a}_N\}$
2. run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \dots, \mathbf{o}_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions $\mathbf{a}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

when we train the policy on the new, augmented dataset, the policy will change. That means that when we run pi_theta, the dataset D_pi will also change.

After doing many rounds of this, this converges.

Ross et al. ‘11

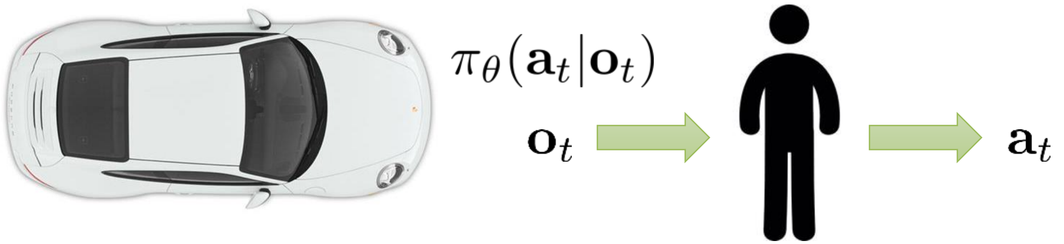
DAgger Example



Ross et al. ‘11

What's the problem?

1. train $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{a}_1, \ldots, \mathbf{o}_N, \mathbf{a}_N\}$
2. run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions $\mathbf{a}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

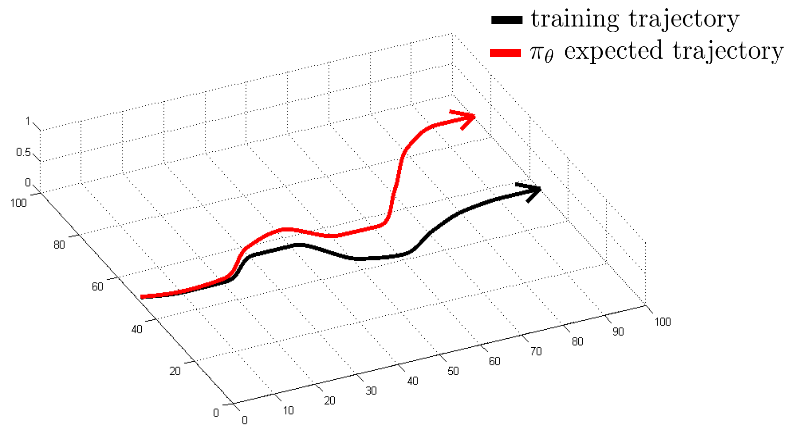


Ross et al. '11

Deep imitation learning in practice

Can we make it work without more data?

- DAgger addresses the problem of distributional “drift”
- What if our model is so good that it doesn’t drift?
- Need to mimic expert behavior very accurately
- But don’t overfit!



Why might we fail to fit the expert?

1. Non-Markovian behavior
2. Multimodal behavior

$$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$$

behavior depends only on current observation

$$\pi_\theta(\mathbf{a}_t|\mathbf{o}_1, ..., \mathbf{o}_t)$$

behavior depends on all past observations

this is how humans actually work

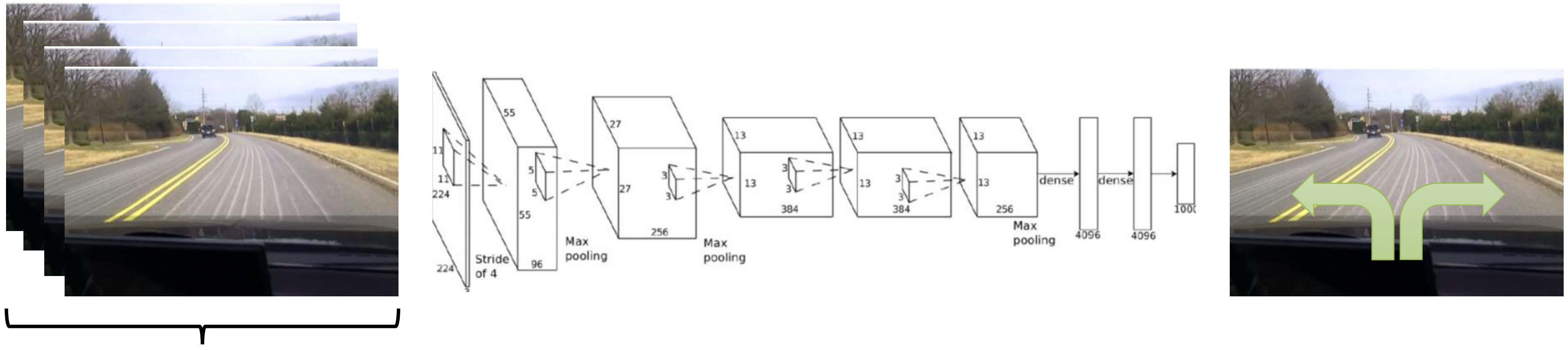
If we see the same thing twice, we do the same thing twice, regardless of what happened before

humans are no consistent!

Often very unnatural for human demonstrators

...to be perfectly Markovian

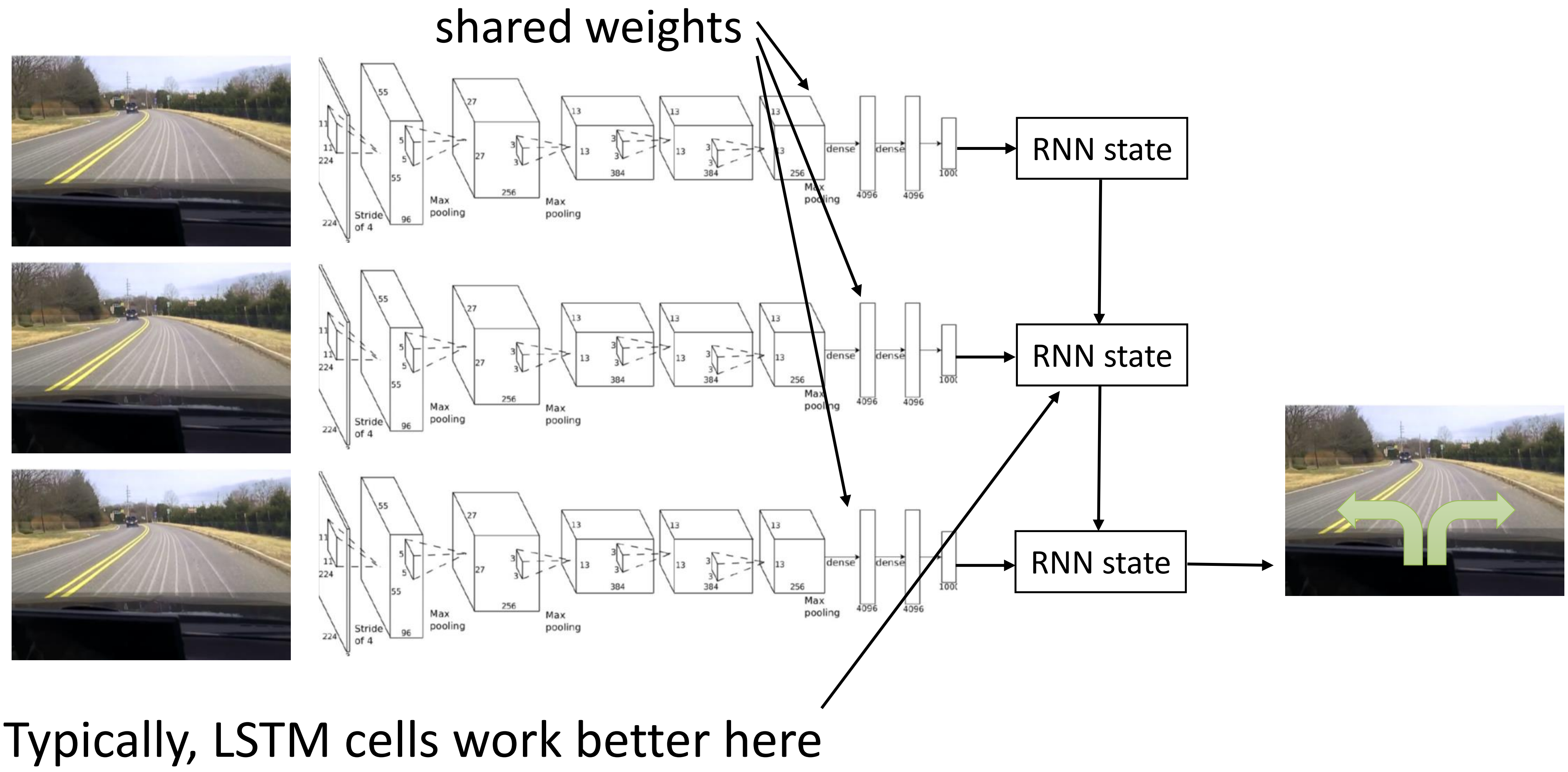
How can we use the whole history?



