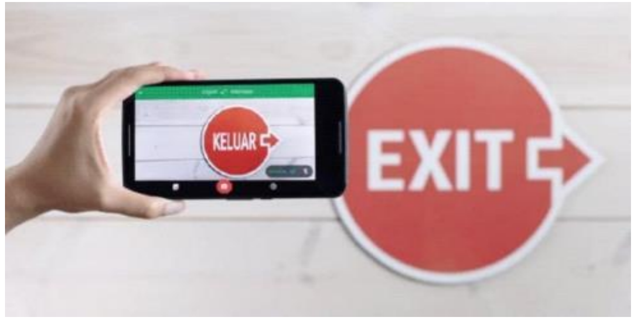
Learning-Based Control & Imitation

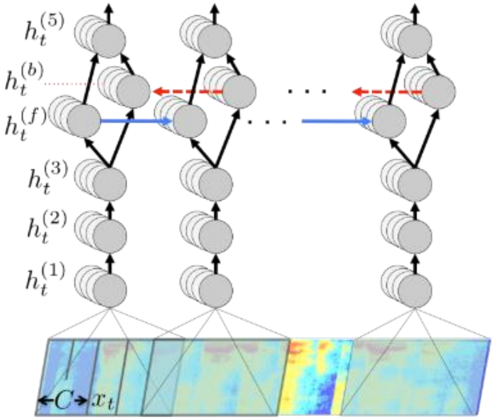
Designing, Visualizing and Understanding Deep Neural Networks

CS W182/282A

Instructor: Sergey Levine
UC Berkeley

So far: learning to *predict*





What about
learning to
control?

From *prediction* to *control*: challenges

i.i.d.: $p(\mathcal{D}) = \prod_i p(y_i|x_i)p(x_i)$

output y_1 does not change x_2

this is **very** important, because it allows us to just focus on getting the highest **average** accuracy over the whole dataset

making the wrong choice here is a disaster

making the wrong choice here is perhaps OK

From *prediction* to *control*: challenges

Ground truth labels:

"puppy"

Abstract goals:

"drive to the grocery store"

> what steering command is that?

From *prediction* to *control*: challenges

- i.i.d. distributed data (each datapoint is independent)
- ground truth supervision
- objective is to predict the right label

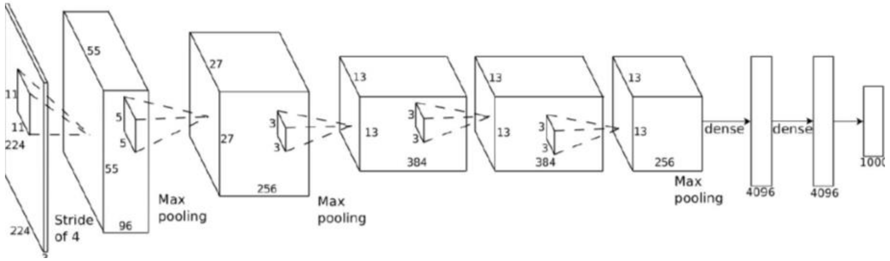
These are not **just** issues for control: in many cases, real-world deployment of ML has these same **feedback** issues

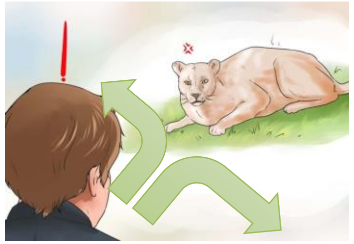
Example: decisions made by a traffic prediction system might affect the route that people take, which changes traffic

- each decision can change future inputs (not independent)
- supervision may be high-level (e.g., a goal)
- objective is to accomplish the task

We will **build up** toward a **reinforcement learning** system that addresses all of these issues, but we'll do so one piece at a time…

Terminology





$\mathbf{o}_t$

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

$\mathbf{a}_t$

used to be $p_\theta(y|x)$

used to be x

used to be y

$\mathbf{s}_t$ – state

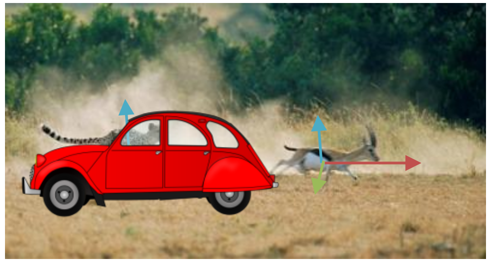
$\mathbf{o}_t$ – observation

$\mathbf{a}_t$ – action

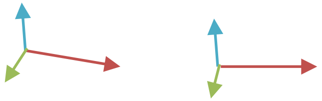
$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ – policy

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

This distinction will very important later, but is not so important today



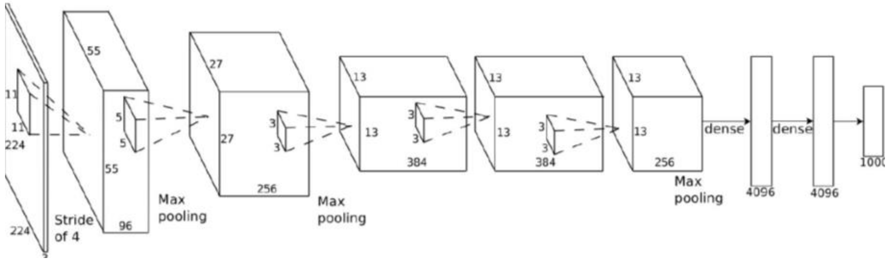
$\mathbf{o}_t$ – observation



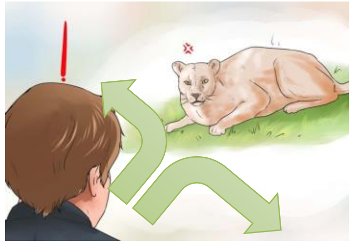
$\mathbf{s}_t$ – state

Terminology

$\mathbf{o}_t$



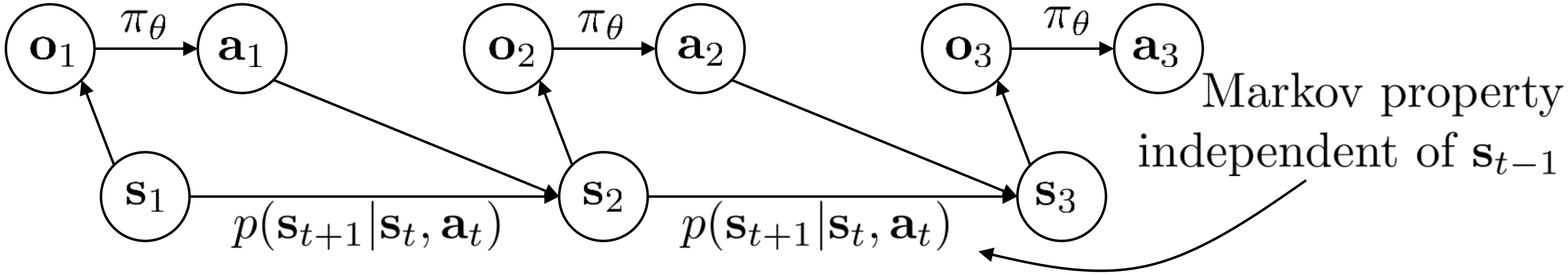
$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$



