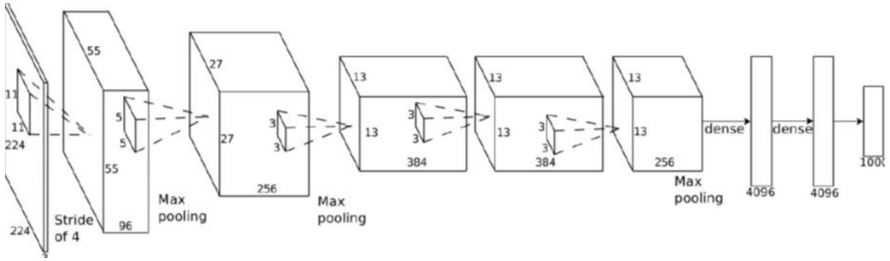
Supervised Learning of Behaviors

CS 285

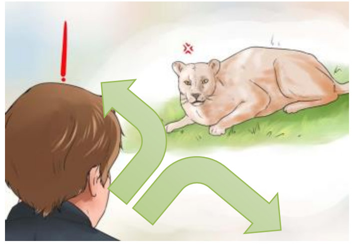
Instructor: Sergey Levine
UC Berkeley

Terminology & notation

$\mathbf{o}_t$



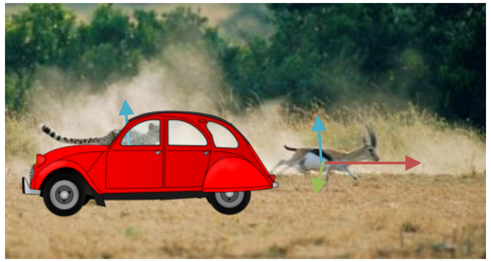
$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$



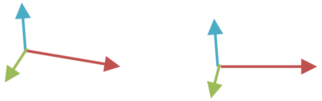
$\mathbf{a}_t$

$\mathbf{s}_t$ – 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)



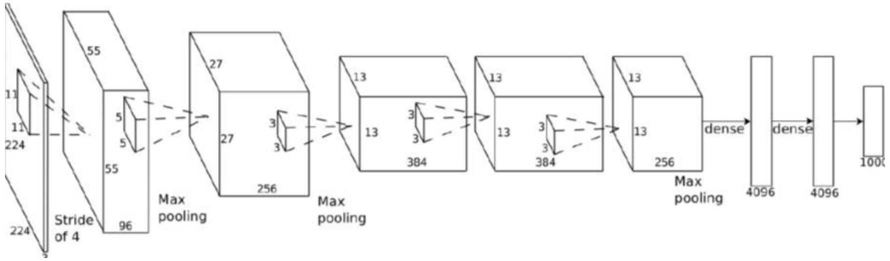
$\mathbf{o}_t$ – observation



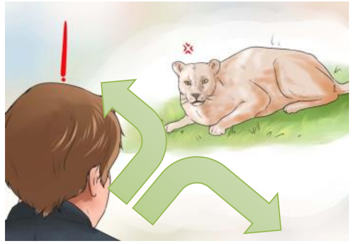
$\mathbf{s}_t$ – state

Terminology & notation

$\mathbf{o}_t$



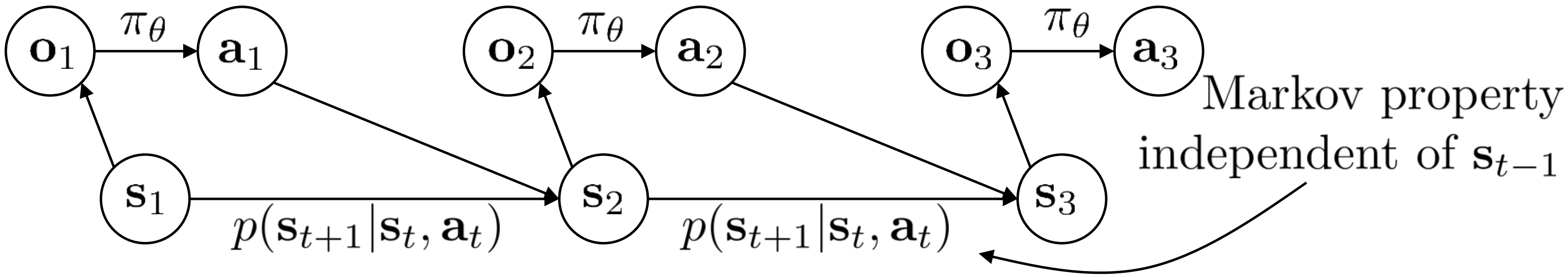
$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$



$\mathbf{a}_t$

$\mathbf{s}_t$ – 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)

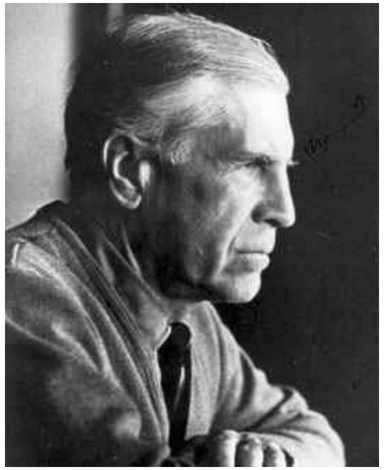


Aside: notation

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

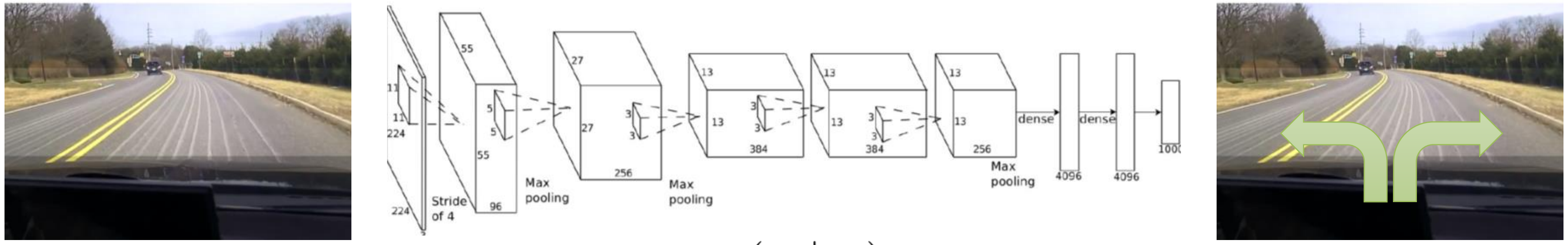
Richard Bellman

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



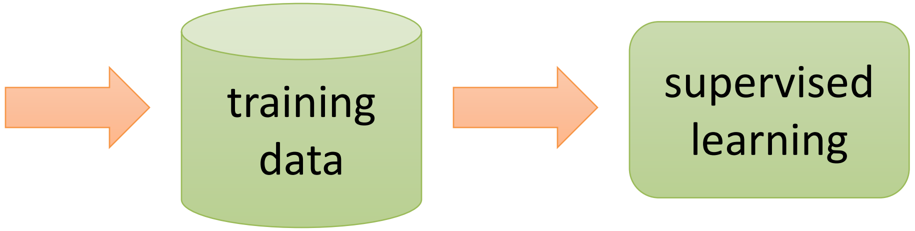
Lev Pontryagin

Imitation Learning



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

$\mathbf{o}_t$
$\mathbf{a}_t$



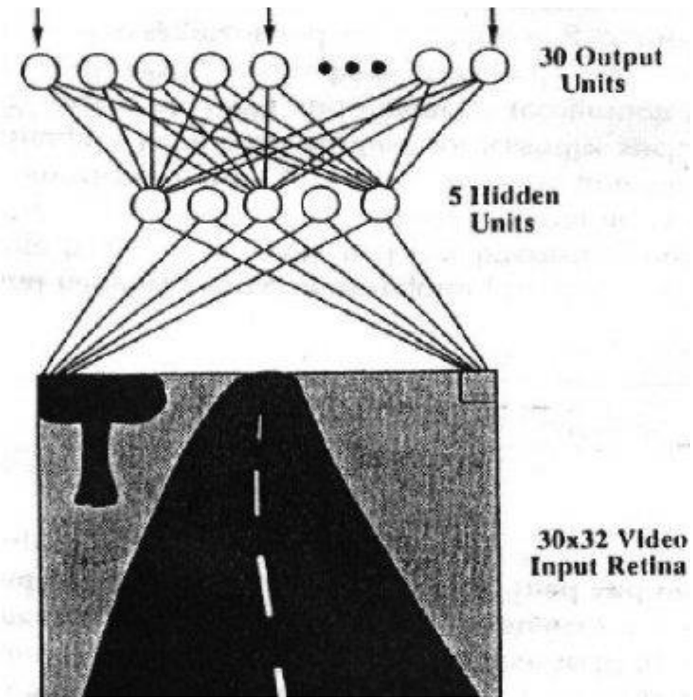
$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

behavioral cloning

Images: Bojarski et al. '16, NVIDIA

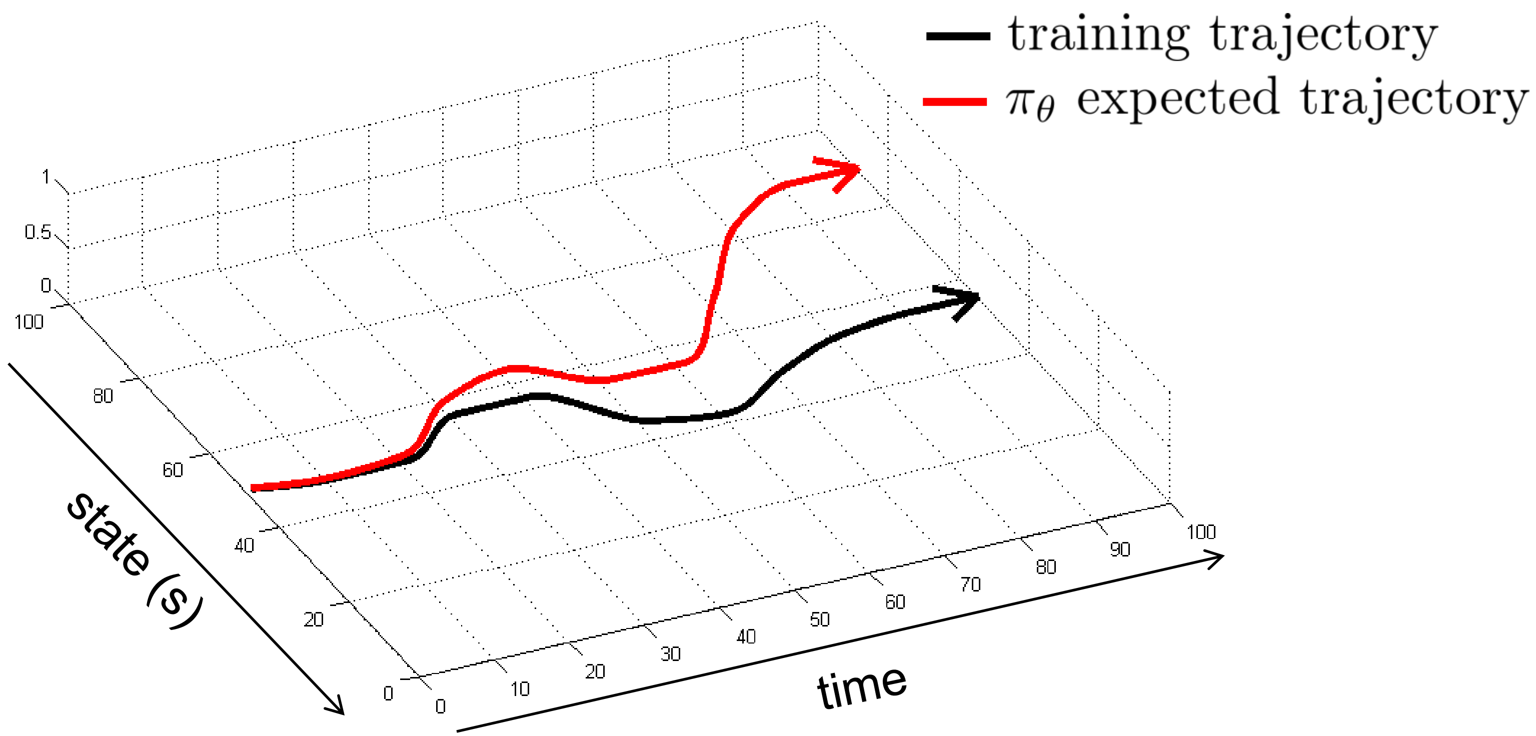
The original deep imitation learning system

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



Does it work?

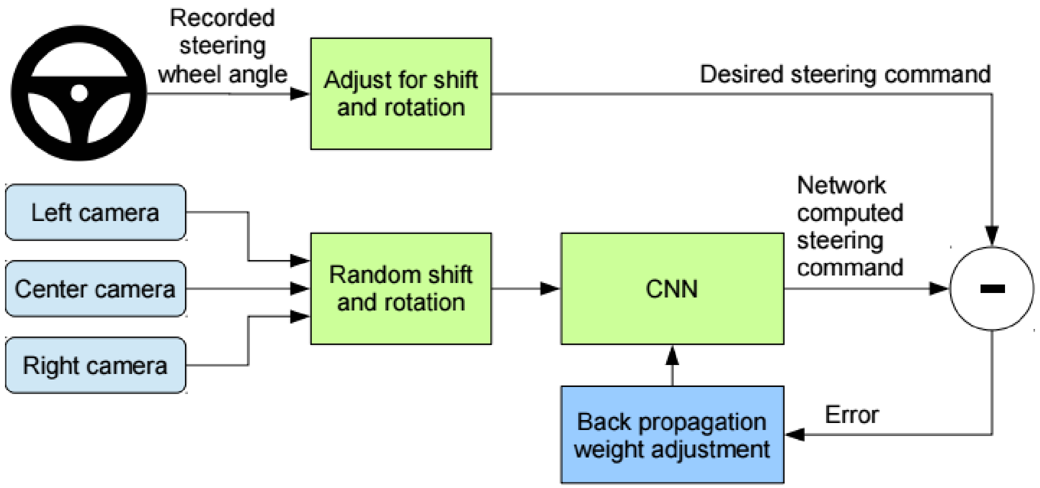
No!

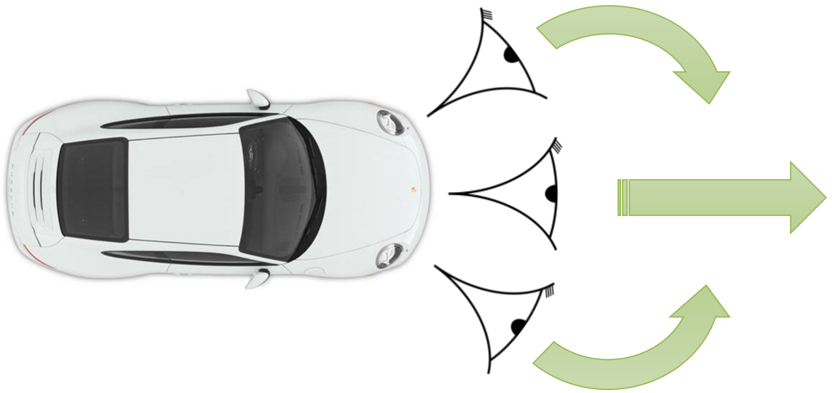


Does it work? Yes!

Video: Bojarski et al. '16, NVIDIA

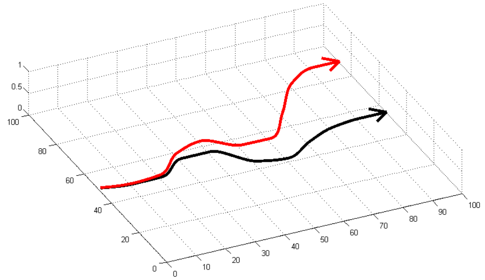
Why did that work?