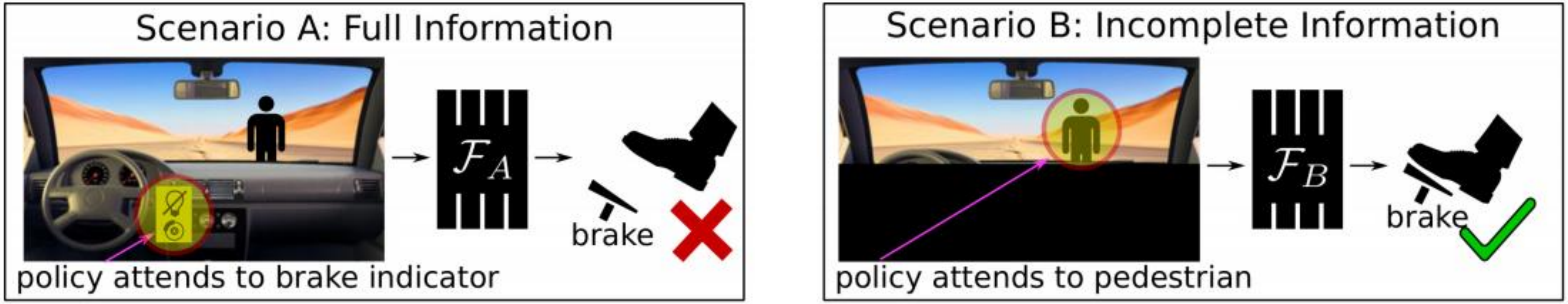
variable number of frames,
too many weights

since the history length differs. Not obvious about how to input this into a CNN

How can we use the whole history?



Aside: why might this work **poorly**?



"causal confusion"

see: de Haan et al., "Causal Confusion in Imitation Learning"

is the breaking caused by the presence of a person, or the light going off?

Question 1: Does including history mitigate causal confusion?

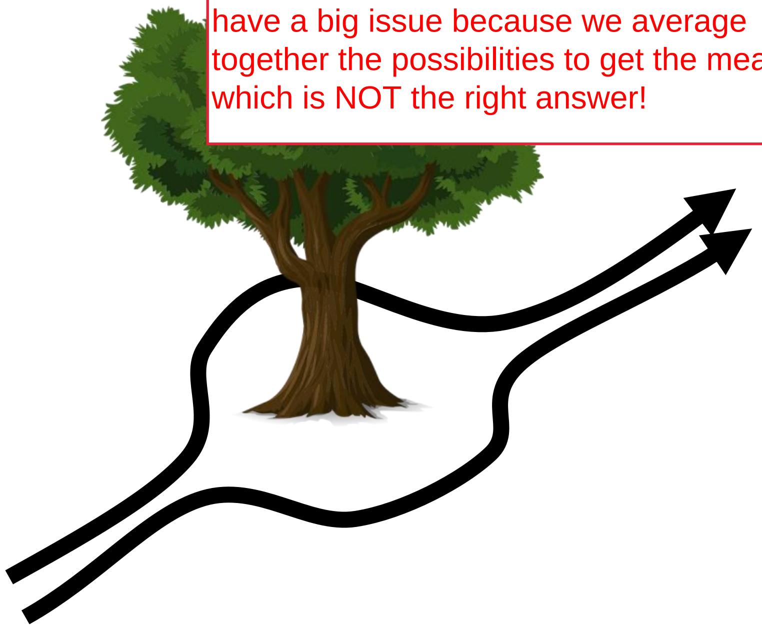
Question 2: Can DAgger mitigate causal confusion?

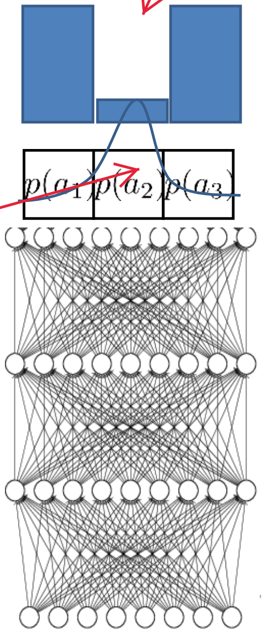
Why might we fail to fit the expert?

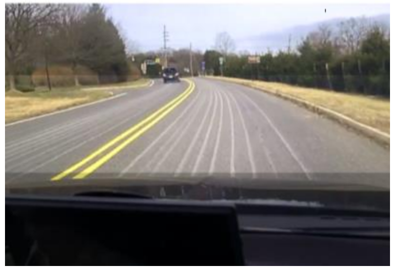
if the action space is discrete, this is not a problem. We use a softmax distribution over the actions, making the "go straight" action highly unlikely

1. Non-Markovian behavior
2. Multimodal behavior

for continuous action space, we parameterize the output distribution as a multi-variate normal distribution determined by its mean and variance. We have a big issue because we average together the possibilities to get the mean, which is NOT the right answer!







SOLUTIONS:

1. Output mixture of Gaussians

instead of outputting the params of of a single gaussian (one mean and variance), output the params of multiple gaussians (multiple means and variances)

2. Latent variable models
3. Autoregressive discretization

Why might we fail to fit the expert?

1. Output mixture of Gaussians

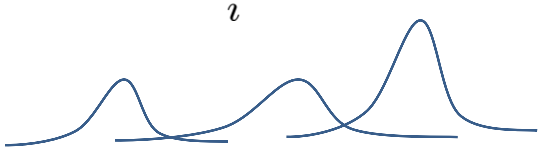
AKA mixture density gaussians

2. Latent variable models
3. Autoregressive discretization

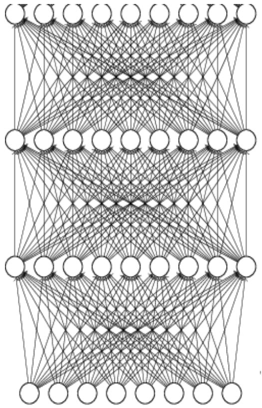
you output N mus, N sigmas, and also N w's.

