Carnegie Mellon
School of Computer Science

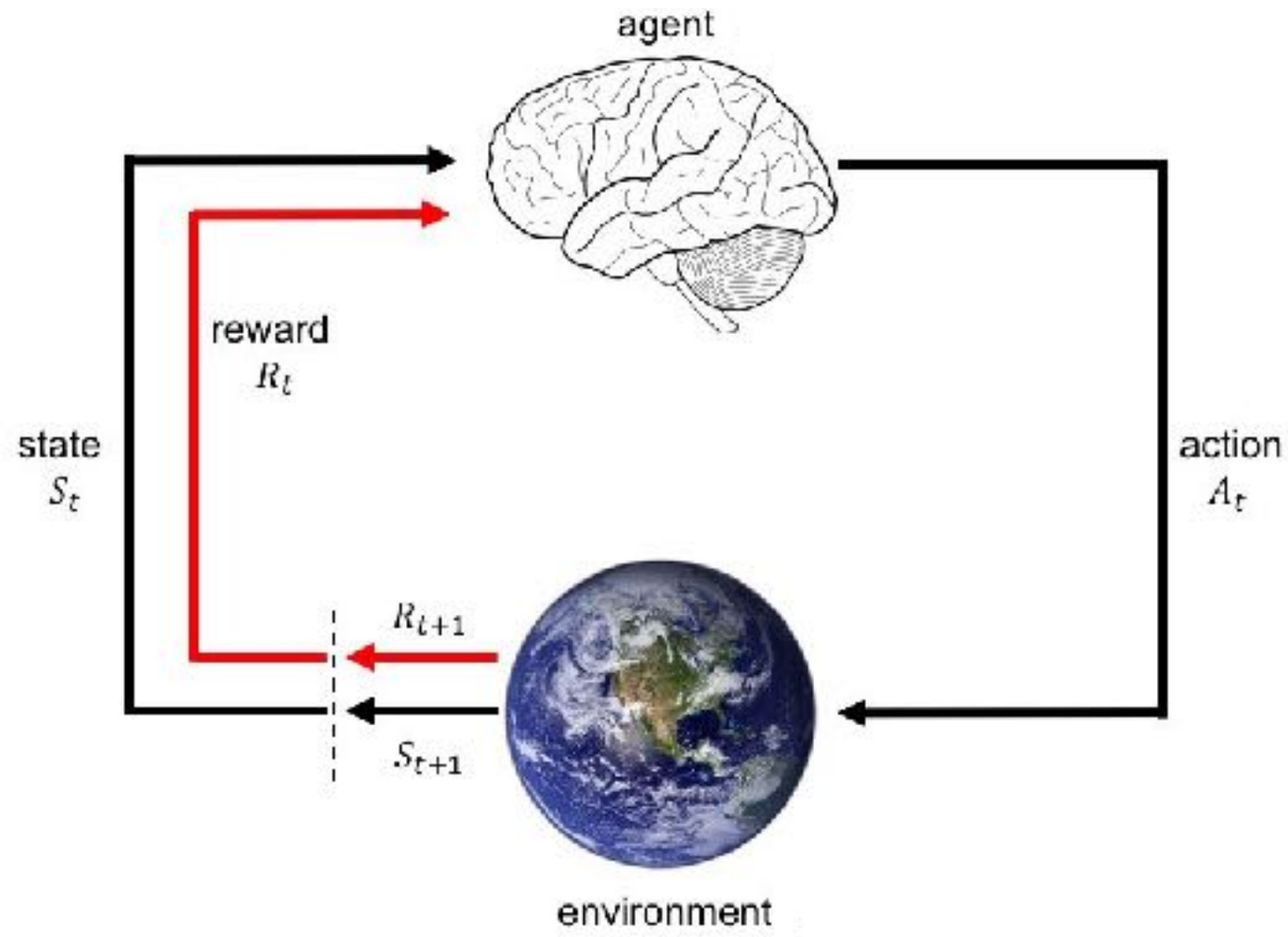
Deep Reinforcement Learning and Control

Imitation Learning

Spring 2019, CMU 10-403

Katerina Fragkiadaki

Reinforcement learning



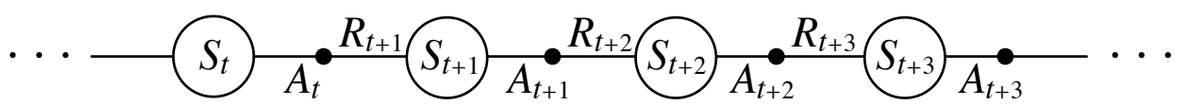
Agent and environment interact at discrete time steps: $t = 0, 1, 2, 3, \ldots$

Agent observes state at step t: $S_t \in \mathcal{S}$

produces action at step t : $A_t \in \mathcal{A}(S_t)$

gets resulting reward: $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$

and resulting next state: $S_{t+1} \in \mathcal{S}^+$



Limitations of Learning by Interaction

- The agent should have the chance to try (and fail) MANY times
- This is impossible when safety is a concern: we cannot afford to fail
- This is also quite impossible in general in real life where each interaction takes time (in contrast to simulation)

Crusher robot

Learning from Demonstration for Autonomous Navigation in Complex Unstructured Terrain, Silver et al. 2010

Imitation Learning (a.k.a. Learning from Demonstrations)

visual imitation

The actions of the teacher need to be inferred from visual sensory input and mapped to the end-effectors to the agent.

Two challenges:

1) visual understanding
2) action mapping, especially when the agent and the teacher do not have the same action space

we will come back to this in a later lecture

kinesthetic imitation

- The teacher takes over the end-effectors of the agent.
- Demonstrated actions can be imitated directly (*cloned*)
- A.k.a. behavior cloning

this lecture!

Imitating Controllers

visual imitation

kinesthetic imitation

- Experts do not need to be humans.
- Machinery that we develop in this lecture can be used for imitating expert policies found through (easier) optimization in a constrained smaller part of the state space.
- Imitation then means distilling knowledge of expert constrained policies into a general policy that can do well in all scenarios the simpler policies do well.

The actions of the teacher need to be inferred from visual sensory input and mapped to the end-effectors to the agent.

Two challenges:

1) visual understanding
2) action mapping, especially when the agent and the teacher do not have the same action space

We will come back to this in a later lecture

- The teacher takes over the end-effectors of the agent.
- Demonstrated actions can be imitated directly (cloned)
- A.k.a. behavior cloning

this lecture!

Notation

Richard Bellman

Lev Pontryagin

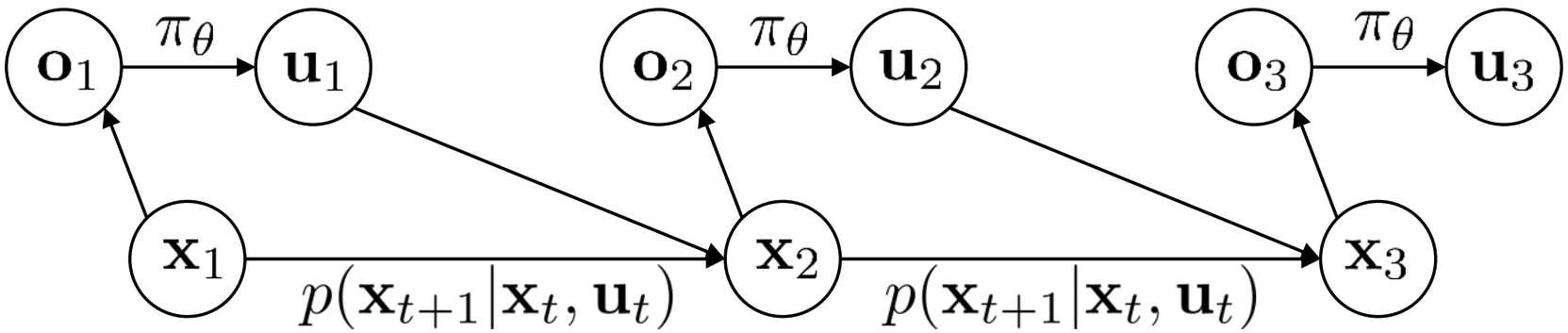
actions	a_t
states	s_t
rewards	r_t
dynamics	$p(s_{t+1} \mid s_t, a_t)$
observations	o_t

actions	u_t
states	x_t
costs	$c(x_t, u_t)$
dynamics	$p(x_{t+1} \mid x_t, u_t)$

Diagram from Sergey Levine

Imitation learning VS Sequence labelling

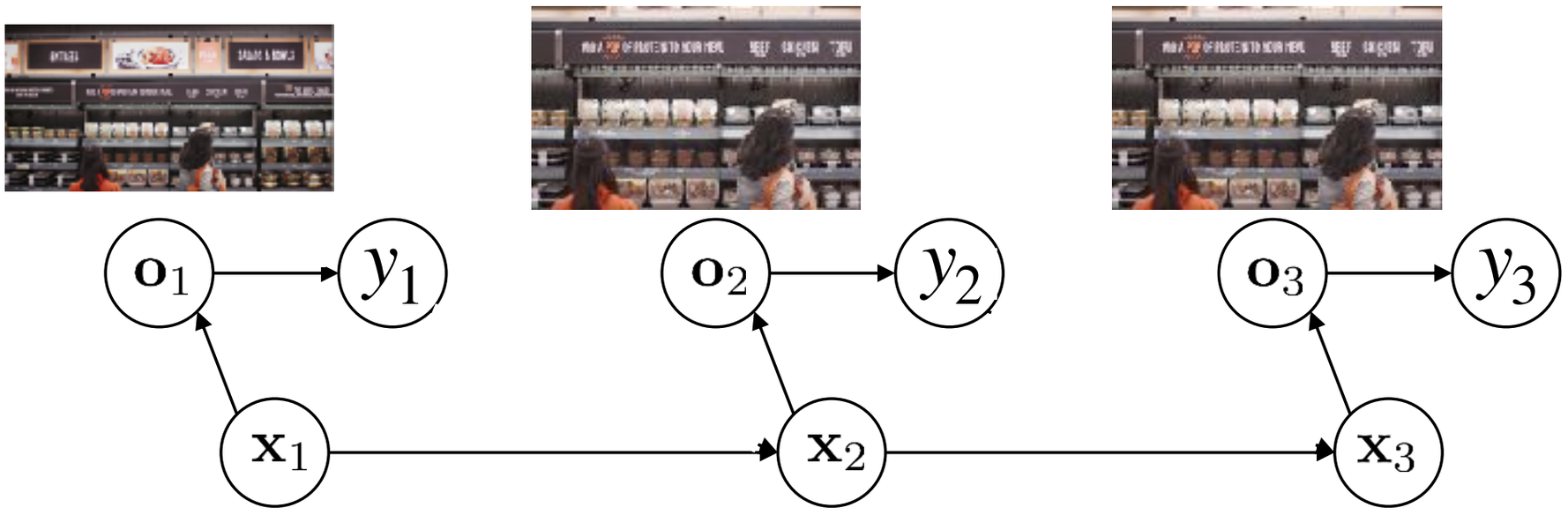
Imitation learning



Training data:

$o_1^1, u_1^1, o_2^1, u_2^1, o_3^1, u_3^1, \ldots.$
$o_1^2, u_1^2, o_2^2, u_2^2, o_3^2, u_3^2, \ldots.$
$o_1^3, u_1^3, o_2^3, u_2^3, o_3^3, u_3^3, \ldots.$

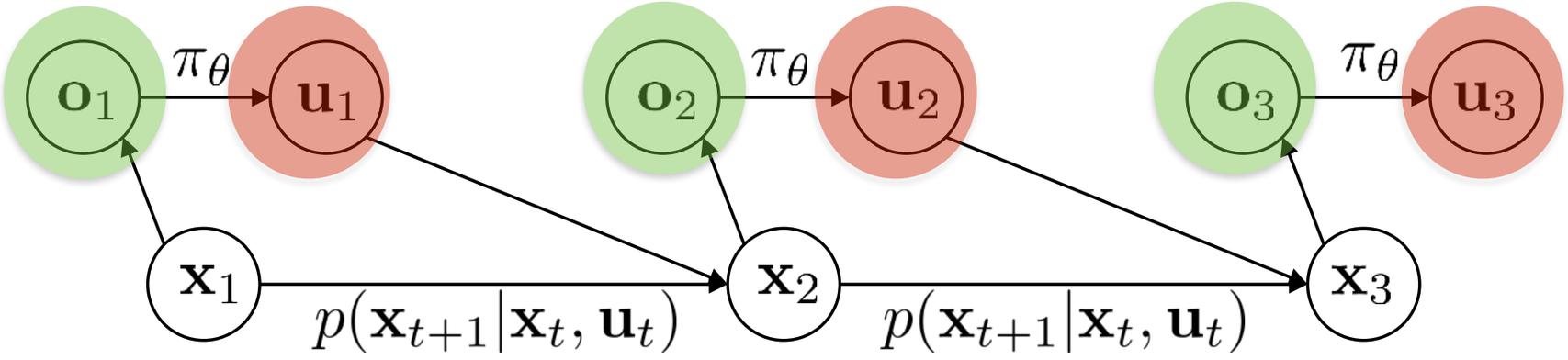
Sequence labelling



y: which product was purchased if any

Imitation learning VS Sequence labelling

Imitation learning



Training data:

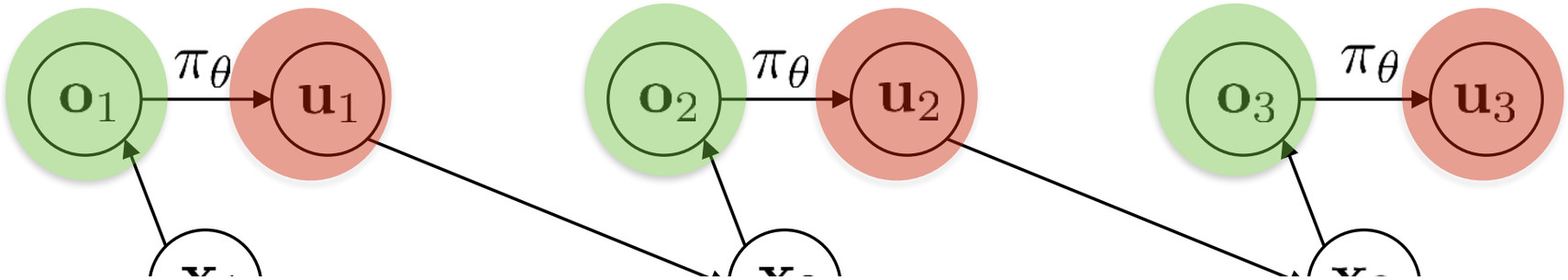
$o_1^1, u_1^1, o_2^1, u_2^1, o_3^1, u_3^1, \ldots.$
$o_1^2, u_1^2, o_2^2, u_2^2, o_3^2, u_3^2, \ldots.$
$o_1^3, u_1^3, o_2^3, u_2^3, o_3^3, u_3^3, \ldots.$