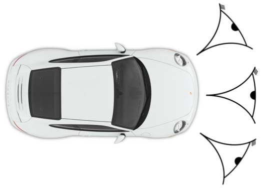
Bojarski et al. ‘16, NVIDIA

The moral of the story, and a list of ideas

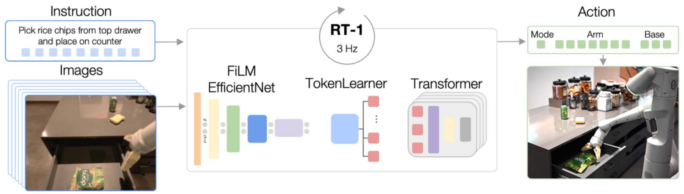


- Imitation learning via behavioral cloning is not guaranteed to work
  - This is **different** from supervised learning
  - The reason: i.i.d. assumption does not hold!

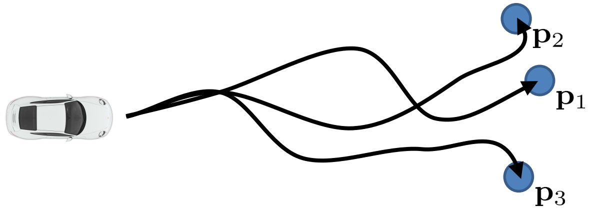
- We can formalize **why** this is and do a bit of theory

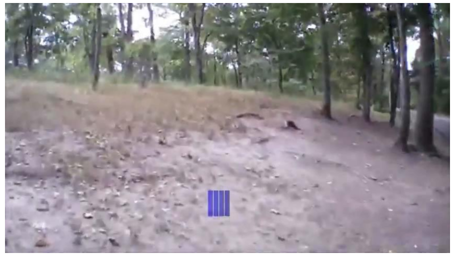


- We can address the problem in a few ways:
  - Be smart about how we collect (and augment) our data
  - Use very powerful models that make very few mistakes



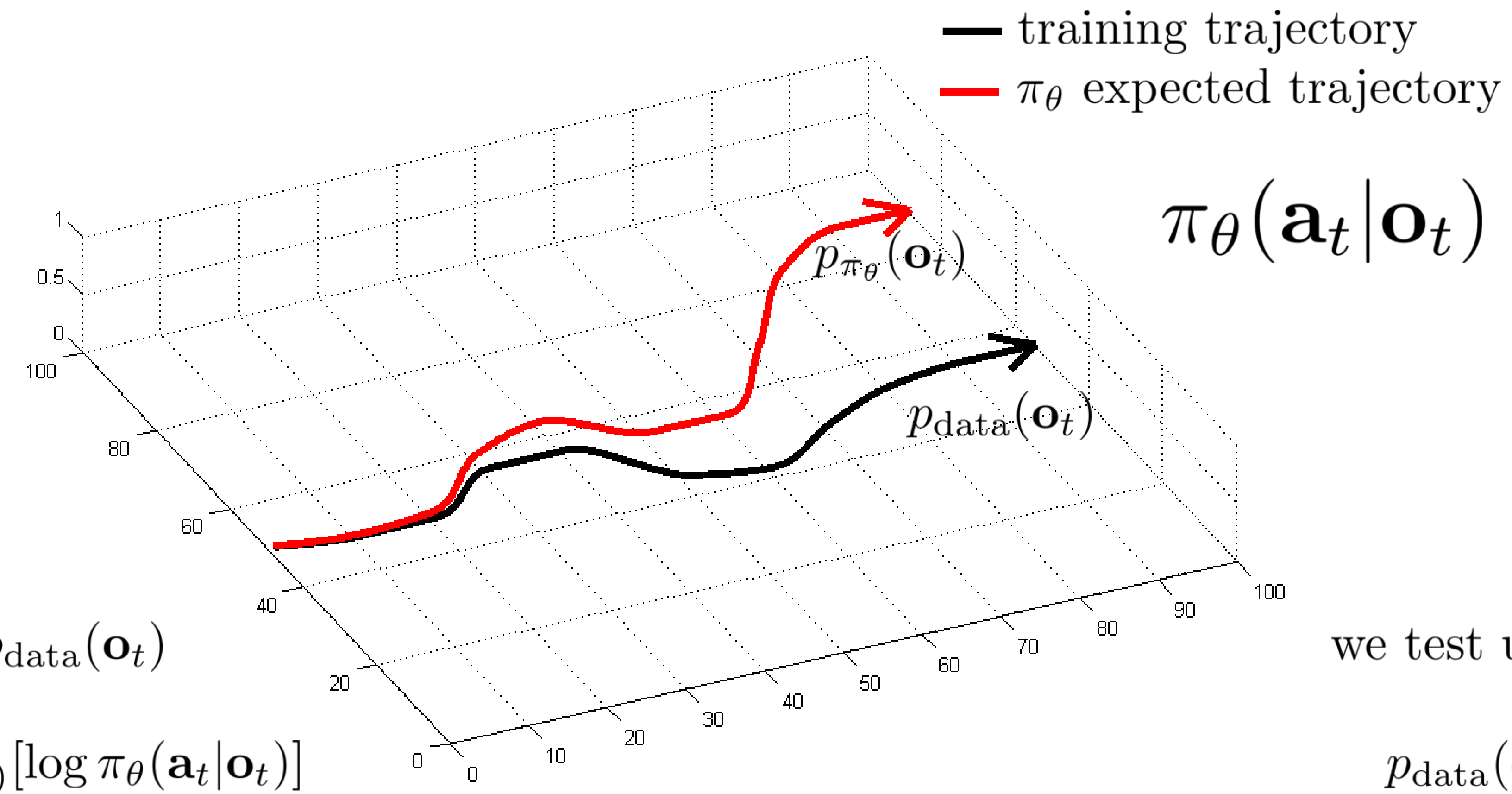
  - Use multi-task learning
  - Change the algorithm (DAgger)





Why does behavioral cloning fail?
A bit of theory

The distributional shift problem



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

we train under $p_{\text{data}}(\mathbf{o}_t)$

$$\max_\theta E_{\mathbf{o}_t \sim p_{\text{data}}(\mathbf{o}_t)}[\log \pi_\theta(\mathbf{a}_t|\mathbf{o}_t)]$$

we test under $p_{\pi_\theta}(\mathbf{o}_t)$

$$p_{\text{data}}(\mathbf{o}_t) \neq p_{\pi_\theta}(\mathbf{o}_t)$$

Let's define more precisely what we want

$\mathbf{o}_t$

$\mathbf{a}_t$

training data

supervised learning

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

What makes a learned $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ good or bad?

$$\cancel{\max_\theta E_{\mathbf{o}_t \sim p_{\text{data}}(\mathbf{o}_t)}[\log \pi_\theta(\mathbf{a}_t|\mathbf{o}_t)]}$$

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

Note: I started mixing up $\mathbf{s}$ and $\mathbf{o}$

I warned you about that...

Goal: minimize $E_{\mathbf{s}_t \sim p_{\pi_\theta}(\mathbf{s}_t)}[c(\mathbf{s}_t, \mathbf{a}_t)]$

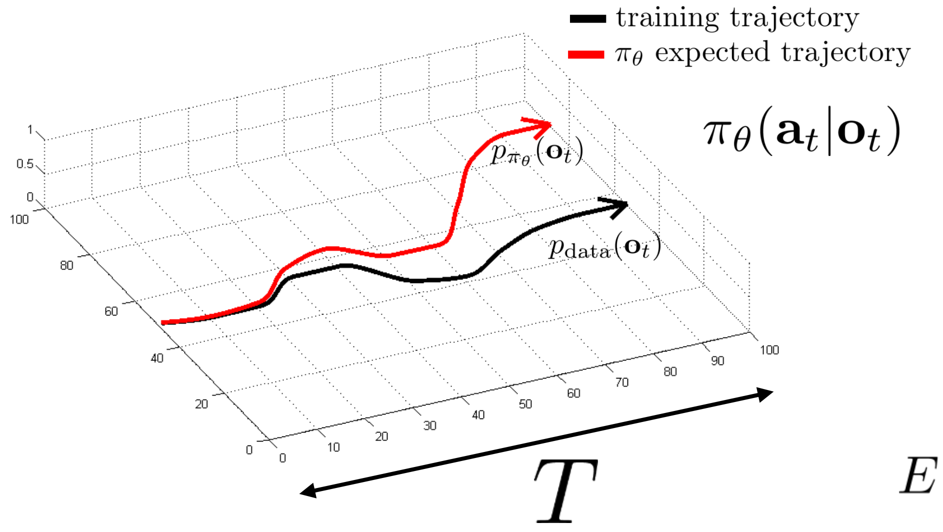
"Minimize the number of mistakes the policy makes when we run it"

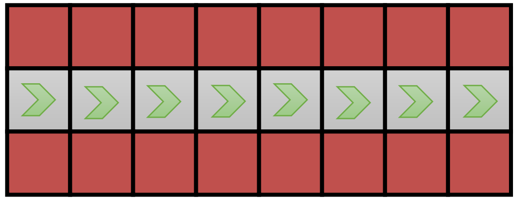
Some analysis

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

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

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





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

$O(\epsilon T^2)$

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

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

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

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 $\epsilon \in [0,1]$ $\qquad \leq 2\epsilon t$

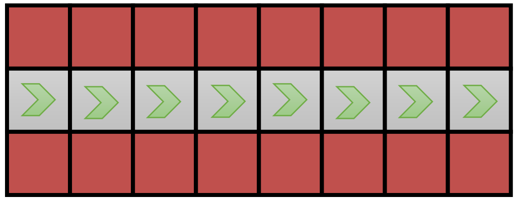
$$\sum_t E_{p_\theta(\mathbf{s}_t)}[c_t] = \sum_t \sum_{\mathbf{s}_t} p_\theta(\mathbf{s}_t) c_t(\mathbf{s}_t) \leq \sum_t \sum_{\mathbf{s}_t} p_{\text{train}}(\mathbf{s}_t) c_t(\mathbf{s}_t) + |p_\theta(\mathbf{s}_t) - p_{\text{train}}(\mathbf{s}_t)| c_{\max}$$

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

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

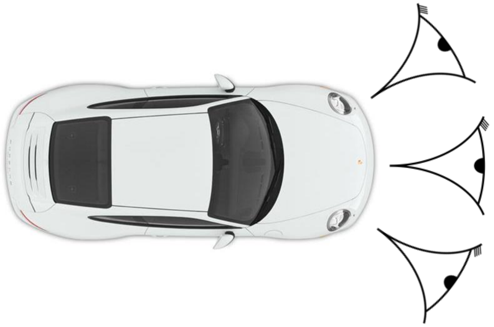
For more analysis, see Ross et al. “A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning”

Why is this rather **pessimistic**?

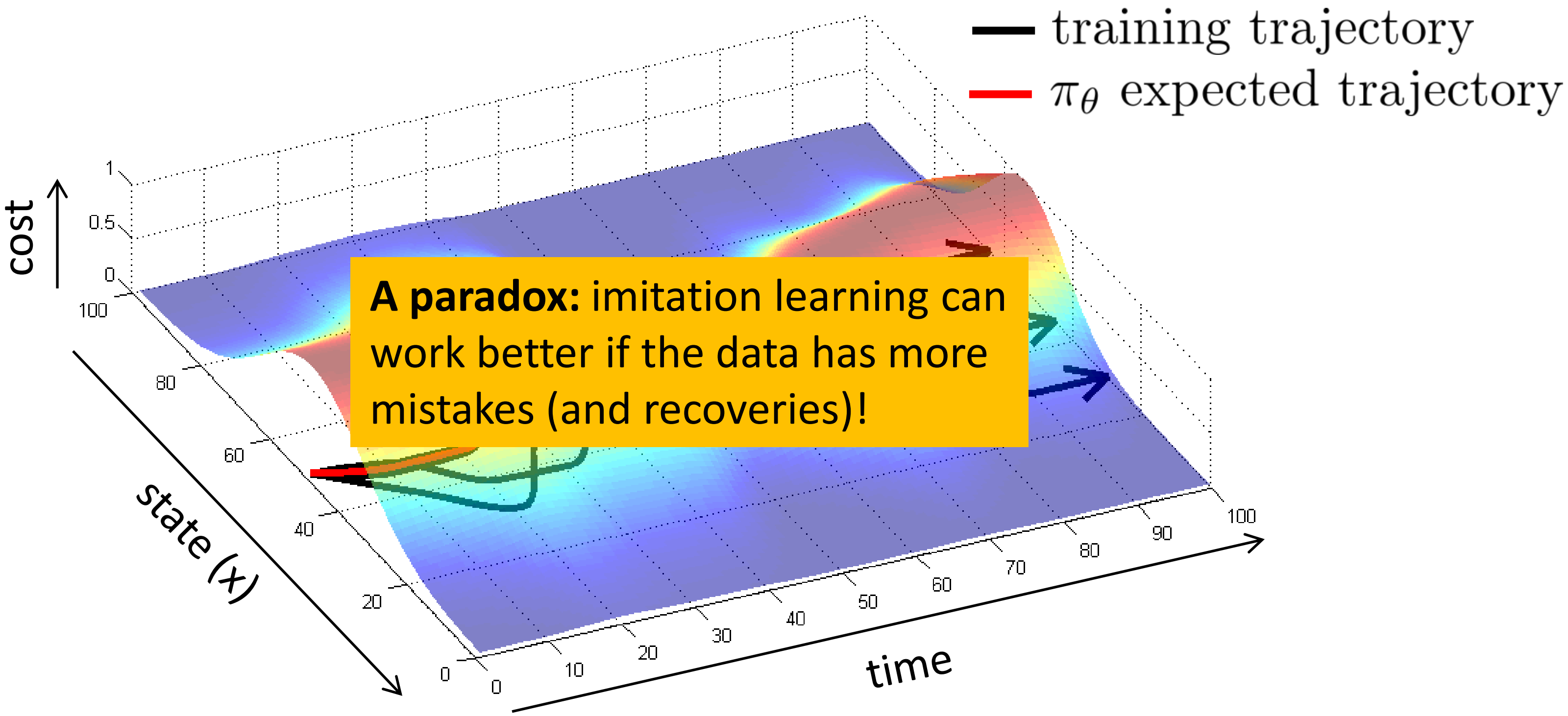


In reality, we can often **recover** from mistakes

But that doesn't mean that **imitation learning** will allow us to learn how to do that!



Why does this work?

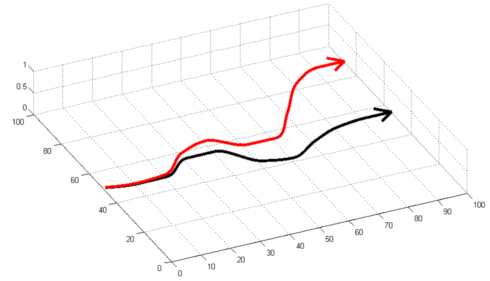


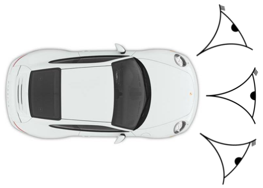
A paradox: imitation learning can work better if the data has more mistakes (and recoveries)!

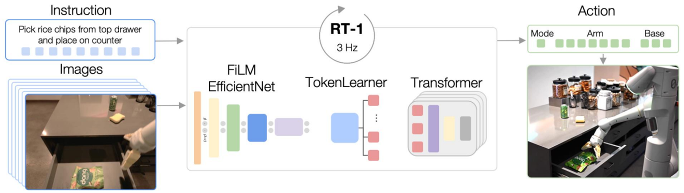
Addressing the problem in practice

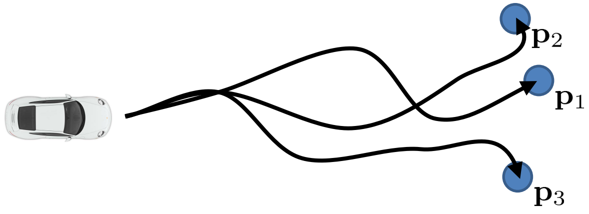
Where are we...