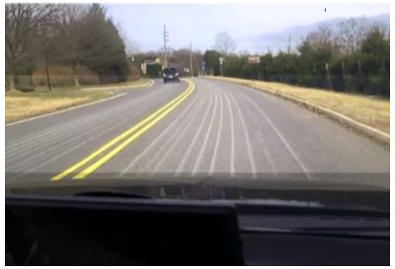
Simple, but suffers in high dimension cases. The higher dimensionality, the more mixture elements you need. And in general, the number of mixture elements needed increases exponentially with dimensionality.

$$\pi(\mathbf{a}|\mathbf{o}) = \sum_i w_i \mathcal{N}(\mu_i, \Sigma_i)$$



$$w_1, \mu_1, \Sigma_1, \ldots, w_N, \mu_N, \sigma_N$$

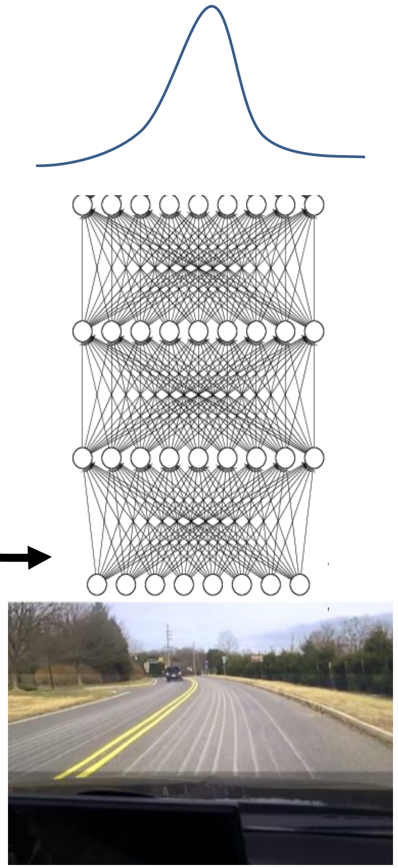




Why might we fail to fit the expert?

1. Output mixture of Gaussians
2. Latent variable models
3. Autoregressive discretization

more sophisticated but more difficult to implement



$\xi \sim \mathcal{N}(0, \mathbf{I})$

in addition to inputting the image, we input a latent variable. Ultimately this is noise.

Look up some of these:

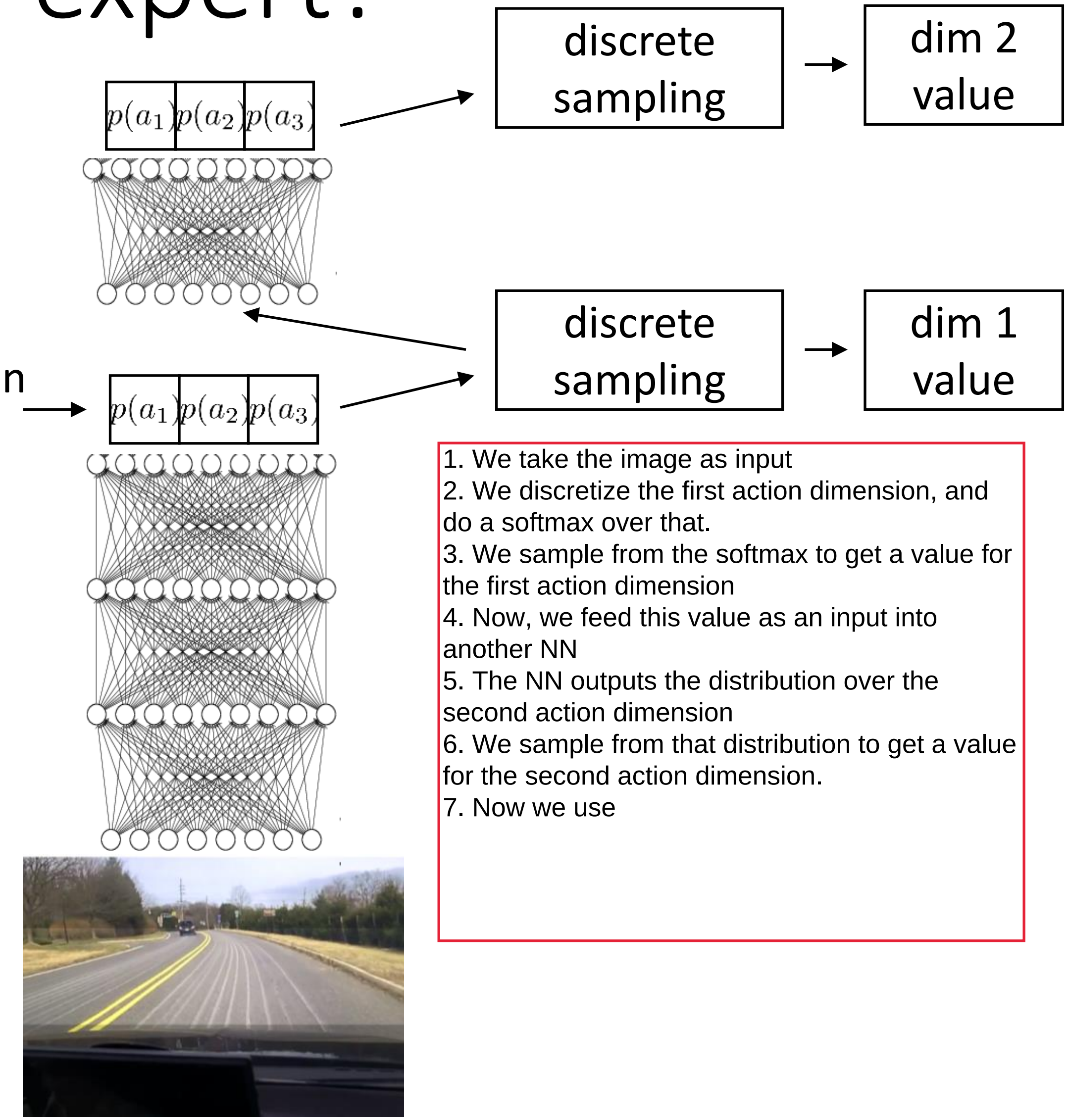
- Conditional variational autoencoder
- Normalizing flow/realNVP
- Stein variational gradient descent

Why might we fail to fit the expert?

1. Output mixture of Gaussians
2. Latent variable models
3. Autoregressive discretization

strikes a good balance of simplicity and expressivity

(discretized) distribution over dimension 1 **only**



recall that with discrete actions, the multiple modality is not an issue. If you have continuous actions, discretizing them could be challenging. The number of states increases exponentially with the dimension.