Sequence labelling

y: which product was purchased if any

Imitation learning VS Sequence labelling

Imitation learning



Action interdependence in imitation learning: the actions we predict will influence the data we will see next, and thus, our future predictions.

g data:

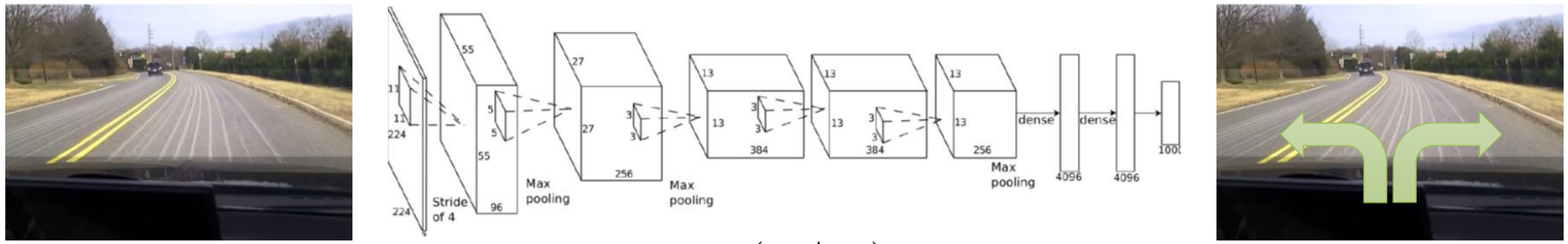
$_3^1, u_3^1, \ldots.$
$_3^2, u_3^2, \ldots.$
$_3^3, u_3^3, \ldots.$

Label interdependence is present in any structured prediction task, e.g, text generation: words we predict influence words we need to predict further down the sentence…

y: which product was purchased if any

Imitation Learning for Driving

Driving policy: a mapping from observations to steering wheel angles

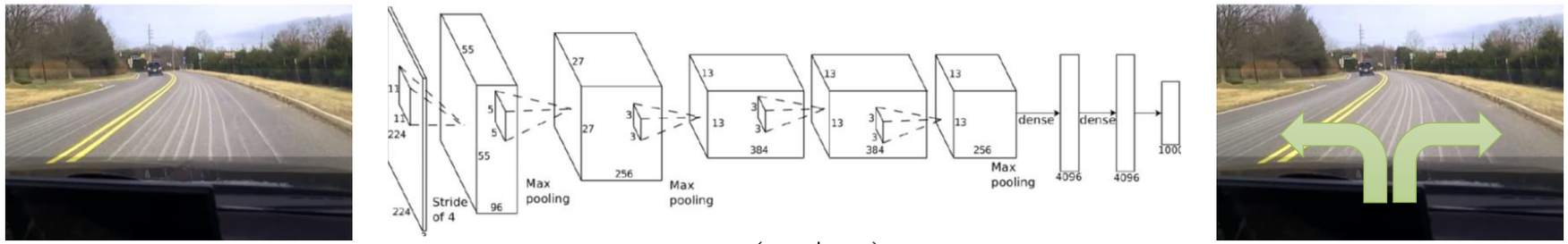


$\mathbf{o}_t$ $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ $\mathbf{u}_t$

End to End Learning for Self-Driving Cars, Bojarski et al. 2016

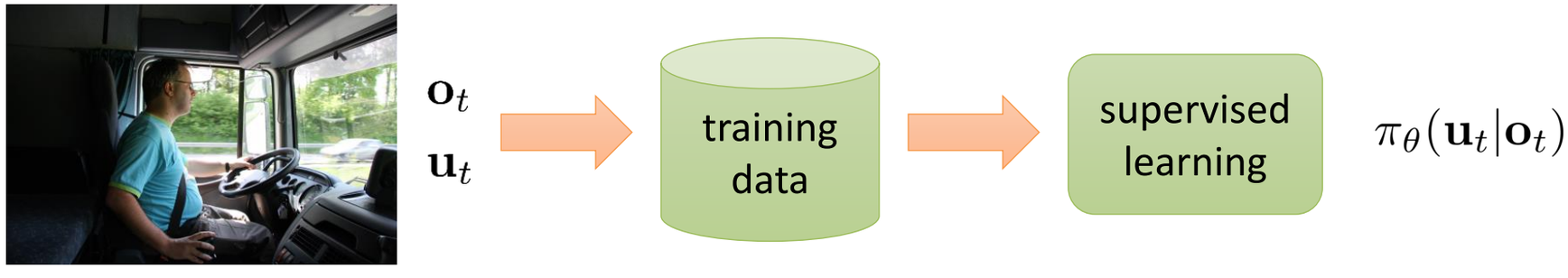
Imitation Learning as Supervised Learning

Driving policy: a mapping from observations to steering wheel angles



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

- Assume actions in the expert trajectories are i.i.d.
- Train a function approximator to map observations to actions at each time step of the trajectory.

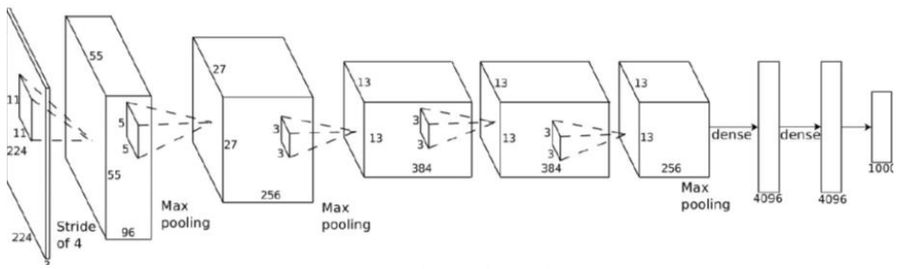


End to End Learning for Self-Driving Cars, Bojarski et al. 2016

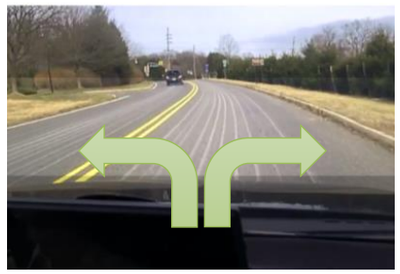
What can go wrong?

- Compounding errors
  Fix: data augmentation
- Stochastic expert actions
  Fix: stochastic latent variable models, action discretiation, gaussian mixture networks
- Non-markovian observations
  Fix: observation concatenation or recurrent models

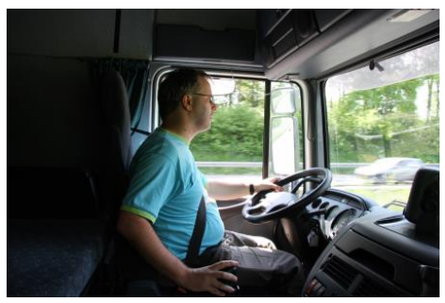
$\mathbf{o}_t$



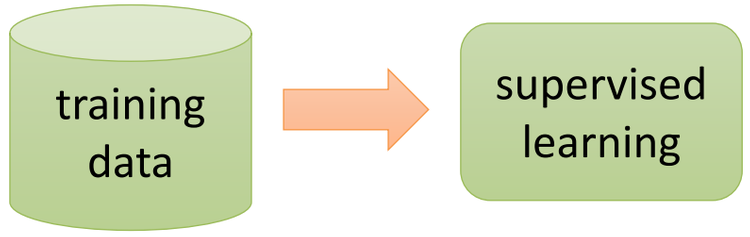
$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$



$\mathbf{u}_t$



$\mathbf{o}_t$
$\mathbf{u}_t$



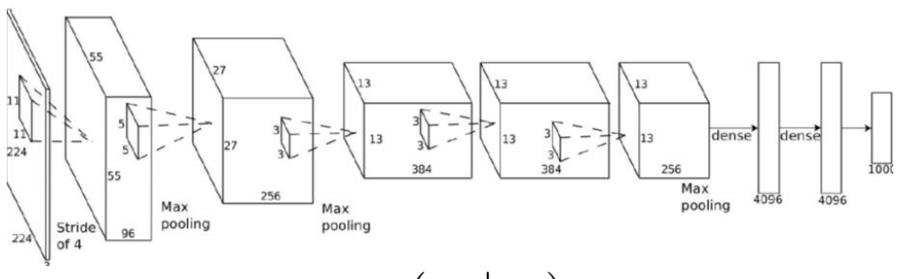
$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$

End to End Learning for Self-Driving Cars, Bojarski et al. 2016

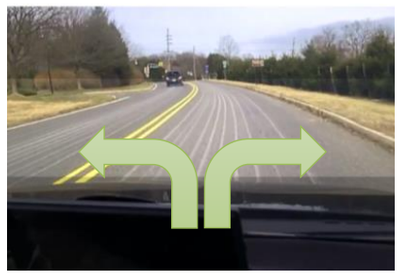
What can go wrong?

- Compounding errors
  Fix: data augmentation
- Stochastic expert actions
  Fix: stochastic latent variable models, action discretiation, gaussian mixture networks
- Non-markovian observations
  Fix: observation concatenation or recurrent models

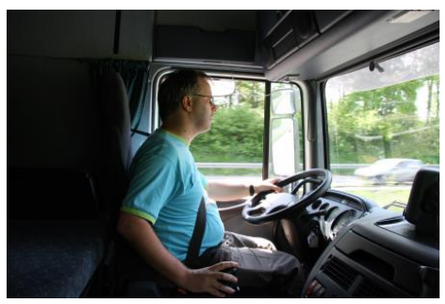
$\mathbf{o}_t$



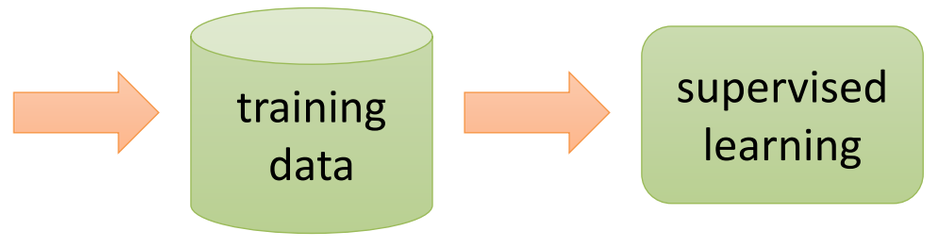
$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$



$\mathbf{u}_t$



$\mathbf{o}_t$
$\mathbf{u}_t$

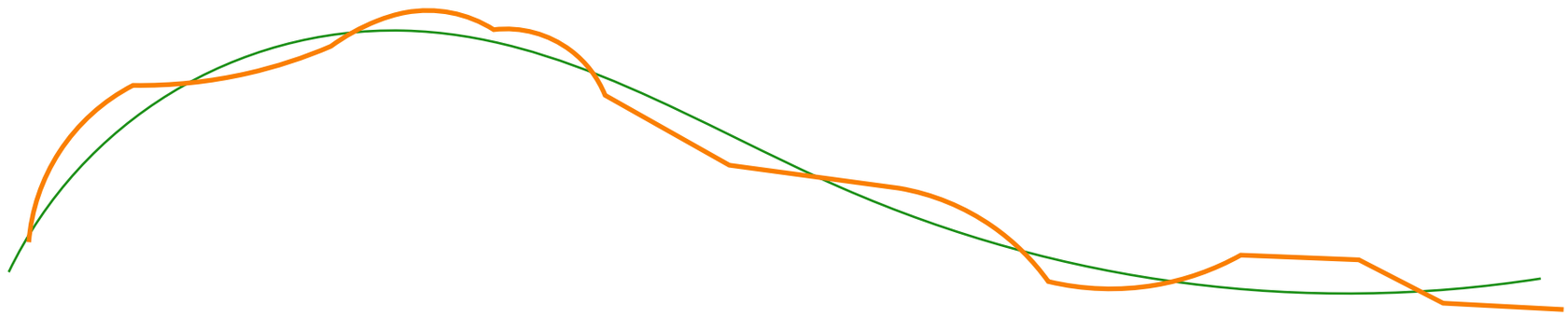


$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$

End to End Learning for Self-Driving Cars, Bojarski et al. 2016

Independent in time errors

This means that at each time step t, the agent wakes up on a state drawn from the data distribution of the expert trajectories, and executes an action

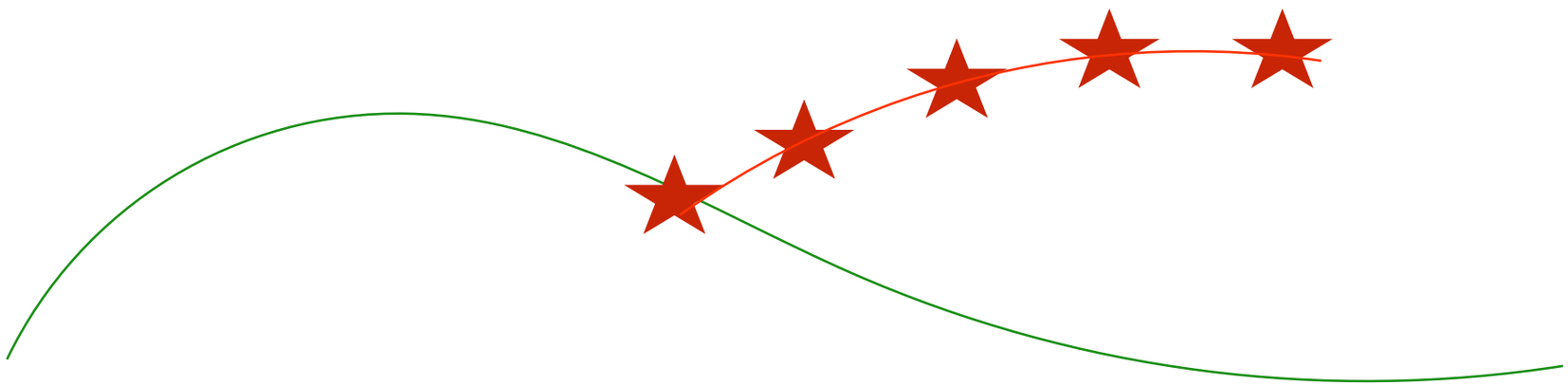


error at time t with probability ε

E[Total errors] ≲ εT

Compounding Errors

This means that at each time step t, the agent wakes up on the state that resulted from executing the action the learned policy suggested in the previous time step.