$\mathbf{a}_t$

$\mathbf{s}_t$ – 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)

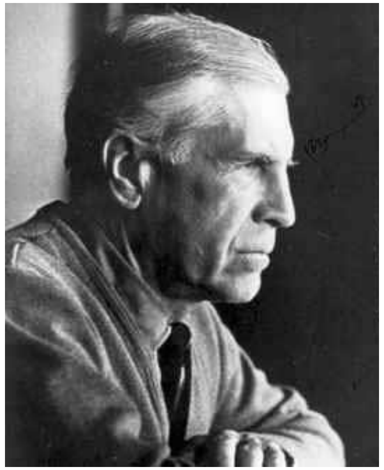


Aside: notation

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

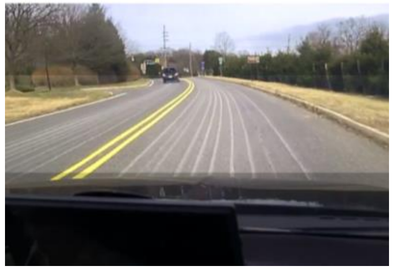
$\mathbf{x}_t$ – state
$\mathbf{u}_t$ – action   управление

Richard Bellman

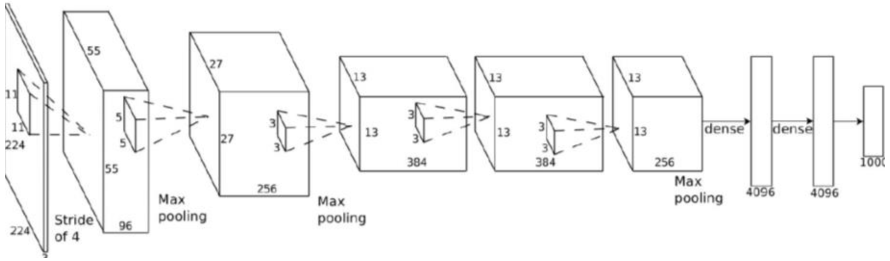


Lev Pontryagin

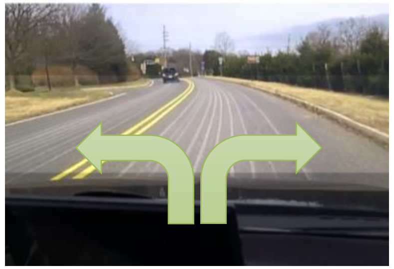
Imitation Learning



$\mathbf{o}_t$



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



$\mathbf{a}_t$

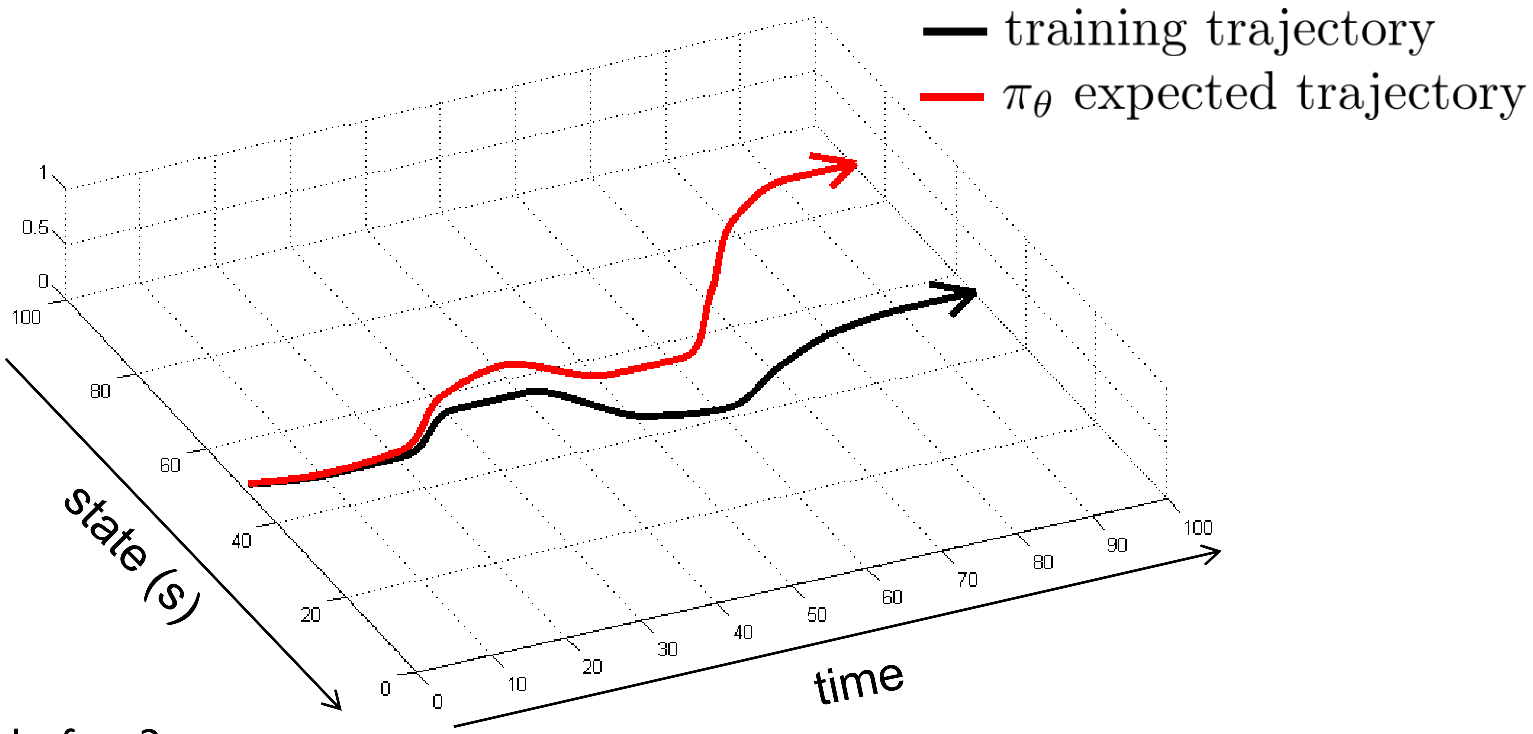
$\mathbf{o}_t$
$\mathbf{a}_t$

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

behavioral cloning

Does it work?

No!