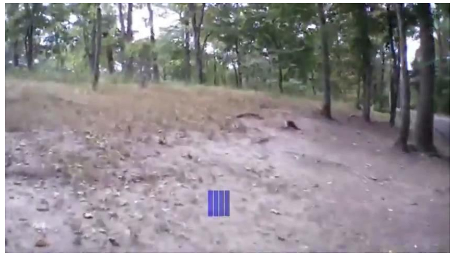
- Imitation learning via behavioral cloning is not guaranteed to work
  - This is **different** from supervised learning
  - The reason: i.i.d. assumption does not hold!
- We can formalize **why** this is and do a bit of theory
- We can address the problem in a few ways:
  - Be smart about how we collect (and augment) our data
  - Use very powerful models that make very few mistakes
  - Use multi-task learning
  - Change the algorithm (DAgger)





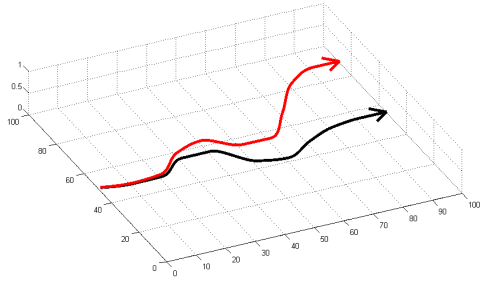


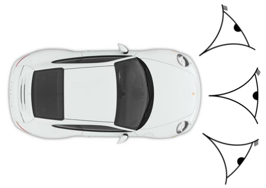


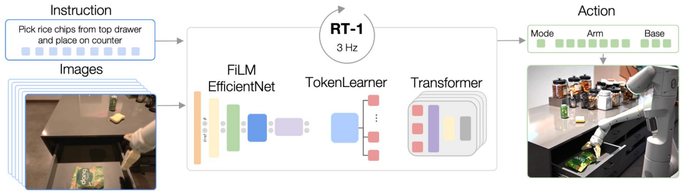


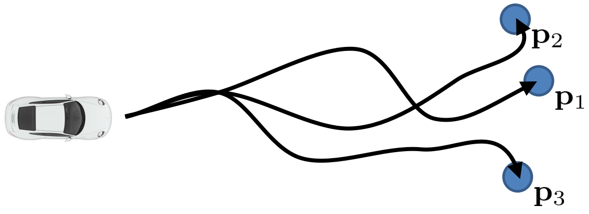
Where are we…

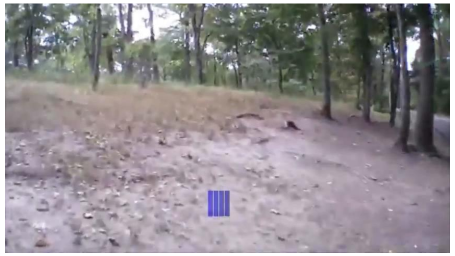
- Imitation learning via behavioral cloning is not guaranteed to work
  - This is **different** from supervised learning
  - The reason: i.i.d. assumption does not hold!
- We can formalize **why** this is and do a bit of theory
- We can address the problem in a few ways:
  - Be smart about how we collect (and augment) our data
  - Use very powerful models that make very few mistakes
  - Use multi-task learning
  - Change the algorithm (DAgger)



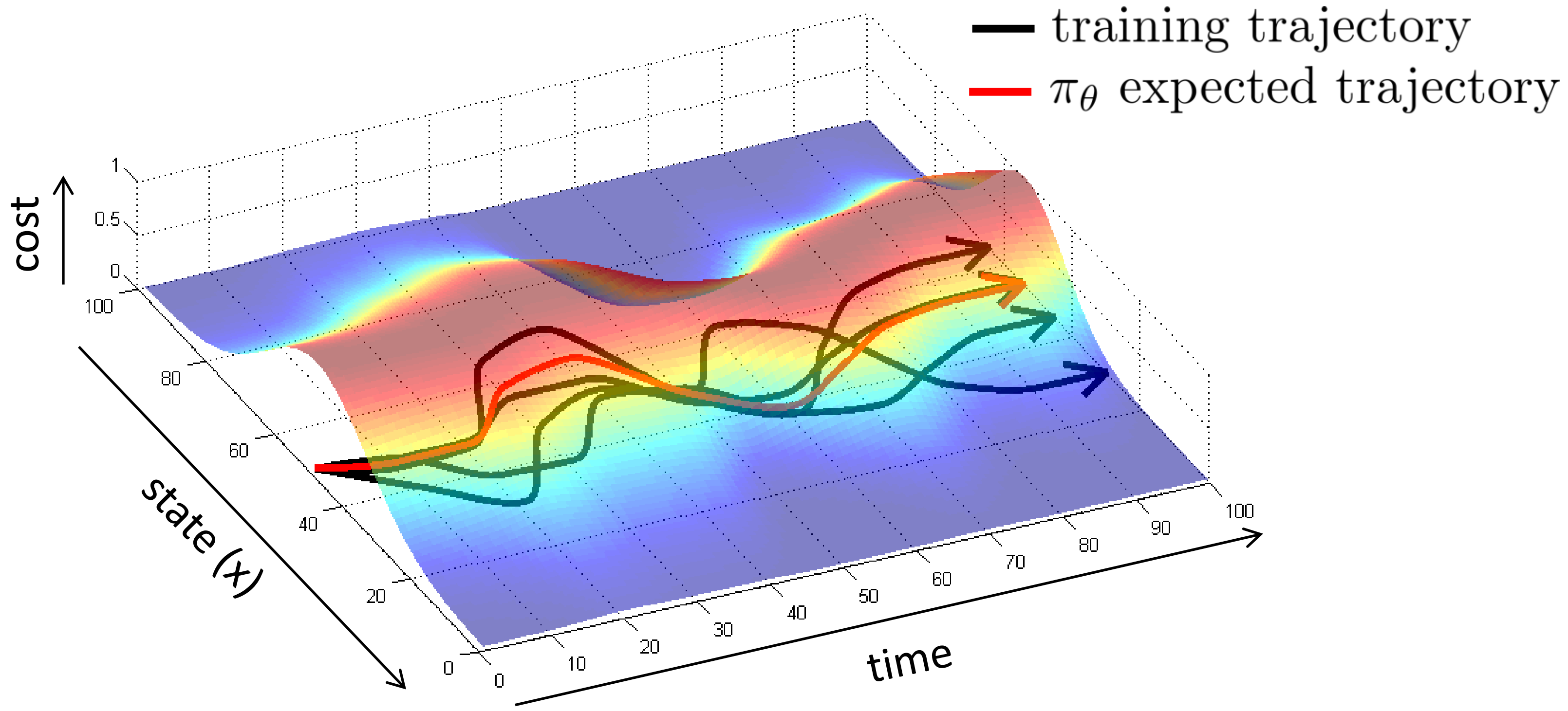








What makes behavioral cloning **easy** and what makes it **hard**?

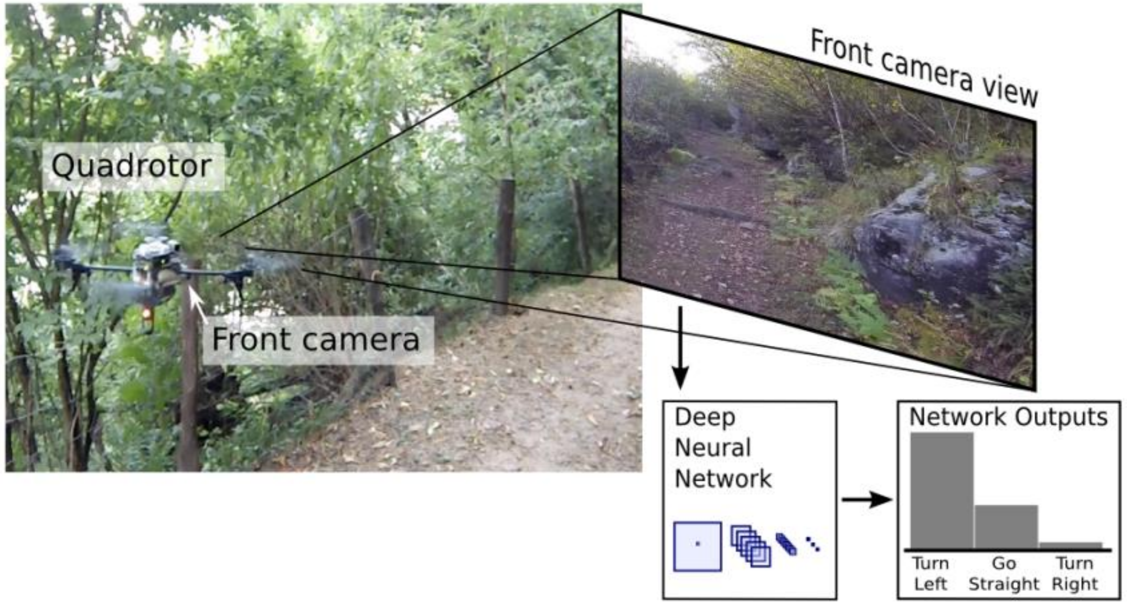


- Intentionally add **mistakes** and **corrections**
  - The mistakes hurt, but the corrections help, often more than the mistakes hurt
- Use **data augmentation**
  - Add some "fake" data that illustrates corrections (e.g., side-facing cameras)

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]

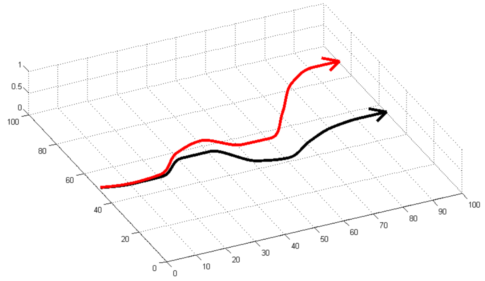


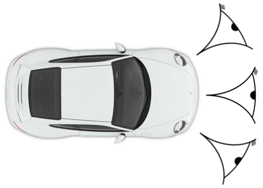
Case study 2: imitation with a cheap robot

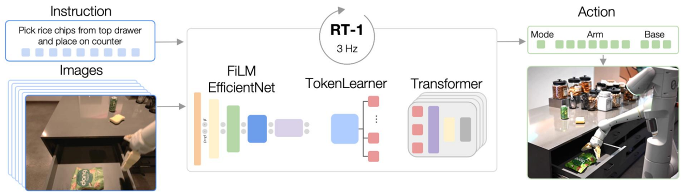
Rouhollah Rahmatizadeh et al., **Vision-Based Multi-Task Manipulation for Inexpensive Robots Using End-To-End Learning from Demonstration**. 2017.

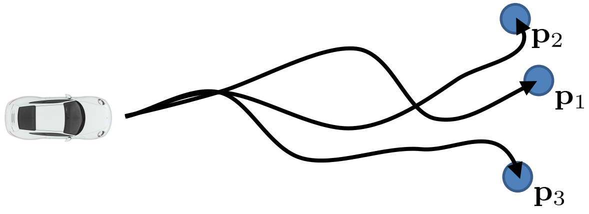
Where are we…

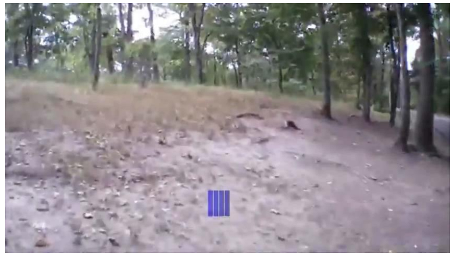
- Imitation learning via behavioral cloning is not guaranteed to work
  - This is **different** from supervised learning
  - The reason: i.i.d. assumption does not hold!
- We can formalize **why** this is and do a bit of theory
- We can address the problem in a few ways:
  - Be smart about how we collect (and augment) our data
  - Use very powerful models that make very few mistakes
  - Use multi-task learning
  - Change the algorithm (DAgger)











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

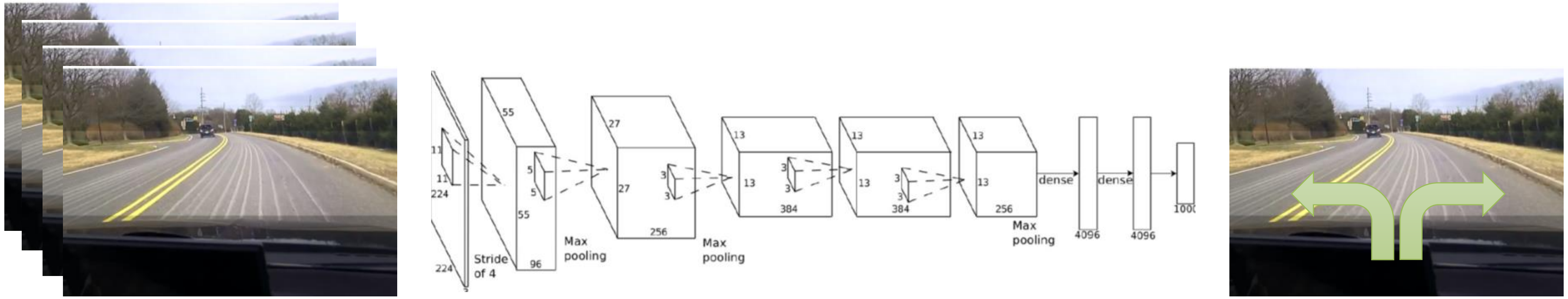
$$\pi_\theta(\mathbf{a}_t|\mathbf{o}_1, ..., \mathbf{o}_t)$$

behavior depends on
all past observations

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

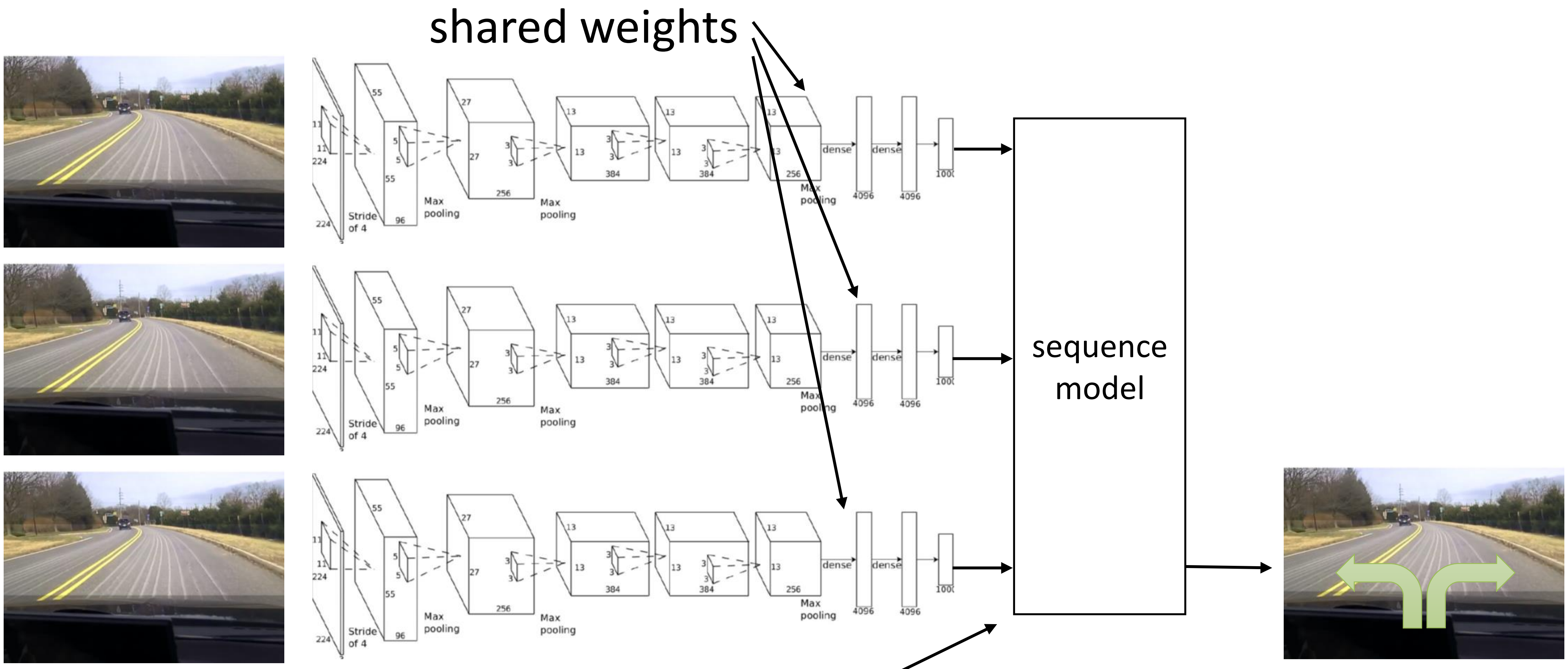
Often very unnatural for human demonstrators

How can we use the whole history?