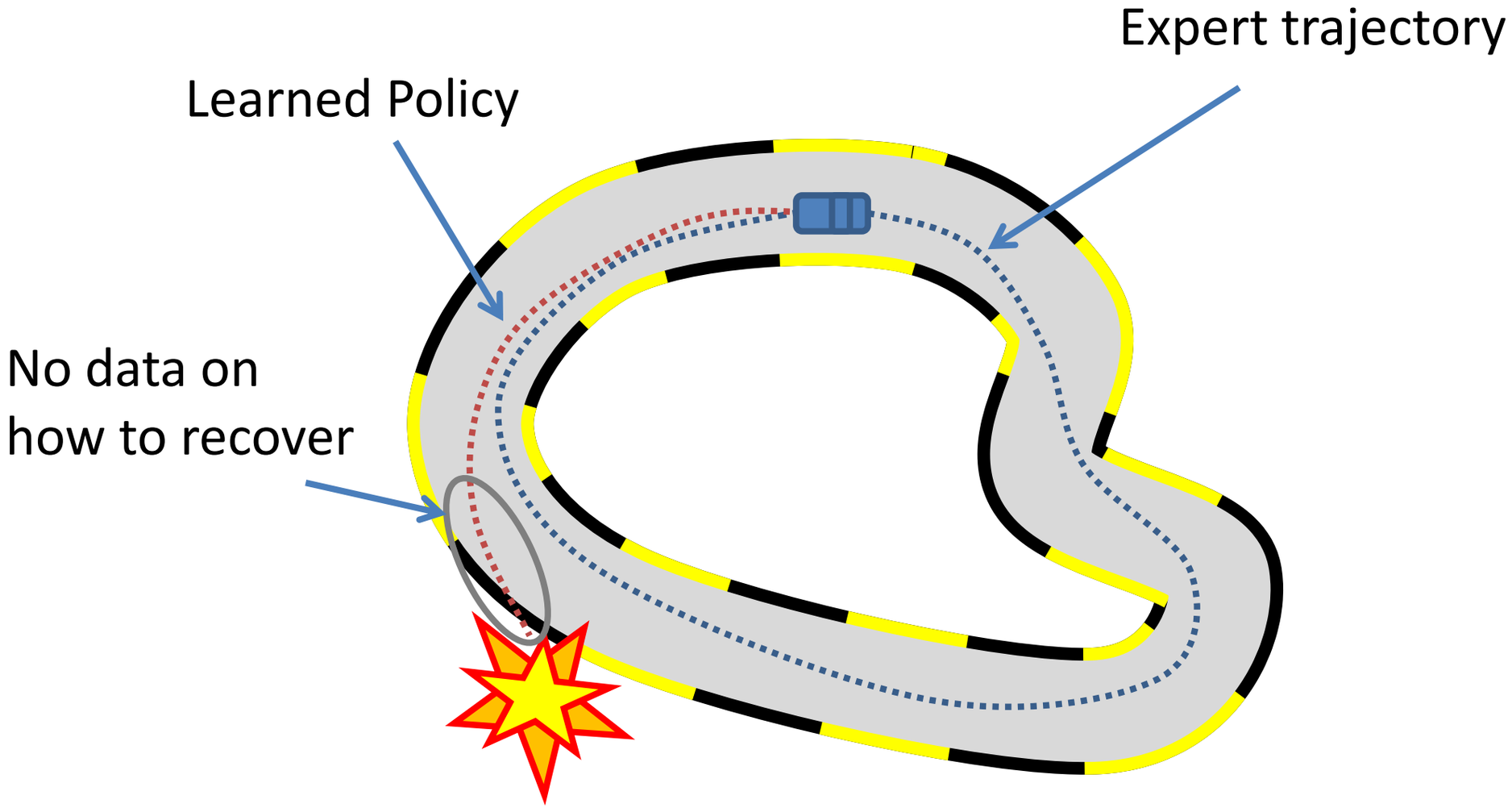
error at time t with probability ε

$E[\text{Total errors}] \lesssim \varepsilon(T + (T-1) + (T-2) + \ldots + 1) \propto \varepsilon T^2$

A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning, Ross et al. 2011

Data Distribution Mismatch!

$$p_{\pi^*}(o_t) \neq p_{\pi_\theta}(o_t)$$



Data Distribution Mismatch!

	supervised learning	supervised learning + control (NAIVE)
train	$(x,y) \sim D$	$s \sim d_{\pi^*}$
test	$(x,y) \sim D$	$s \sim d_{\pi}$

SL succeeds when training and test data distributions match, that is a fundamental assumption.

Solution: data augmentations

Change $p_{\pi^*}(o_t)$ using demonstration augmentation!! Add examples in expert demonstration trajectories to cover the states/observations points where the agent will land when trying out its own policy. How?

- Synthetically in simulation or by clever hardware
- Interactively with experts in the loop (DAGGER)

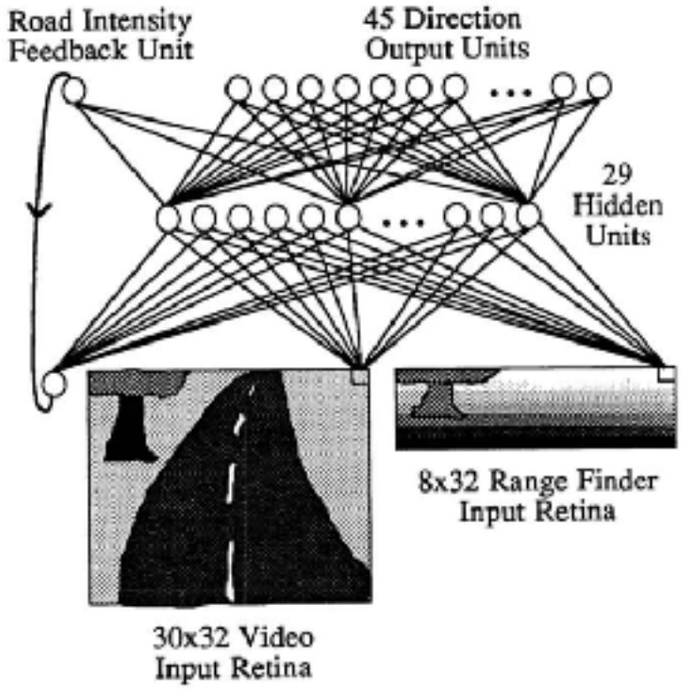
Solution: data augmentations

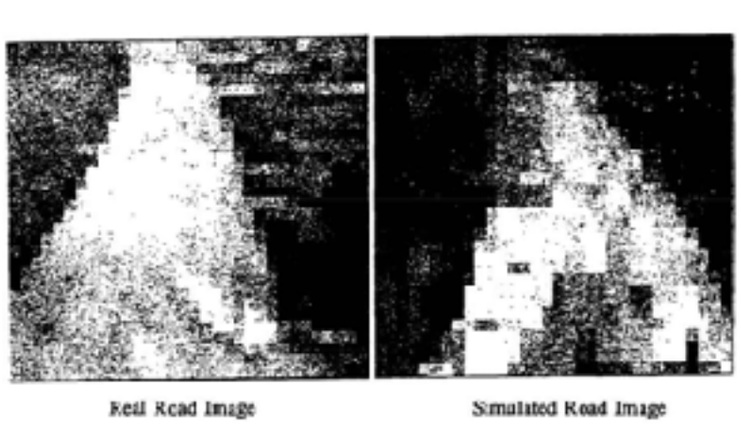
Change the training data distribution $p_{\pi^*}(o_t)$ using demonstration augmentation: add examples in expert demonstration trajectories to cover the states/observations where the agent will land when trying out its own policy.

	supervised learning	supervised learning + control (NAIVE)
train	(x,y) ~ D	s ~ d_{π^*}
test	(x,y) ~ D	s ~ d_{π}

Demonstration Augmentation: ALVINN 1989

Road follower

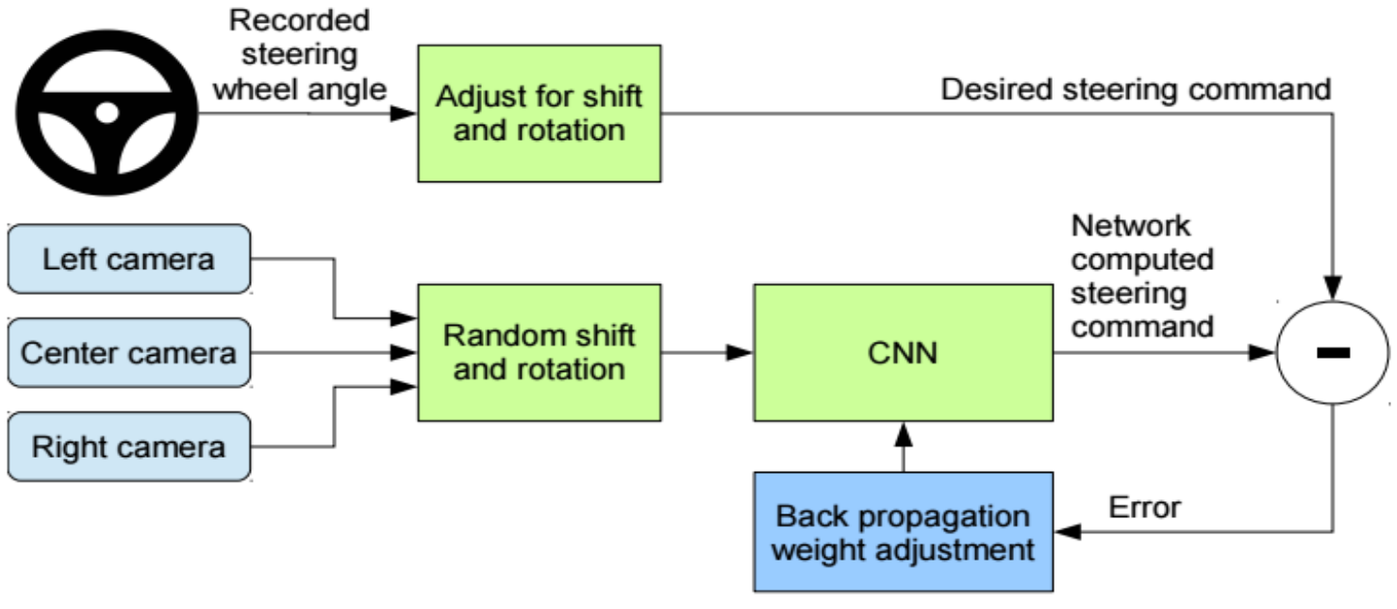




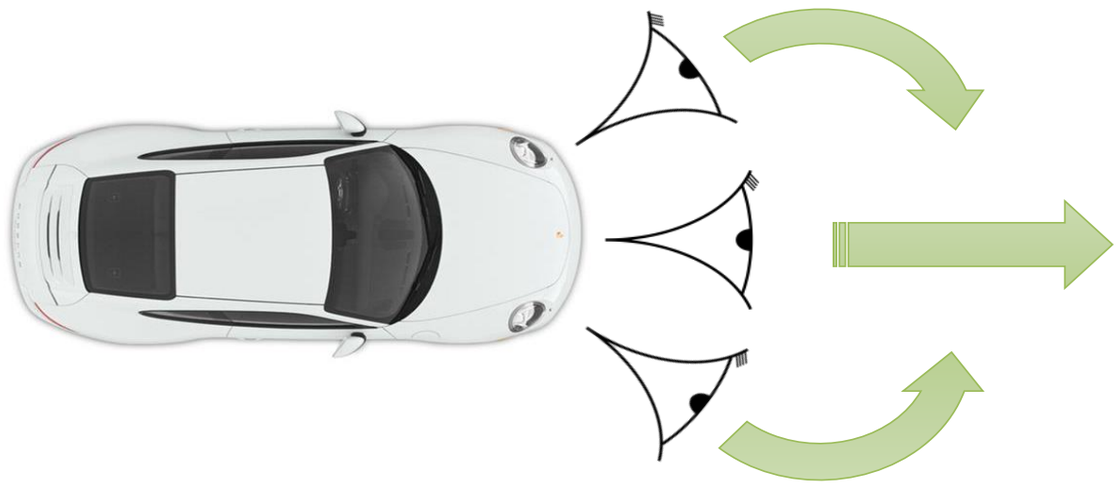
- Using graphics simulator for road images and corresponding steering angle ground-truth
- Online adaptation to human driver steering angle control
- 3 layers, fully connected layers, very low resolution input from camera

> *"In addition, the network must not solely be shown examples of accurate driving, but also how to recover (i.e. return to the road center) once a mistake has been made. Partial initial training on a variety of simulated road images should help eliminate these difficulties and facilitate better performance."*
> *ALVINN: An autonomous Land vehicle in a neural Network", Pomerleau 1989*

Demonstration Augmentation: NVIDIA 2016



Additional, left and right cameras with automatic grant-truth labels to recover from mistakes



“DAVE-2 was inspired by the pioneering work of Pomerleau [6] who in 1989 built the Autonomous Land Vehicle in a Neural Network (ALVINN) system. Training with data from only the human driver is not sufficient. The network must learn how to recover from mistakes. …”
End to End Learning for Self-Driving Cars , Bojarski et al. 2016

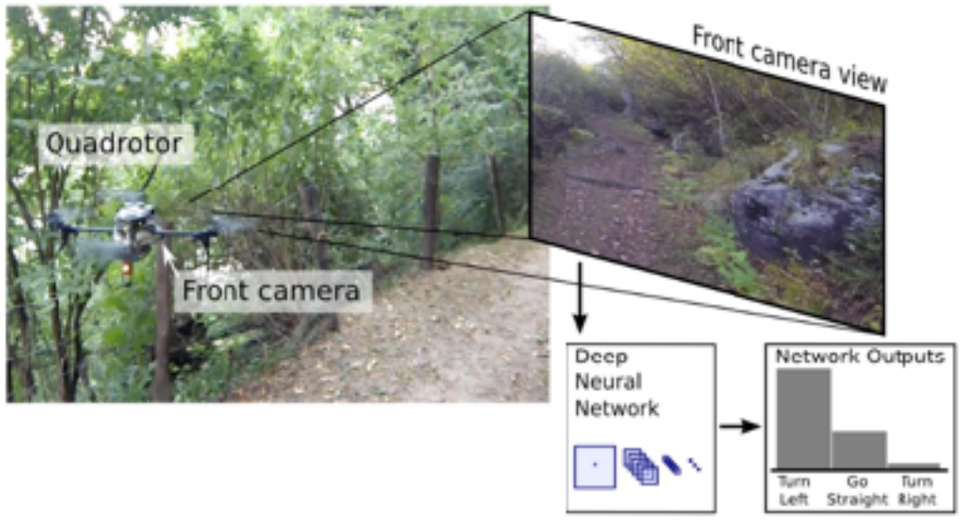
Data Augmentation (2): NVIDIA 2016

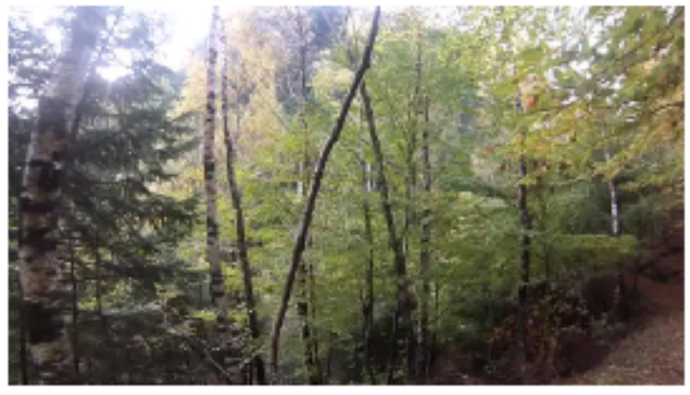
DAVE 2 Driving a Lincoln

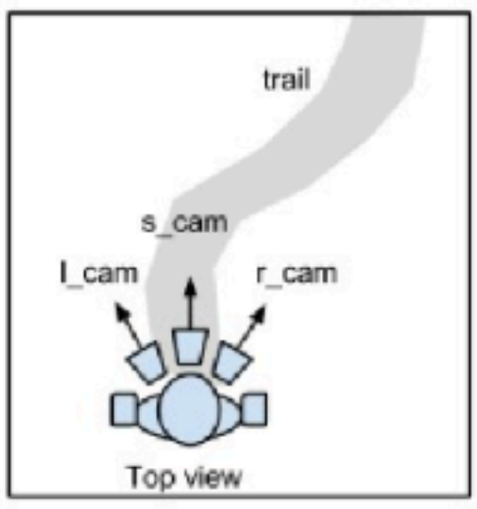
- A convolutional neural network
- Trained by human drivers
- Learns perception, path planning, and control
  "pixel in, action out"
- Front-facing camera is the only sensor

