Carnegie Mellon

School of Computer Science

Deep Reinforcement Learning and Control

Imitation Learning

Spring 2020, CMU 10-403

Katerina Fragkiadaki

Limitations of Learning by Interaction

- The agent should have the chance to try (and fail) MANY times
- This is hard when safety is a concern: we cannot afford to fail
- This is also quite hard 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 action space of the agent.
Two challenges:

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

(later lecture)

kinesthetic imitation

- The teacher takes over the end-effectors of the agent.
- Demonstrated actions are in the action space of the imitator and can be imitated directly)

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

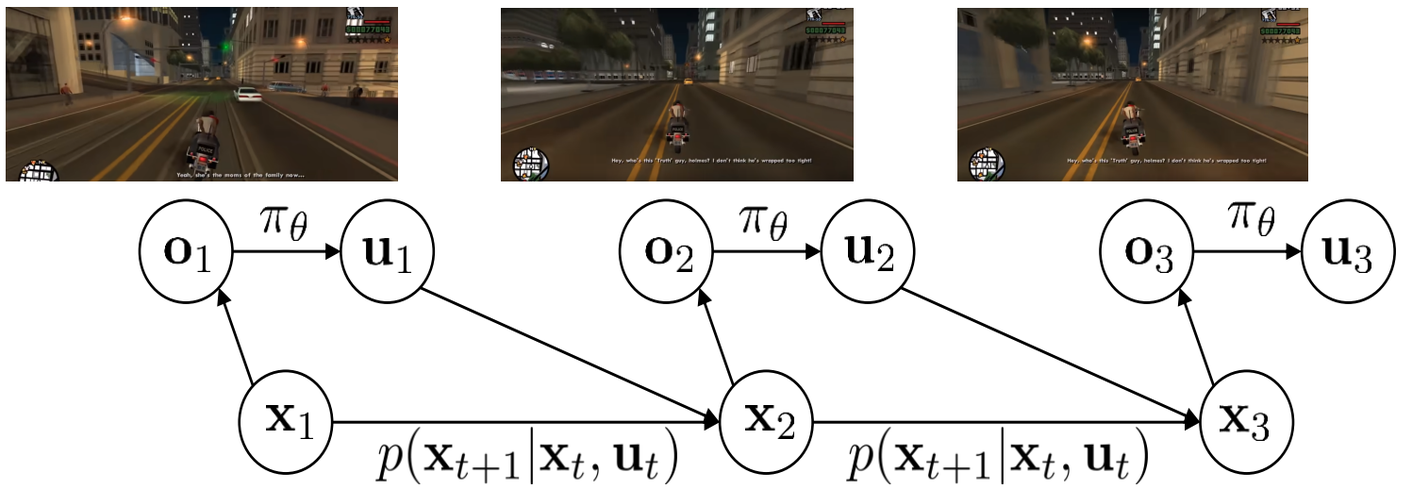
costs $c(x_t, u_t)$

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

Diagram from Sergey Levine

Imitation learning VS Sequence labelling

Imitation learning

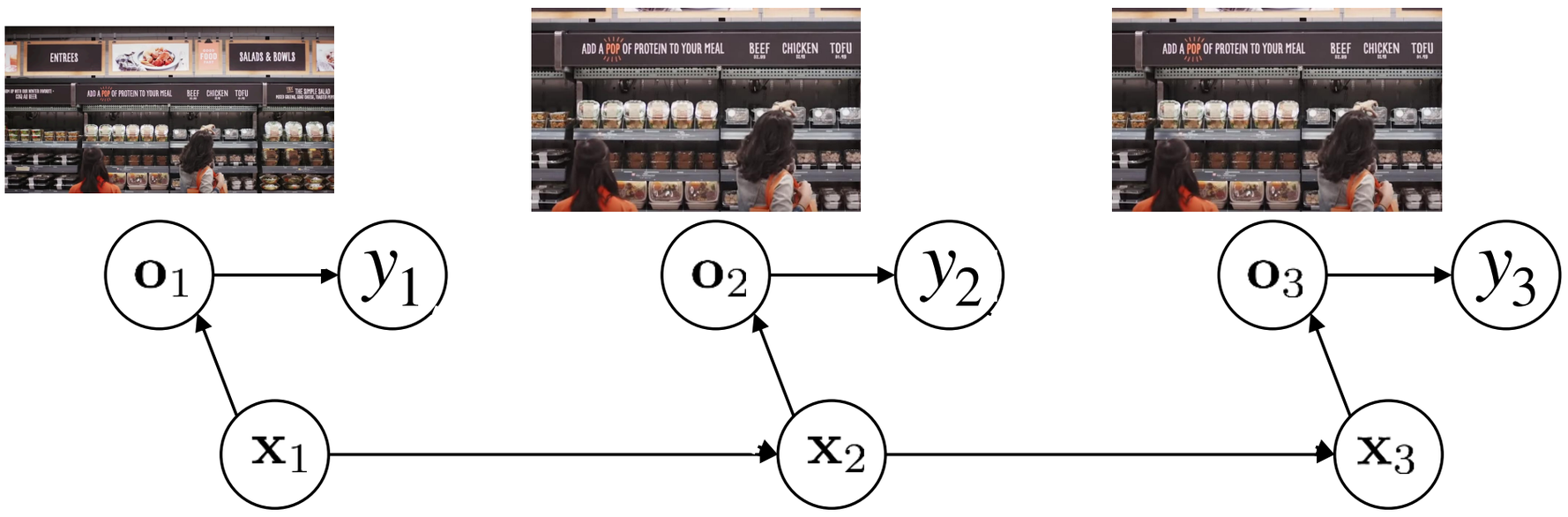


Training data:

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

$\mathbf{u}_t$:the action at time t $\quad$ $\mathbf{o}_t$:the observation at time t $\quad$ $\mathbf{x}_t$:the state at time t

Sequence labelling



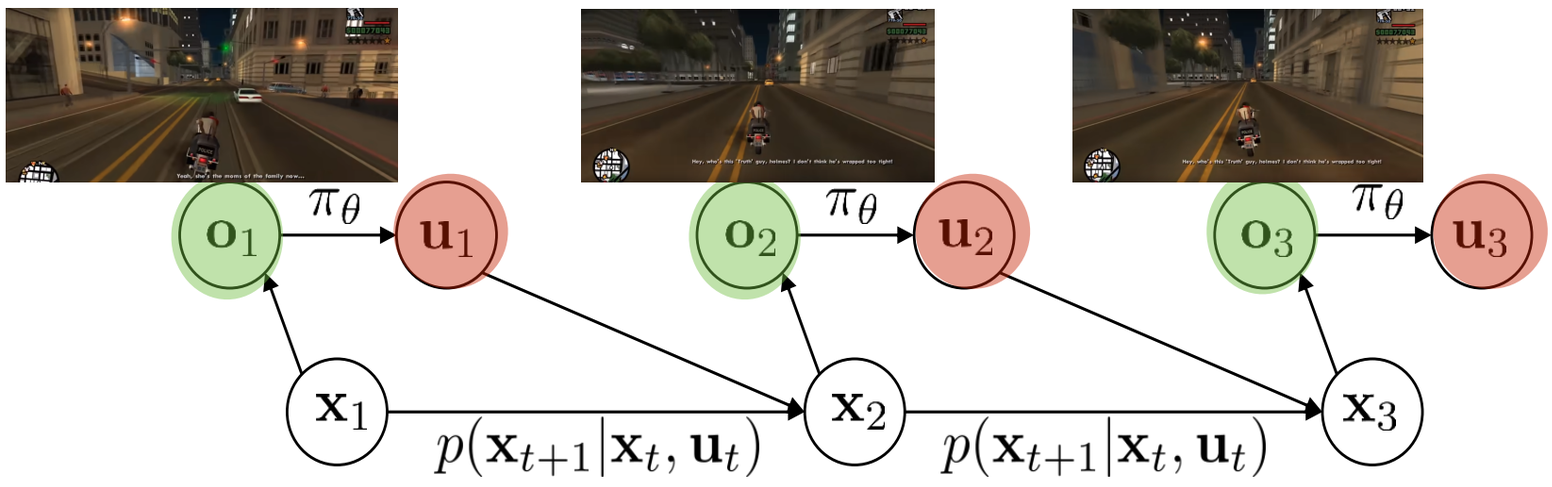
Training data:

$$o_1^1, y_1^1, o_2^1, y_2^1, o_3^1, y_3^1, \ldots.$$
$$o_1^2, y_1^2, o_2^2, y_2^2, o_3^2, y_3^2, \ldots.$$
$$o_1^3, y_1^3, o_2^3, y_2^3, o_3^3, y_3^3, \ldots.$$

y_t : which product was purchased at frame t (if any) $\quad$ $\mathbf{o}_t$:the observation at time t $\quad$ $\mathbf{x}_t$:the state at time t