Autoregressive discretization discretizes one dimension at a time but can still represent arbitrary distributions by using a clever NN trick

1. We take the image as input
2. We discretize the first action dimension, and do a softmax over that.
3. We sample from the softmax to get a value for the first action dimension
4. Now, we feed this value as an input into another NN
5. The NN outputs the distribution over the second action dimension
6. We sample from that distribution to get a value for the second action dimension.
7. Now we use

Imitation learning: recap

$\mathbf{o}_t$

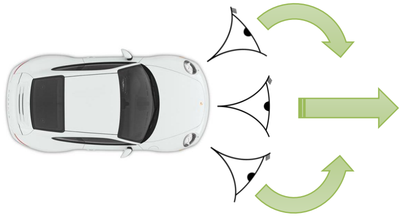
$\mathbf{a}_t$

training data

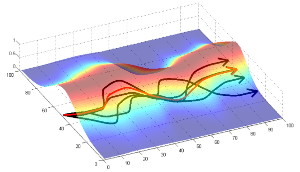
supervised learning

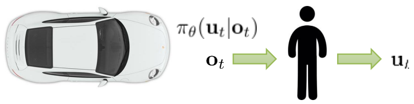
$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

- Often (but not always) insufficient by itself
  - Distribution mismatch problem



- Sometimes works well
  - Hacks (e.g. left/right images)
  - Samples from a stable trajectory distribution
  - Add more **on-policy** data, e.g. using Dagger
  - Better models that fit more accurately



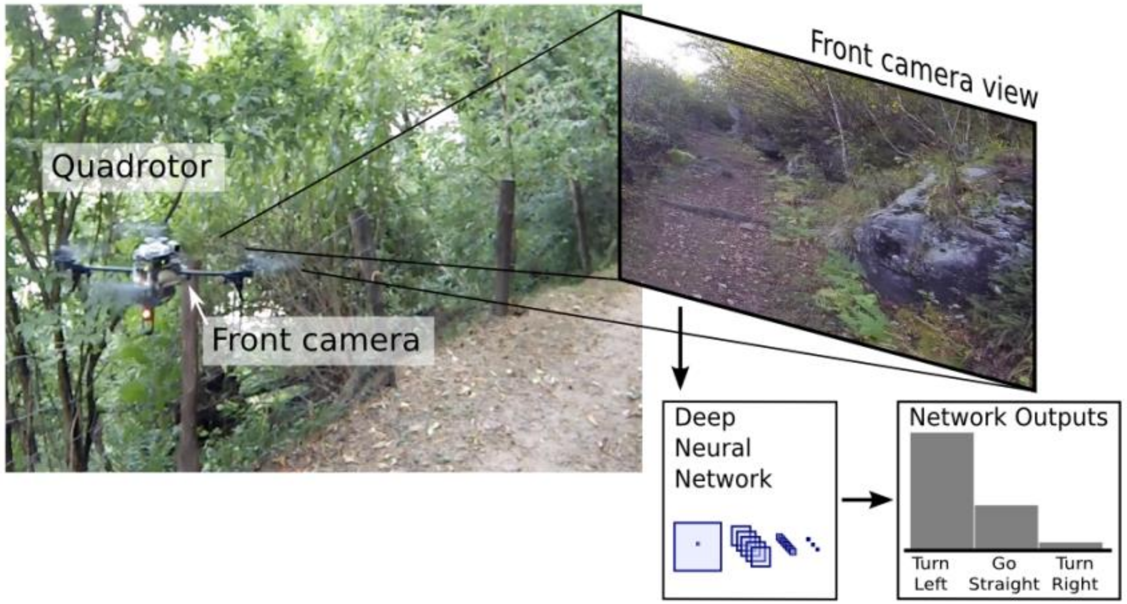


A case study: trail following from human demonstration data

Case study 1: trail following as classification

A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots

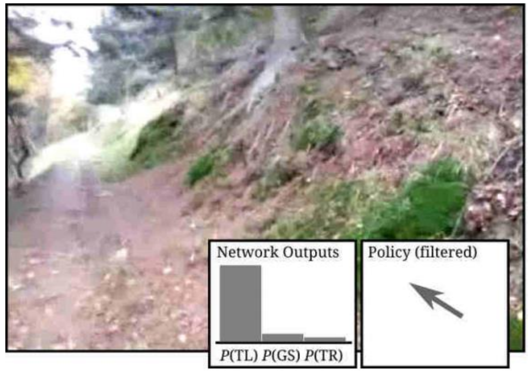
Alessandro Giusti[1], Jérôme Guzzi[1], Dan C. Cireşan[1], Fang-Lin He[1], Juan P. Rodríguez[1]
Flavio Fontana[2], Matthias Faessler[2], Christian Forster[2]
Jürgen Schmidhuber[1], Gianni Di Caro[1], Davide Scaramuzza[2], Luca M. Gambardella[1]

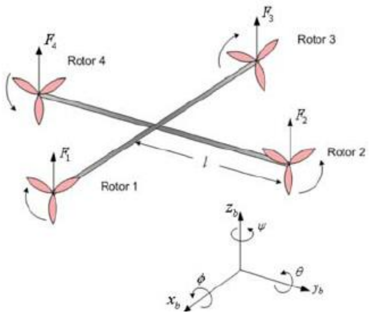


Cost functions, reward functions, and a bit of theory

Imitation learning: what's the problem?

- Humans need to provide data, which is typically finite
  - Deep learning works best when data is plentiful
- Humans are not good at providing some kinds of actions

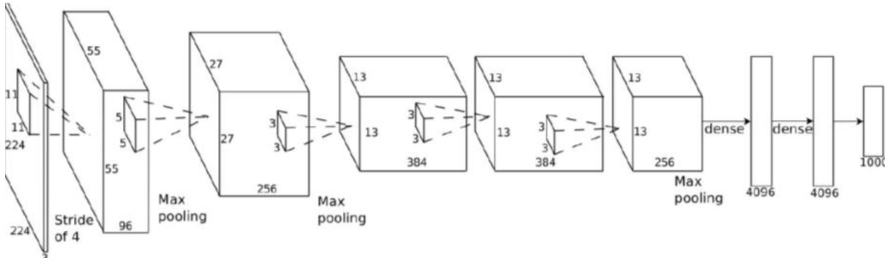




- Humans can learn autonomously; can our machines do the same?
  - Unlimited data from own experience
  - Continuous self-improvement

Terminology & notation

$\mathbf{o}_t$



$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

$\mathbf{a}_t$

$\mathbf{s}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{a}_t$ – action

$c(\mathbf{s}_t, \mathbf{a}_t)$ – cost function
$r(\mathbf{s}_t, \mathbf{a}_t)$ – reward function

$$\min_\theta E_{\mathbf{a}\sim\pi_\theta(\mathbf{a}|\mathbf{s}),\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})}[\delta(\mathbf{s}' = \text{eaten by tiger})]$$

$$\min_\theta E_{\mathbf{s}_{1:T},\mathbf{a}_{1:T}}\left[\sum_t \delta(\mathbf{s}_t = \text{eaten by tiger})\right]$$

$$\min_\theta E_{\mathbf{s}_{1:T},\mathbf{a}_{1:T}}\left[\sum_t c(\mathbf{s}_t, \mathbf{a}_t)\right]$$

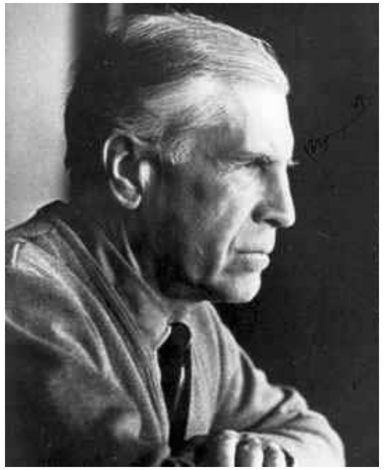
Aside: notation

$\mathbf{s}_t$ – state
$\mathbf{a}_t$ – action
$r(\mathbf{s}, \mathbf{a})$ – reward function