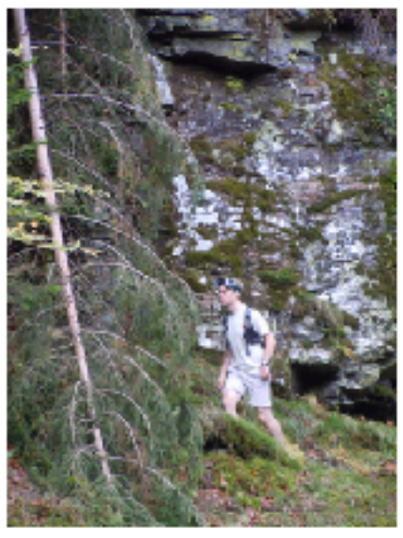
“DAVE-2 was inspired by the pioneering work of Pomerleau [6] who in 1989 built the Autonomous Land Vehicle in a Neural Network (ALVINN) system. Training with data from only the human driver is not sufficient. The network must learn how to recover from mistakes. …”, End to End Learning for Self-Driving Cars , Bojarski et al. 2016

Data Augmentation (3): Trails 2015



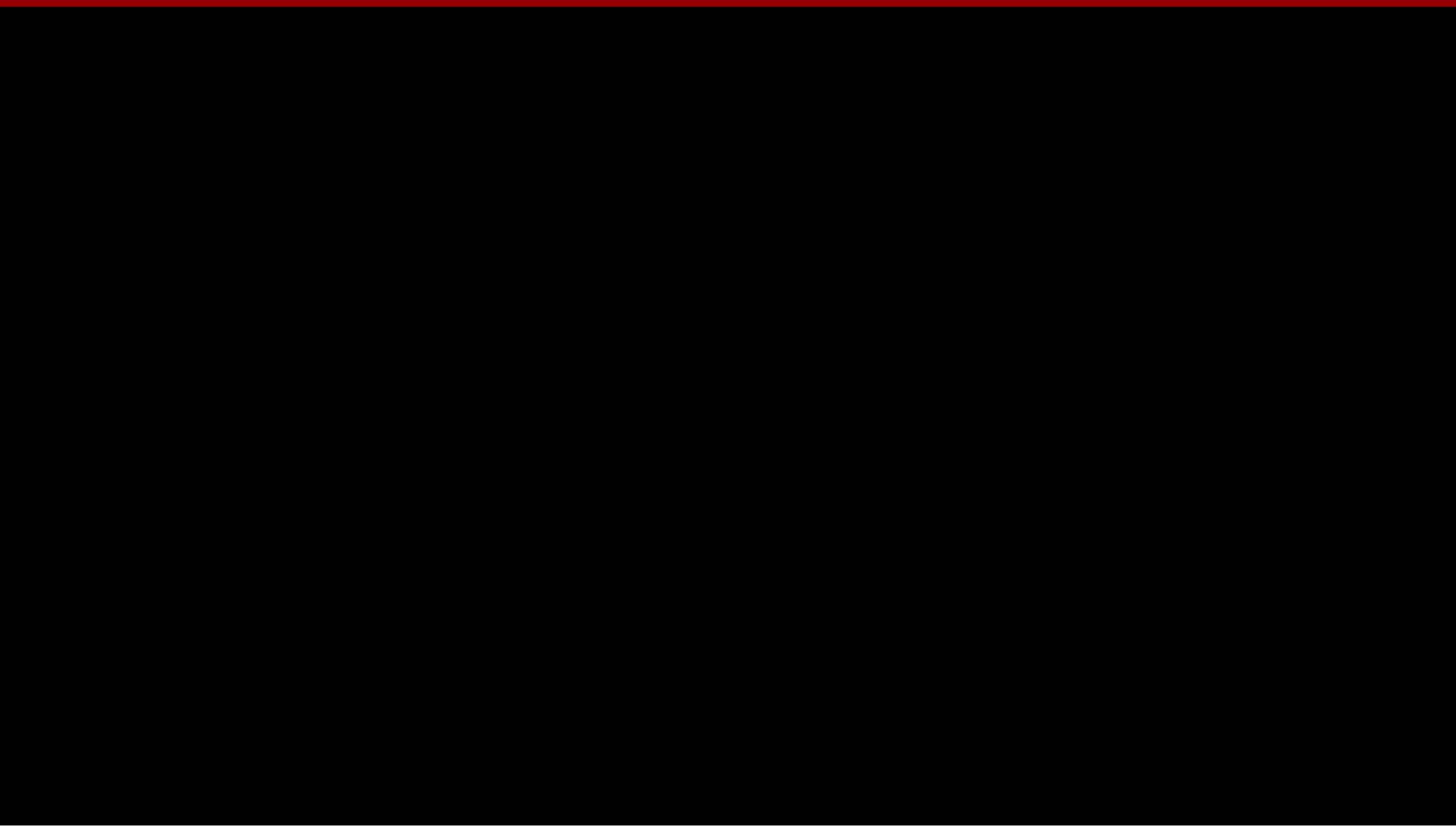






A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots Giusti et al.

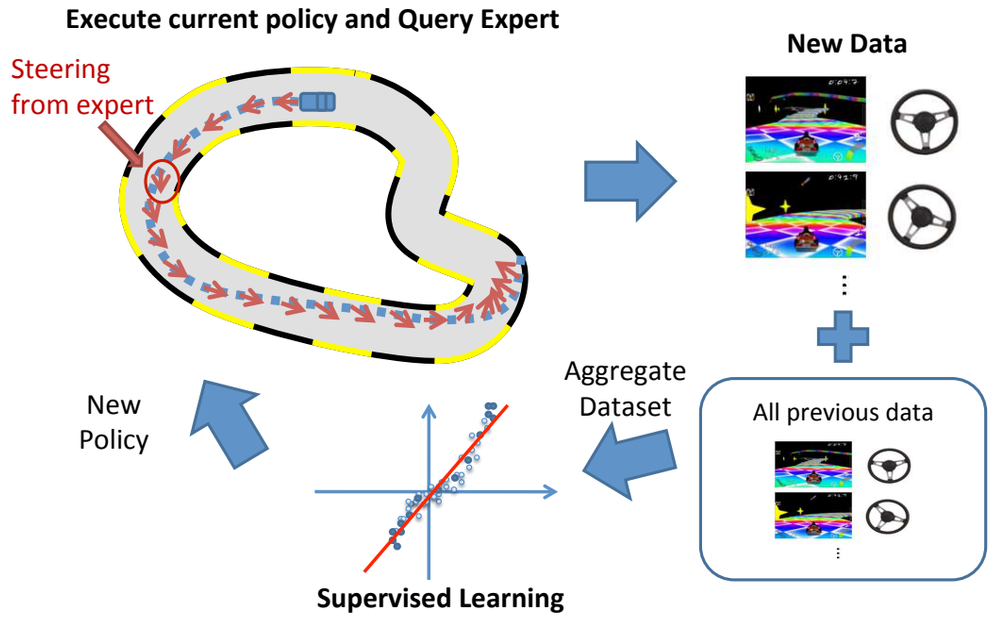
Data Augmentation (3): Trails 2015



A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots Giusti et al.

DAGGER (in simulation)

Dataset AGGregation: bring learner's and expert's trajectory distributions closer by (asking uman experts to provide) labelling additional data points resulting from applying the current policy



A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning, Ross et al. 2011

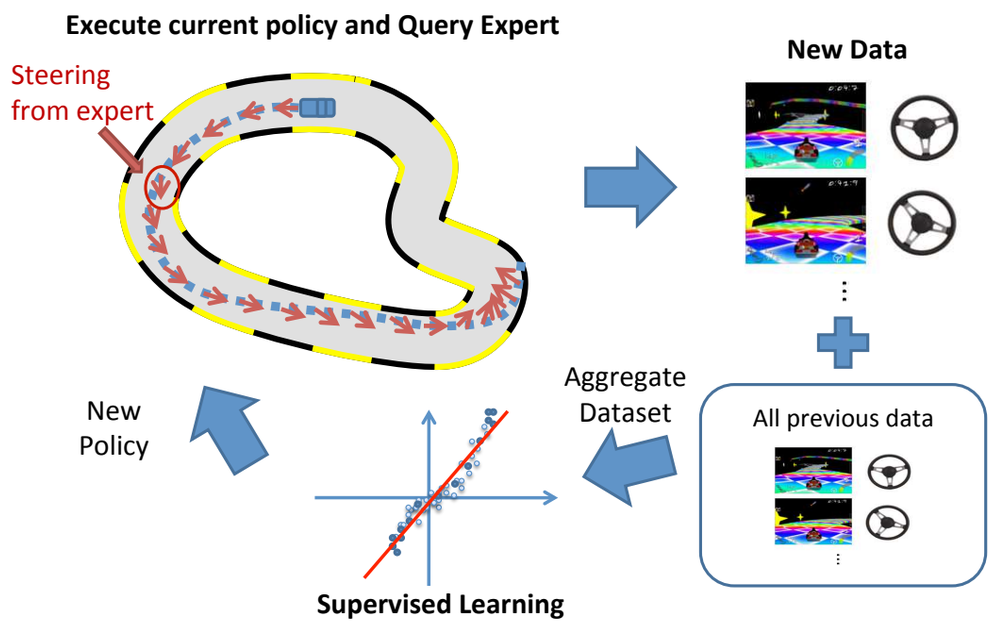
DAGGER (in simulation)

Dataset AGGregation: bring learner's and expert's trajectory distributions closer by (asking uman experts to provide) labelling additional data points resulting from applying the current policy

1. train $\pi_\theta(u_t|o_t)$ from human data $\mathcal{D}_{\pi^*} = \{o_1, u_1, ..., o_N, u_N\}$
2. run $\pi_\theta(u_t|o_t)$ to get dataset $\mathcal{D}_\pi = \{o_1, ..., o_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions u_t
4. Aggregate: $\mathcal{D}_{\pi^*} \leftarrow \mathcal{D}_{\pi^*} \cup \mathcal{D}_\pi$
5. GOTO step 1.

Problems:

- execute an unsafe/partially trained policy
- repeatedly query the expert



A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning, Ross et al. 2011

DAGGER (on a real platform)

Application on drones: given RGB from the drone camera predict steering angles

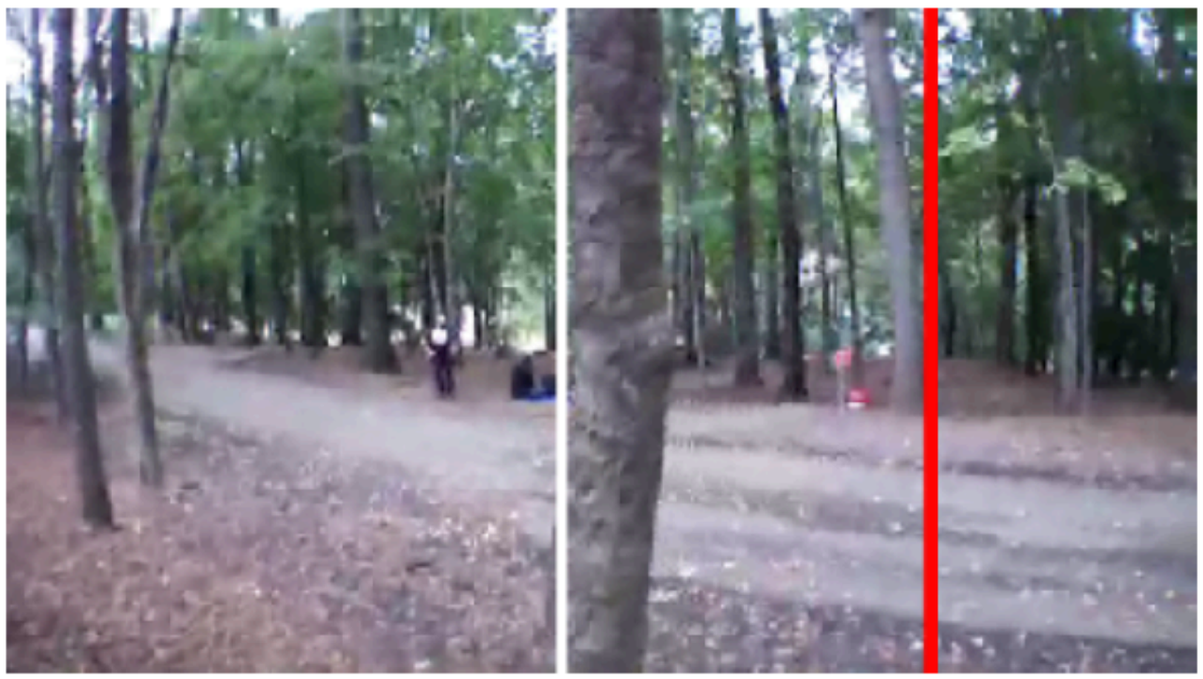
Learning monocular reactive UAV control in cluttered natural environments, Ross et al. 2013

DAGGER (on a real platform)

Application on drones : given RGB from the drone camera predict steering angle

Caveats:

1. It is hard for the expert to provide the right magnitude for the turn without feedback of his own actions! Solution: provide him with visual feedback



Learning monocular reactive UAV control in cluttered natural environments, Ross et al. 2013

DAGGER (on a real platform)

Caveats:

1. Is hard for the expert to provide the right magnitude for the turn without feedback of his own actions! Solution: provide him with his visual feedback

2. The expert's reaction time to the drone's behavior is large, this causes imperfect actions to be commanded. Solution: play-back in slow motion offline and record their actions.