Imitation learning VS Sequence labelling

Imitation learning

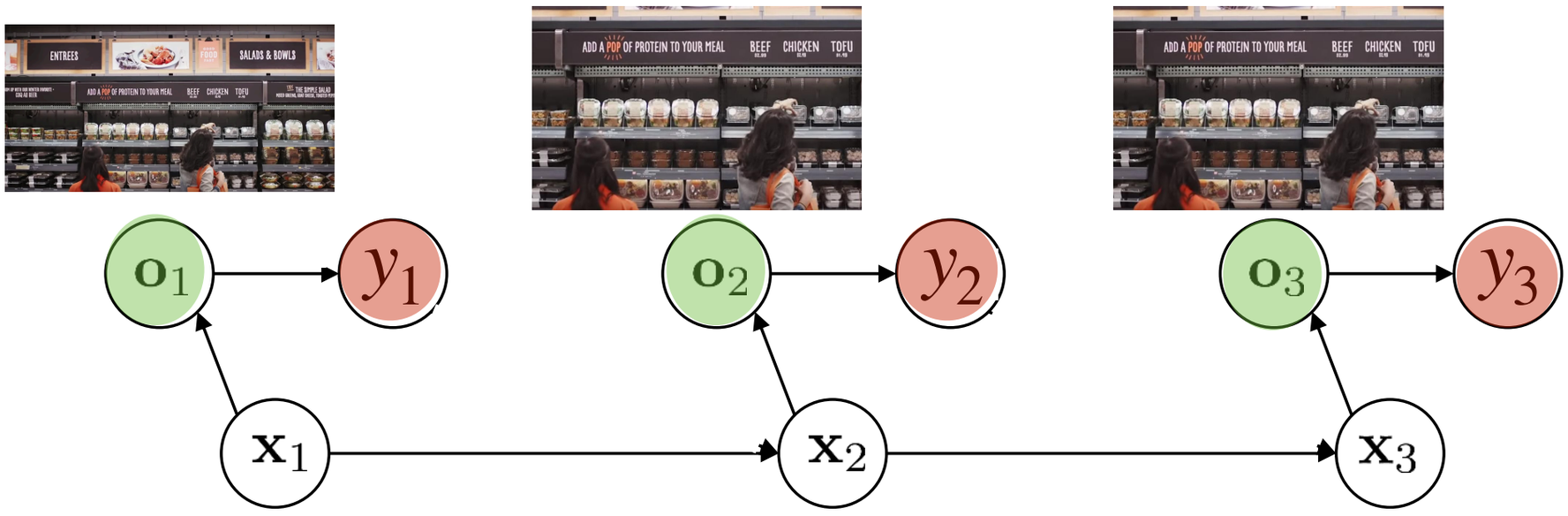


Training data:

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

$\mathbf{u}_t$:the action at time t $\mathbf{o}_t$:the observation at time t $\mathbf{x}_t$:the state at time t

Sequence labelling



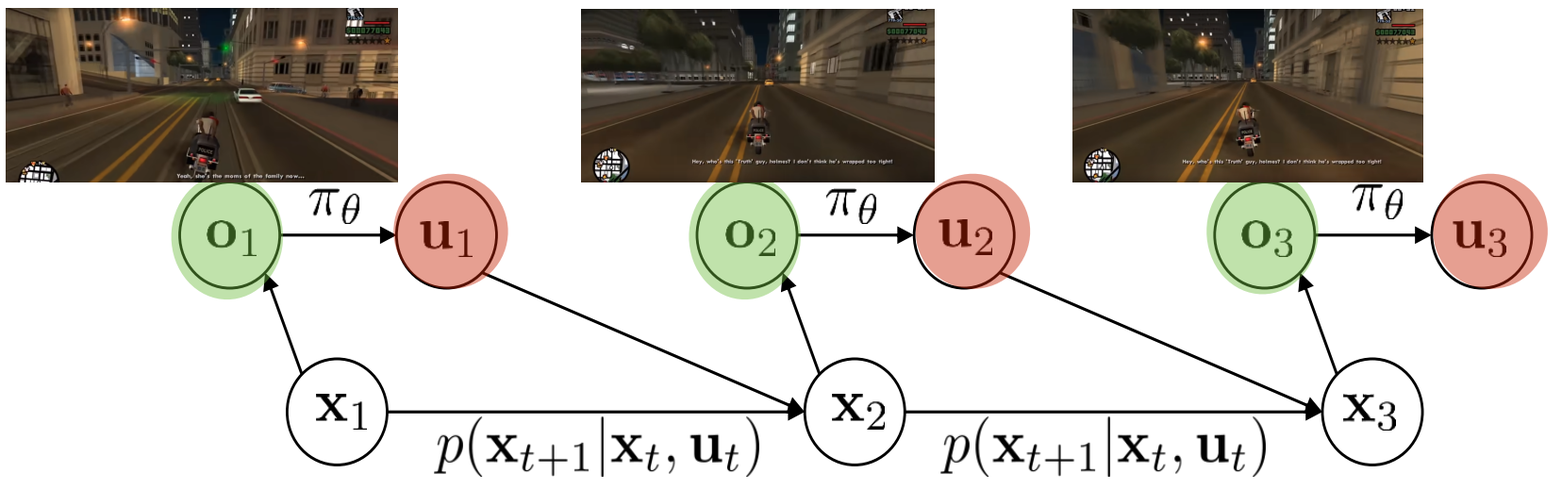
Training data:

$o_1^1, y_1^1, o_2^1, y_2^1, o_3^1, y_3^1, \ldots.$
$o_1^2, y_1^2, o_2^2, y_2^2, o_3^2, y_3^2, \ldots.$
$o_1^3, y_1^3, o_2^3, y_2^3, o_3^3, y_3^3, \ldots.$

y_t : which product was purchased at frame t (if any) $\mathbf{o}_t$:the observation at time t $\mathbf{x}_t$:the state at time t

Imitation learning VS Sequence labelling

Imitation learning

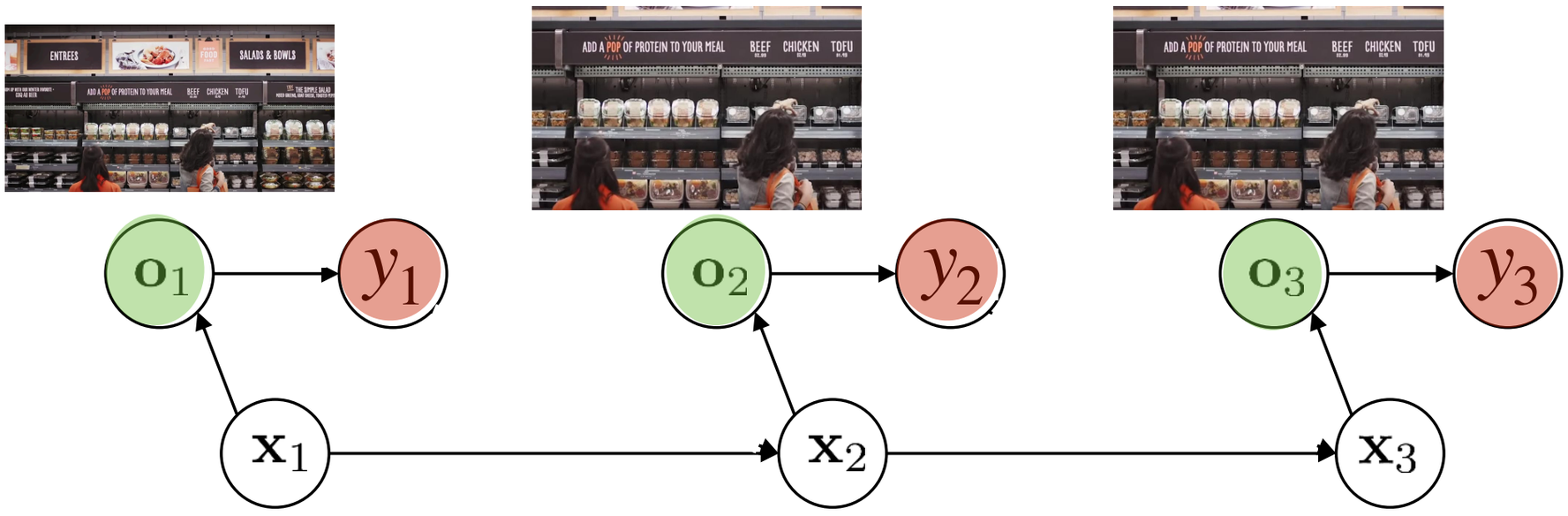


Training data:

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

$\mathbf{u}_t$:the action at time t $\mathbf{o}_t$:the observation at time t $\mathbf{x}_t$:the state at time t