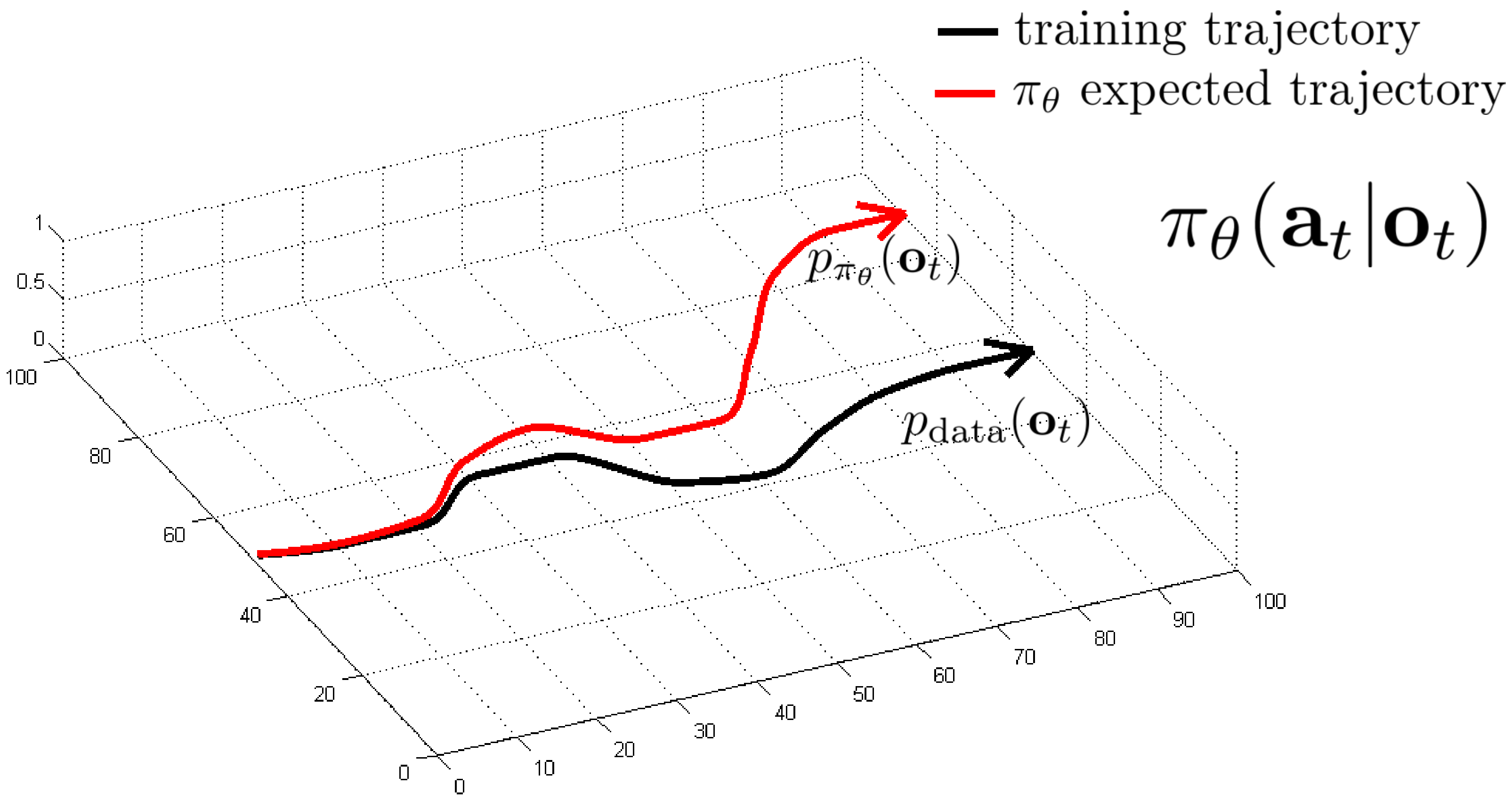
Where have we seen this before?

Does it work? Yes!

Video: Bojarski et al. ‘16, NVIDIA

Getting behavioral cloning to work

What is the problem?

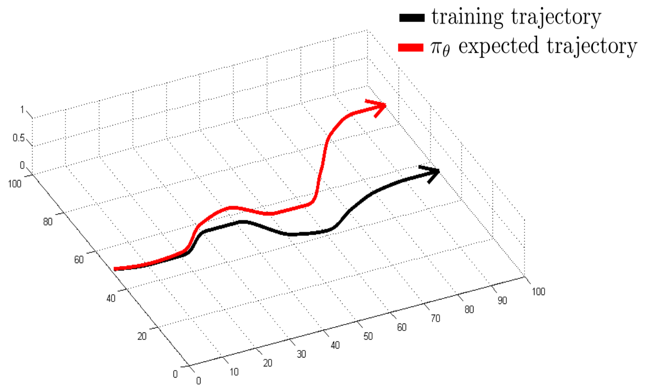


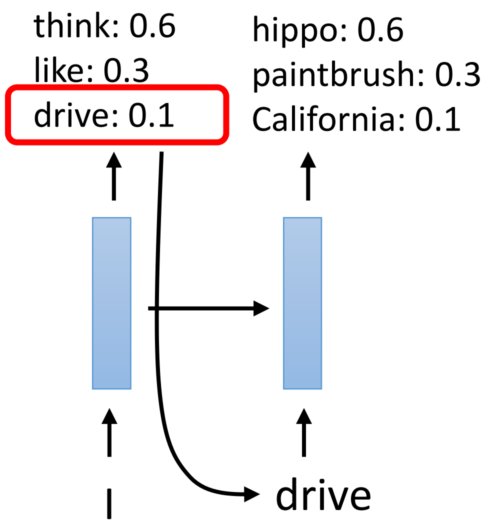
$$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$$

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

What is the problem?

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





unlikely but possible mistake

complete nonsense, because the network never saw inputs remotely like this

This is the same problem!

The problem: this is a training/test discrepancy: the network always saw **true** sequences as inputs, but at test-time it gets as input its own (potentially incorrect) predictions

we got unlucky, but now the model is completely confused

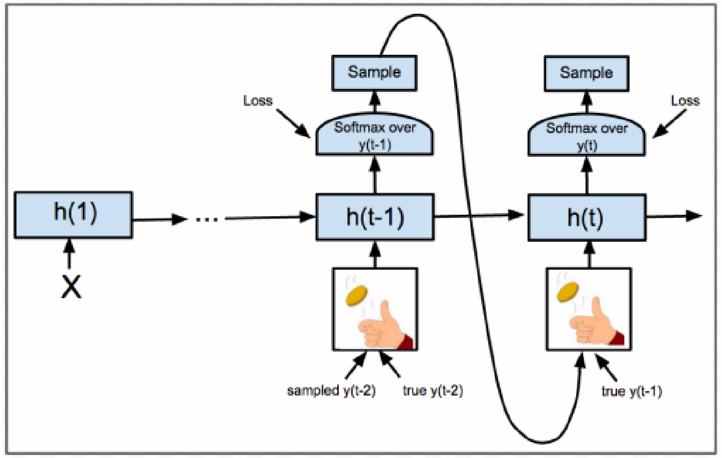
it never saw “I drive” before

This is called **distributional shift**, because the input distribution **shifts** from true strings (at training) to synthetic strings (at test time)

Why not use the same solution?