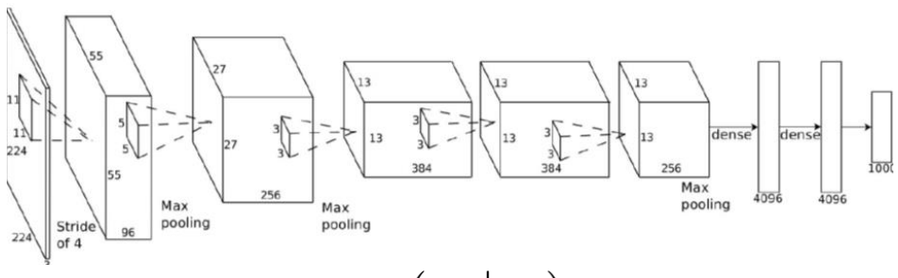
3. Executing an imperfect policy causes accidents, crashes into obstacles. Solution: safety measures which make again the data distribution matching imperfect between train and test, but good enough..

Learning monocular reactive UAV control in cluttered natural environments, Ross et al. 2013

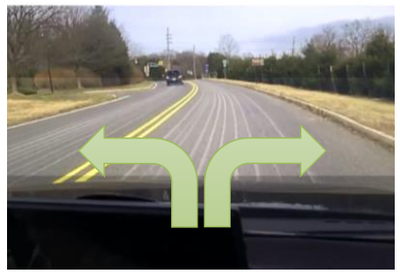
What can go wrong?

- Compounding errors
  Fix: data augmentation
- Stochastic expert actions
  Fix: stochastic latent variable models, action discretiation, gaussian mixture networks
- Non-markovian observations
  Fix: observation concatenation or recurrent models

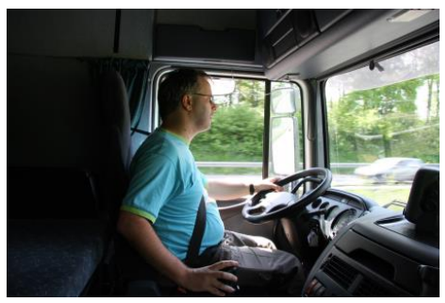
$\mathbf{o}_t$



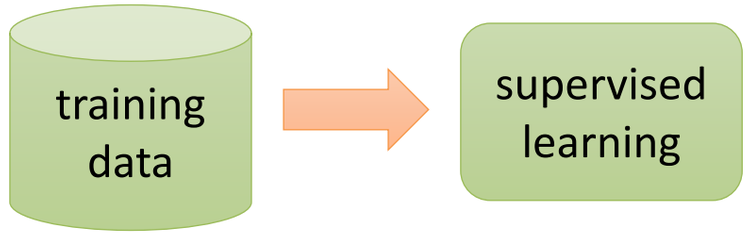
$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$



$\mathbf{u}_t$



$\mathbf{o}_t$
$\mathbf{u}_t$

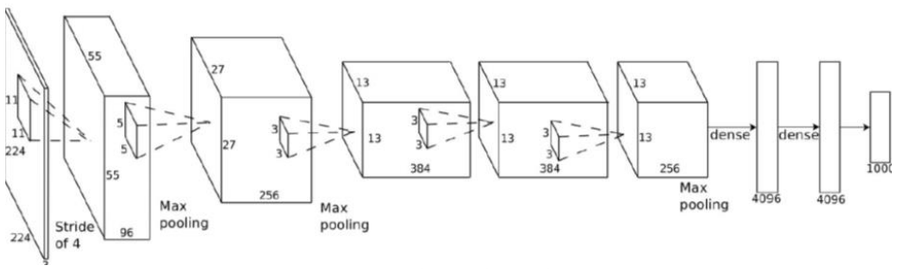


$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$

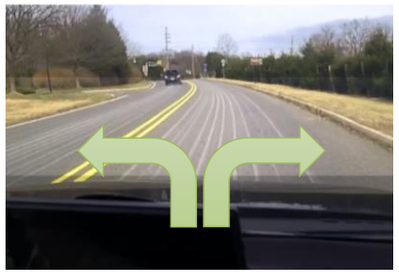
End to End Learning for Self-Driving Cars, Bojarski et al. 2016

Non-markovian observations

$\mathbf{o}_t$



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



$\mathbf{u}_t$

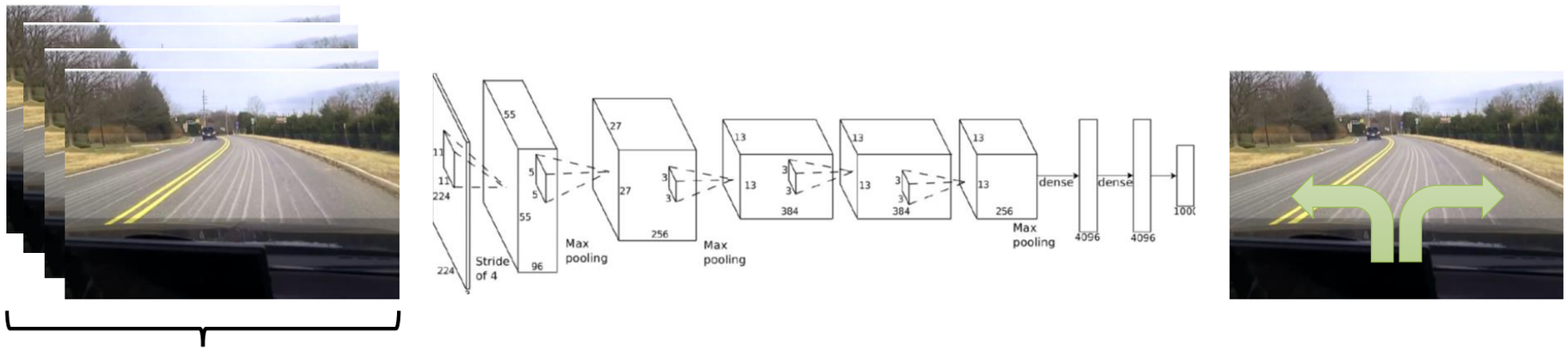
$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$

behavior depends only
on current observation

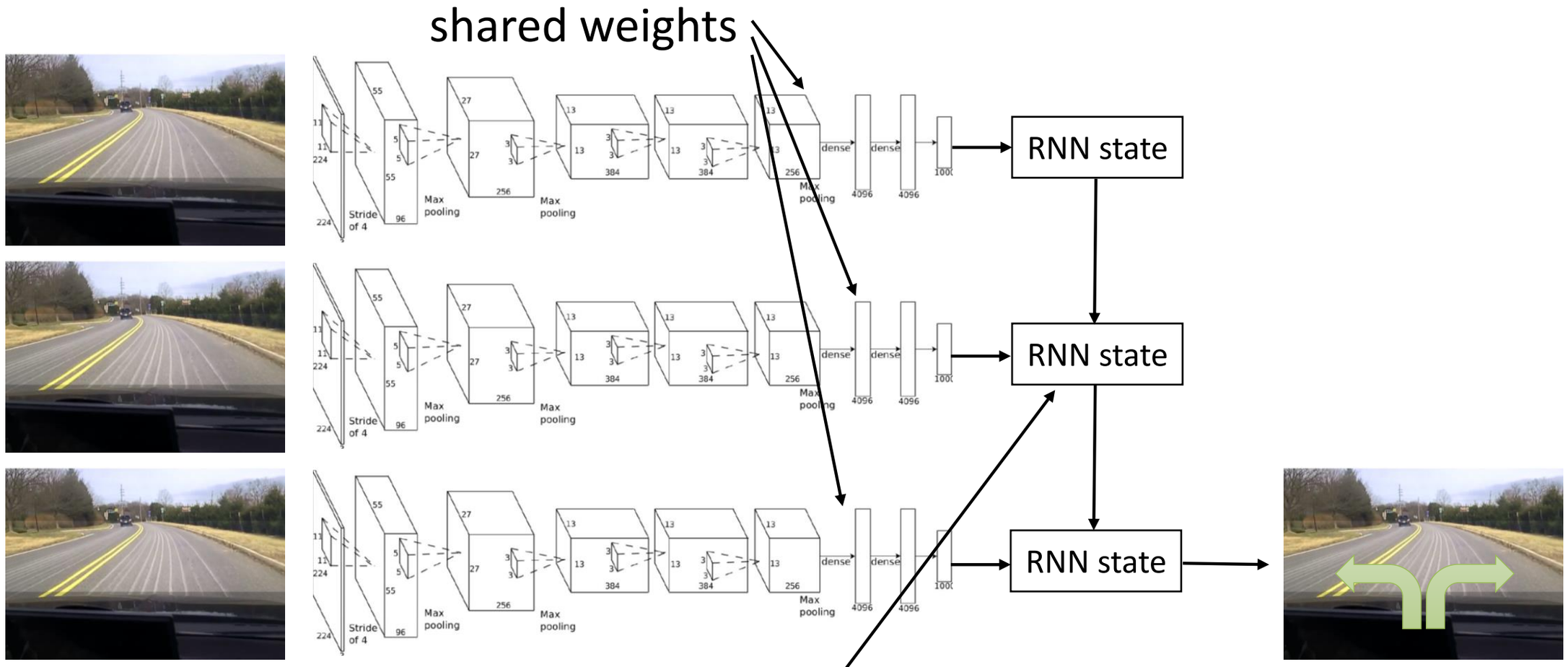
$\pi_\theta(\mathbf{u}_t|\mathbf{o}_1, \ldots, \mathbf{o}_t)$

behavior depends on
all past observations

Fix 1: concatenate observations



Fix 2: use recurrent networks



Typically, LSTM cells work better here

Diagram from Sergey Levine

Recurrent Neural Networks (RNNs)

- RNNs tie the weights at each time step
- Condition the neural network on all previous inputs
- In principle, any interdependencies can be modeled across time steps.
- In practice, limitations from SGD training, capacity, initialization etc.

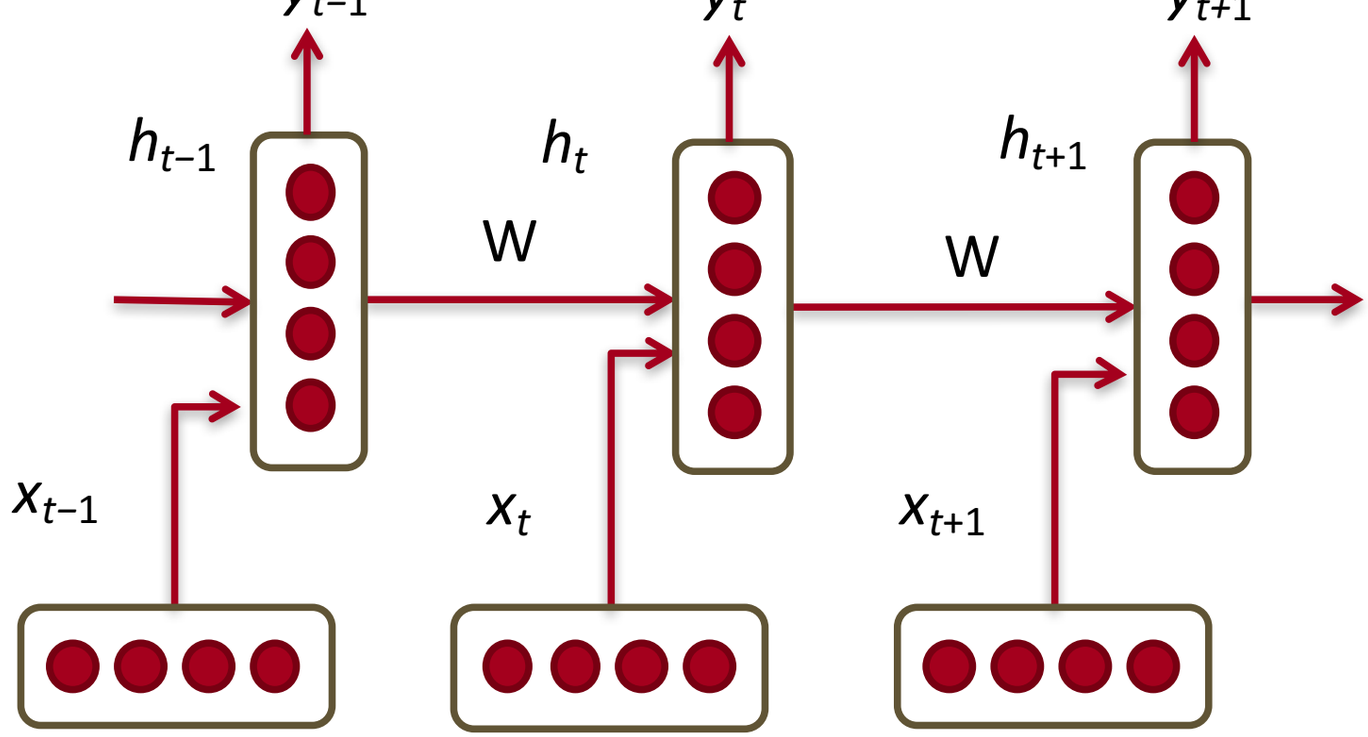


Diagram from Richard Socher

Recurrent Neural Network (single hidden layer)

- Given list of **vectors**: $x_1, ..., x_{t-1}, x_t, x_{t+1}, ..., x_T$
- At a single time step:

$$h_t = \sigma\left(W^{(hh)} h_{t-1} + W^{(hx)} x_{[t]}\right)$$

$$\hat{y}_t = \text{softmax}\left(W^{(S)} h_t\right)$$

(in case of discrete labels)