Sequence labelling



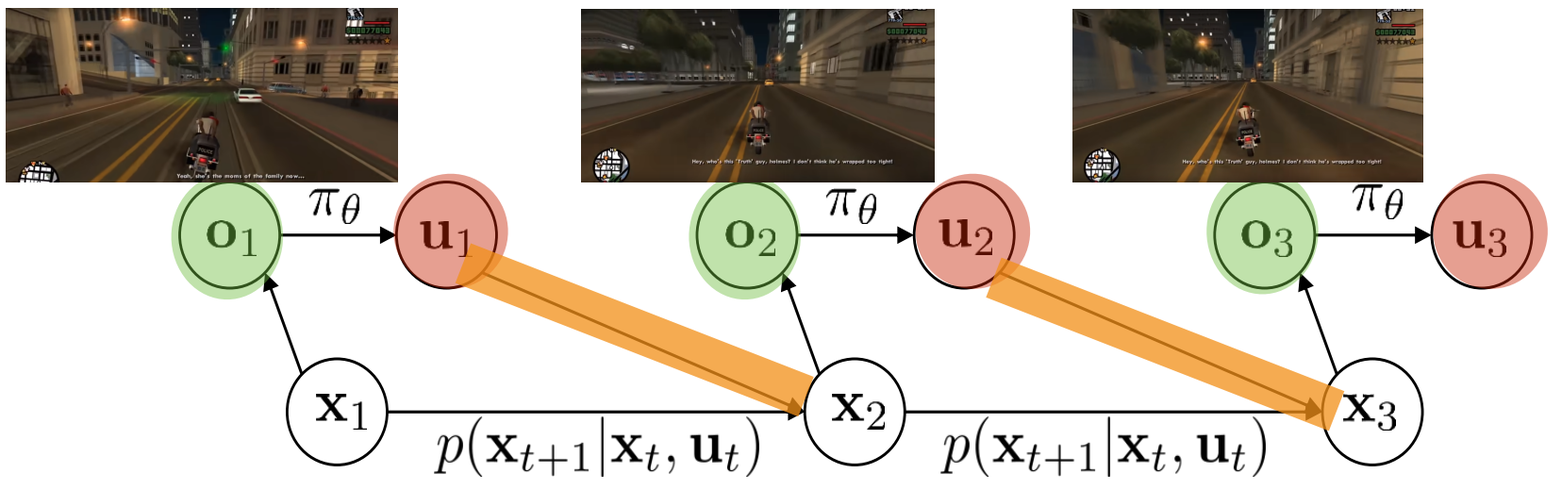
Training data:

$o_1^1, y_1^1, o_2^1, y_2^1, o_3^1, y_3^1, \ldots.$
$o_1^2, y_1^2, o_2^2, y_2^2, o_3^2, y_3^2, \ldots.$
$o_1^3, y_1^3, o_2^3, y_2^3, o_3^3, y_3^3, \ldots.$

y_t : which product was purchased at frame t (if any) $\mathbf{o}_t$:the observation at time t $\mathbf{x}_t$:the state at time t

Imitation learning VS Sequence labelling

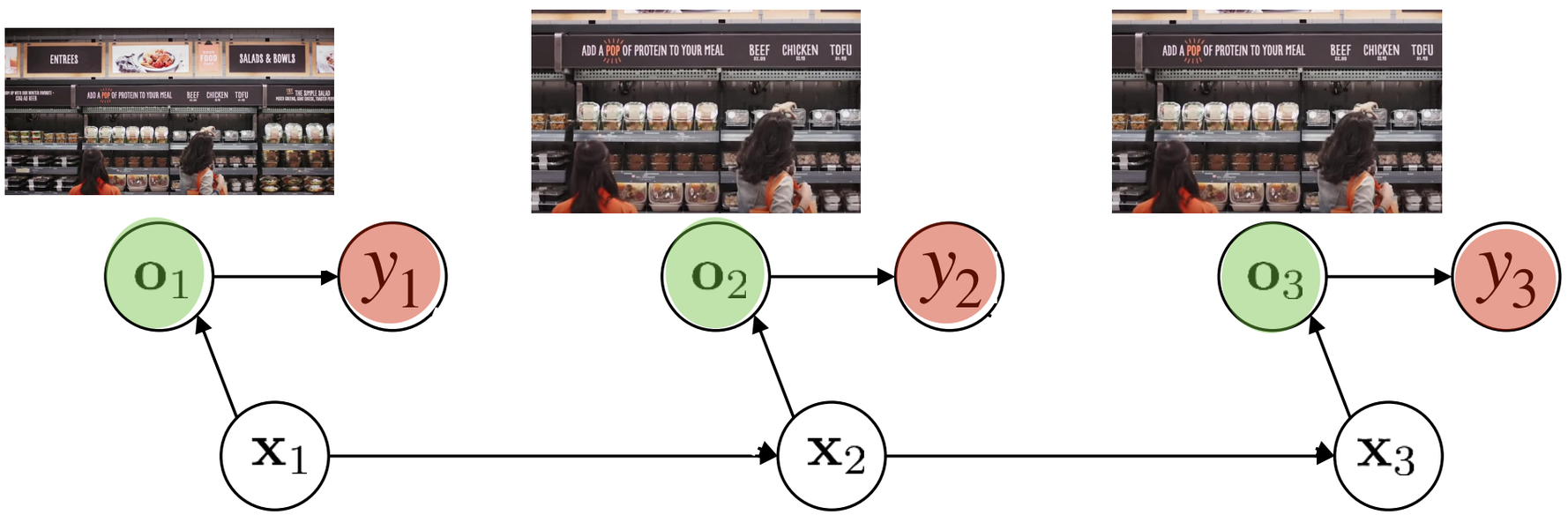
Imitation learning



- In RL, our actions will influence our future state, and thus our future data.
- In sequence labelling, our labels won't influence the future frames.

$\mathbf{u}_t$:the action at time t $\mathbf{o}_t$:the observation at time t $\mathbf{x}_t$:the stat

Sequence labelling



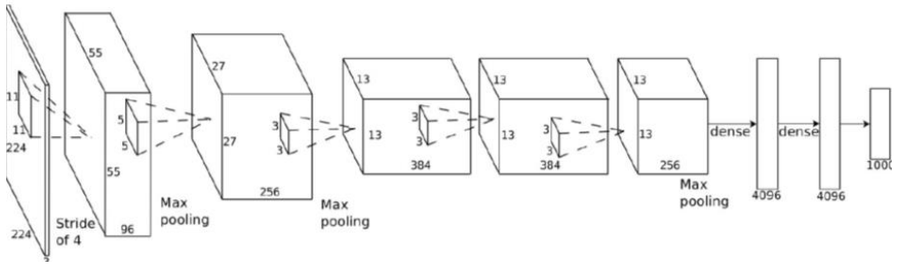
Training data:

$o_1^1, y_1^1, o_2^1, y_2^1, o_3^1, y_3^1, \ldots.$
$o_1^2, y_1^2, o_2^2, y_2^2, o_3^2, y_3^2, \ldots.$
$o_1^3, y_1^3, o_2^3, y_2^3, o_3^3, y_3^3, \ldots.$

y_t : which product was purchased at frame t (if any)

Video sequence labelling

Action labelling: a mapping from states/observations to action labels



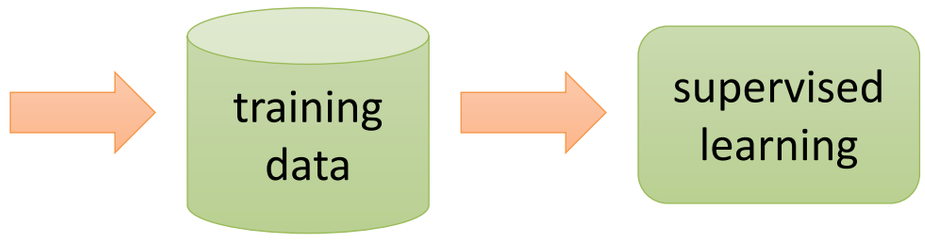
$$f_\theta(y_t \mid \mathbf{o}_t)$$

"picking up chicken A"

- Assume action labels in an annotated video are i.i.d. (independent and identically distributed).
- Train a classifier to map observations to labels at each time step of the trajectory

$\mathbf{o}_t$

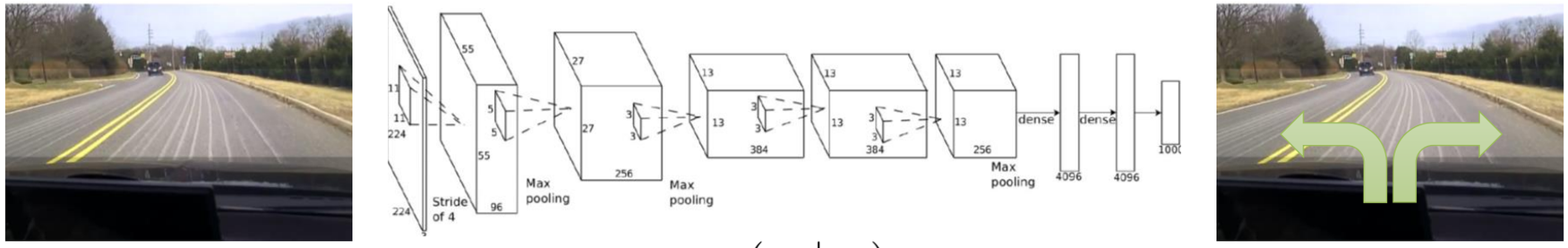
y_t



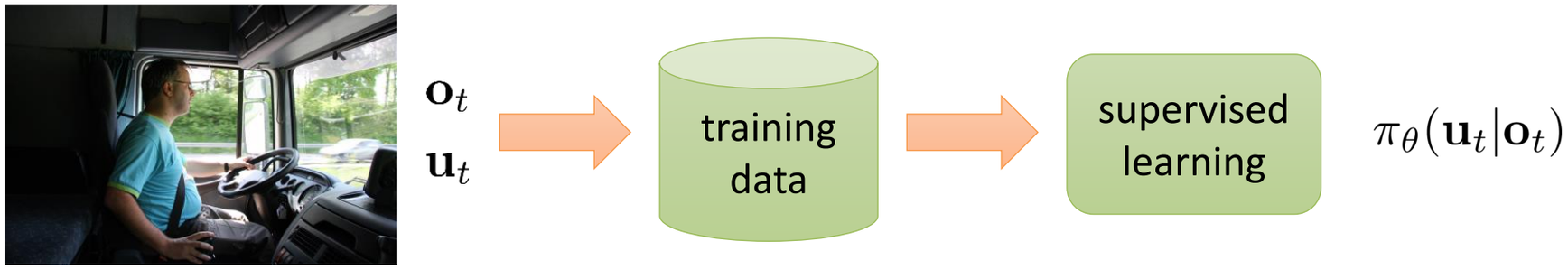
$$f_\theta(y_t \mid \mathbf{o}_t)$$

Imitation Learning

Policy: a mapping from observations to actions

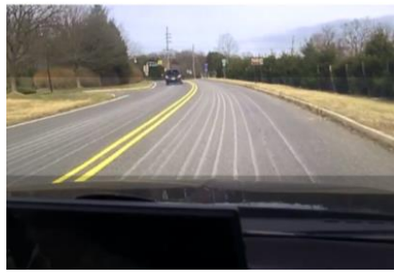


- Assume actions in the expert trajectories are i.i.d. (independent and identically distributed)
- Train a function to map observations/states to actions at each time step of the trajectory