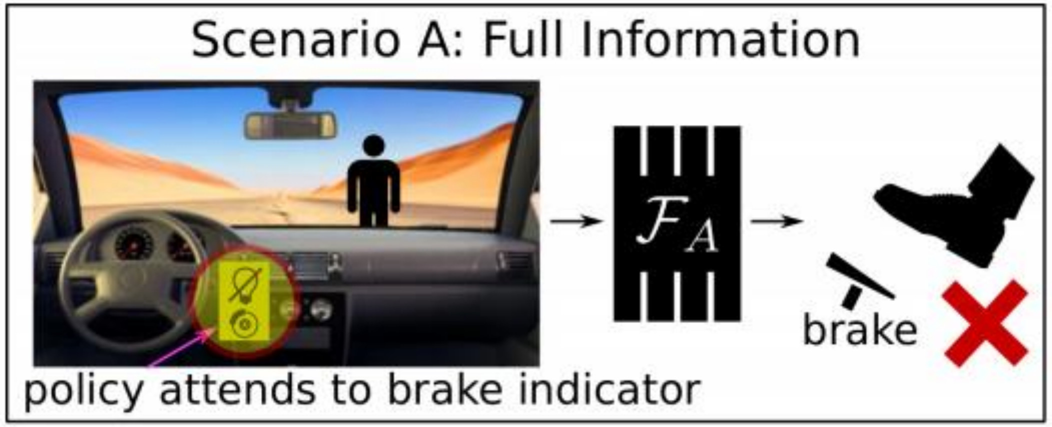
variable number of frames,
too many weights

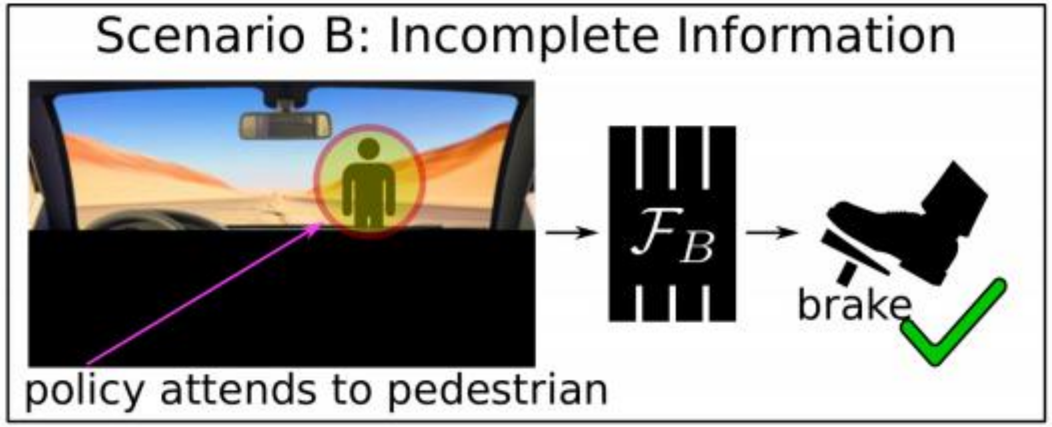
How can we use the whole history?



Can be done with Transformers, LSTM cells, etc.

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





"causal confusion"

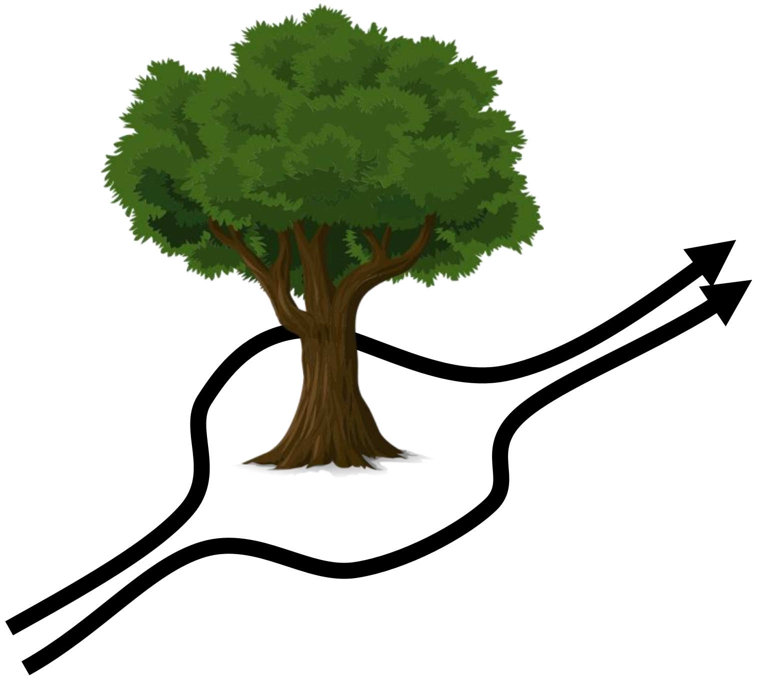
see: de Haan et al., "Causal Confusion in Imitation Learning"

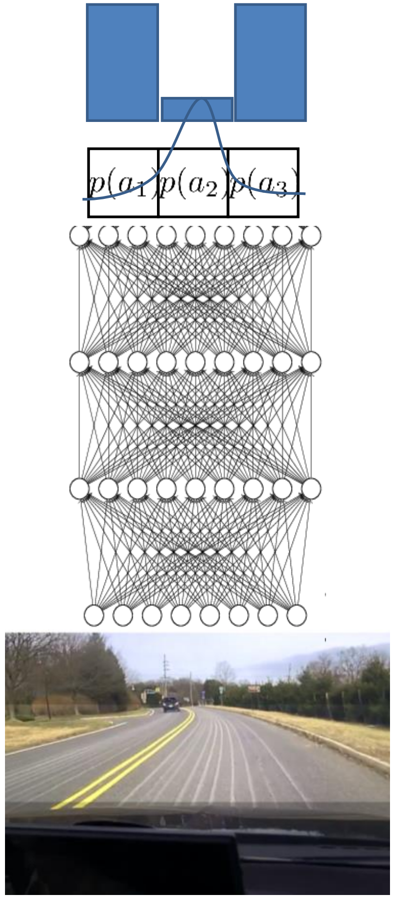
Question 1: Does including history mitigate causal confusion?

Question 2: Can DAgger mitigate causal confusion?

Why might we fail to fit the expert?

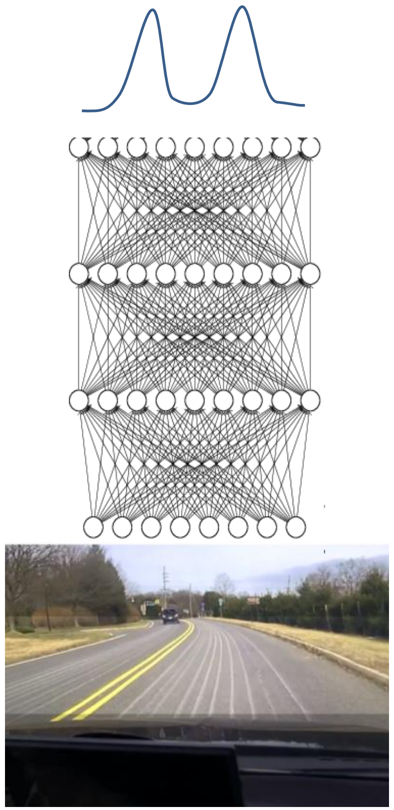
1. Non-Markovian behavior
2. Multimodal behavior





1. More expressive continuous distributions
2. Discretization with high-dimensional action spaces

Expressive continuous distributions



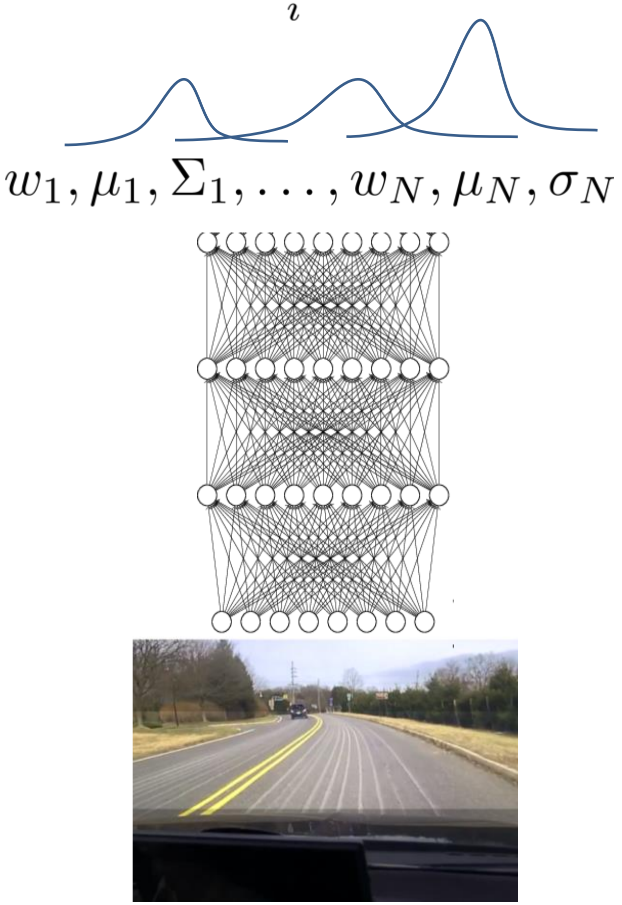
Quite a few options, many ways to make things work:

1. mixture of Gaussians
2. latent variable models
3. diffusion models

Expressive continuous distributions

1. mixture of Gaussians
2. latent variable models
3. diffusion models

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

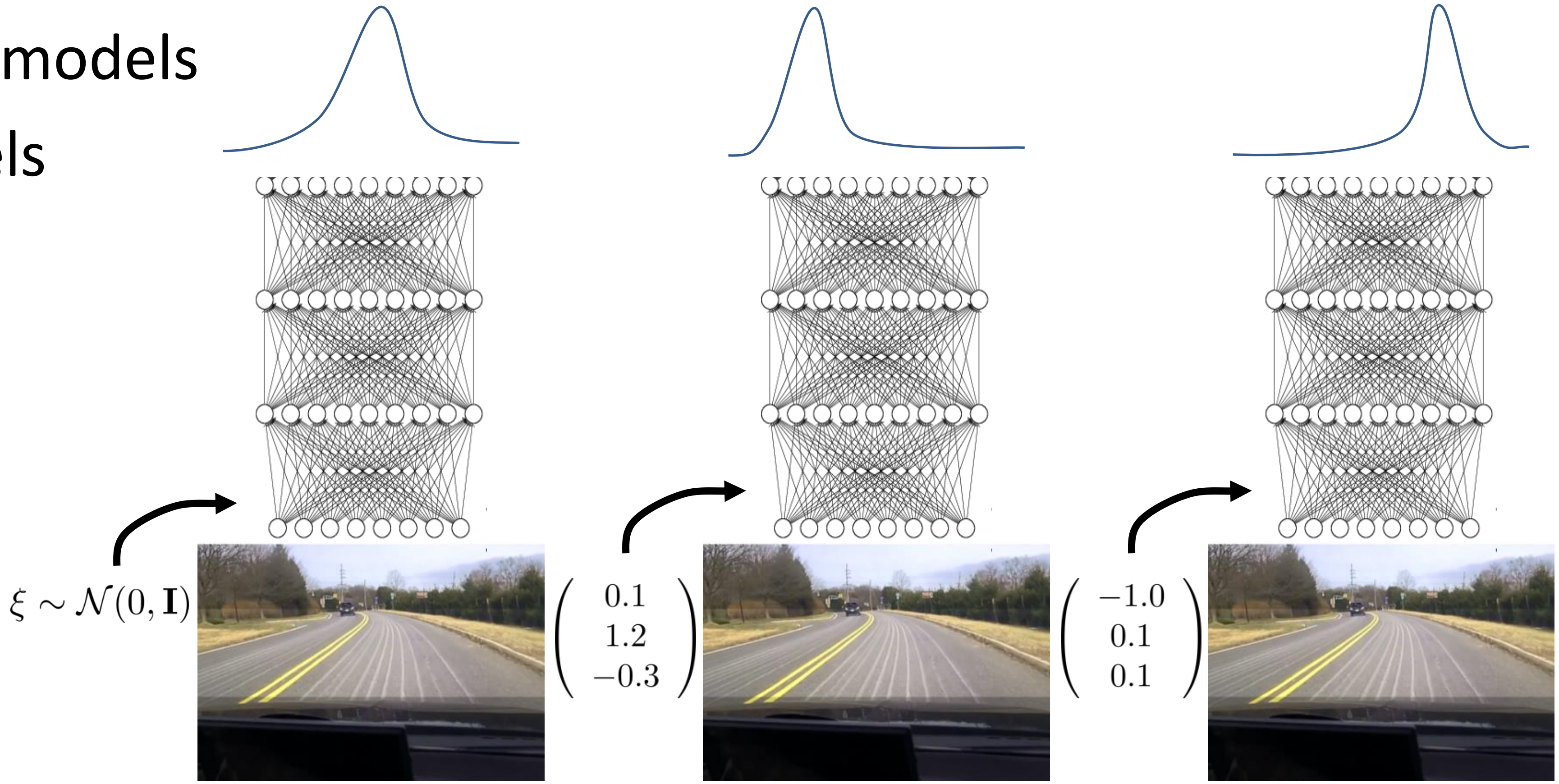


Expressive continuous distributions

1. mixture of Gaussians
2. latent variable models
3. diffusion models

The most widely used type of model of this sort is the (conditional) variational autoencoder

We'll learn about such models later in the course



Expressive continuous distributions

1. mixture of Gaussians
2. latent variable models
3. diffusion models

$\mathbf{x_0}$ = true image

$\mathbf{x_{i+1}} = \mathbf{x}_i + \text{noise}$

Learned network: $f(\mathbf{x}_i) = \mathbf{x}_{i-1}$

(actually use $f(\mathbf{x}_i) = \text{noise}$)