$\mathbf{x}_t$ – state
$\mathbf{u}_t$ – action
$c(\mathbf{x}, \mathbf{u})$ – cost function

$$r(\mathbf{s}, \mathbf{a}) = -c(\mathbf{x}, \mathbf{u})$$

Richard Bellman



Lev Pontryagin

Cost functions, reward functions, and a bit of theory

A cost function for imitation?

the reward must be evaluated in expectation under the learned policy, not the policy of the expert. So we want to match the experts actions in the states that WE actually visit, not just the states that the expert visited. This is why BC is not actually optimizing the correct objective. BC maximizes the log-likelihood in expectation under the state distribution of the EXPERT. We want to maximize it in expectation under the state distribution of the POLICY. Distributional mismatch problem! This is the what DAGGER tries to correct

$\mathbf{o}_t$

$\mathbf{a}_t$

training data → supervised learning → $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

$$r(\mathbf{s},\mathbf{a}) = \log p(\mathbf{a} = \pi^\star(\mathbf{s})|\mathbf{s})$$

log-likelihood. the reward is the log probability of the action taken by the demonstrator

$$c(\mathbf{s},\mathbf{a}) = \begin{cases} 0 \text{ if } \mathbf{a} = \pi^\star(\mathbf{s}) \\ 1 \text{ otherwise} \end{cases}$$

1. train $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{a}_1, \dots, \mathbf{o}_N, \mathbf{a}_N\}$
2. run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \dots, \mathbf{o}_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions $\mathbf{a}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

Ross et al. '11

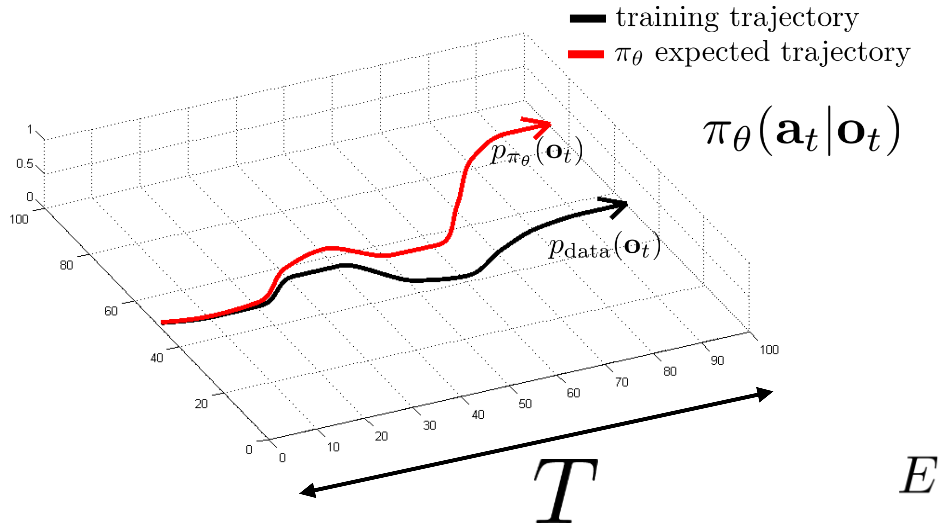
Some analysis

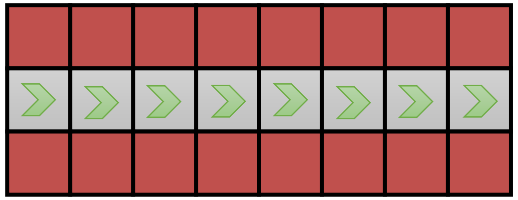
$$c(\mathbf{s}, \mathbf{a}) = \begin{cases} 0 \text{ if } \mathbf{a} = \pi^\star(\mathbf{s}) \\ 1 \text{ otherwise} \end{cases}$$

we are accurate for the states we trained on

assume: $\pi_\theta(\mathbf{a} \neq \pi^\star(\mathbf{s})|\mathbf{s}) \leq \epsilon$

for all $\mathbf{s} \in \mathcal{D}_{\text{train}}$





$$E\left[\sum_t c(\mathbf{s}_t, \mathbf{a}_t)\right] \leq \underbrace{\epsilon T + (1-\epsilon)(\epsilon(T-1) + (1-\epsilon)(\ldots))}_{T \text{ terms, each } O(\epsilon T)}$$

$O(\epsilon T^2)$

as the length of our trajectory increases, our bound increases quadratically. Bad!

tightrope walker.

The walker cannot make one mistake or they fall off. If the policy makes a mistake and falls into the red zone, we have no data on what to do in the red zone because the walker never fell off.

for the first timestep, if they fall off they will make an addition T mistakes

More general analysis

$$c(\mathbf{s},\mathbf{a}) = \begin{cases} 0 \text{ if } \mathbf{a} = \pi^\star(\mathbf{s}) \\ 1 \text{ otherwise} \end{cases}$$

assume: $\pi_\theta(\mathbf{a} \neq \pi^\star(\mathbf{s})|\mathbf{s}) \leq \epsilon$

~~for all $\mathbf{s} \in \mathcal{D}_{\text{train}}$~~ for $\mathbf{s} \sim p_{\text{train}}(\mathbf{s})$

actually enough for $E_{p_{\text{train}}(\mathbf{s})}[\pi_\theta(\mathbf{a} \neq \pi^\star(\mathbf{s})|\mathbf{s})] \leq \epsilon$

with DAgger, $p_{\text{train}}(\mathbf{s}) \rightarrow p_\theta(\mathbf{s})$

$$E\left[\sum_t c(\mathbf{s}_t,\mathbf{a}_t)\right] \leq \epsilon T$$

if $p_{\text{train}}(\mathbf{s}) \neq p_\theta(\mathbf{s})$:

$$p_\theta(\mathbf{s}_t) = \underbrace{(1-\epsilon)^t}_{\text{probability we made no mistakes}} p_{\text{train}}(\mathbf{s}_t) + (1-(1-\epsilon)^t))\underbrace{p_{\text{mistake}}(\mathbf{s}_t)}_{\text{some } \textit{other} \text{ distribution}}$$

probability we made no mistakes

up until now, time=t

some *other* distribution

For more analysis, see Ross et al. "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning"

More general analysis

assume: $\pi_\theta(\mathbf{a} \neq \pi^\star(\mathbf{s})|\mathbf{s}) \leq \epsilon$

for all $\mathbf{s} \in \mathcal{D}_{\text{train}}$ for $\mathbf{s} \sim p_{\text{train}}(\mathbf{s})$

$$p_\theta(\mathbf{s}_t) = \underbrace{(1-\epsilon)^t}_{\text{probability we made no mistakes}} p_{\text{train}}(\mathbf{s}_t) + (1 - (1-\epsilon)^t)) \underbrace{p_{\text{mistake}}(\mathbf{s}_t)}_{\text{some } \textit{other} \text{ distribution}}$$

$$|p_\theta(\mathbf{s}_t) - p_{\text{train}}(\mathbf{s}_t)| = (1-(1-\epsilon)^t)|p_{\text{mistake}}(\mathbf{s}_t) - p_{\text{train}}(\mathbf{s}_t)| \leq 2(1-(1-\epsilon)^t)$$

useful identity: $(1-\epsilon)^t \geq 1 - \epsilon t$ for $\epsilon \in [0,1]$ $\qquad \leq 2\epsilon t$