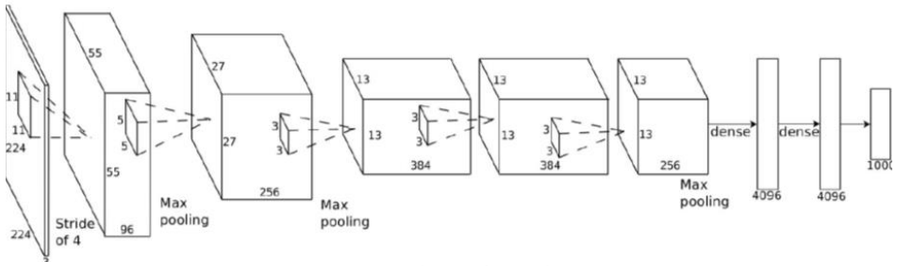


What can go wrong?

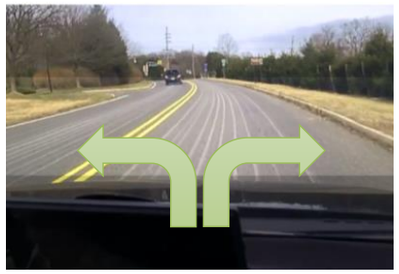
- Compounding errors
  Fix: data augmentation
- Non-markovian observations
  Fix: observation concatenation or recurrent models
- Stochastic expert actions
  Fix: generative modelling (GANs,stochastic latent variable models, action discretisation, gaussian mixture networks)



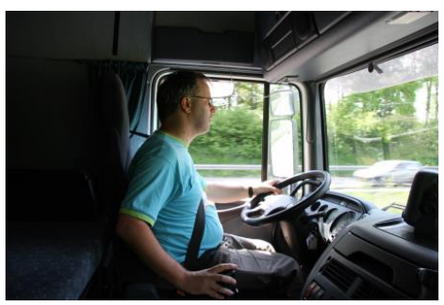
$\mathbf{o}_t$



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

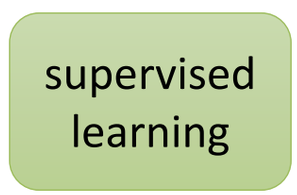


$\mathbf{u}_t$



$\mathbf{o}_t$

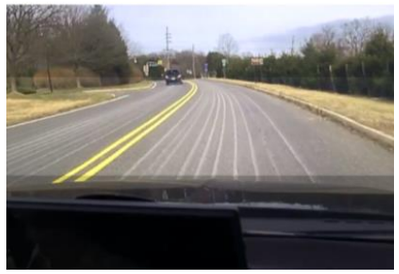
$\mathbf{u}_t$



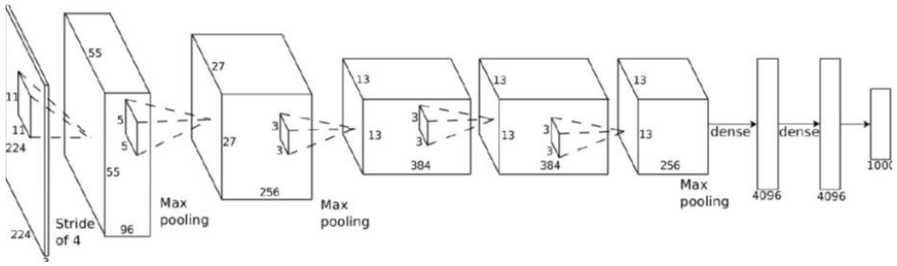
$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$

What can go wrong?

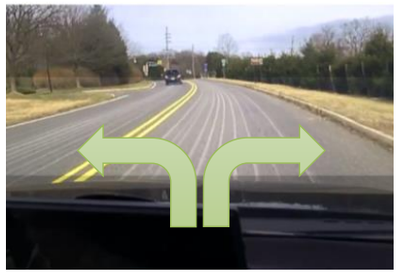
- Compounding errors
  Fix: data augmentation
- Non-markovian observations
  Fix: observation concatenation or recurrent models
- Stochastic expert actions
  Fix: generative modelling (GANs,stochastic latent variable models, action discretisation, gaussian mixture networks)