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

Before: scheduled sampling

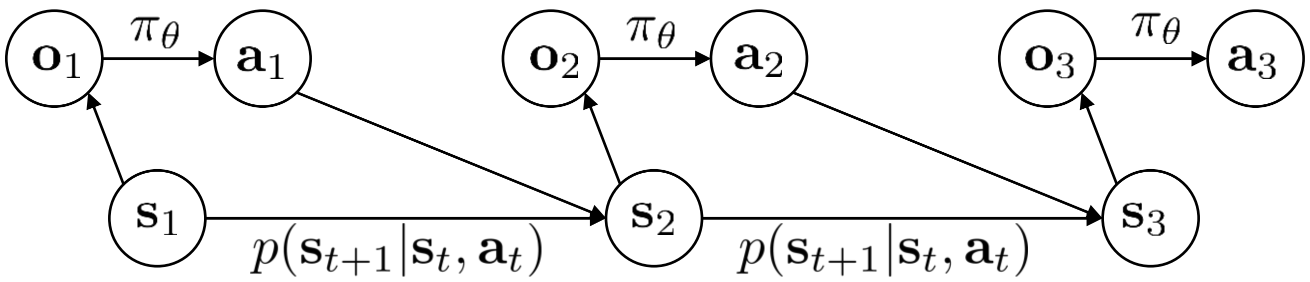


Now: control

we *could* take the predicted action $\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ and observe the resulting $\mathbf{o}_{t+1}$

but this requires interacting with the world!
why?

we don't know $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$!



Can we **mitigate** the problem?

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

if $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ is *very* accurate

maybe $p_{\text{data}}(\mathbf{o}_t) \approx p_\theta(\mathbf{o}_t)$

Why **might** we fail to fit the expert?

1. Non-Markovian behavior
2. Multimodal behavior

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

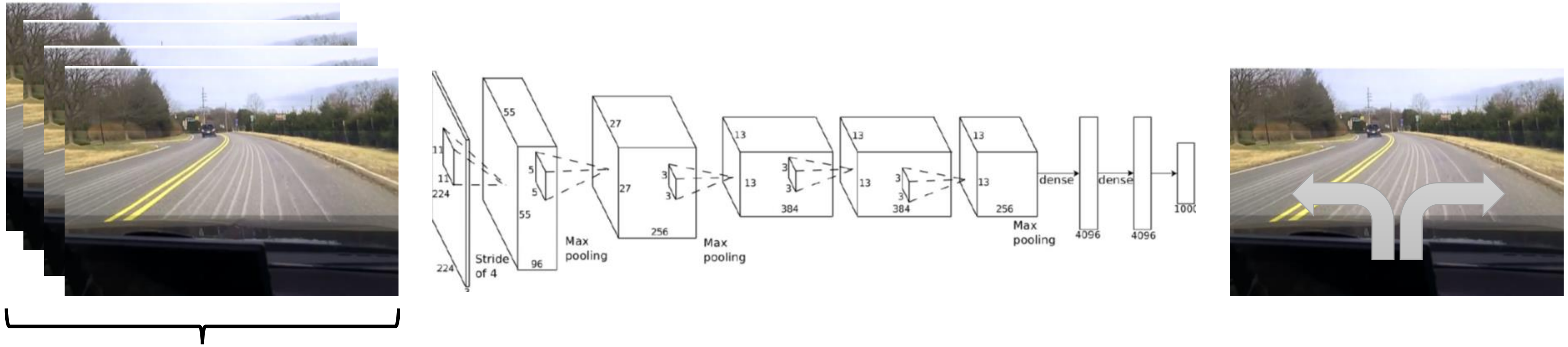
behavior depends only on current observation

behavior depends on all past observations

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

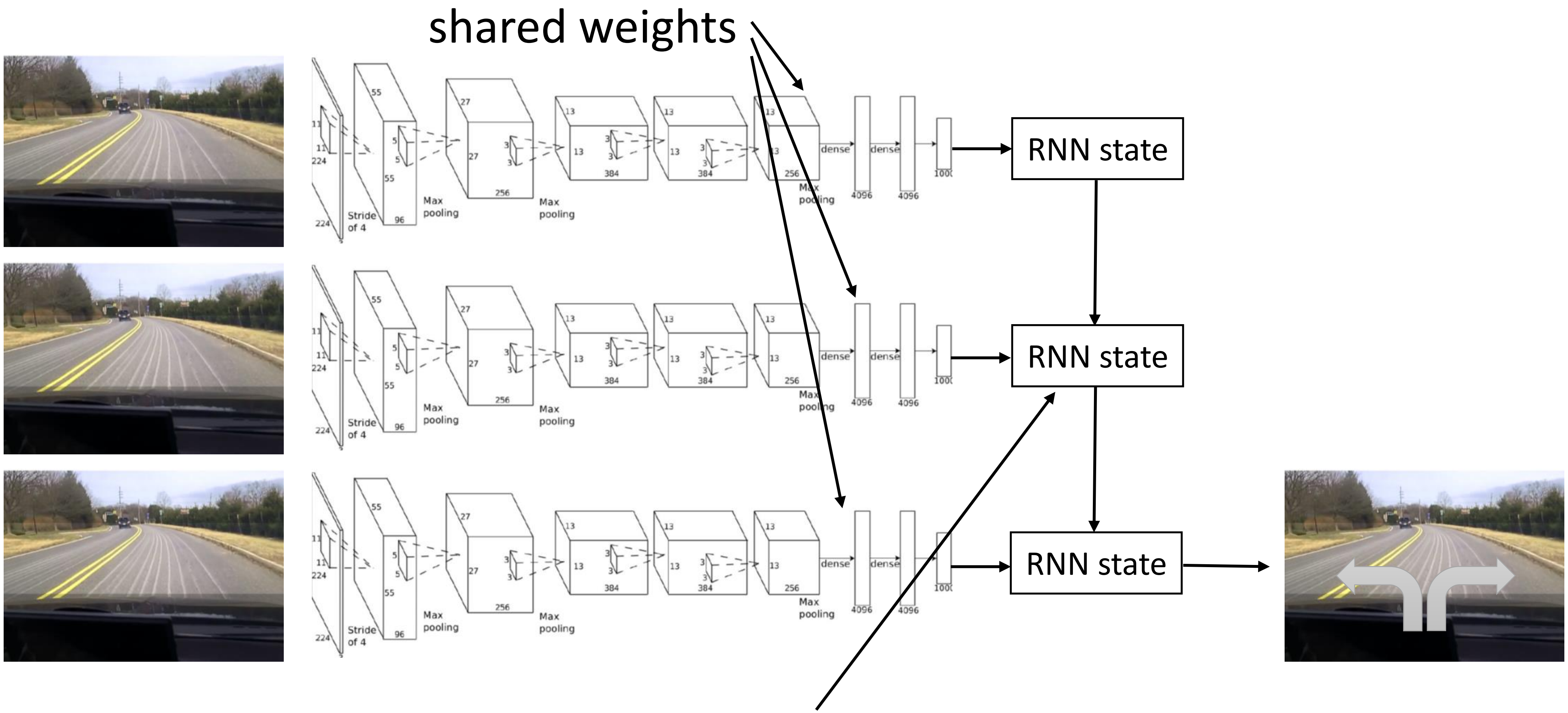
Often very unnatural for human demonstrators