$$\sum_t E_{p_\theta(\mathbf{s}_t)}[c_t] = \sum_t \sum_{\mathbf{s}_t} p_\theta(\mathbf{s}_t) c_t(\mathbf{s}_t) \leq \sum_t \sum_{\mathbf{s}_t} p_{\text{train}}(\mathbf{s}_t) c_t(\mathbf{s}_t) + |p_\theta(\mathbf{s}_t) - p_{\text{train}}(\mathbf{s}_t)| c_{\max}$$

$$\leq \sum_t \epsilon + 2\epsilon t$$

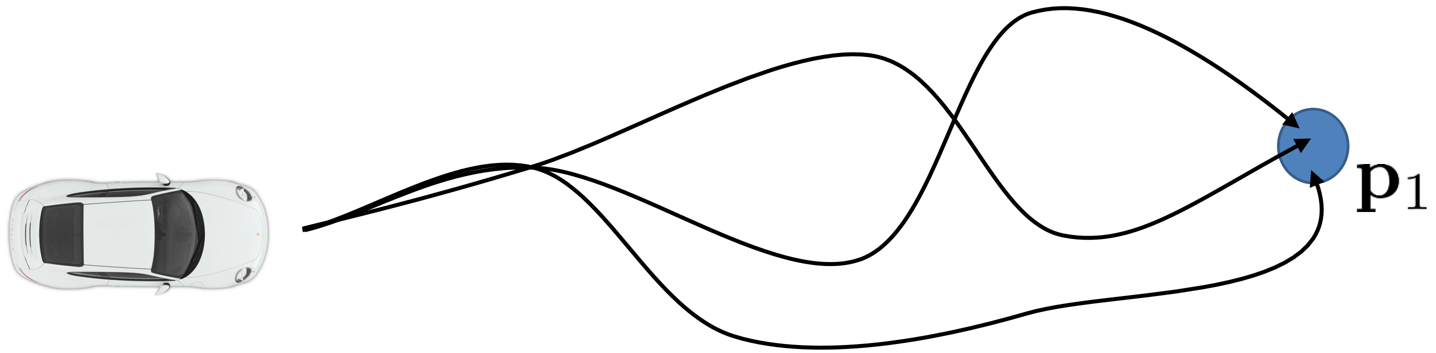
$$O(\epsilon T^2)$$

For more analysis, see Ross et al. “A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning”

Another way to imitate

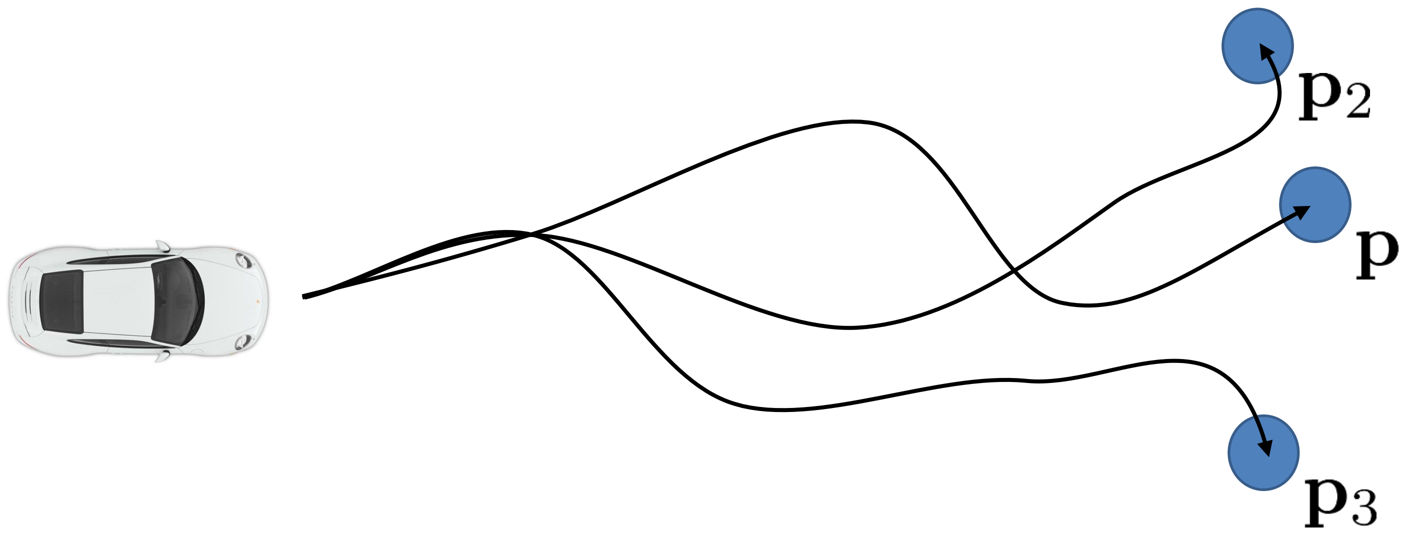
data that's not optimal for one task might be optimal from another

Another imitation idea



$$\pi_\theta(\mathbf{a}|\mathbf{s})$$

policy for reaching $\mathbf{p}_1$



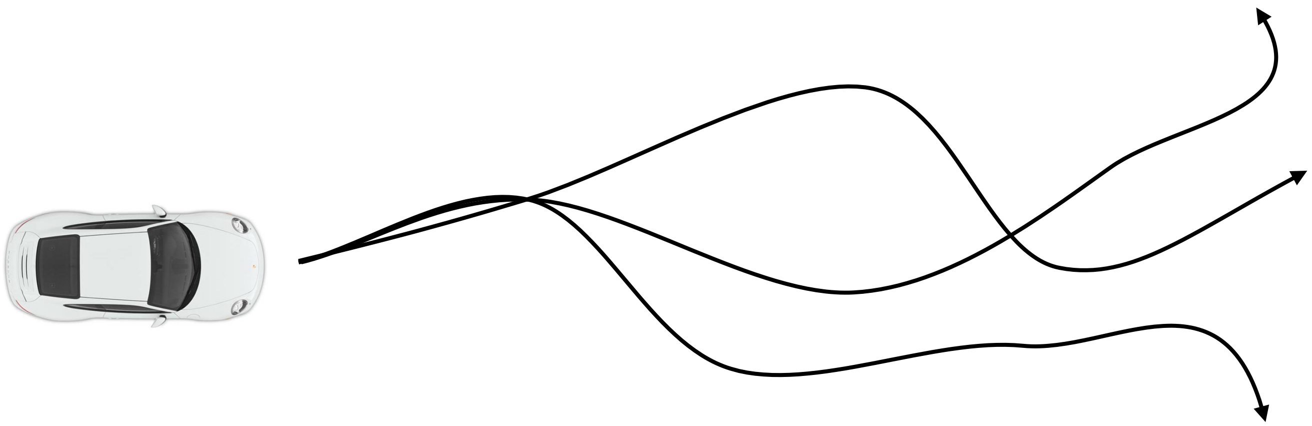
$$\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{p})$$

policy for reaching *any* $\mathbf{p}$

you might not have enough demonstrations for any one point itself. What if you condition your policy on the one that it's reached?