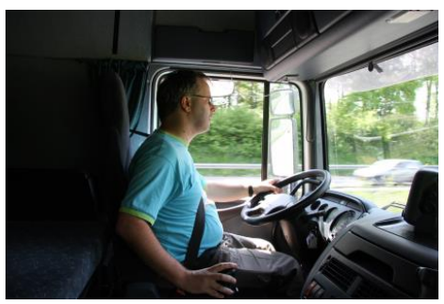
$\mathbf{o}_t$



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

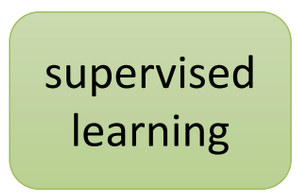


$\mathbf{u}_t$



$\mathbf{o}_t$

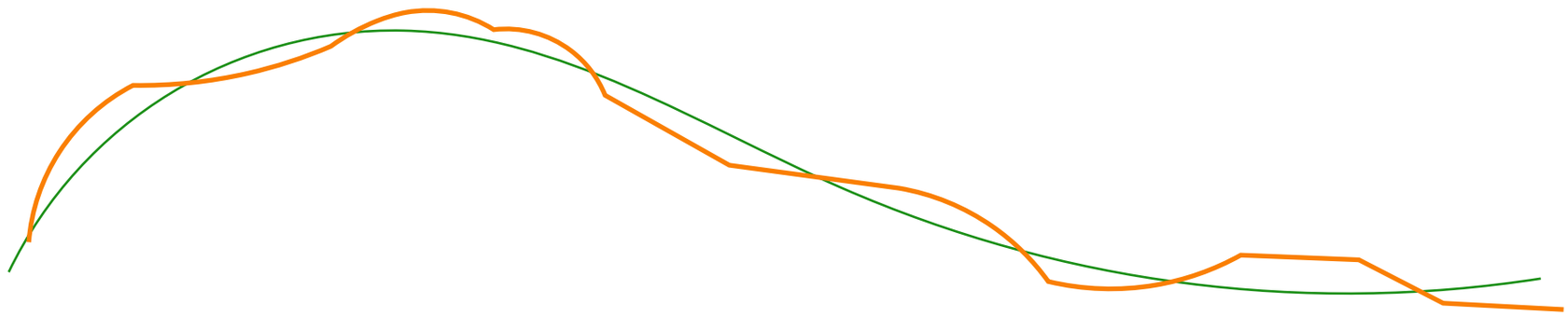
$\mathbf{u}_t$



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

Independent in time errors

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

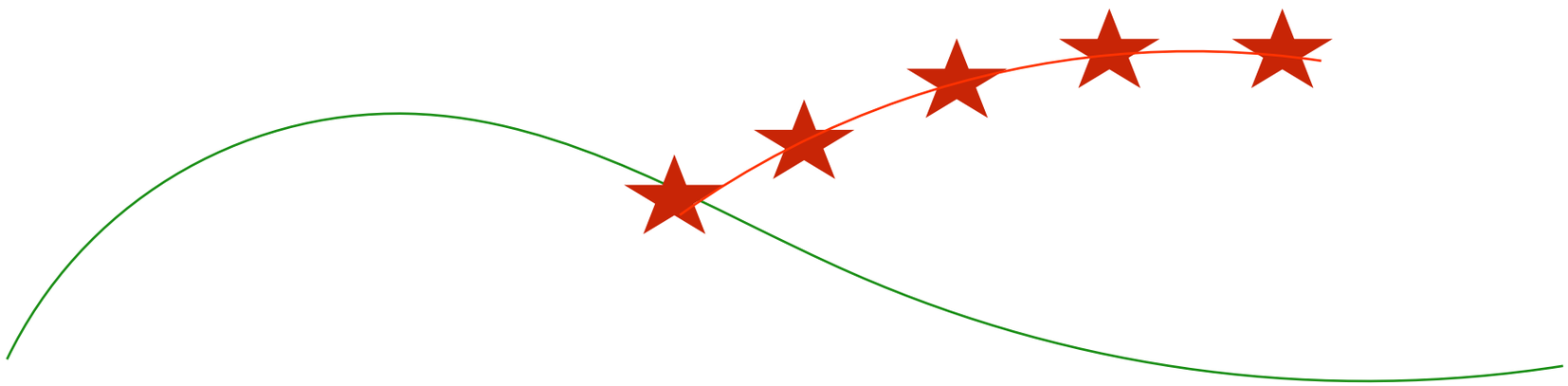


error at time t with probability ε

E[Total errors] ≲ εT, T the length of the trajectory

Compounding Errors

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



error at time t with probability ε

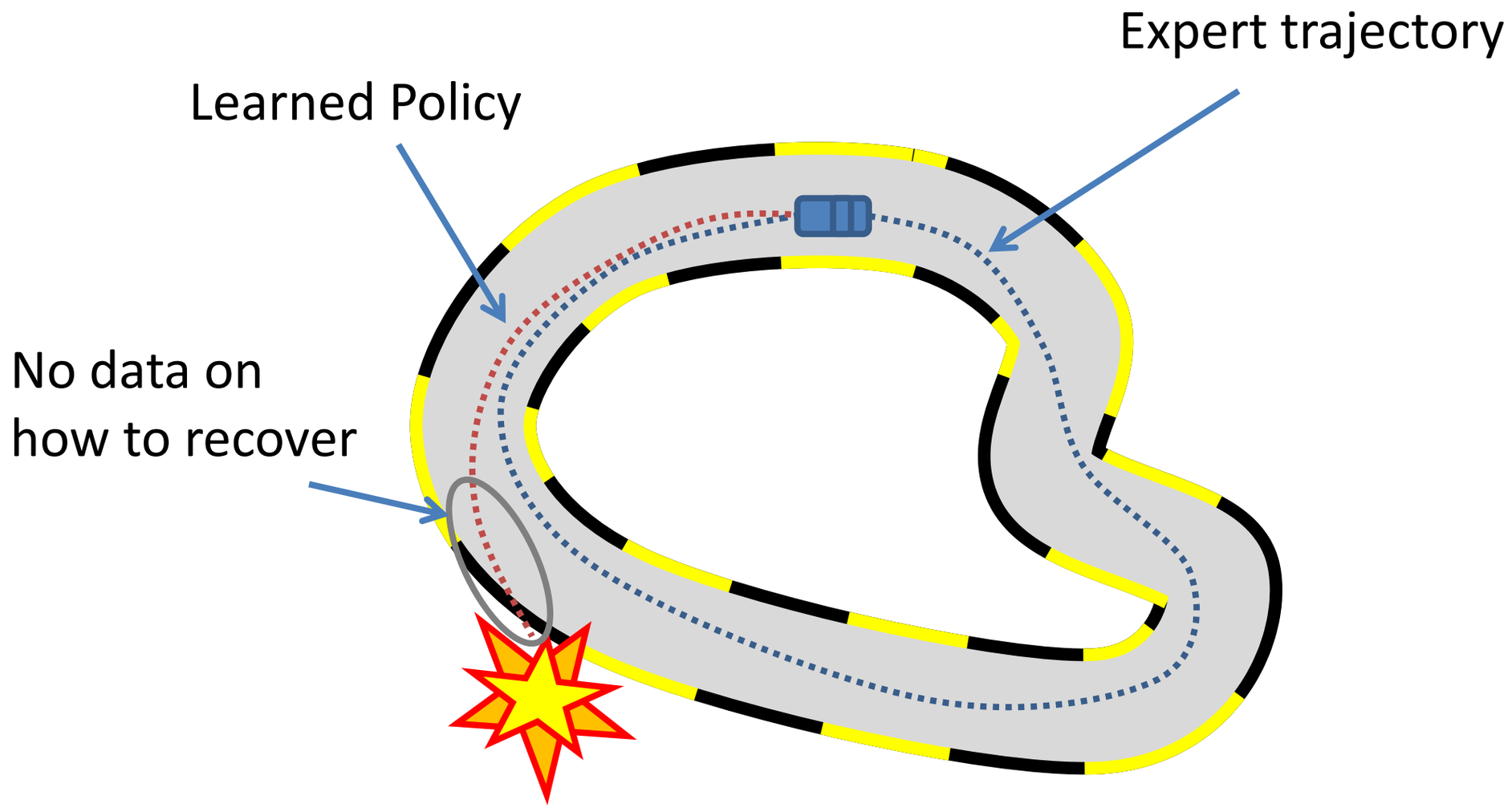
$E[\text{Total errors}] \lesssim \varepsilon(T + (T-1) + (T-2) + \ldots + 1) \propto \varepsilon T^2$

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

Distribution mismatch (distribution shift)

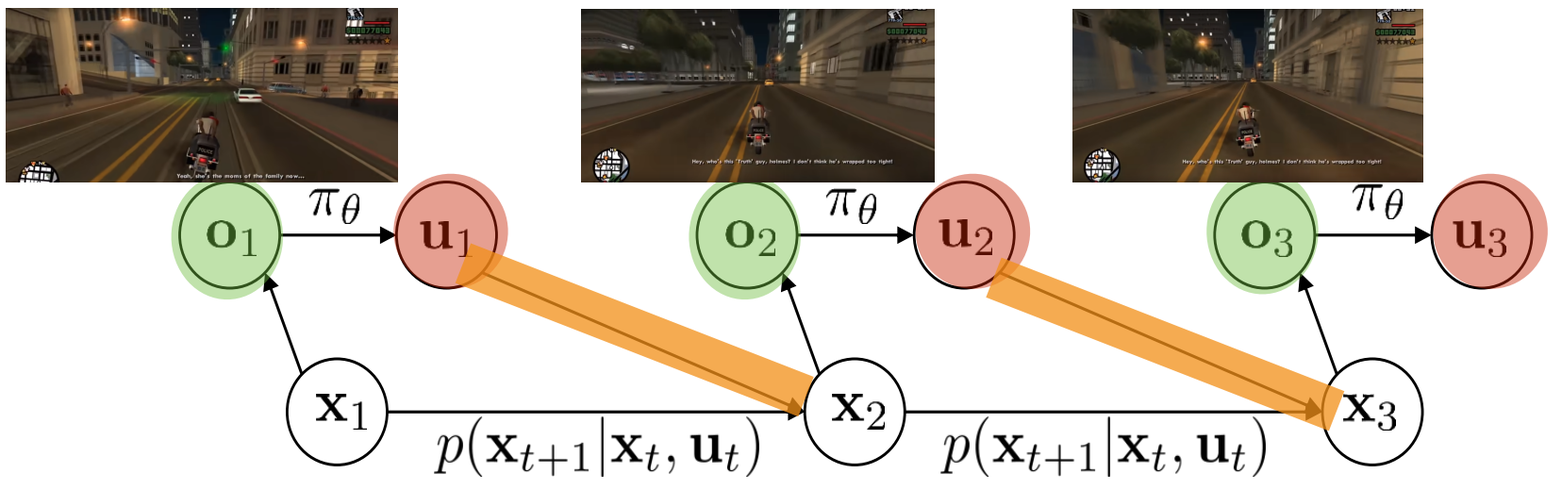
Due to the interdependence between our action at time step t and the state at t+1, states seen at test time may come from a different distribution than those seen at training time.

$$P_{\pi^*}(\mathbf{o}_t) \neq P_{\pi_\theta}(\mathbf{o}_t)$$



Imitation learning VS Sequence Generation

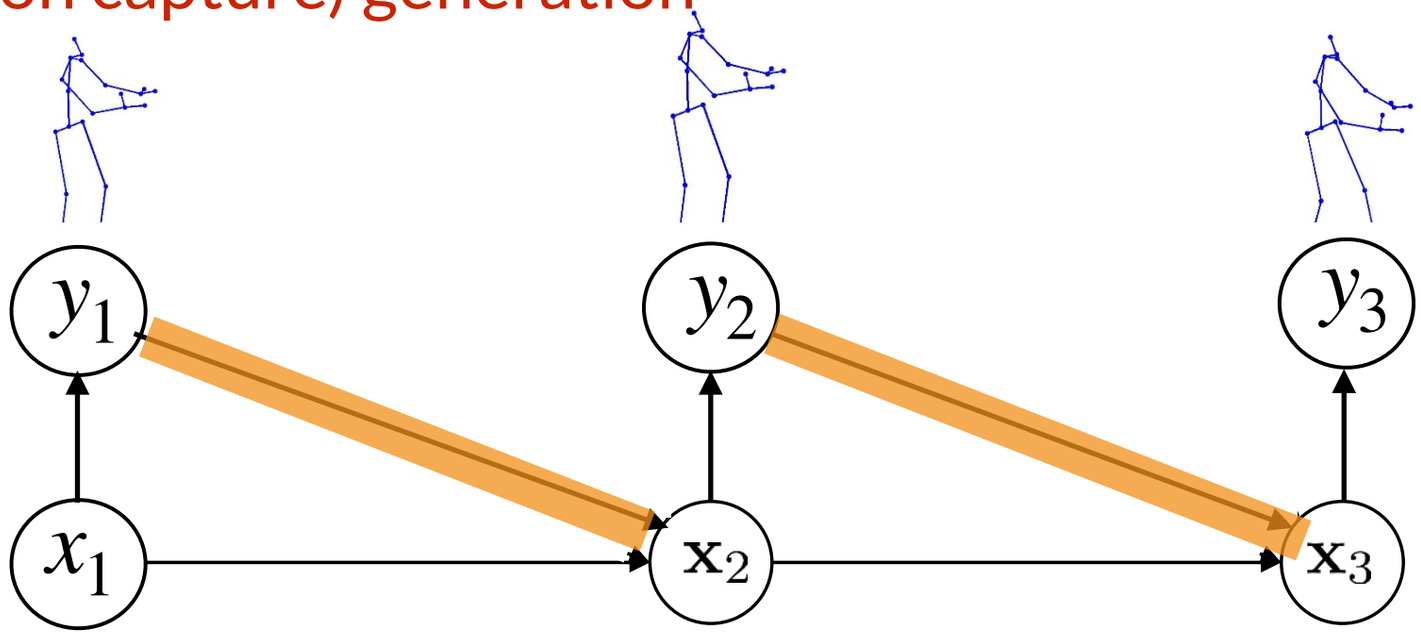
Imitation learning



- In RL, our actions will influence our future state, and thus our future data.
- In sequence generation, the token we generate at time t will influence what we should generate at time t+1.