$\mathbf{x}_{i-1} = \mathbf{x}_i - f(\mathbf{x}_i)$

Expressive continuous distributions

1. mixture of Gaussians
2. latent variable models
3. diffusion models

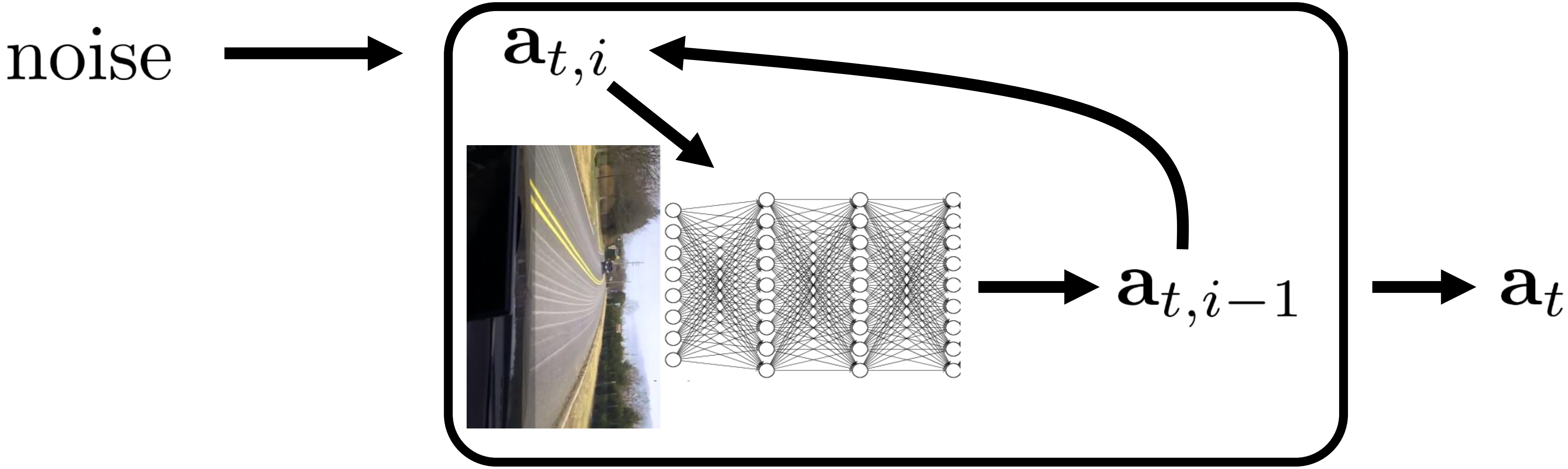
$\mathbf{a_{t,0}} = \text{true action}$

$\mathbf{a_{t,i+1}} = \mathbf{a}_{t,i} + \text{noise}$

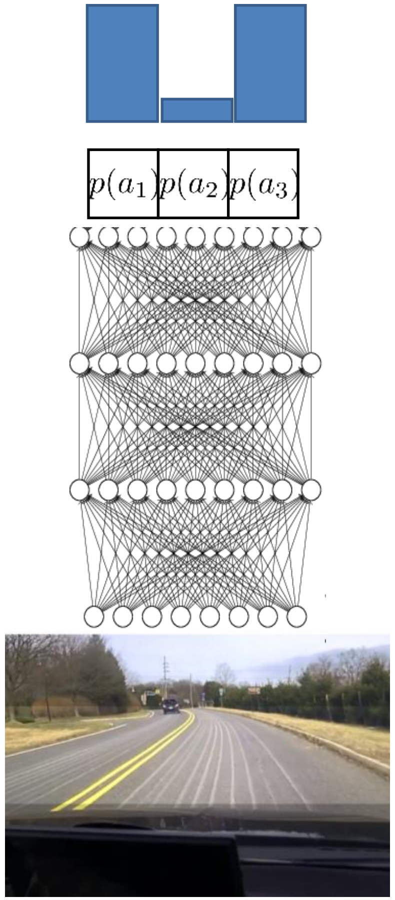
Learned network: $f(\mathbf{s}_t, \mathbf{a}_{t,i}) = \mathbf{a}_{t,i-1}$

(actually use $f(\mathbf{s}_t, \mathbf{a}_{t,i}) = \text{noise}$)

$\mathbf{a}_{t,i-1} = \mathbf{a}_{t,i} - f(\mathbf{s}_t, \mathbf{a}_{t,i})$



What about **discretization**?



Problem: this is great for 1D actions, but in higher dimensions, discretizing the full space is impractical

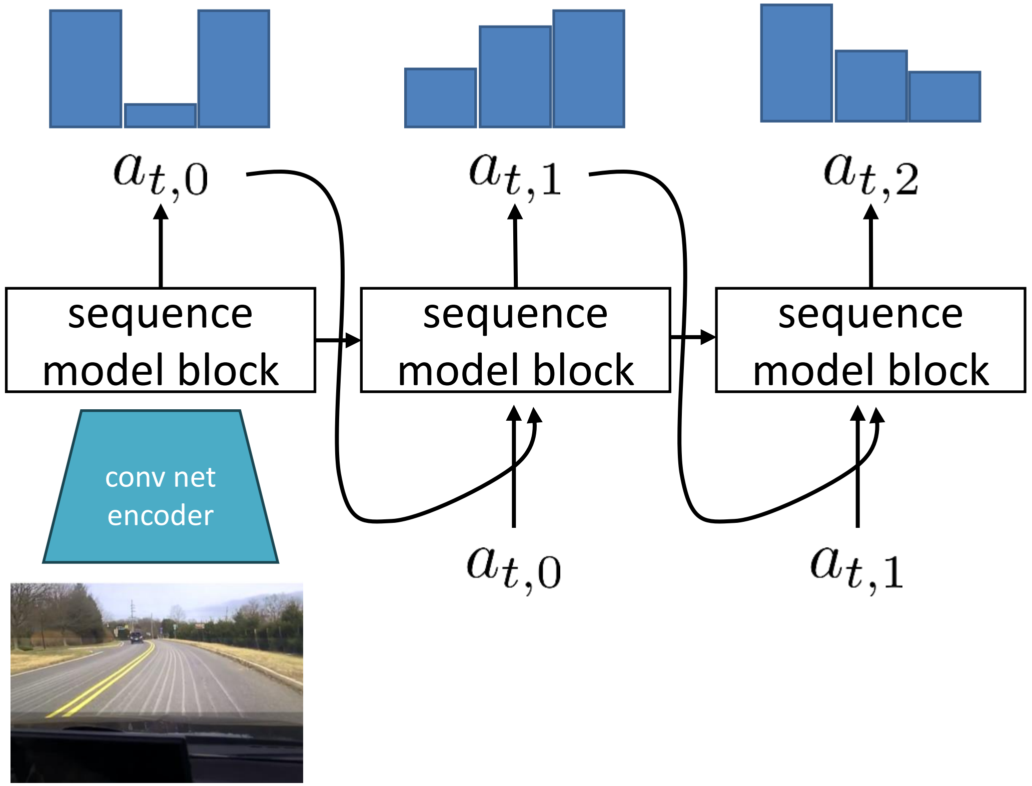
Solution: discretize one dimension at a time

Autoregressive discretization

$$\mathbf{a}_t = \begin{array}{cc} \begin{pmatrix} 0.1 \\ 1.2 \\ -0.3 \end{pmatrix} & \begin{array}{c} a_{t,0} \\ a_{t,1} \\ a_{t,2} \end{array} \end{array}$$

Why does this work?

use LSTM or Transformer



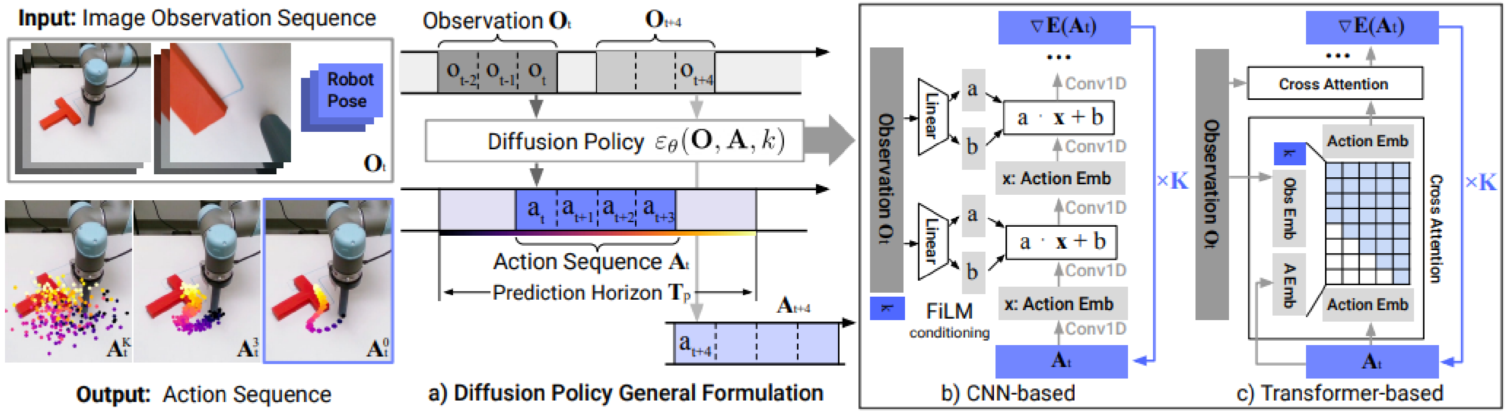
first step: $p(a_{t,0}|\mathbf{s}_t)$

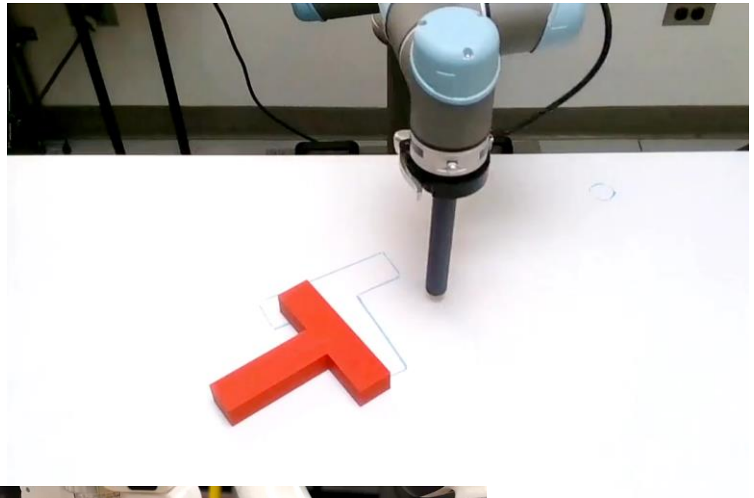
second step: $p(a_{t,1}|\mathbf{s}_t, a_{t,0})$

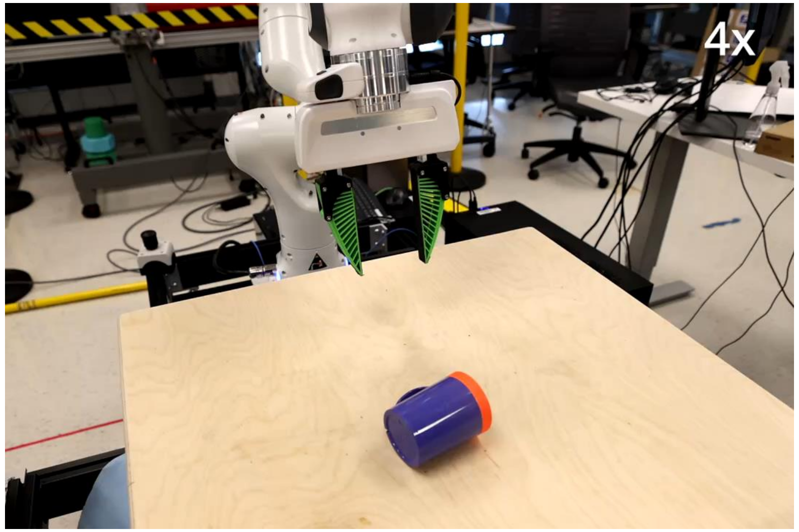
third step: $p(a_{t,2}|\mathbf{s}_t, a_{t,0}, a_{t,1})$

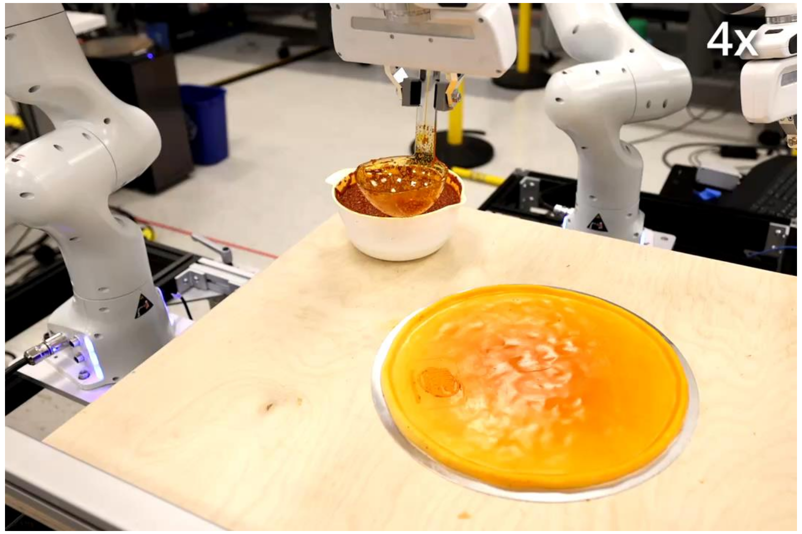
$$p(a_{t,2}|\mathbf{s}_t, a_{t,0}, a_{t,1})p(a_{t,1}|\mathbf{s}_t, a_{t,0})p(a_{t,0}|\mathbf{s}_t)$$
$$= p(a_{t,0}, a_{t,1}, a_{t,2}|\mathbf{s}_t)$$
$$= p(\mathbf{a}_t|\mathbf{s}_t)$$