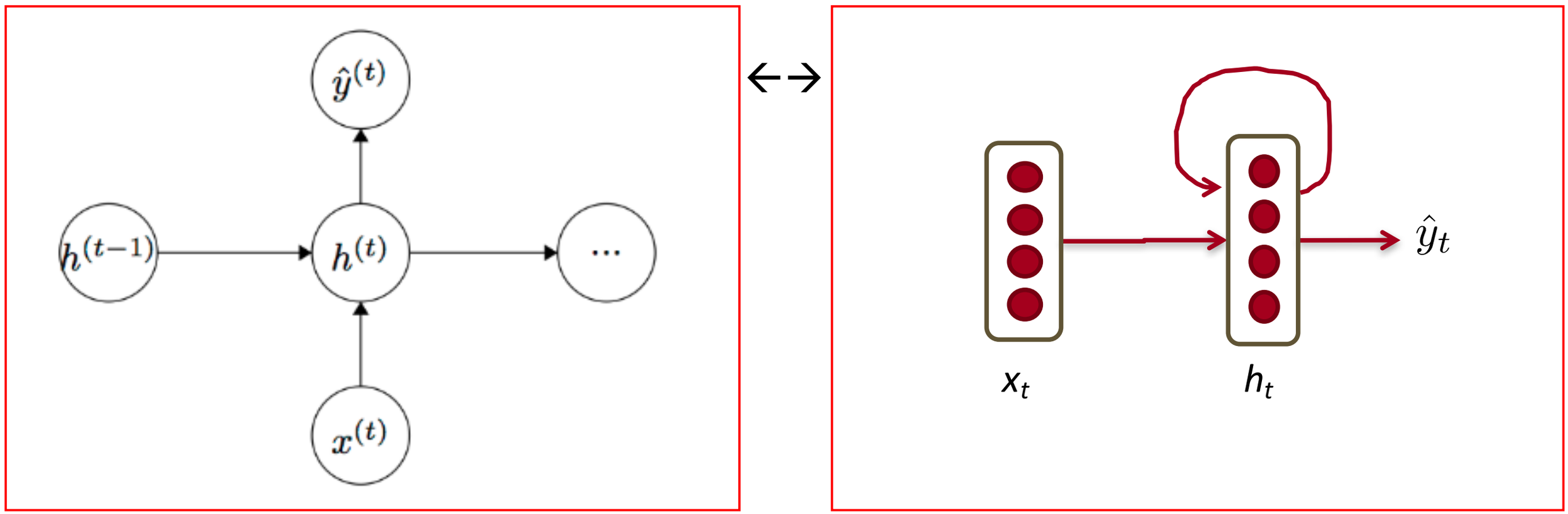
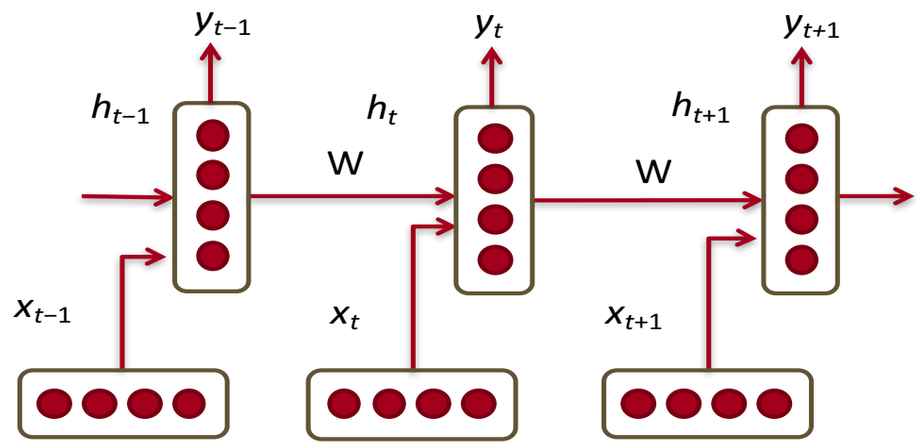


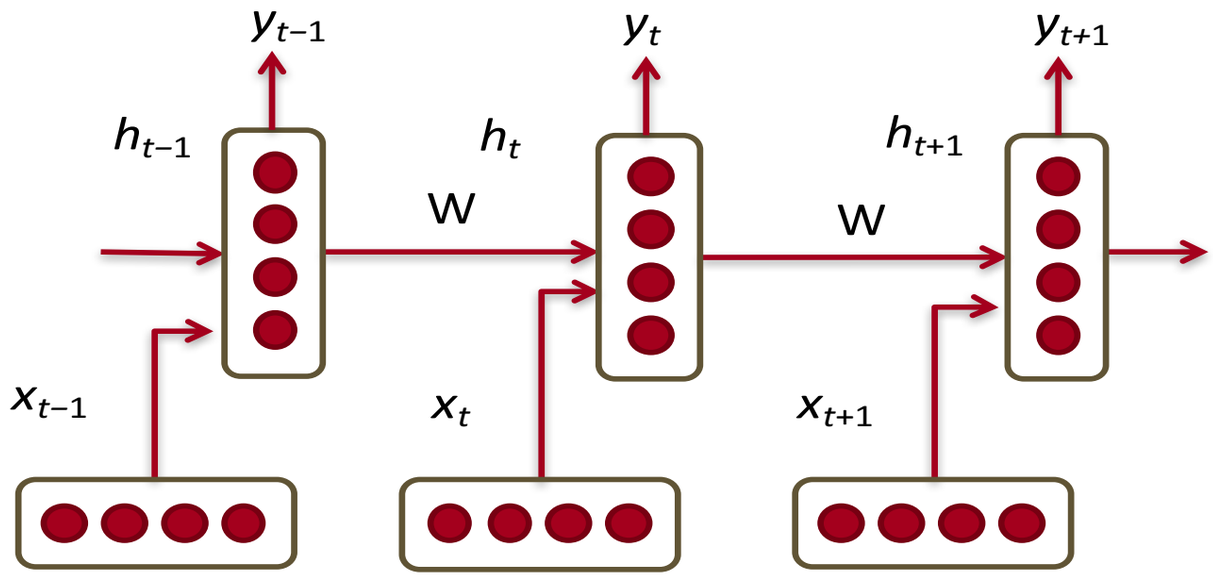
Diagram from Richard Socher

Recurrent Neural Networks

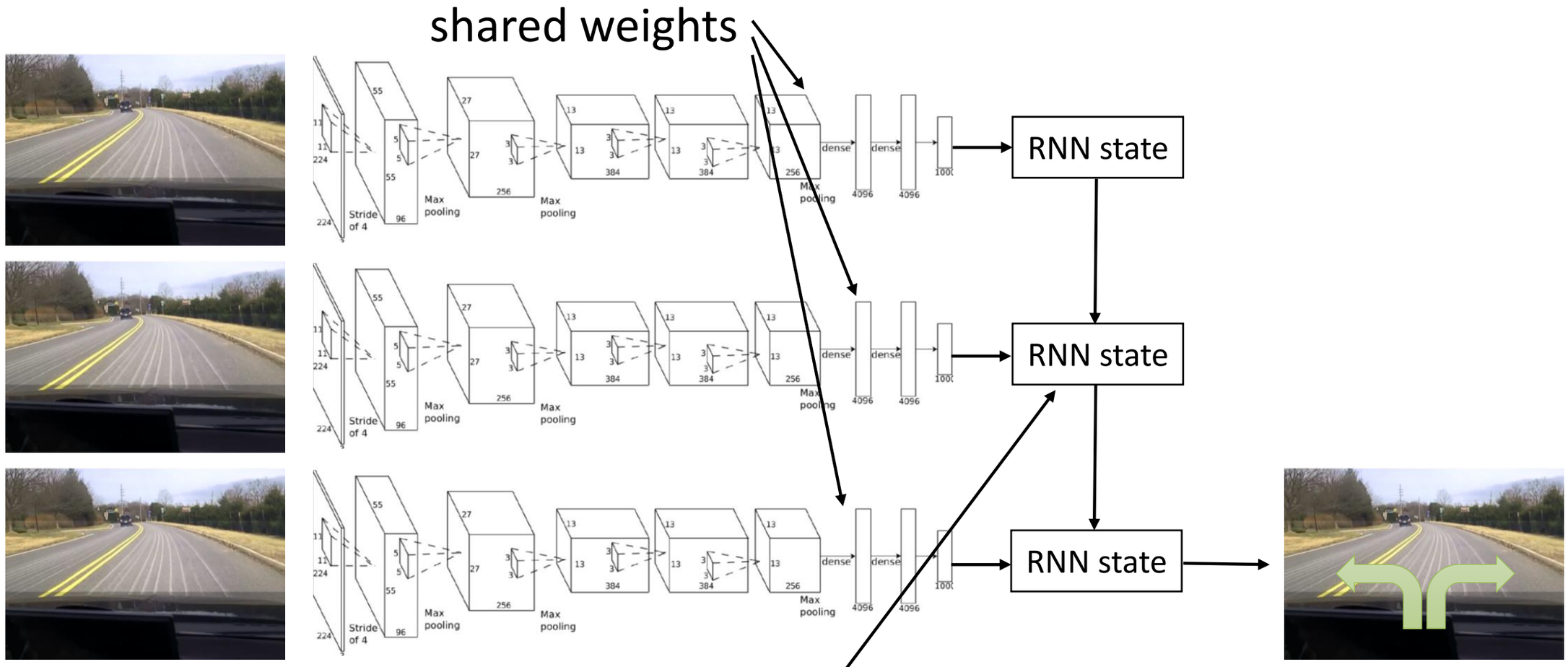
For sequence labelling problems, actions of the labelling policies are y_t, e.g., part of speech tags



For sequence generation, actions of the labelling policies are $y_t = x_{t+1}$, e.g., word in answer generation $\hat{P}(x_{t+1} = v_j | x_t, ..., x_1) = \hat{y}_{t,j}$



Fix 2: use recurrent networks

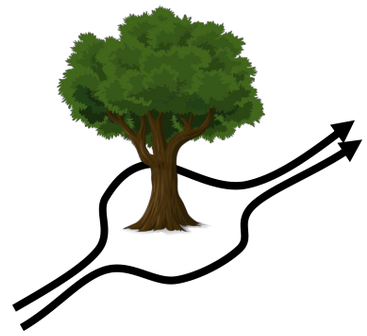


Typically, LSTM cells work better here

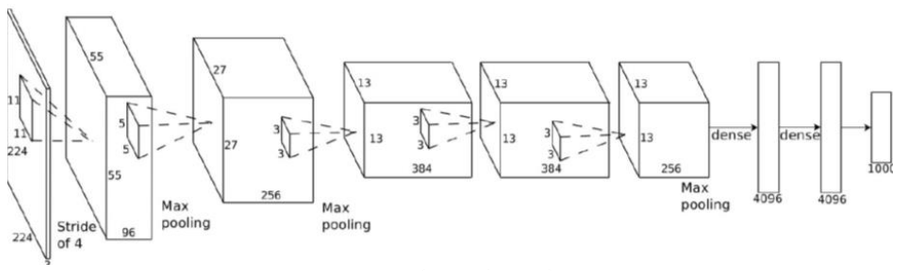
Diagram rom Sergey Levine

What can go wrong?

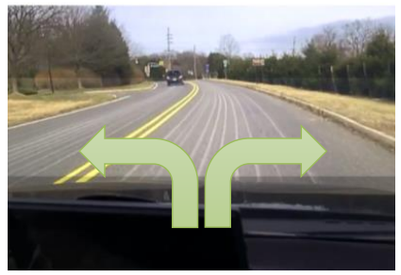
- Compounding errors
  - Fix: data augmentation
- Stochastic expert actions
  - Fix: stochastic latent variable models, action discretiation, gaussian mixture networks
- Non-markovian observations
  - Fix: observation concatenation or recurrent models



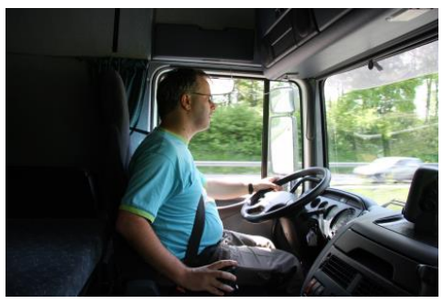
$\mathbf{o}_t$



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

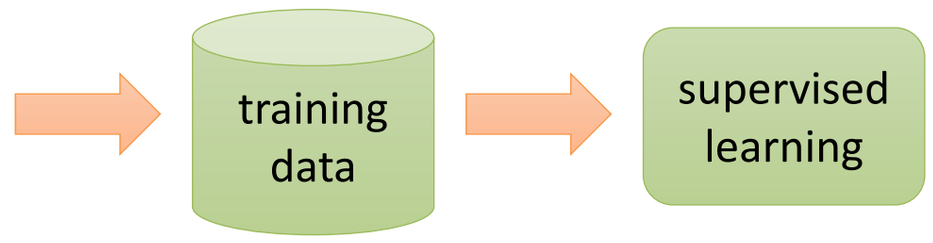


$\mathbf{u}_t$



$\mathbf{o}_t$

$\mathbf{u}_t$

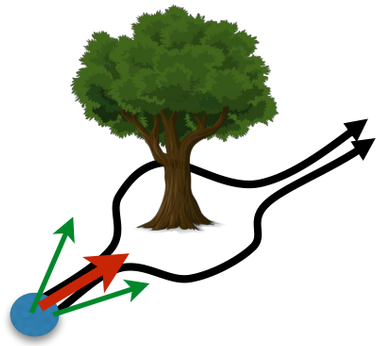


$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$

End to End Learning for Self-Driving Cars, Bojarski et al. 2016

Regression fails under multimodality

The answer that minimizes the mean square error is the average which is not a valid prediction



groundtruth streering angles

predicted streering angles

Stochastic expert actions: Fixes

- Discretize the action space and use a classifier (e.g., softmax output and cross-entropy loss)
- Use gaussian mixture model as an output layer, mixture components weights, means and variances are parametrized at the output of a neural net, minimize GMM loss, (e.g., Handwriting generation Graves 2013)
- Stochastic neural networks (later lecture)

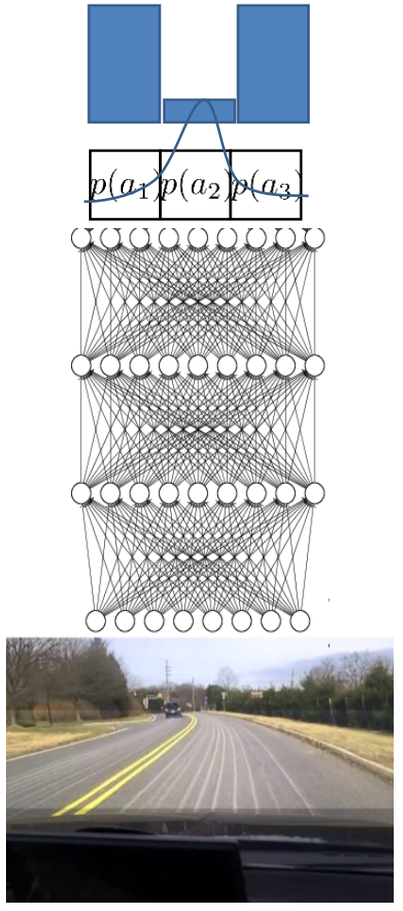
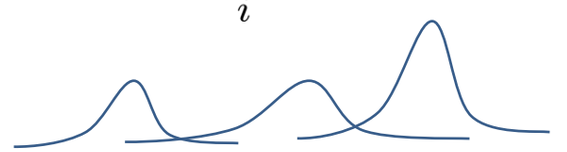