How can we use the whole history?



variable number of frames,
too many weights

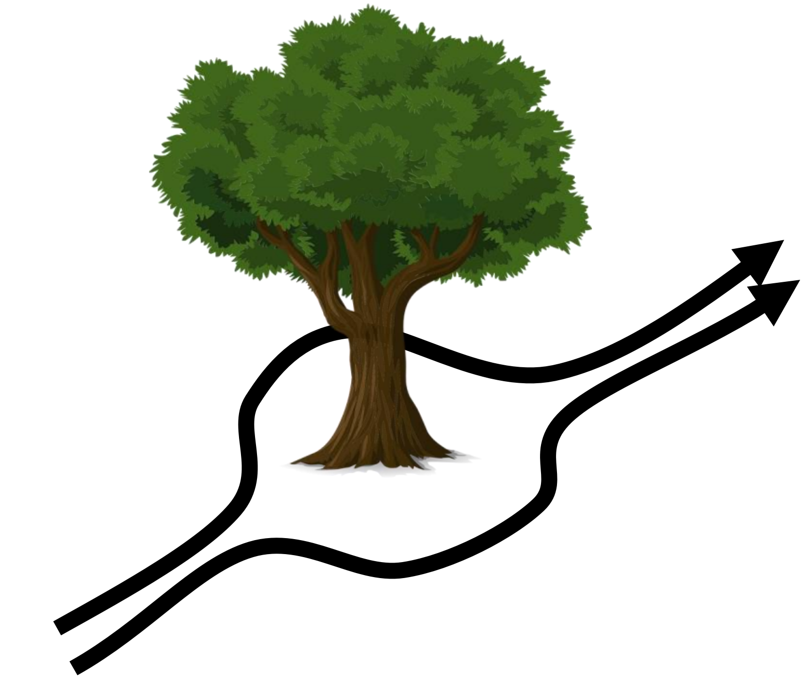
How can we use the whole history?

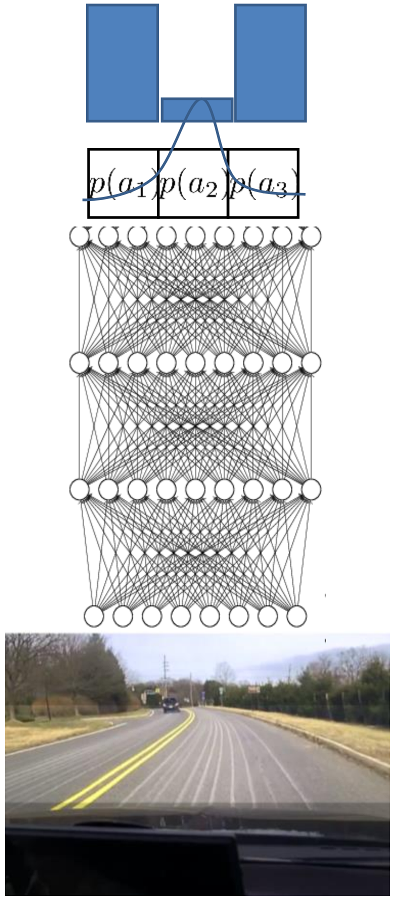


Typically, LSTM cells work better here

Why might we fail to fit the expert?

1. Non-Markovian behavior
2. Multimodal behavior



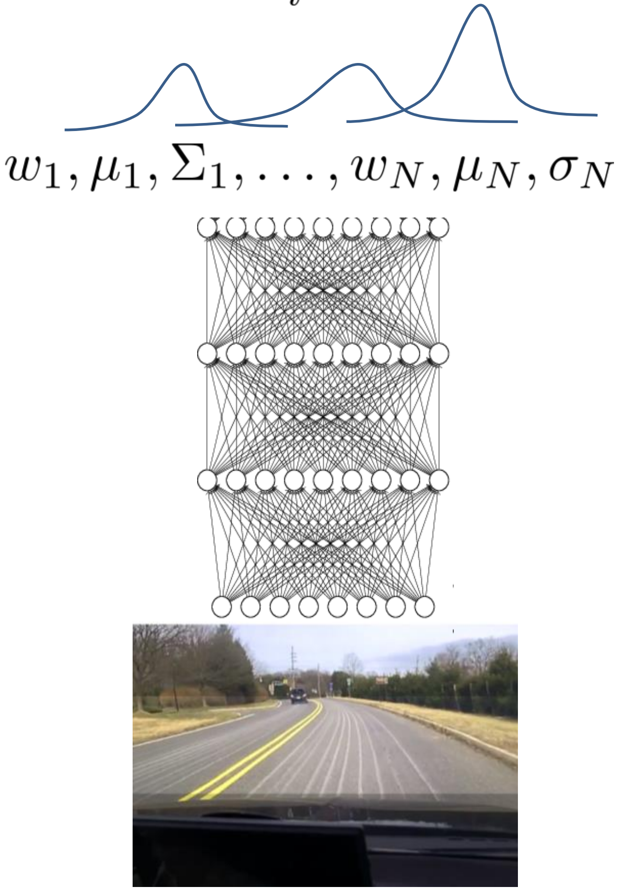


1. Output mixture of Gaussians
2. Latent variable models
3. Autoregressive discretization

Why might we fail to fit the expert?