Case study 3: imitation with diffusion models



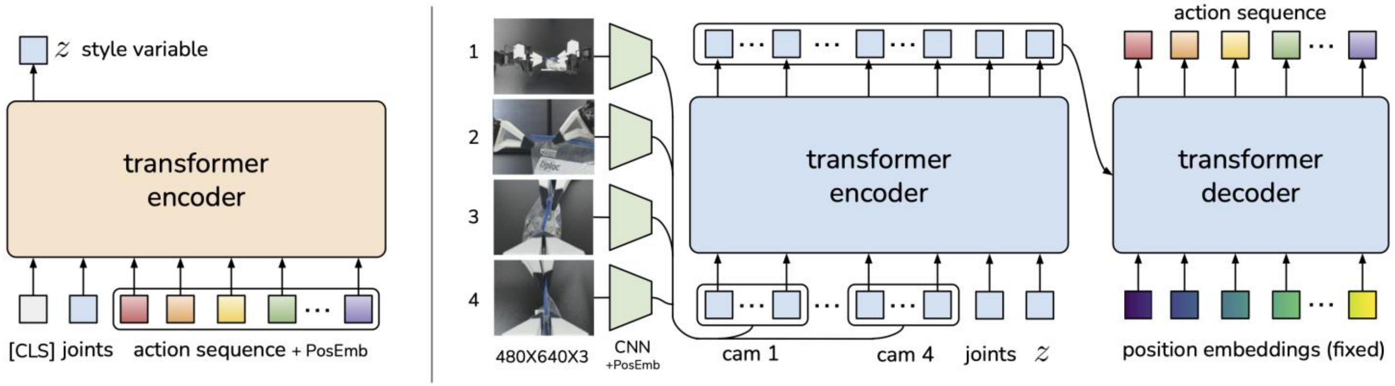


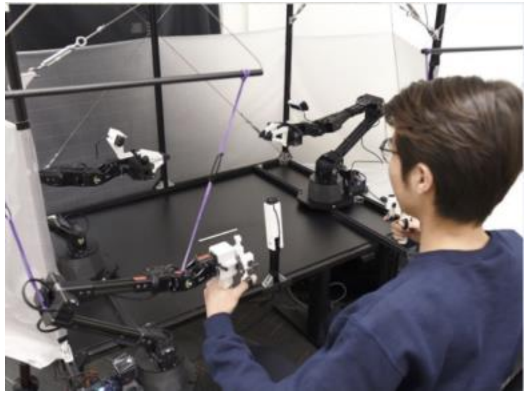


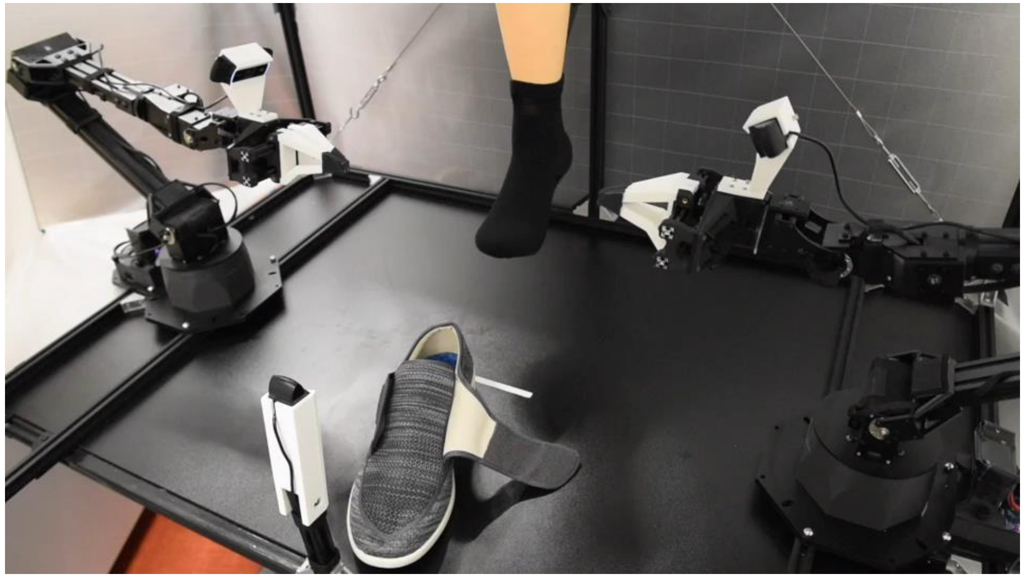


Chi et al. **Diffusion Policy: Visuomotor Policy Learning via Action Diffusion.** 2023

Case study 4: imitation with latent variables

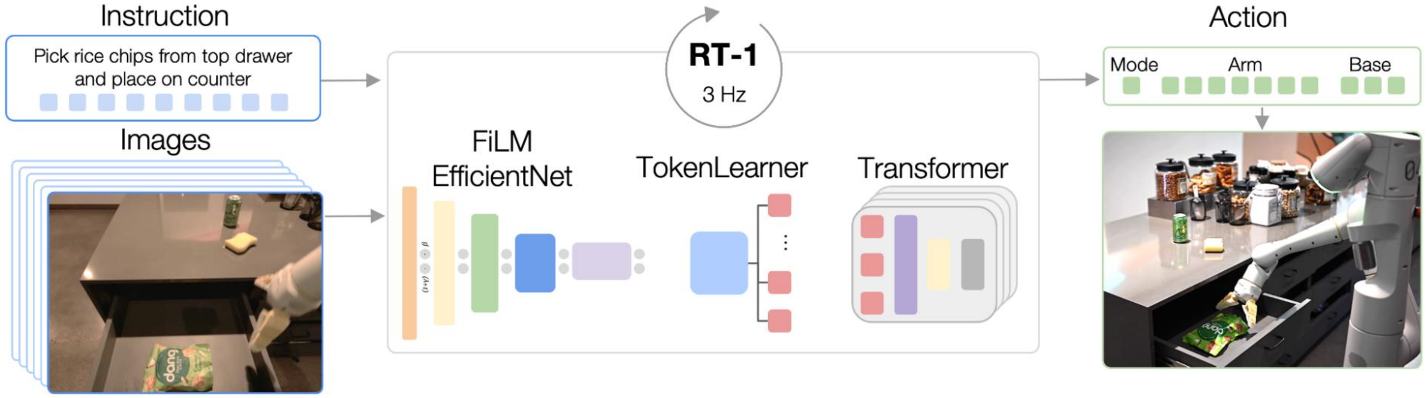


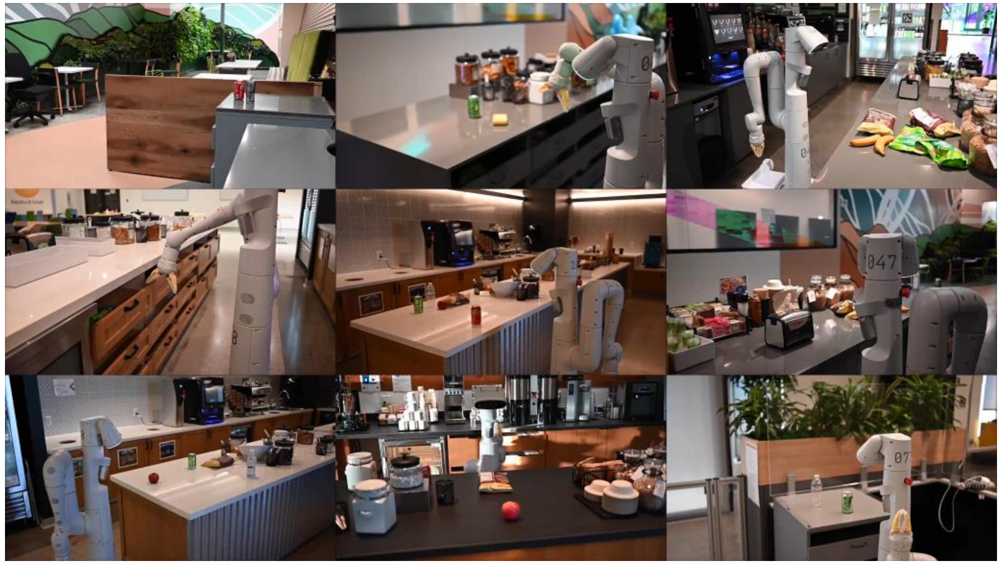




Zhao et al. **Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware.** 2023

Case study 5: imitation with Transformers





RT-1: Robotics Transformer
for Real-World Control at Scale

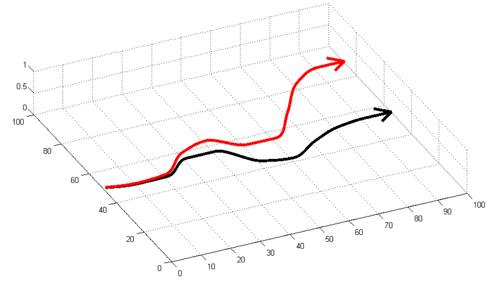
Robotics at Google
Everyday Robots
Google Research

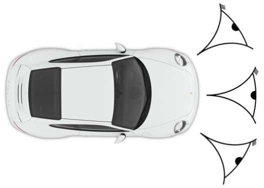
https://robotics-transformer.github.io/

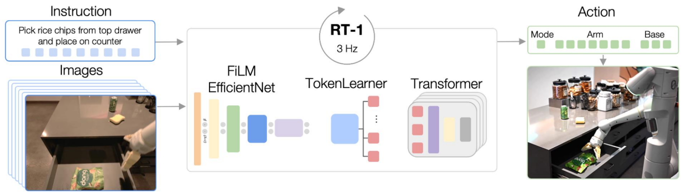
Brohan et al. **RT-1: Robotics Transformer.** 2023.

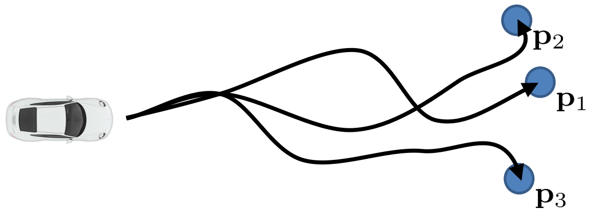
Where are we…

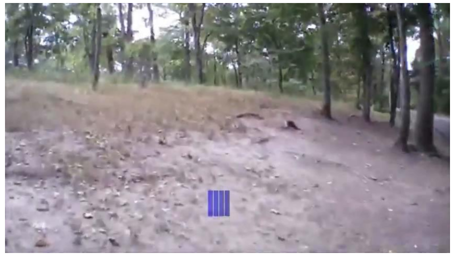
- Imitation learning via behavioral cloning is not guaranteed to work
  - This is **different** from supervised learning
  - The reason: i.i.d. assumption does not hold!
- We can formalize **why** this is and do a bit of theory
- We can address the problem in a few ways:
  - Be smart about how we collect (and augment) our data
  - Use very powerful models that make very few mistakes
  - Use multi-task learning
  - Change the algorithm (DAgger)



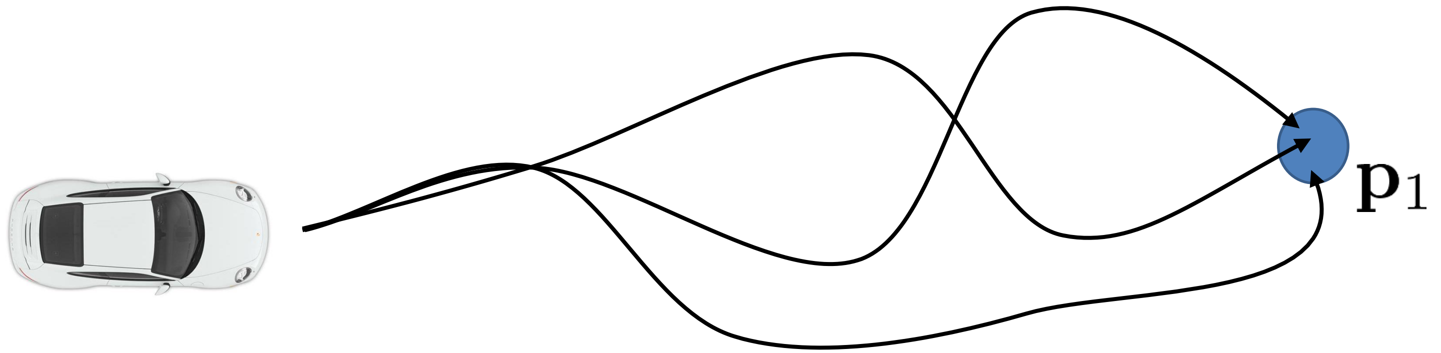






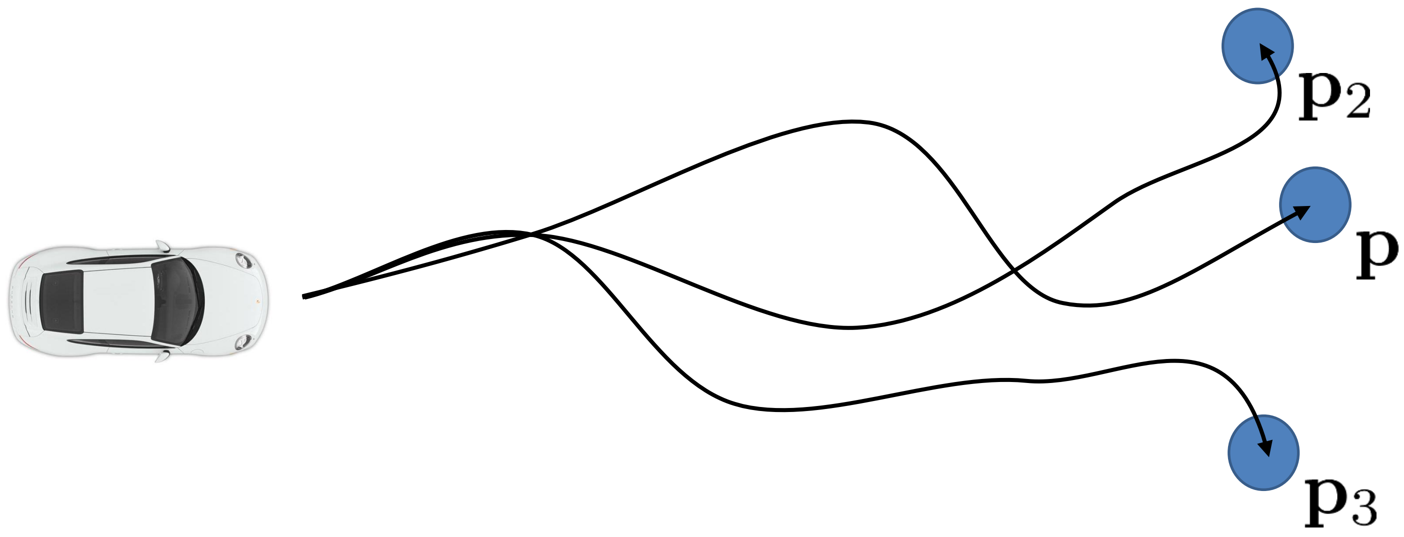


Does learning **many** tasks become easier?



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

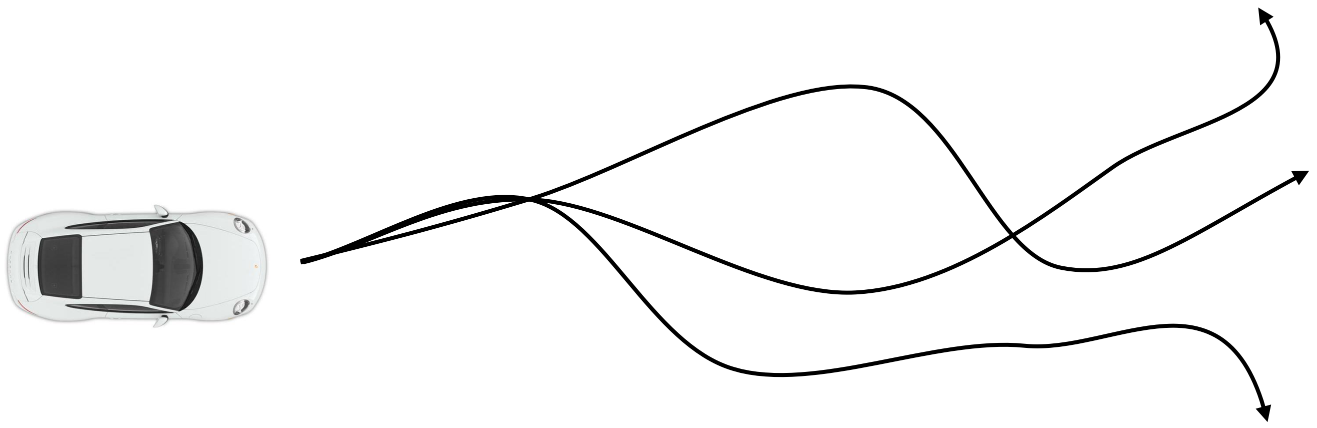
policy for reaching $\mathbf{p}_1$



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

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

Goal-conditioned behavioral cloning



training time:

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

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

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

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

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

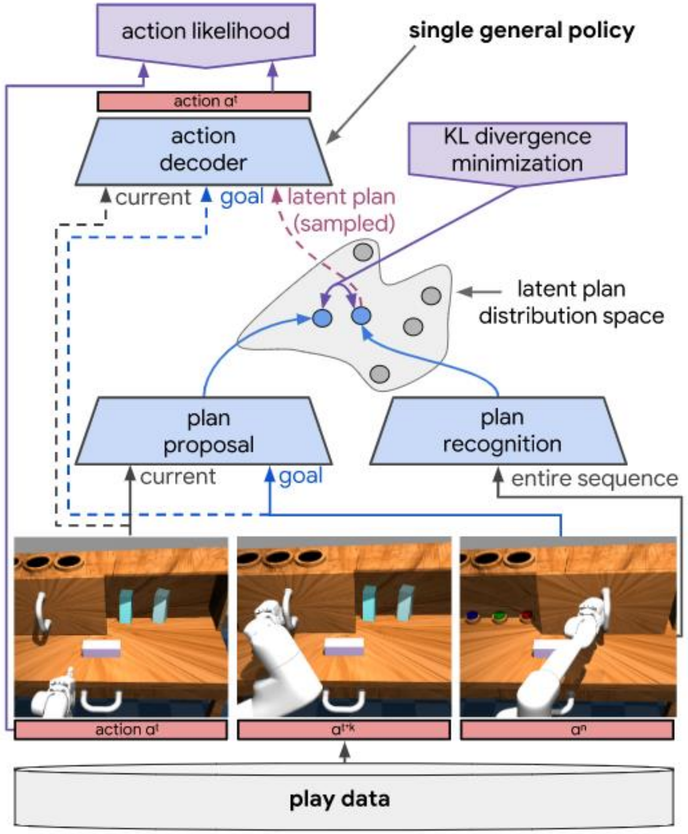
maximize $\log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i, \mathbf{g} = \mathbf{s}_T^i)$

We see distributional shift in **two** places here!