Diagram from Sergey Levine

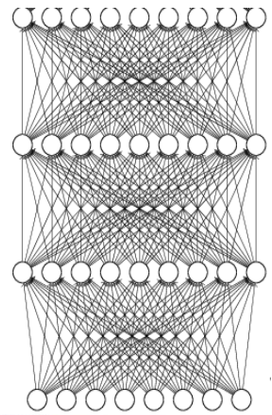
Stochastic expert actions: Fixes

- Discretize the action space and use a classifier (e.g., softmax output and cross-entropy loss)
- Use gaussian mixture model as an output layer, mixture components weights, means and variances are parametrized at the output of a neural net, minimize GMM loss, (e.g., Handwriting generation Graves 2013)
- Stochastic neural networks (later lecture)

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



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



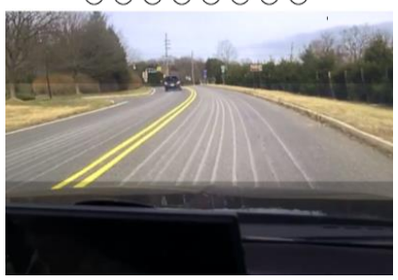


Diagram from Sergey Levine

Stochastic expert actions: Fixes

- Discretize the action space and use a classifier (e.g., softmax output and cross-entropy loss)
- Use gaussian mixture model as an output layer, mixture components weights, means and variances are parametrized at the output of a neural net, minimize GMM loss, (e.g., Handwriting generation Graves 2013)
- Stochastic neural networks (later lecture)

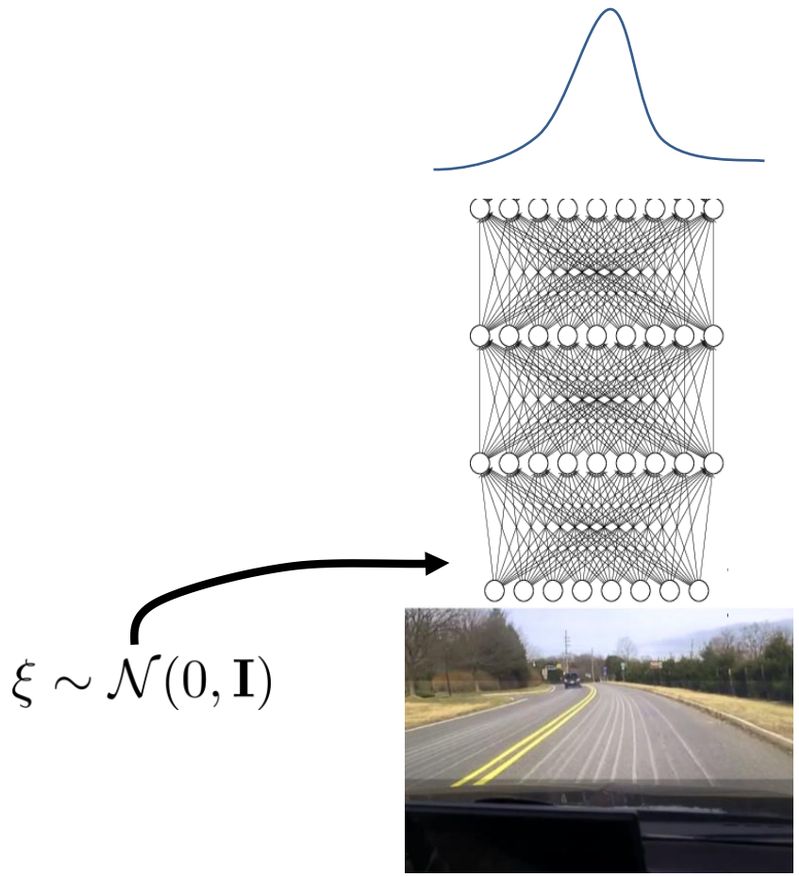


Diagram from Sergey Levine

Structured prediction

Structured prediction: a learner makes predictions over a set of interdependent output variables and observes a joint loss.

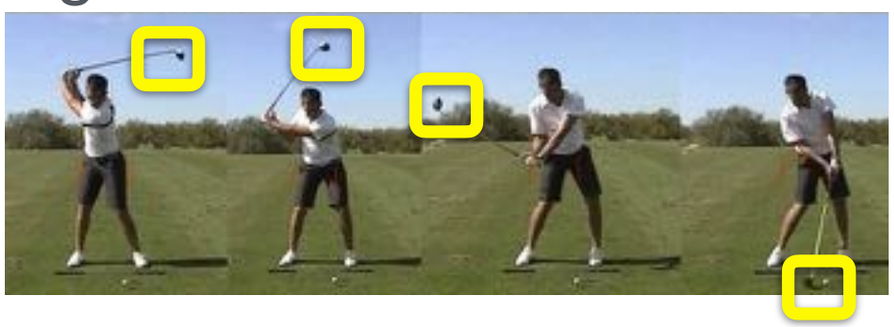
part-of-speech tagging

```
x = the monster ate the sandwich
y = Dt    Nn    Vb  Dt      Nn
```

NER (Name Entity Recognition)

```
x = Yesterday I traveled to Lille
y =       -     PER    -        -   LOC
```

tracking



captioning

“A blue monster is eating a cookie”

Machine translation

Few images from Hall Daume III

Recurrent Neural Networks

The regular training procedure of RNNs treat true labels y_t as actions while making forward passes. Hence, the learning agent follows trajectories generated by the reference policy rather than the learned policy. In other words, it learns:

$$\theta^{sup} = \arg\min_{\theta} \mathbb{E}_{h \sim d_{\pi^*}} [l_\theta(h)]$$

However, our true goal is to learn a policy that minimizes error under its own induced state distribution:

$$\hat{\theta} = \arg\min_{\theta} \mathbb{E}_{h \sim d_\theta} [l_\theta(h)]$$

Imitation Learning with Recurrent Neural Networks, Nyuyen 2016

Mocap generation

$\mathbf{x}_{t+1}$

$\mathbf{x}_{t+2}$

decode1
lstm3
lstm2
lstm1
encode1

decode1
lstm3
lstm2
lstm1
encode1

noise

noise

$\mathbf{x}_t$

$\mathbf{x}_{t+1}$

DAGGER for sequence labelling/generation

Q: what we be feeding the groundtruth x,y or the predicted x,y during training?

1: **function** TRAIN(N, α)
2: Intialize $\alpha = 1$.
3: Initialize model parameters θ.
4: **for** $i = 1..N$ **do**
5: Set $\alpha = \alpha \cdot p$.
6: Randomize a batch of labeled examples.
7: **for** each example (x, y) in the batch **do**
8: Initialize $h_0 = \Phi(X)$.
9: Initialize $\mathcal{D} = \{(h_0, y_0)\}$.
10: **for** $t = 1 \ldots |Y|$ **do**
11: Uniformly randomize a floating-number $\beta \in [0, 1)$.
12: **if** $\alpha < \beta$ **then**
13: Use true label $\tilde{y}_{t-1} = y_{t-1}$
14: **else**
15: Use predicted label: $\tilde{y}_{t-1} = \arg\max_y P(y \mid h_{t-1}; \theta)$.
16: **end if**
17: Compute the next state: $h_t = f_\theta(h_{t-1}, \tilde{y}_{t-1})$.
18: Add example: $\mathcal{D} = \mathcal{D} \cup \{(h_t, y_t)\}$.
19: **end for**
20: **end for**
21: Online update θ by $\mathcal{D}$ (mini-batch back-propagation).
22: **end for**
23: **end function**

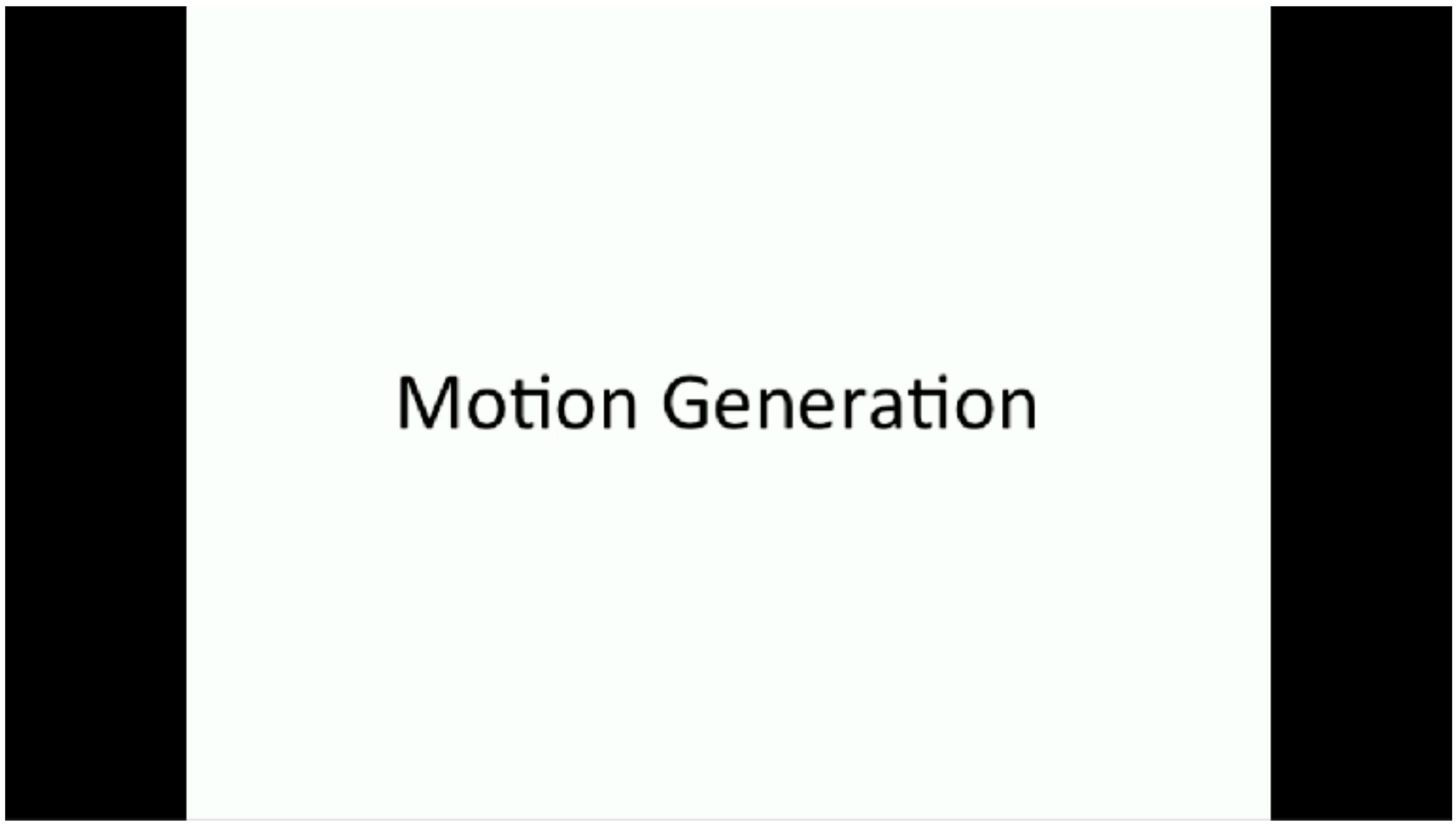
Teacher forcing

Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks, Bengio(Samy) et al.

Imitation Learning with Recurrent Neural Networks, Nyuyen

Mocap generation

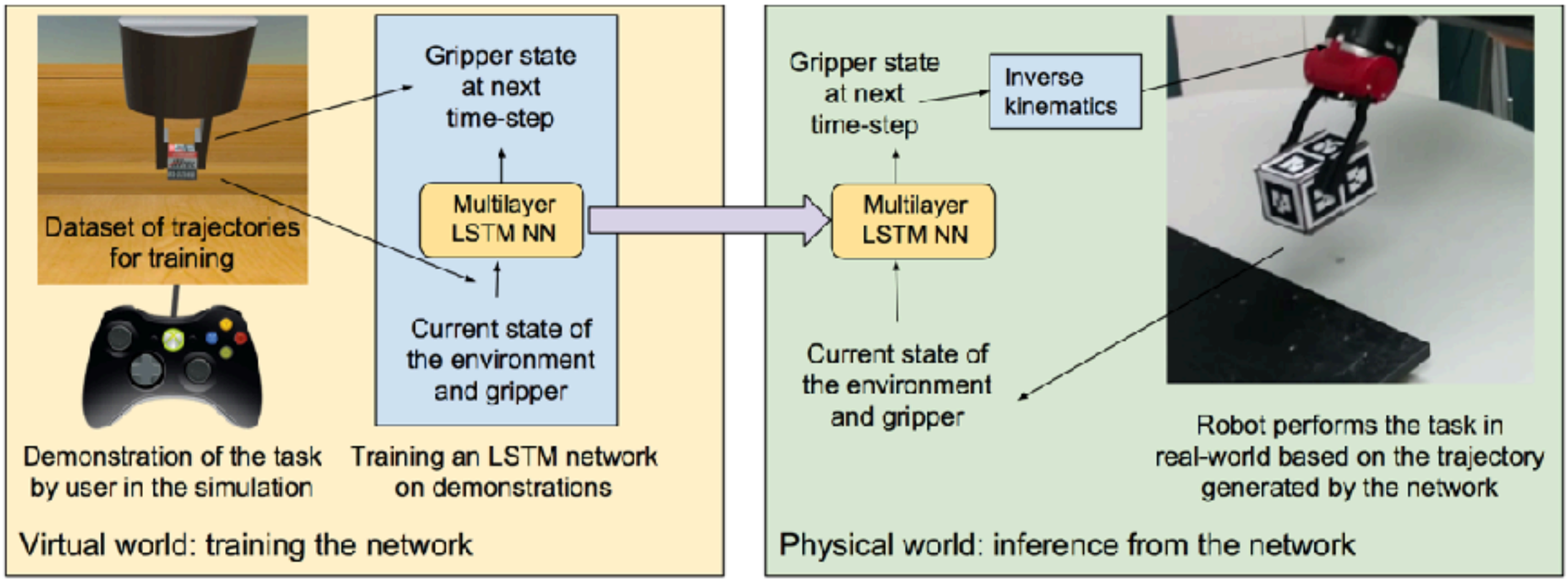
- Right: no augmentation, using only ground-truth state input
- Left: augmentation by adding Gaussian noise to the state (not to the prediction)



When adding noise to the input, despite the per frame prediction error being larger, the long term prediction error is lower.

Learning human dynamics with recurrent neural networks, Fragkiadaki et al.