$\mathbf{u}_t$:the action at time t $\mathbf{o}_t$:the observation at time t $\mathbf{x}_t$:the stat

MoCap (motion capture) generation



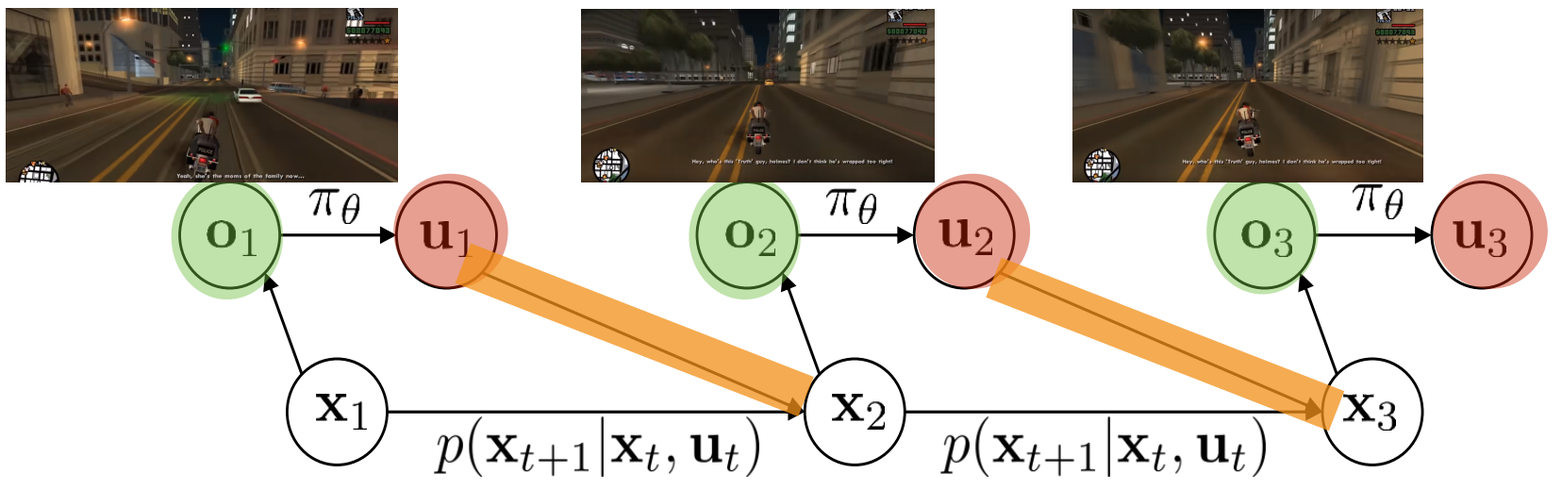
Training data:

$y_1^1, y_2^1, y_3^1, \ldots.$
$y_1^2, y_2^2, y_3^2, \ldots.$
$y_1^3, y_2^3, y_3^3, \ldots.$

y_t : the human skeleton at the tth timestep

Imitation learning VS Sequence Generation

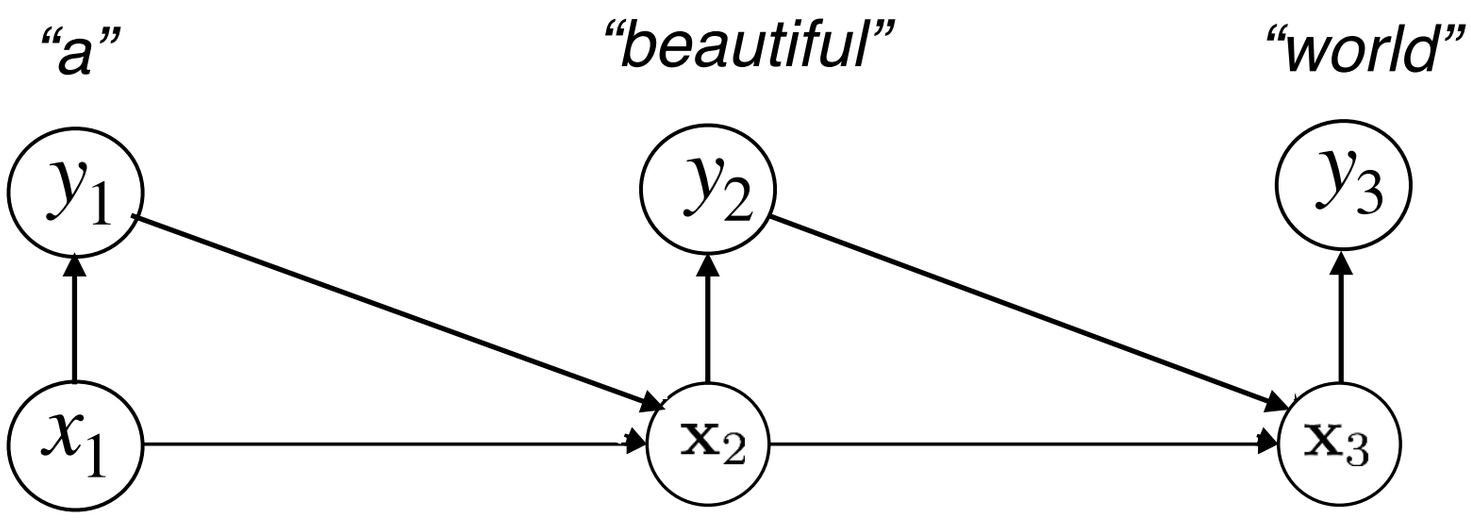
Imitation learning



- In RL, our actions will influence our future state, and thus our future data.
- In sequence generation, the token we generate at time t will influence what we should generate at time t+1.

$\mathbf{u}_t$:the action at time t $\mathbf{o}_t$:the observation at time t $\mathbf{x}_t$:the stat

Language generation



Training data:

$y_1^1, y_2^1, y_3^1, \ldots.$
$y_1^2, y_2^2, y_3^2, \ldots.$
$y_1^3, y_2^3, y_3^3, \ldots.$

y_t : the word at the tth timestep

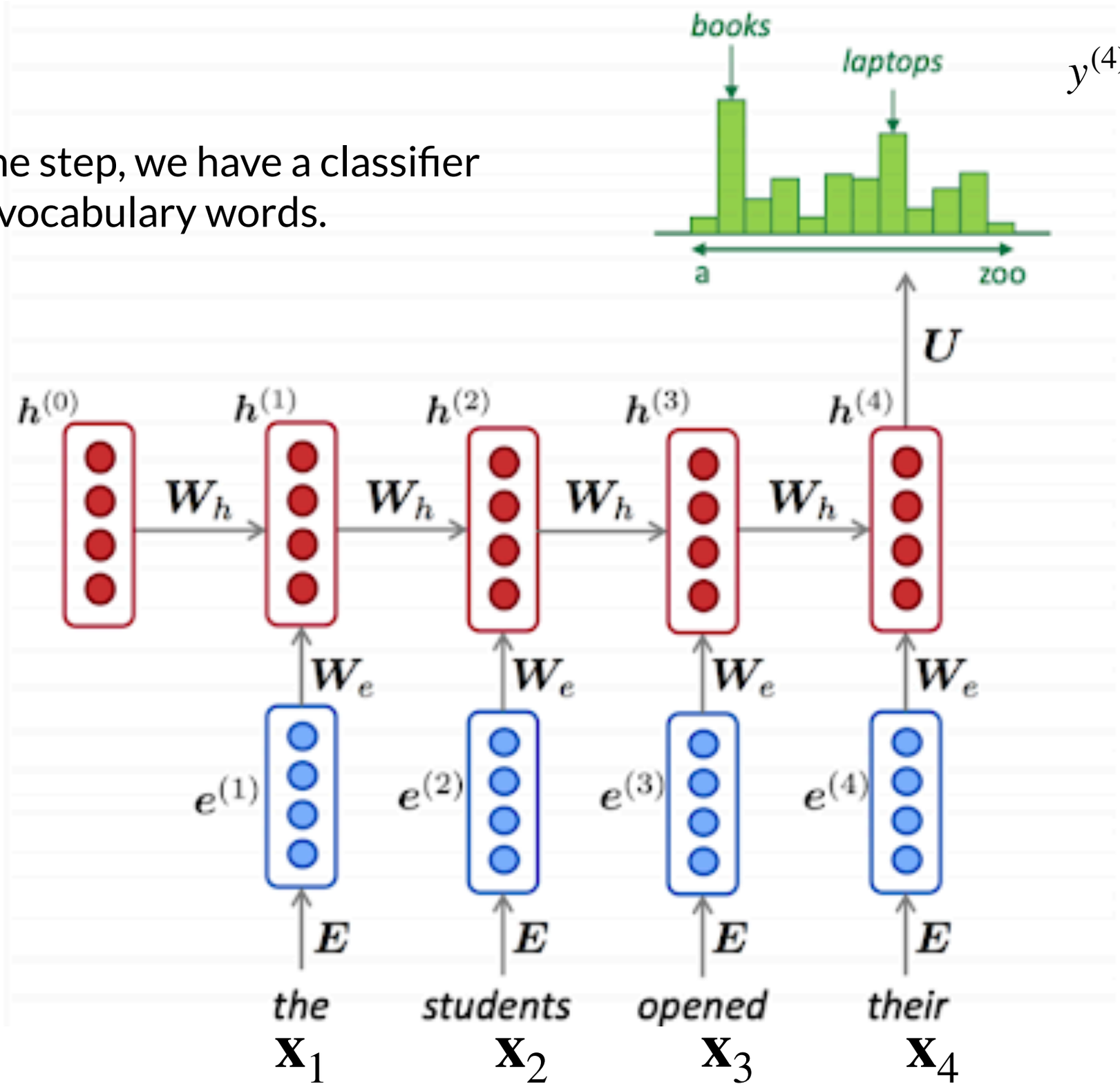
Distribution mismatch (distribution shift)