Can you figure out what the second place is?

Learning Latent Plans from Play

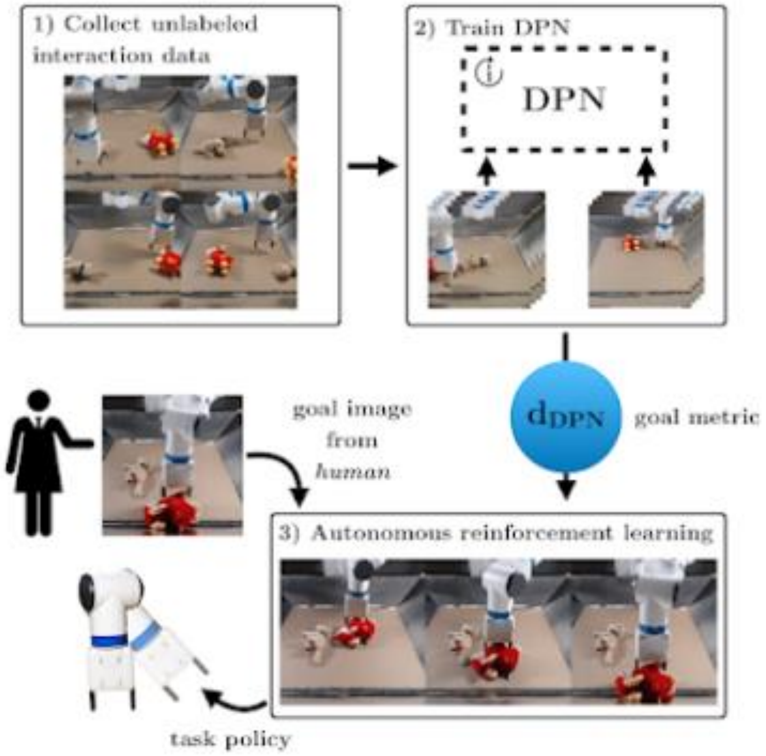
COREY LYNCH Google Brain
MOHI KHANSARI Google X
TED XIAO Google Brain
VIKASH KUMAR Google Brain
JONATHAN TOMPSON Google Brain
SERGEY LEVINE Google Brain
PIERRE SERMANET Google Brain



Unsupervised Visuomotor Control through Distributional Planning Networks

Tianhe Yu, Gleb Shevchuk, Dorsa Sadigh, Chelsea Finn

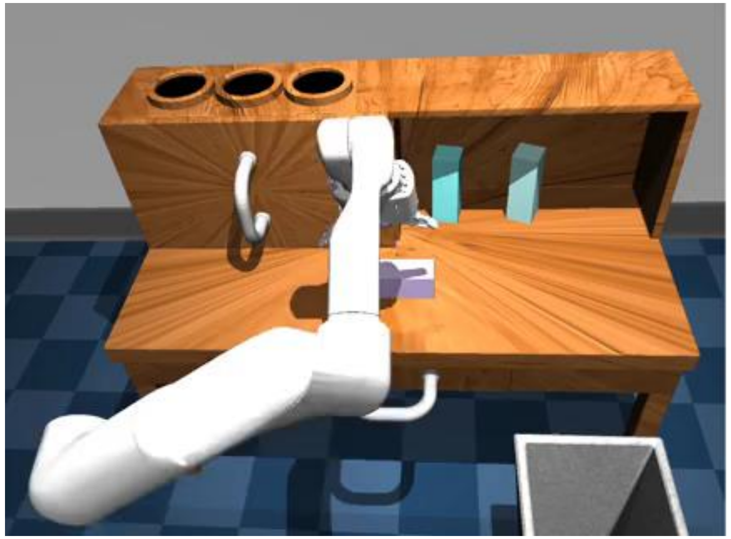
Stanford University



Learning Latent Plans from Play

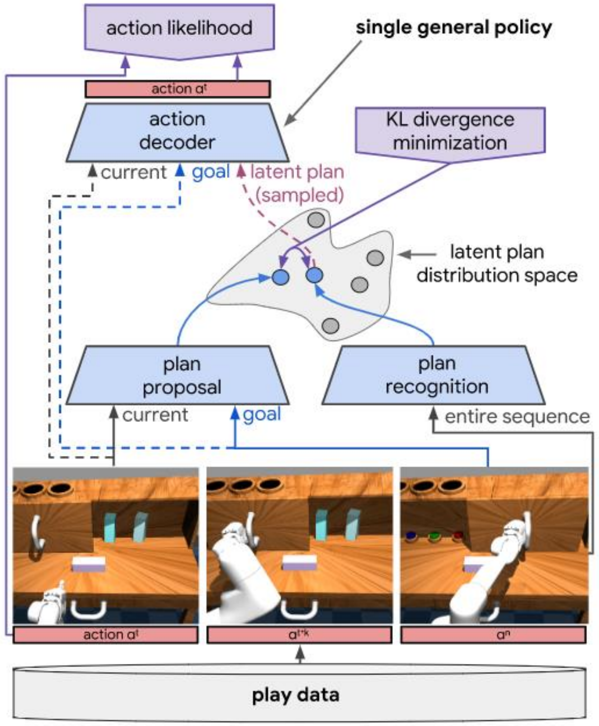
COREY LYNCH Google Brain | MOHI KHANSARI Google X | TED XIAO Google Brain | VIKASH KUMAR Google Brain | JONATHAN TOMPSON Google Brain | SERGEY LEVINE Google Brain | PIERRE SERMANET Google Brain

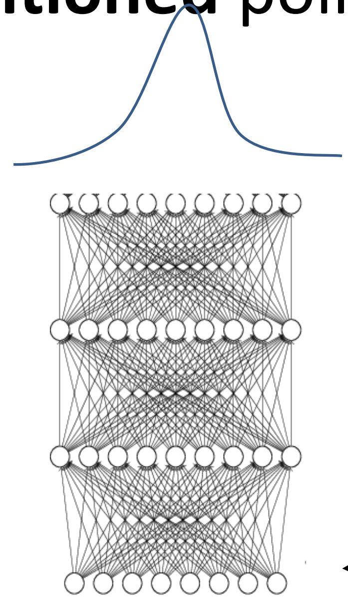
1. Collect **data**



2.5x speedup

2. Train **goal conditioned** policy



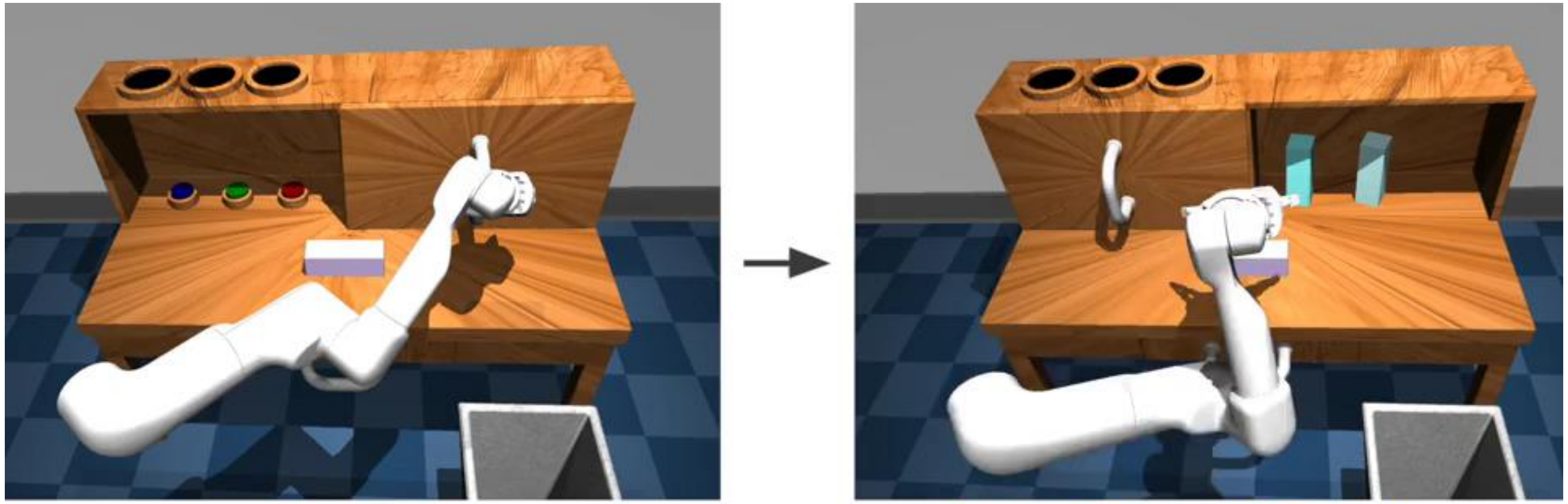


$\xi \sim \mathcal{N}(0, \mathbf{I})$

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

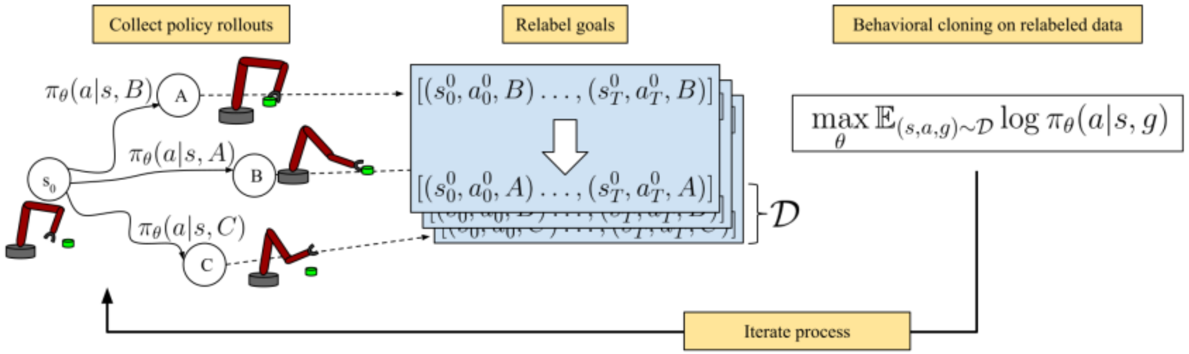


Goal → Single Play-LMP policy

Going **beyond** just imitation?

Learning to Reach Goals via Iterated Supervised Learning

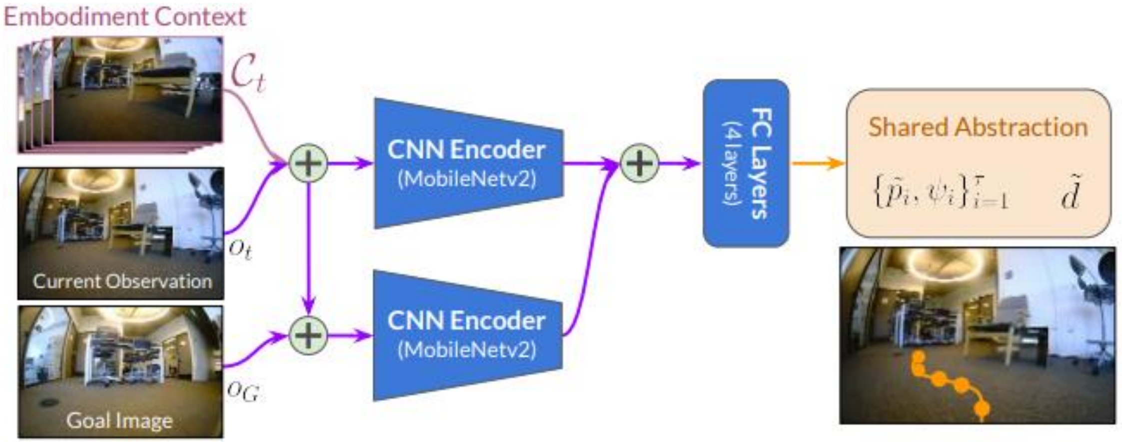
Dibya Ghosh* UC Berkeley
Abhishek Gupta* UC Berkeley
Ashwin Reddy UC Berkeley
Justin Fu UC Berkeley
Coline Devin UC Berkeley
Benjamin Eysenbach Carnegie Mellon University
Sergey Levine UC Berkeley



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

Goal-conditioned BC at a **huge** scale

	Dataset	Platform	Speed	Amt.	Environment
1	GoStanford [26]	TurtleBot2	0.5m/s	14h	office
2	RECON [32]	Jackal	1m/s	25h	off-road
3	CoryHall [35]	RC Car	1.2m/s	2h	hallways
4	Berkeley [33]	Jackal	2m/s	4h	suburban
5	SCAND-S [36]	Spot	1.5m/s	8h	sidewalks
6	SCAND-J [36]	Jackal	2m/s	1h	sidewalks
7	Seattle [37]	Warthog	5m/s	1h	off-road
8	TartanDrive [38]	ATV	10m/s	5h	off-road
	Ours			60h	




RC-Car
(Kahn et al. 2018)


TurtleBot
(Hirose et al. 2019)


Jackal
(Shah et al. 2021, 2022)


Spot
(Karnan et al. 2022)


Warthog
(Shaban et al. 2021)


ATV
(Triest et al. 2022)

Shah*, Sridhar*, Bhorkar, Hirose, Levine. **GNM: A General Navigation Model to Drive Any Robot**. 2022.