By conditioning our policy on p, we can train our policy to achieve some task, even if we don't have enough data for that exact task

Goal-conditioned behavioral cloning



during training we observe a bunch of demonstrations that in general all do different things. We treat each demonstration as a successful example for doing whatever the demonstration succeeded at.

training time:

demo 1: $\{\mathbf{s}_1, \mathbf{a}_t, \ldots, \mathbf{s}_{T-1}, \mathbf{a}_{T-1}, \mathbf{s}_T\}$ ⟵ successful demo for reaching $\mathbf{s}_T$

demo 2: $\{\mathbf{s}_1, \mathbf{a}_t, \ldots, \mathbf{s}_{T-1}, \mathbf{a}_{T-1}, \mathbf{s}_T\}$

demo 3: $\{\mathbf{s}_1, \mathbf{a}_t, \ldots, \mathbf{s}_{T-1}, \mathbf{a}_{T-1}, \mathbf{s}_T\}$

learn $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{g})$ ⟵ goal state

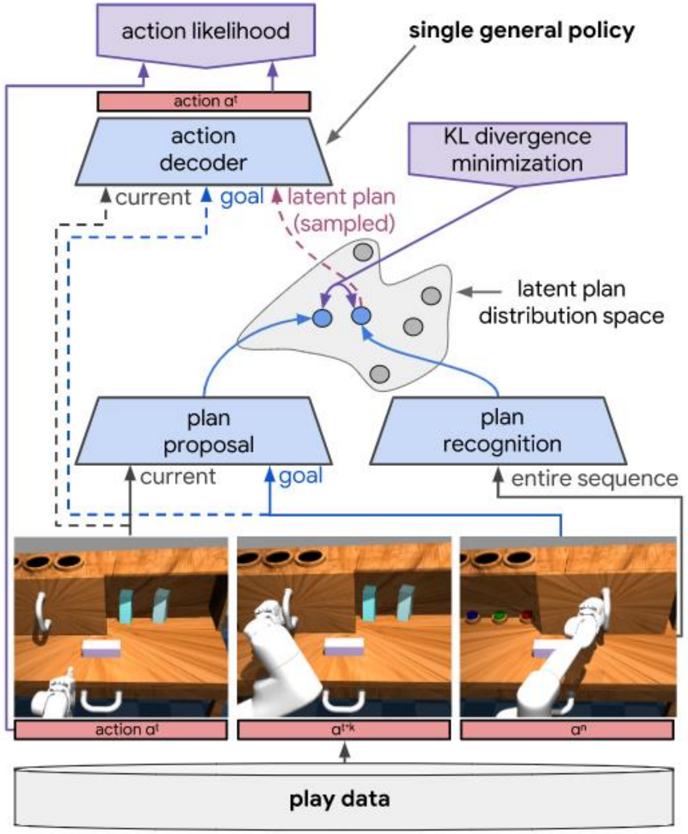
select the goal to be the last state that it reached, then just do regular behavior cloning

for each demo $\{\mathbf{s}_1^i, \mathbf{a}_1^i, \ldots, \mathbf{s}_{T-1}^i, \mathbf{a}_{T-1}^i, \mathbf{s}_T^i\}$

maximize $\log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i, \mathbf{g} = \mathbf{s}_T^i)$

Learning Latent Plans from Play

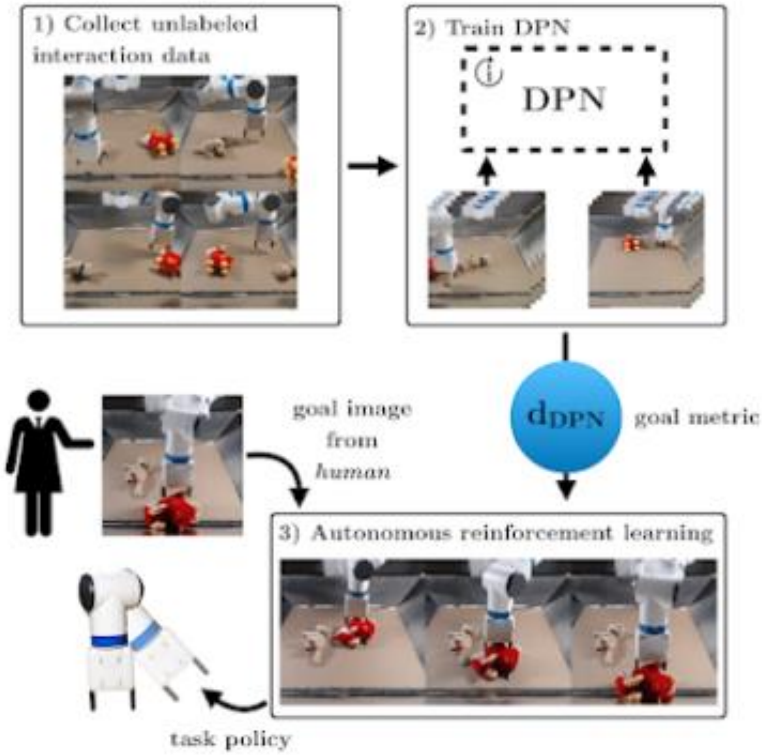
COREY LYNCH Google Brain
MOHI KHANSARI Google X
TED XIAO Google Brain
VIKASH KUMAR Google Brain
JONATHAN TOMPSON Google Brain
SERGEY LEVINE Google Brain
PIERRE SERMANET Google Brain