- Similar to imitation learning, in sequence generation there is distribution shift.
- We can fight is with scheduled sampling.
- In the begining of the training, the conditioning states come from the ``teacher'' (our training data).
- Later in training states are sampled from the output of the model. The ground-truth for the next time step does not change.
- In this way, the model learns to handle its mistakes: pushing deviating generated sequences back to the right track.

Recurrent Neural Networks for Sequence generation

$$y^{(4)} = P(\mathbf{x}_5 \,|\, \text{the students opened their})$$

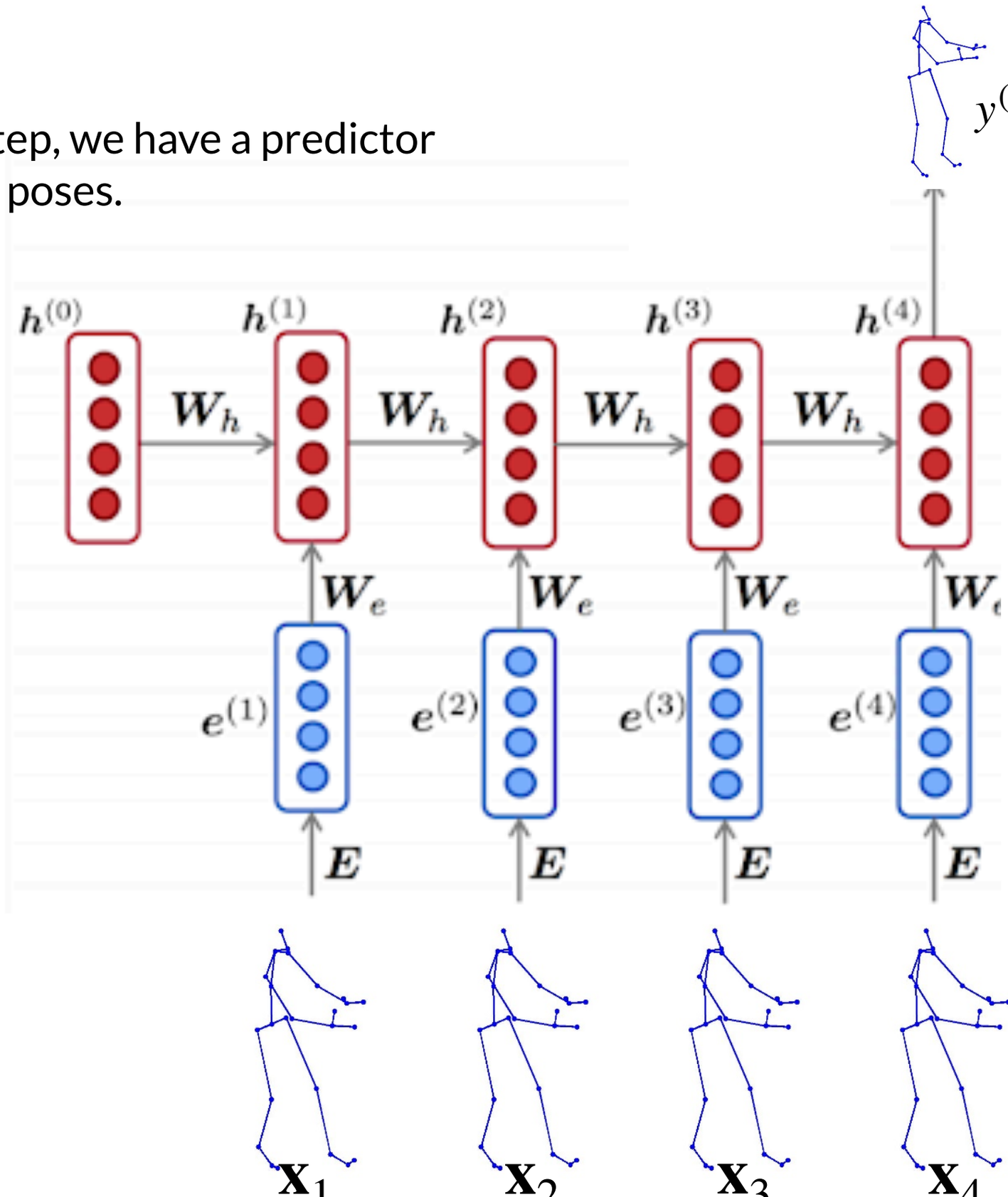
At every time step, we have a classifier over all the vocabulary words.



$$y^{(4)} = P(\mathbf{x}_5 \,|\, \mathbf{x}_1 \ldots \mathbf{x}_4)$$

Recurrent Neural Networks for Sequence generation

At every time step, we have a predictor over 3D human poses.



Scheduled sampling for sequence generation

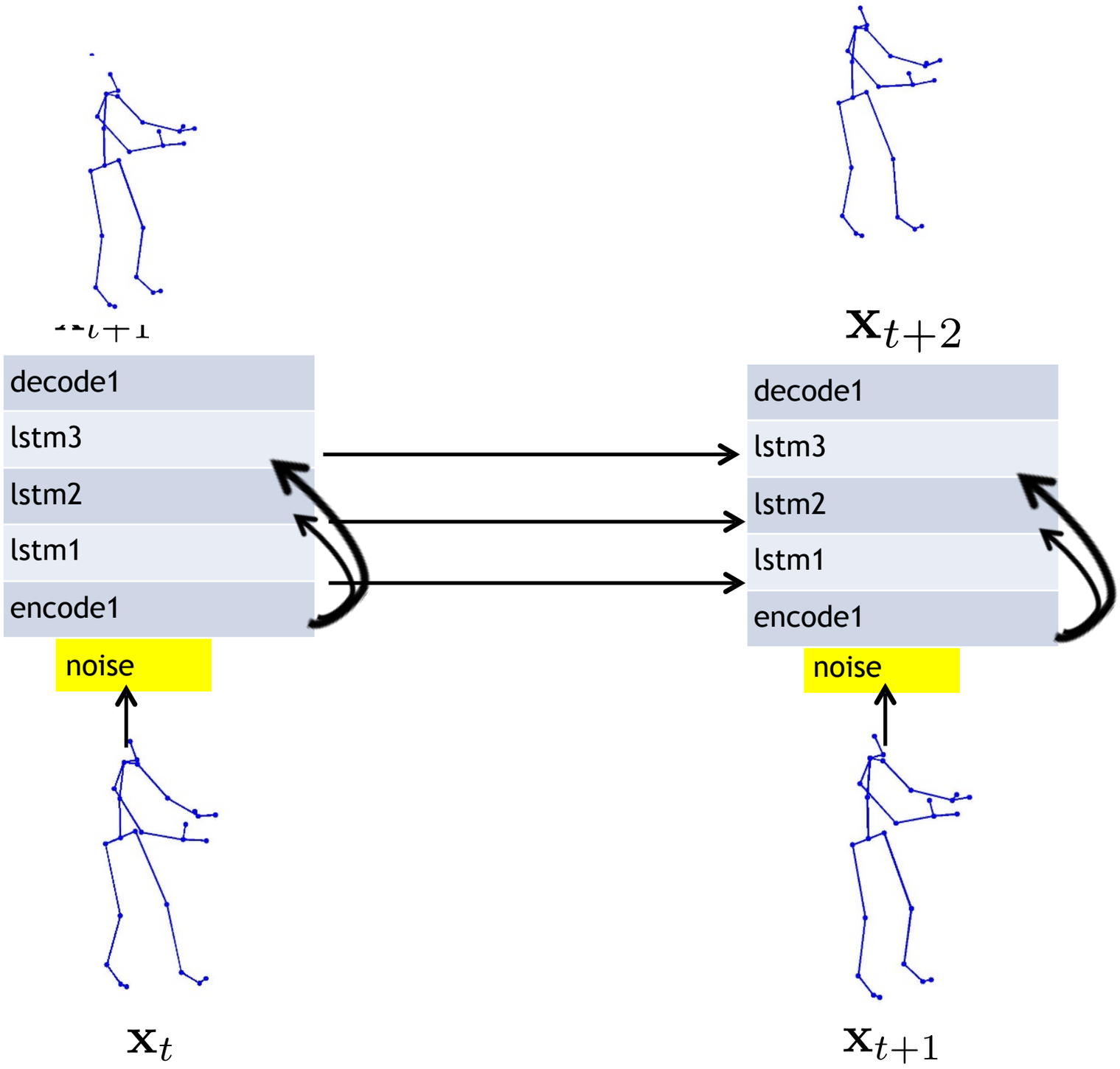
Q: what we be feeding the groundtruth x,y or the predicted x,y during training?