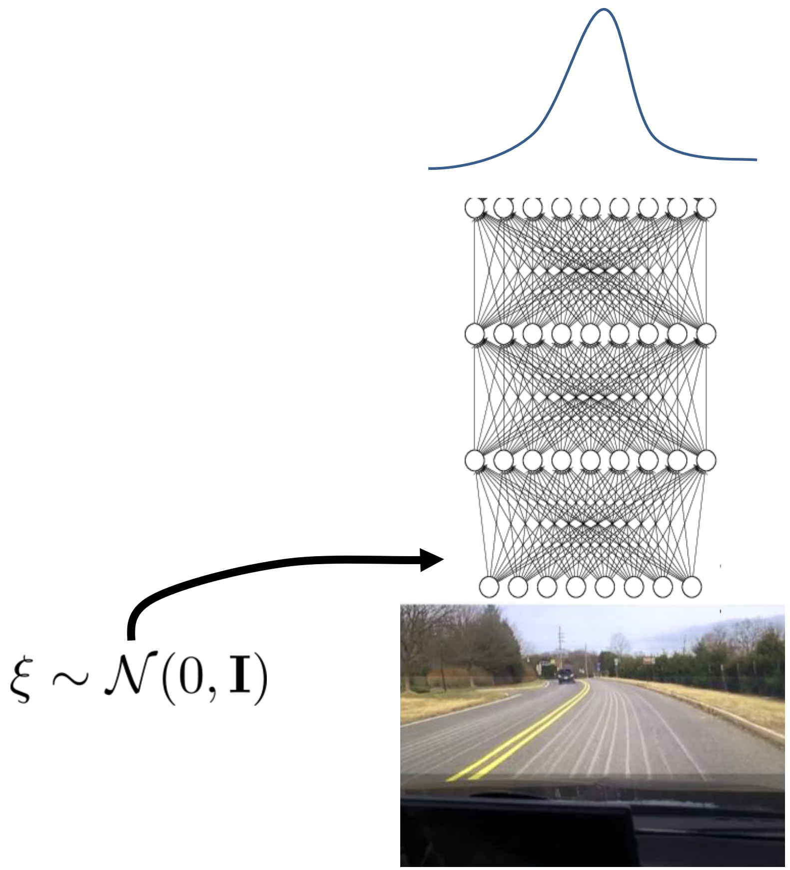
1. Output mixture of Gaussians
2. Latent variable models
3. Autoregressive discretization

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



Why might we fail to fit the expert?

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

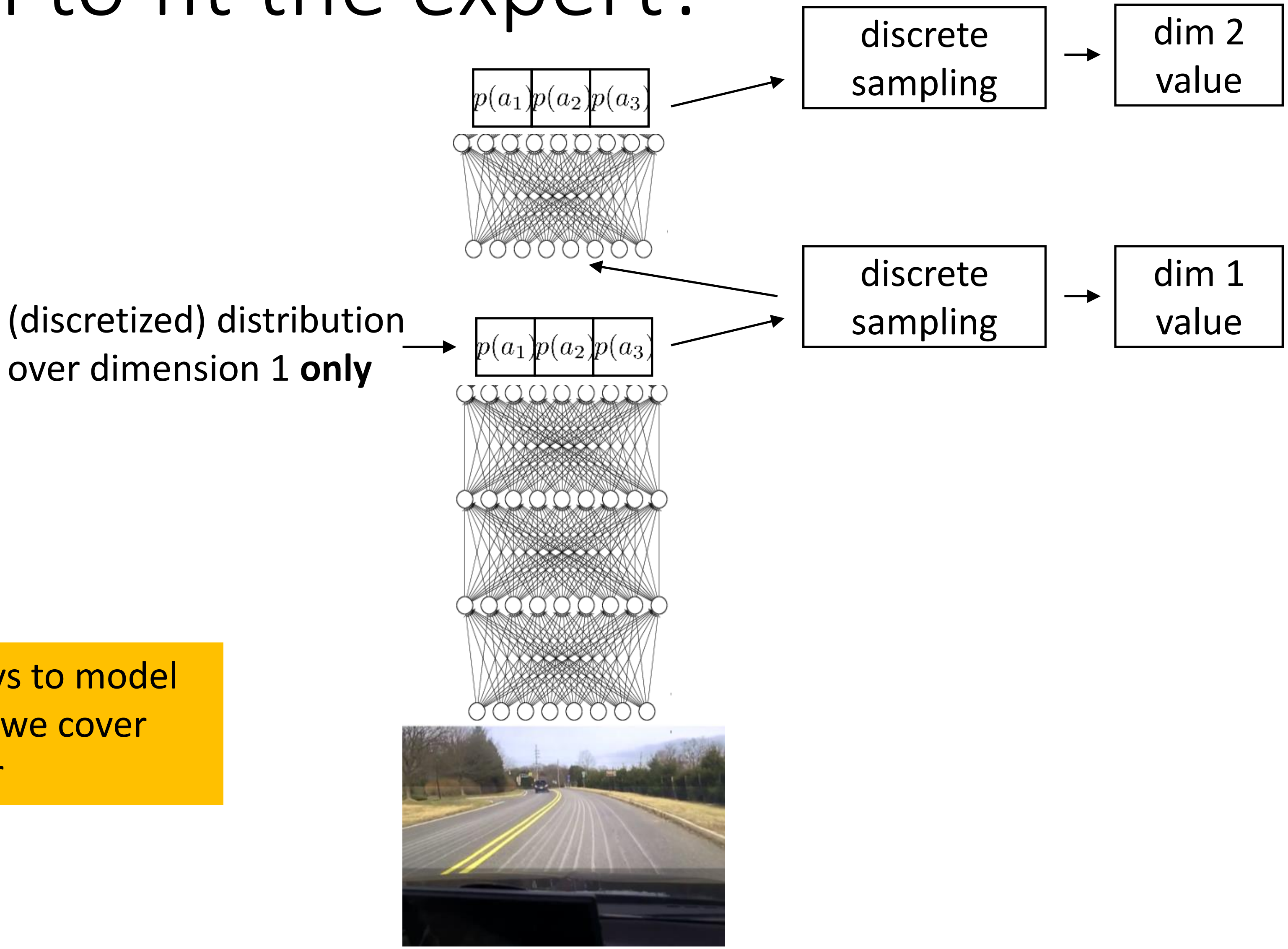


Look up some of these:

- 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

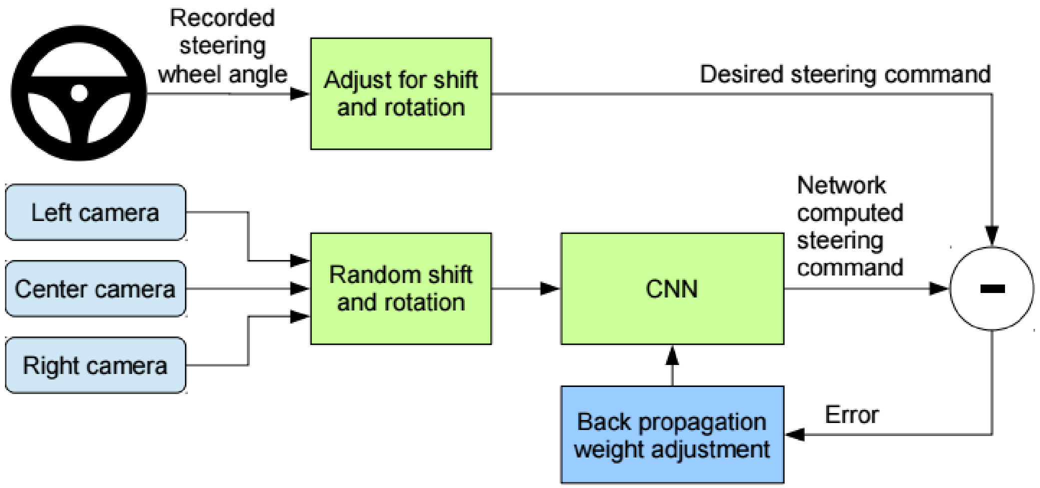


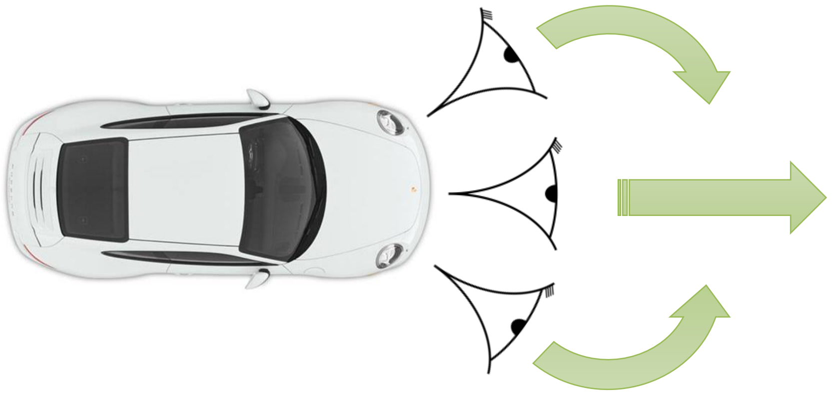
We'll learn more about better ways to model multi-modal distributions when we cover generative models later

Does it work? Yes!

Video: Bojarski et al. ‘16, NVIDIA

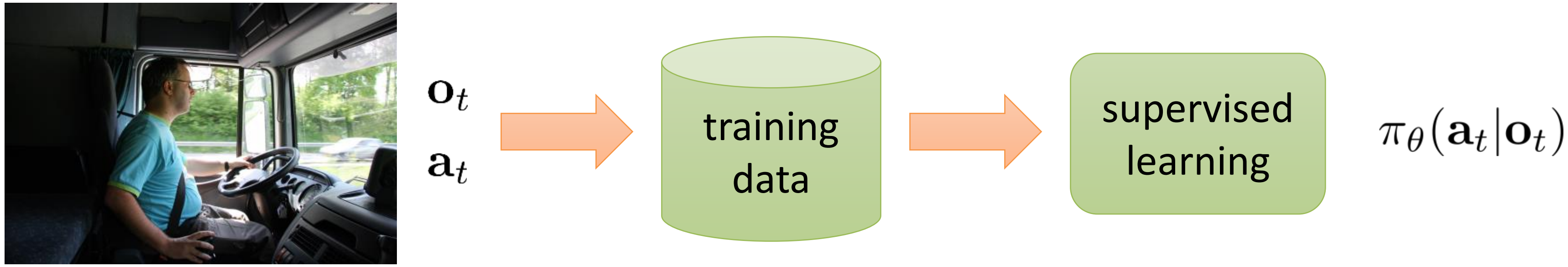
Why did that work?



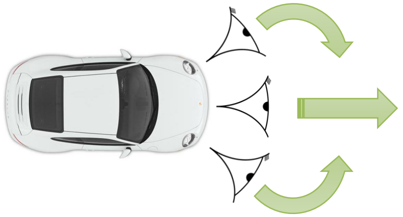


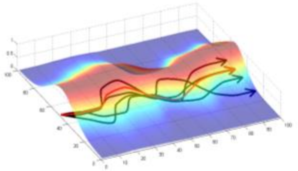
Bojarski et al. ‘16, NVIDIA

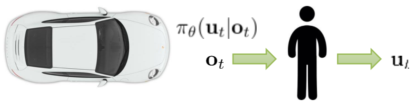
Summary



- In principle it should not work
  - Distribution mismatch problem
- Sometimes works well
  - Hacks (e.g. left/right images)
  - Models with memory (i.e., RNNs)
  - Better distribution modeling
  - Generally taking care to get high accuracy