Unsupervised Visuomotor Control through Distributional Planning Networks

Tianhe Yu, Gleb Shevchuk, Dorsa Sadigh, Chelsea Finn

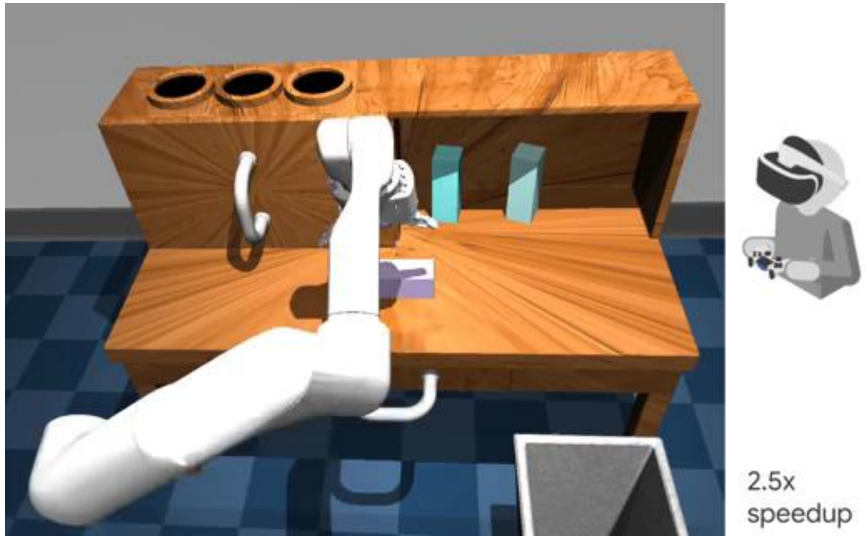
Stanford University



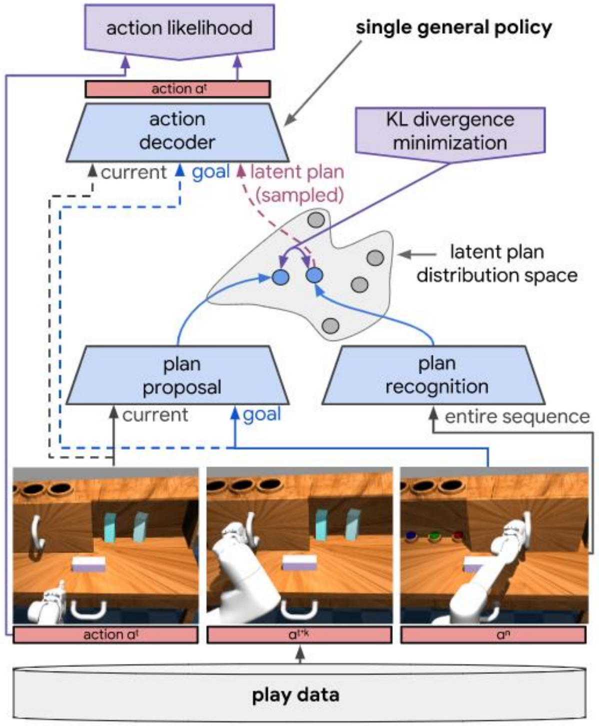
Learning Latent Plans from Play

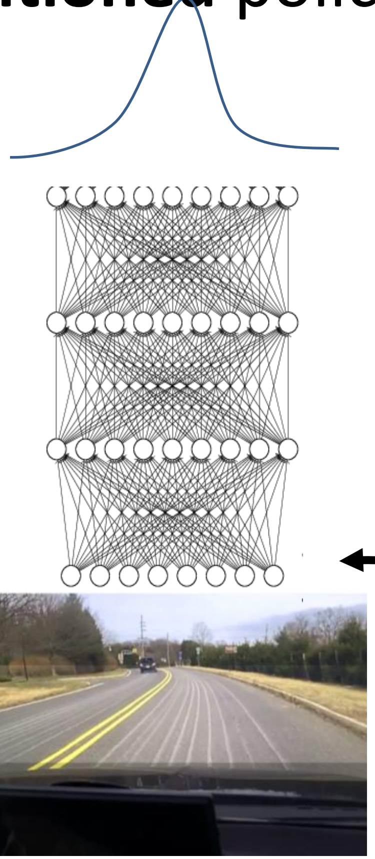
COREY LYNCH — Google Brain
MOHI KHANSARI — Google X
TED XIAO — Google Brain
VIKASH KUMAR — Google Brain
JONATHAN TOMPSON — Google Brain
SERGEY LEVINE — Google Brain
PIERRE SERMANET — Google Brain

1. Collect **data**



2. Train **goal conditioned** policy



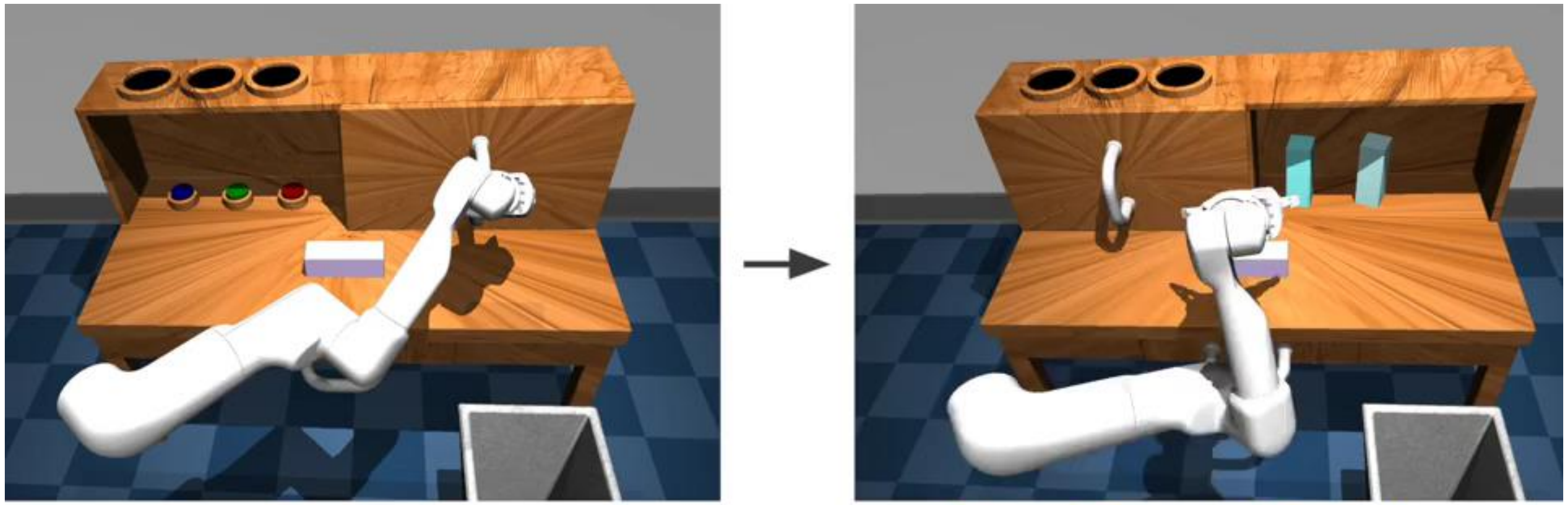


$\xi \sim \mathcal{N}(0, \mathbf{I})$

Learning Latent Plans from Play

COREY LYNCH Google Brain
MOHI KHANSARI Google X
TED XIAO Google Brain
VIKASH KUMAR Google Brain
JONATHAN TOMPSON Google Brain
SERGEY LEVINE Google Brain
PIERRE SERMANET Google Brain

3. Reach goals

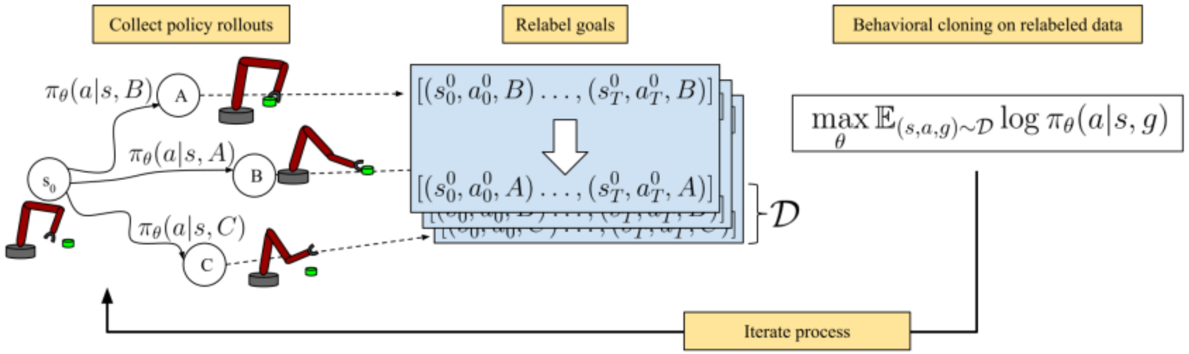


Goal → Single Play-LMP policy

Going **beyond** just imitation?

Learning to Reach Goals via Iterated Supervised Learning

Dibya Ghosh* UC Berkeley
Abhishek Gupta* UC Berkeley
Ashwin Reddy UC Berkeley
Justin Fu UC Berkeley
Coline Devin UC Berkeley
Benjamin Eysenbach Carnegie Mellon University
Sergey Levine UC Berkeley



do we even need the data to come from demonstrations? Can we just use bad random data? Yes!

- Start with a **random** policy

we start from scratch! Don't need human-labeled data

- Collect data with **random** goals
- Treat this data as "demonstrations" for the goals that were reached
- Use this to improve the policy
- Repeat