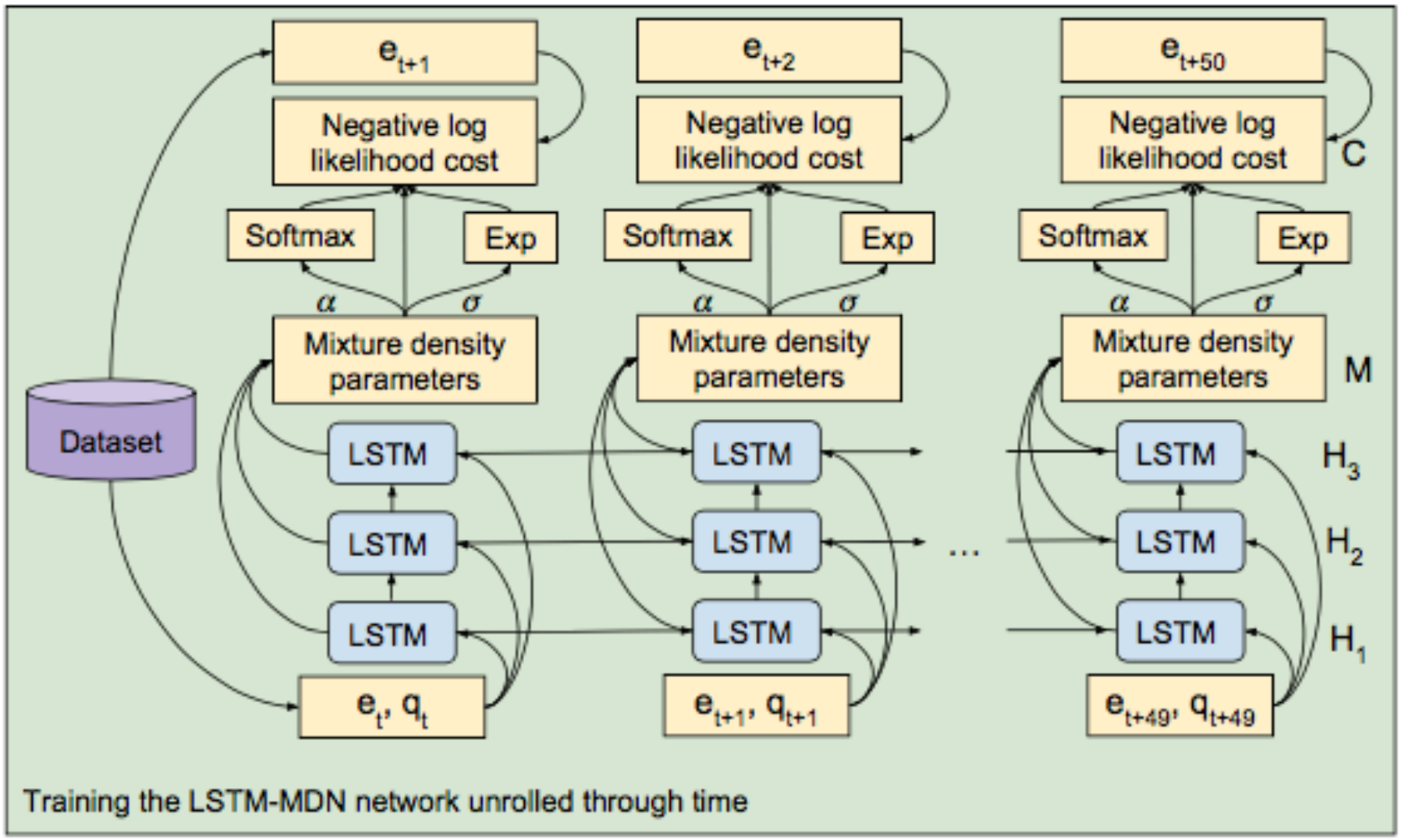
Case study: learning from virtual demonstrations



- Two tasks considered: pick and place, move to desired pose
- State representation x: the poses of all the objects in the seen (rotations, translations) and the pose of the end effector
- Output y: the desired next pose of the end effector
- Supervision: expert trajectories in the simulator
- Demonstation augmentation: consider multiple trajectories by subsampling in time the expert ones, and by translating in space the end effector

Learning real manipulation tasks from virtual demonstrations using LSTM, Rahmatizadeh et al 2016

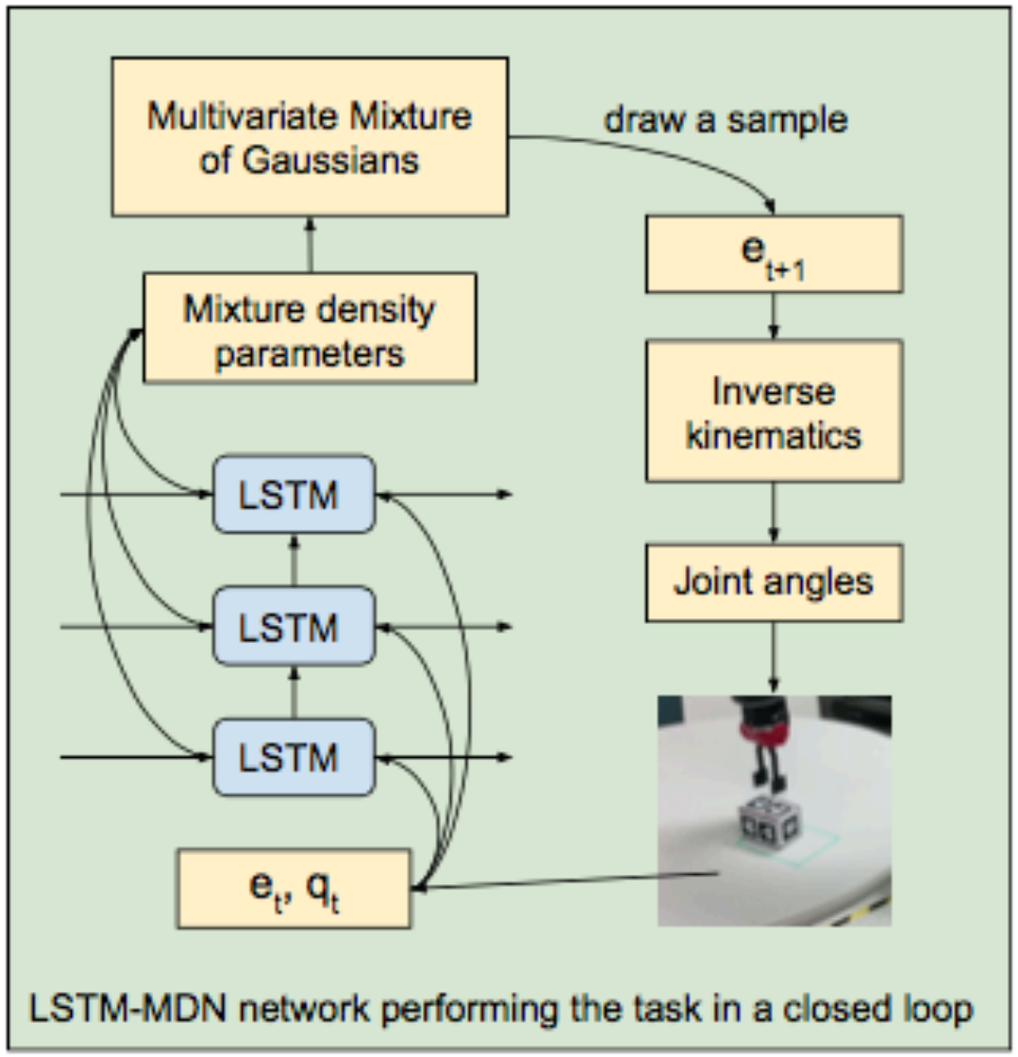
Case study: learning from virtual demonstrations



- Multimodality of actions-> GMM loss!
- Predict mixture weights over a Gaussian Mixture Model at the output (alphas) and mean and variances for the mixture components.

Learning real manipulation tasks from virtual demonstrations using LSTM, Rahmatizadeh et al 2016

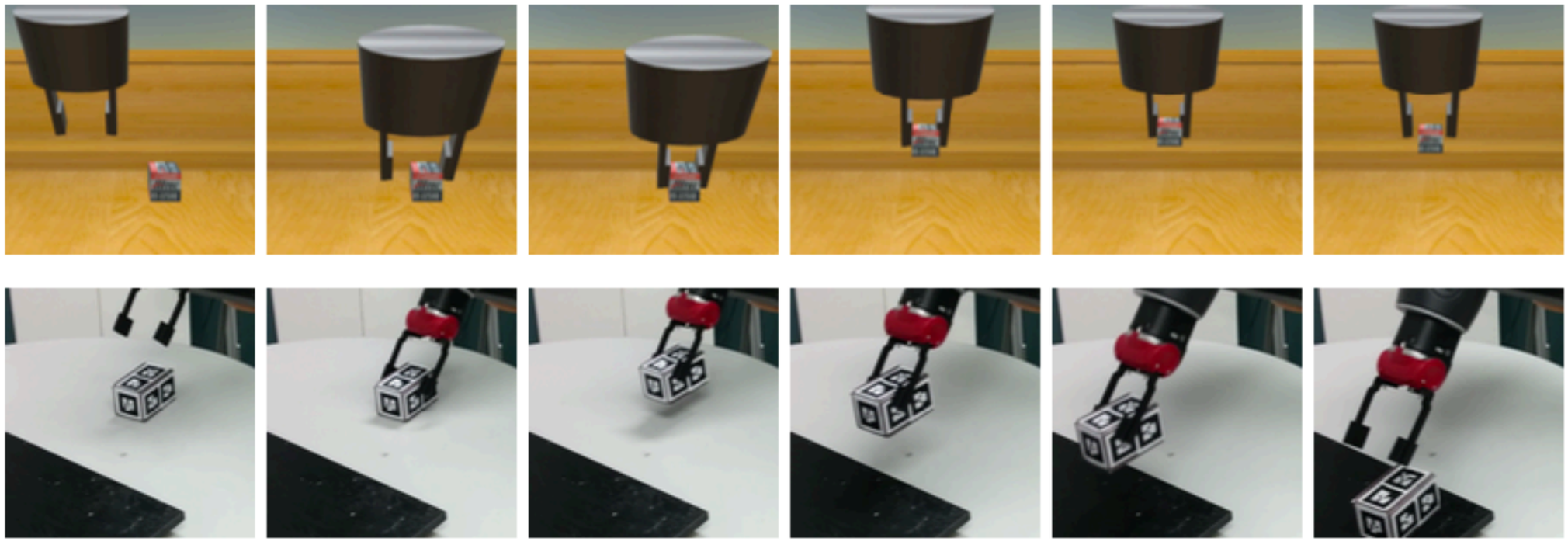
Case study: learning from virtual demonstrations



- Multimodality: predict mixture weights over a Gaussian Mixture Model at the output (alphas) and mean and variances for the mixture components. Minimize a GMM loss.

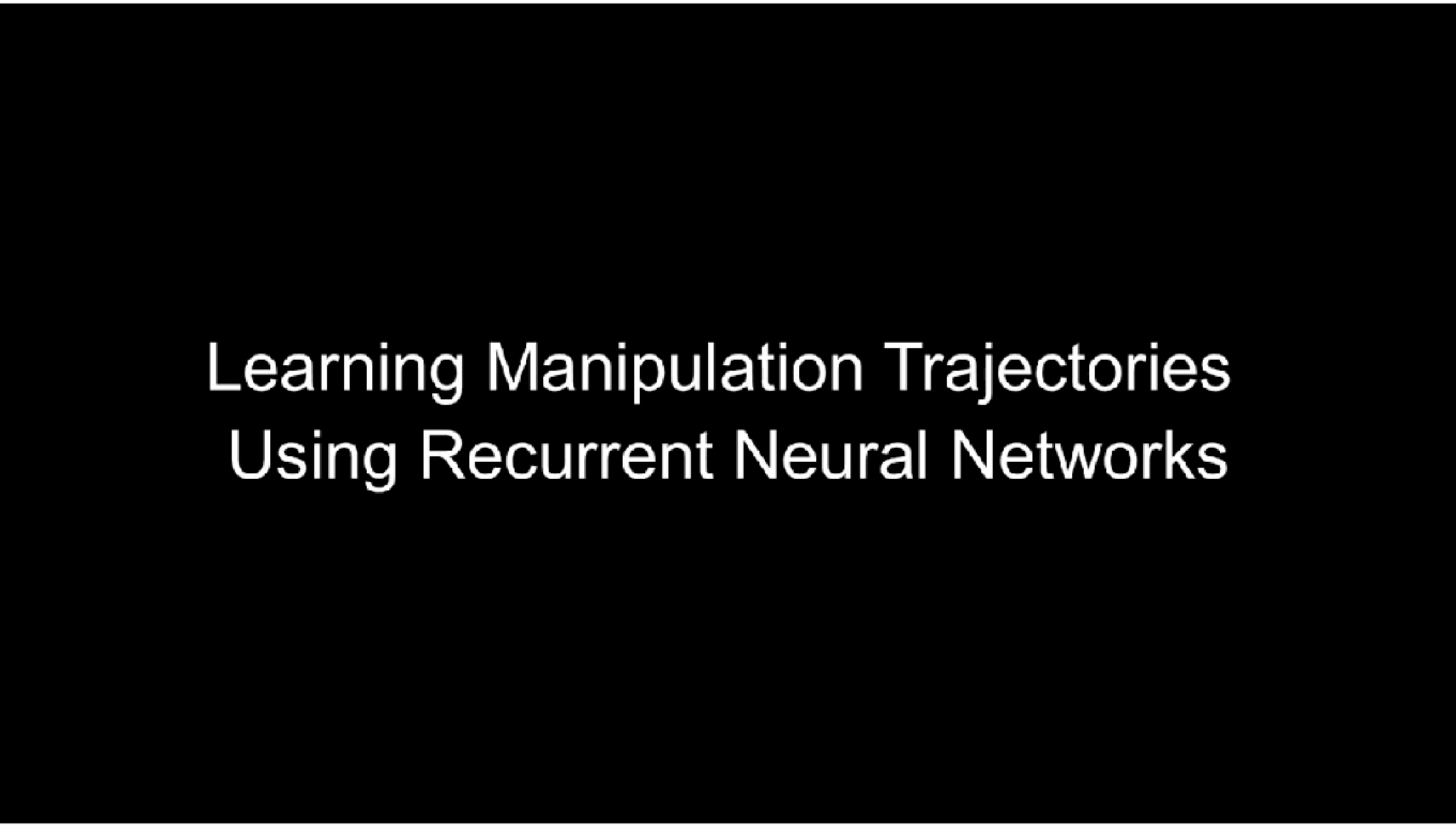
Learning real manipulation tasks from virtual demonstrations using LSTM, Rahmatizadeh et al 2016

Case study: learning from virtual demonstrations



Learning real manipulation tasks from virtual demonstrations using LSTM, Rahmatizadeh et al 2016

Case study: learning from virtual demonstrations



Learning real manipulation tasks from virtual demonstrations using LSTM, Rahmatizadeh et al 2016