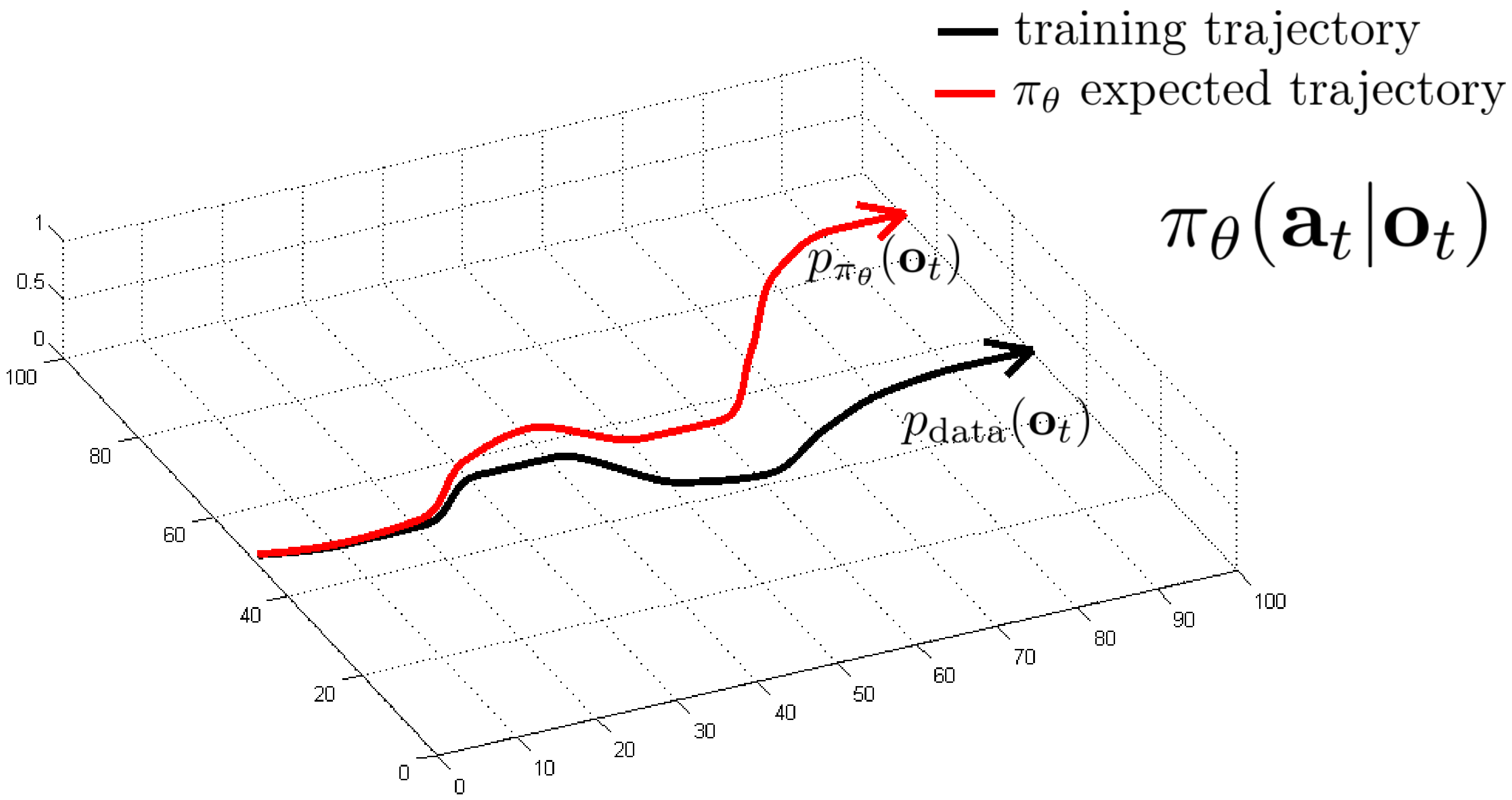






A (perhaps) better approach

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

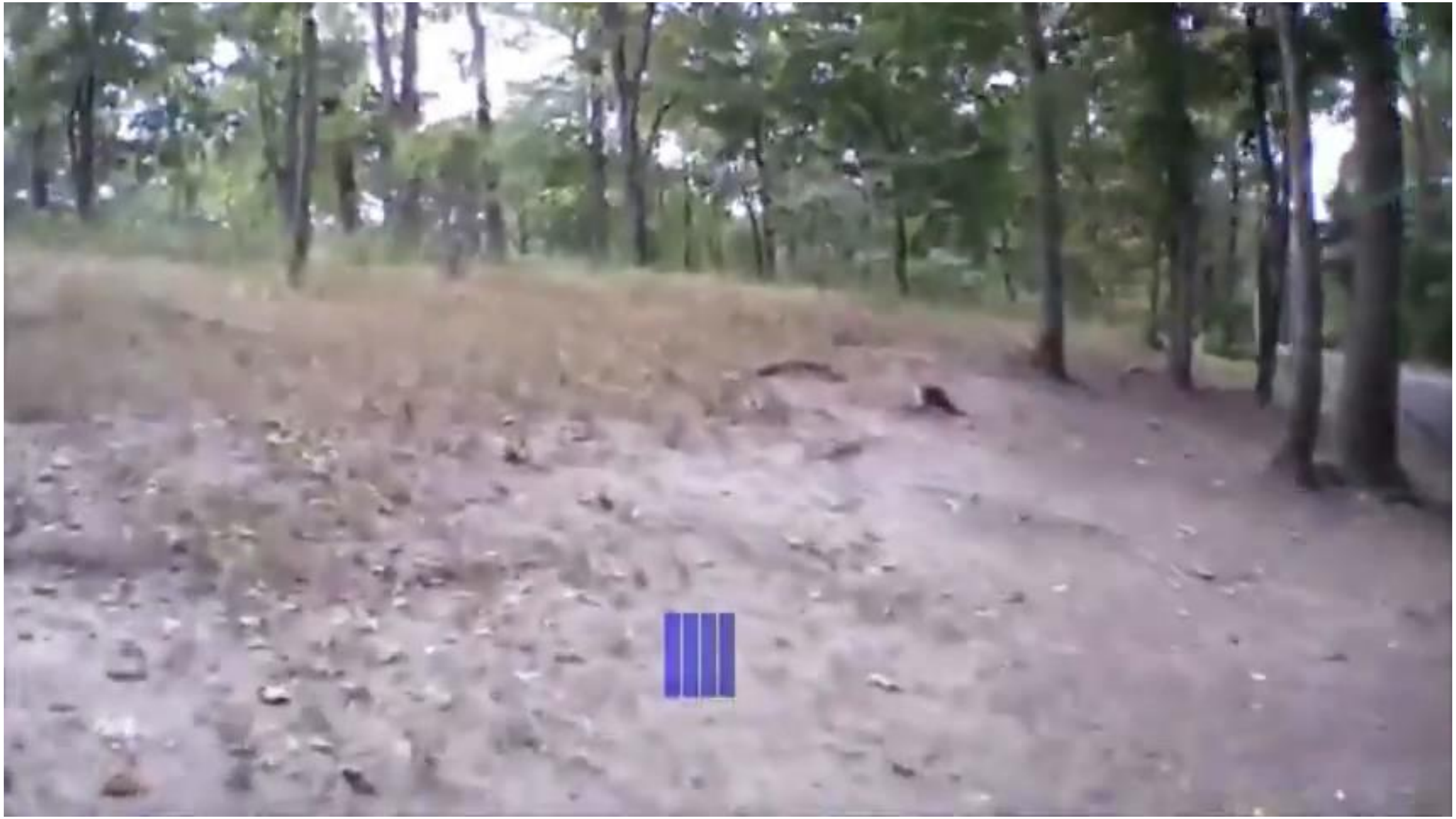
how? just run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

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

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

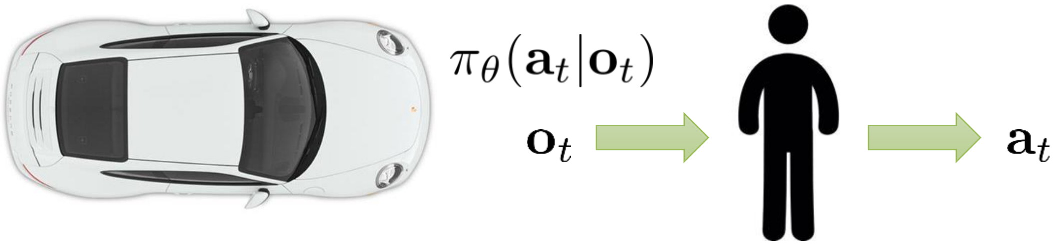
DAgger Example



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

Summary and takeaways

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

- In principle it should not work
  - Distribution mismatch problem
  - **DAgger can address this, but requires costly data collection and labeling**
- Sometimes works well
  - Requires a bit of (heuristic) hacks, and very good (high-accuracy) models

My recommendation: try behavioral cloning first, but prepare to be disappointed

Next time

- i.i.d. distributed data (each datapoint is independent)
- ground truth supervision
- objective is to predict the right label

- each decision can change future inputs (not independent)
- supervision may be high-level (e.g., a goal)
- objective is to accomplish the task

We'll tackle these issues with **reinforcement learning**