Also related (for later…)

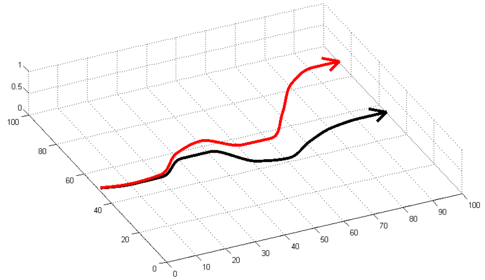
Hindsight Experience Replay

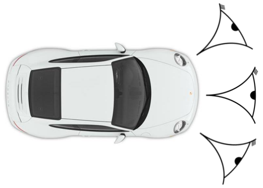
Marcin Andrychowicz[*], Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel[†], Wojciech Zaremba[†]
OpenAI

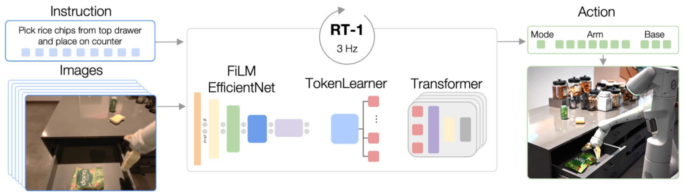
- Similar principle but with reinforcement learning
- This will make more sense later once we cover off-policy value-based RL algorithms
- Worth mentioning because this idea has been used widely outside of imitation (and was arguably first proposed there)

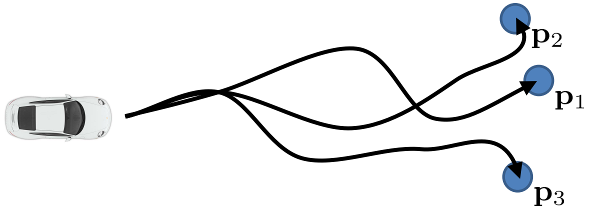
Where are we…

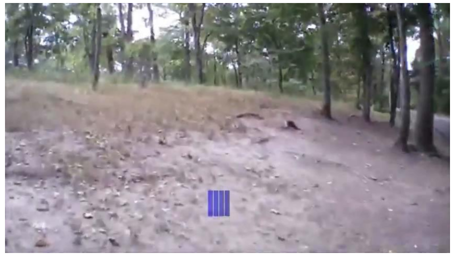
- Imitation learning via behavioral cloning is not guaranteed to work
  - This is **different** from supervised learning
  - The reason: i.i.d. assumption does not hold!
- We can formalize **why** this is and do a bit of theory
- We can address the problem in a few ways:
  - Be smart about how we collect (and augment) our data
  - Use very powerful models that make very few mistakes
  - Use multi-task learning
  - Change the algorithm (DAgger)



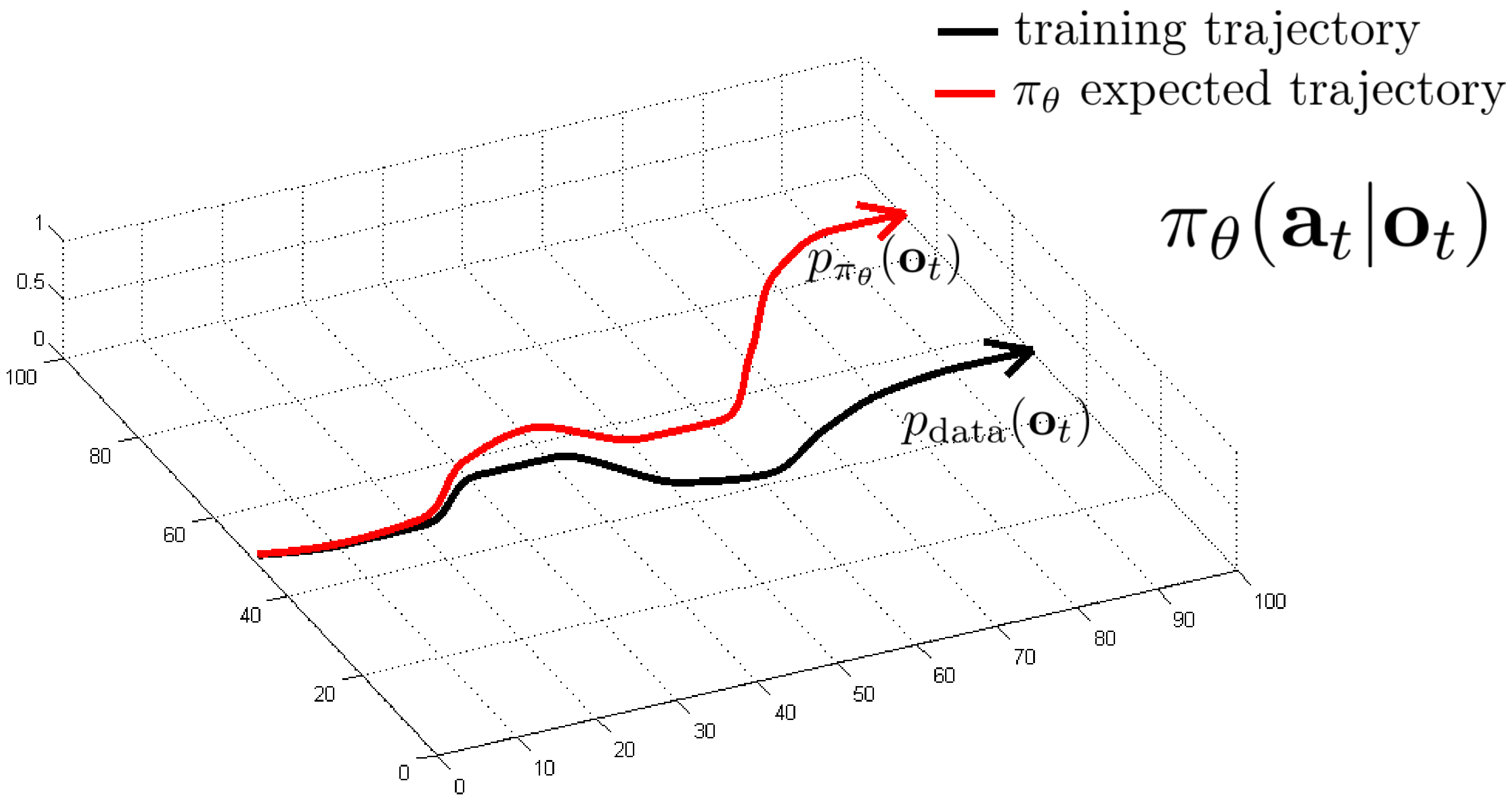








Can we make it work more often?



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

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 $p_{\text{data}}(\mathbf{o}_t)$!

DAgger: **D**ataset **Agg**regation

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

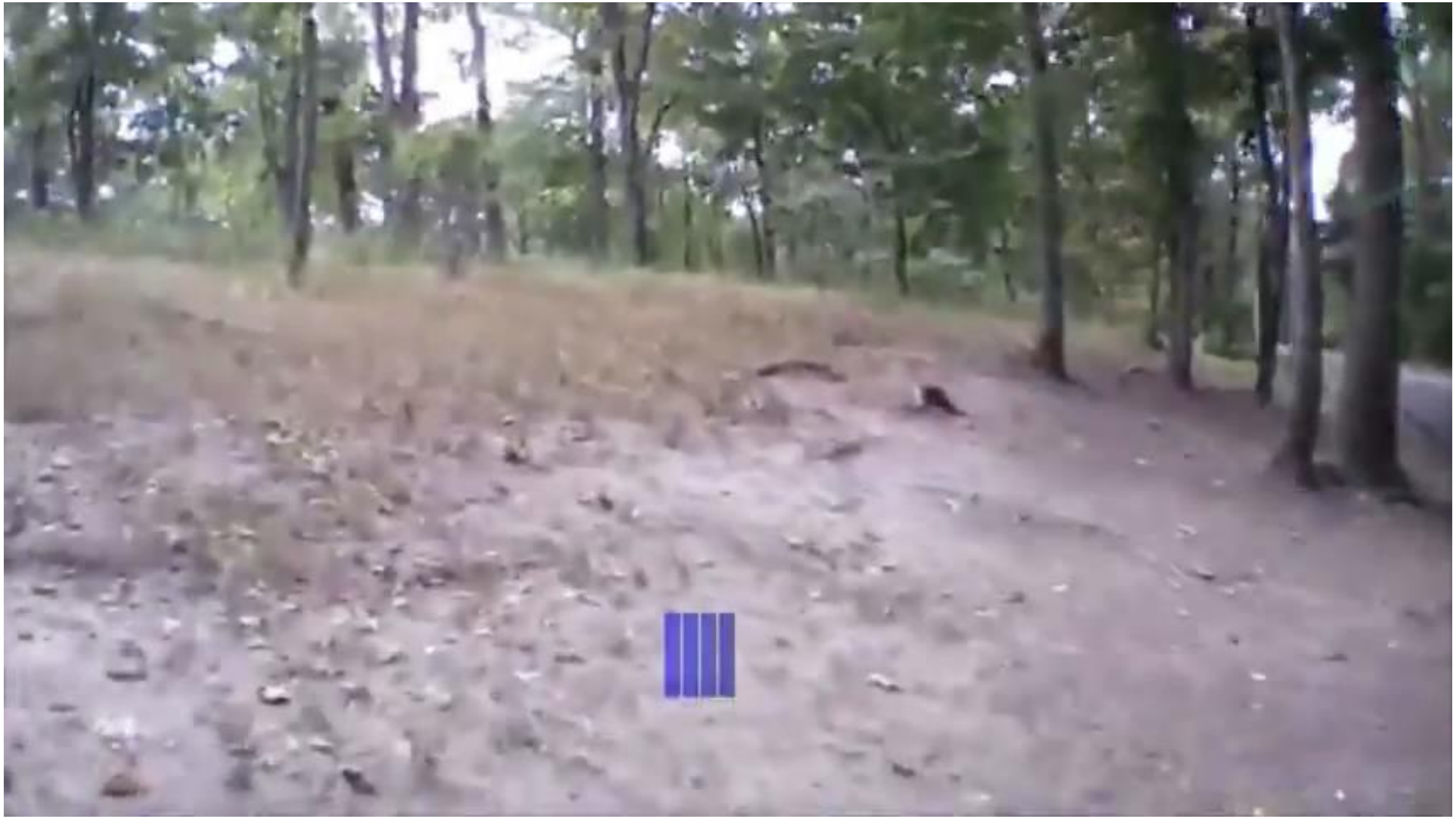
how? just run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

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

1. train $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{a}_1, \ldots, \mathbf{o}_N, \mathbf{a}_N\}$
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

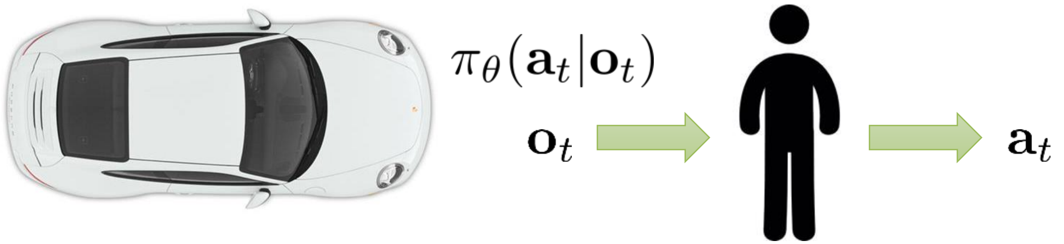
DAgger Example



Ross et al. ‘11

What's the problem?

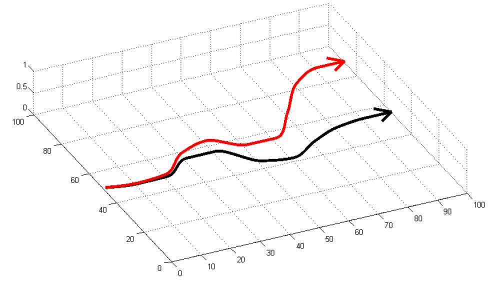
1. train $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{a}_1, \ldots, \mathbf{o}_N, \mathbf{a}_N\}$
2. run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask human to label $\mathcal{D}_\pi$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

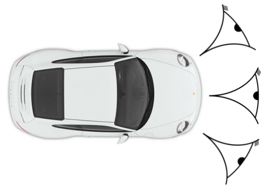


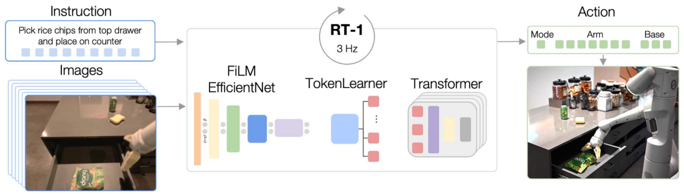
Ross et al. '11

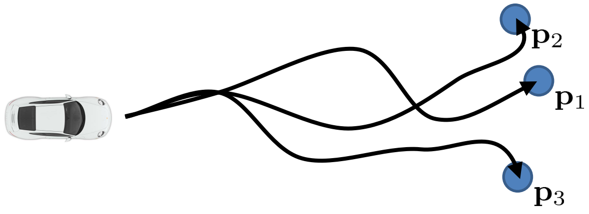
Recap

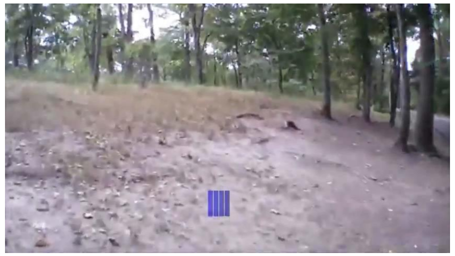
- Imitation learning via behavioral cloning is not guaranteed to work
  - This is **different** from supervised learning
  - The reason: i.i.d. assumption does not hold!
- We can formalize **why** this is and do a bit of theory
- We can address the problem in a few ways:
  - Be smart about how we collect (and augment) our data
  - Use very powerful models that make very few mistakes
  - Use multi-task learning
  - Change the algorithm (DAgger)







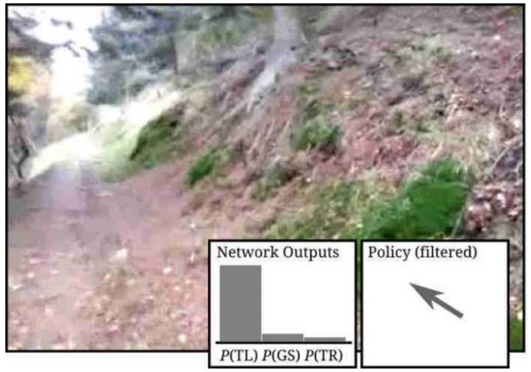


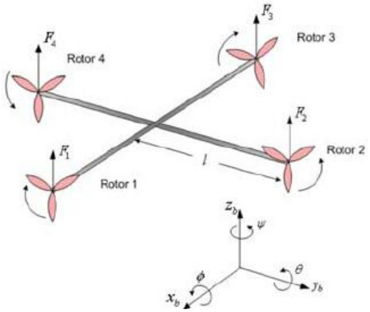


Cost functions and reward functions, a preview of what comes next

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

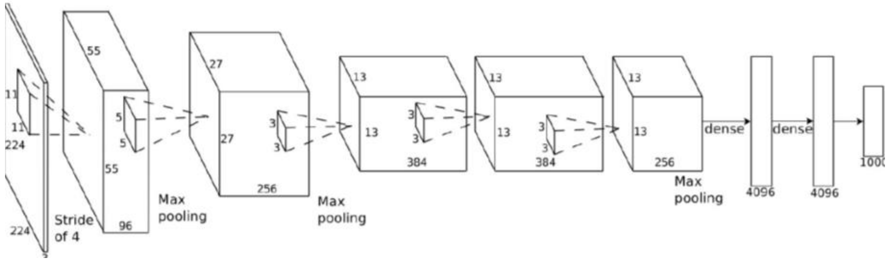




- Humans can learn autonomously; can our machines do the same?
  - Unlimited data from own experience
  - Continuous self-improvement

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

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

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

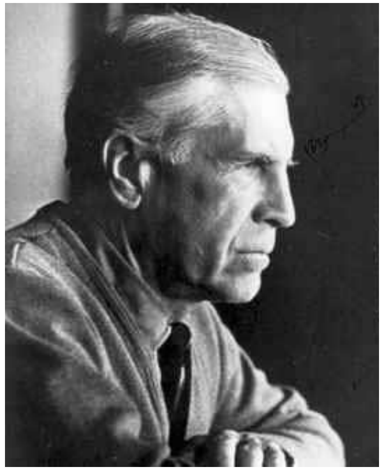
Aside: notation

$\mathbf{s}_t$ – 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

A cost function for imitation?

$\mathbf{o}_t$

$\mathbf{a}_t$

training data

supervised learning

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

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

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

$$\text{Goal: minimize } E_{\mathbf{s}_t \sim p_{\pi_\theta}(\mathbf{s}_t)}[c(\mathbf{s}_t, \mathbf{a}_t)]$$

$$\text{Goal: maximize } E_{\mathbf{s}_t \sim p_{\pi_\theta}(\mathbf{s}_t)}[r(\mathbf{s}_t, \mathbf{a}_t)]$$

Imitation learning algorithms **do** maximize reward when they work well!

For a very particular choice of reward