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



- Back to imitation learning

Distribution mismatch (distribution shift)

	supervised learning	supervised learning + control (NAIVE)
train	(x,y) ~ D	$\mathbf{o}_t \sim P_{\pi^*}(\mathbf{o}_t)$
test	(x,y) ~ D	$\mathbf{o}_t \sim P_{\pi_\theta}(\mathbf{o}_t)$

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

Solution: data augmentations

Change $P_{\pi^*}(\mathbf{o}_t)$ by augmenting the expert demonstration trajectories.
This measn: add examples in expert demonstration trajectories to cover the states/ observations points where the agent will land when trying out its own policy. How?

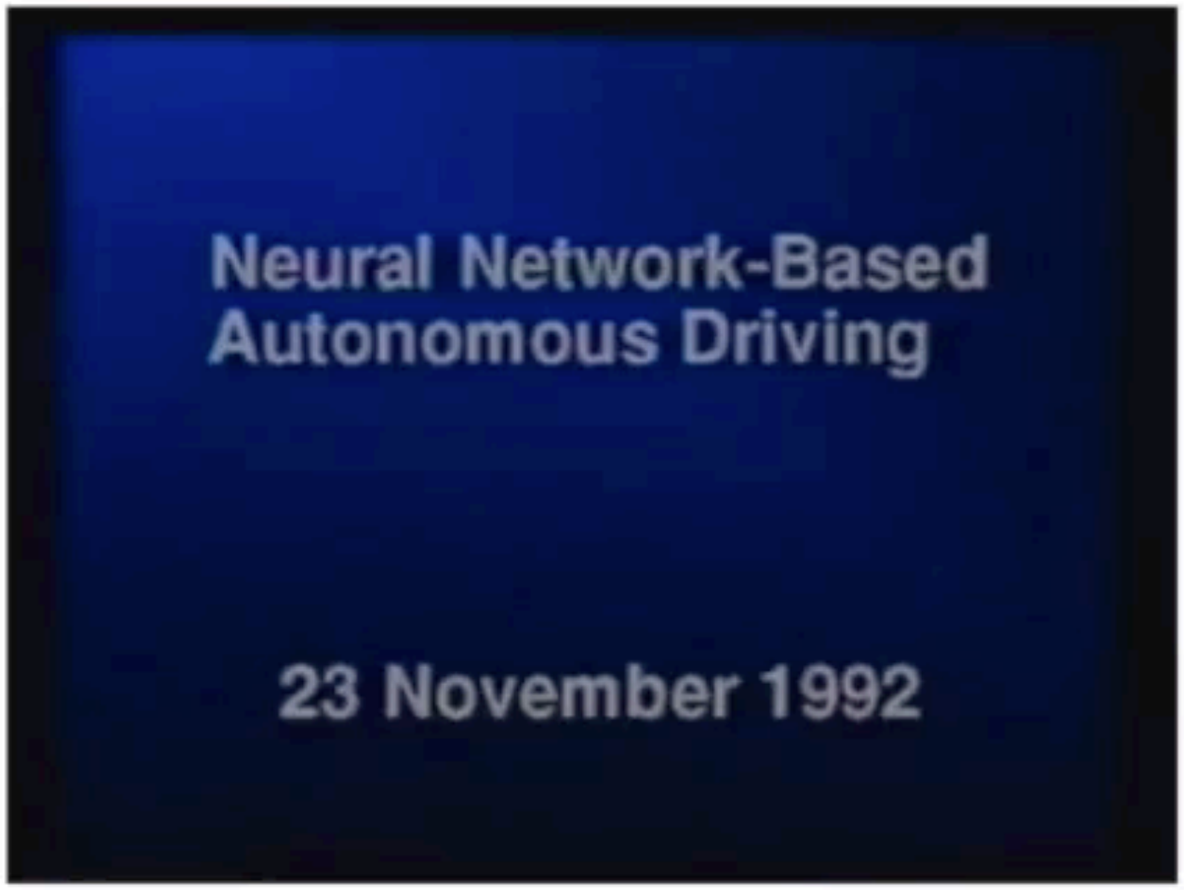
1. By generating synthetic data in simulation
2. By collecting additional data via clever hardware
3. By interactively querying the experts in additional datapoints

Solution: data augmentations

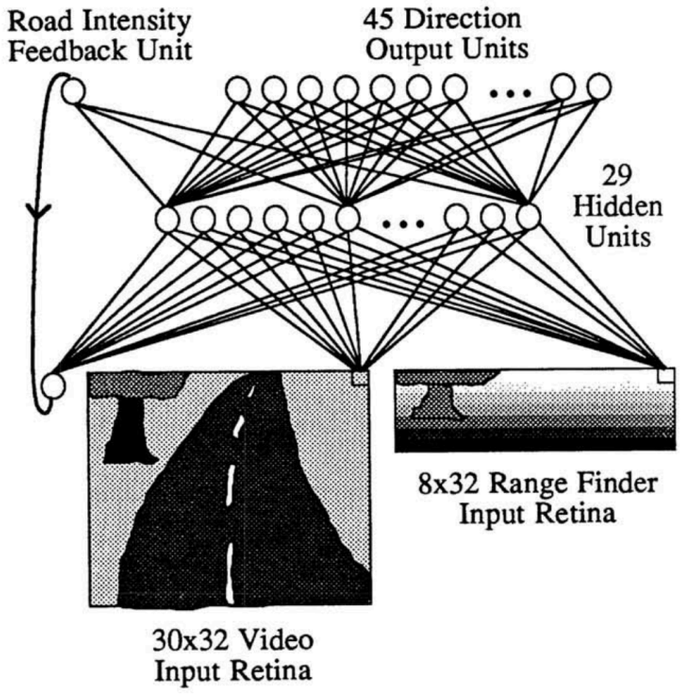
Change $P_{\pi^*}(\mathbf{o}_t)$ by augmenting the expert demonstration trajectories.
Add examples in expert demonstration trajectories to cover the states/ observations points where the agent will land when trying out its own policy. How?

1. By generating synthetic data in simulation
2. By collecting additional data via clever hardware
3. By interactively querying the experts in additional datapoints

Demonstration Augmentation: ALVINN 1989



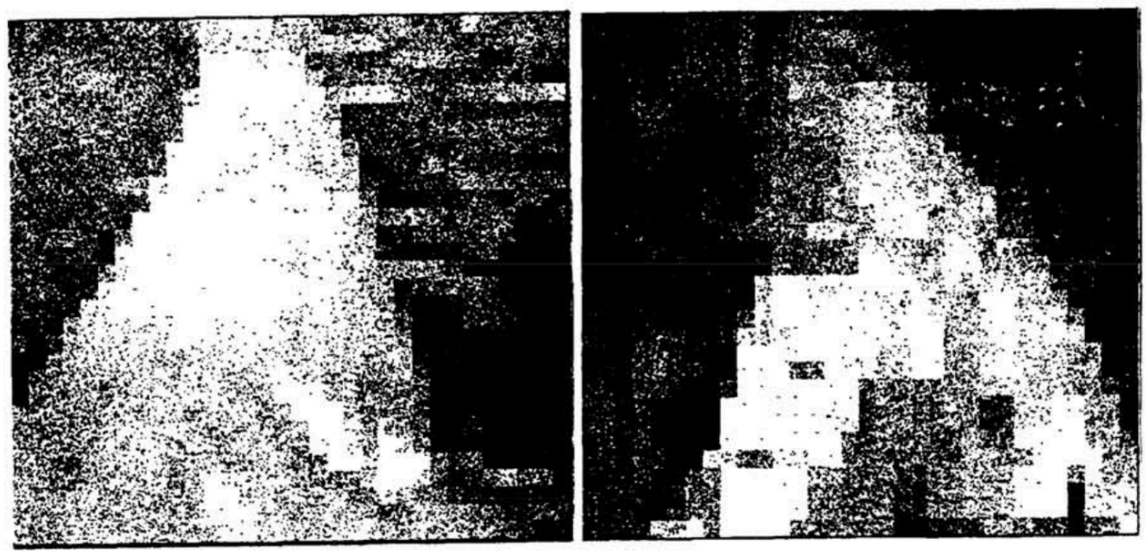
[Courtesy of Dean Pomerleau]



Demonstration Augmentation: ALVINN 1989

“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: ALVINN 1989



Real Road Image Simulated Road Image

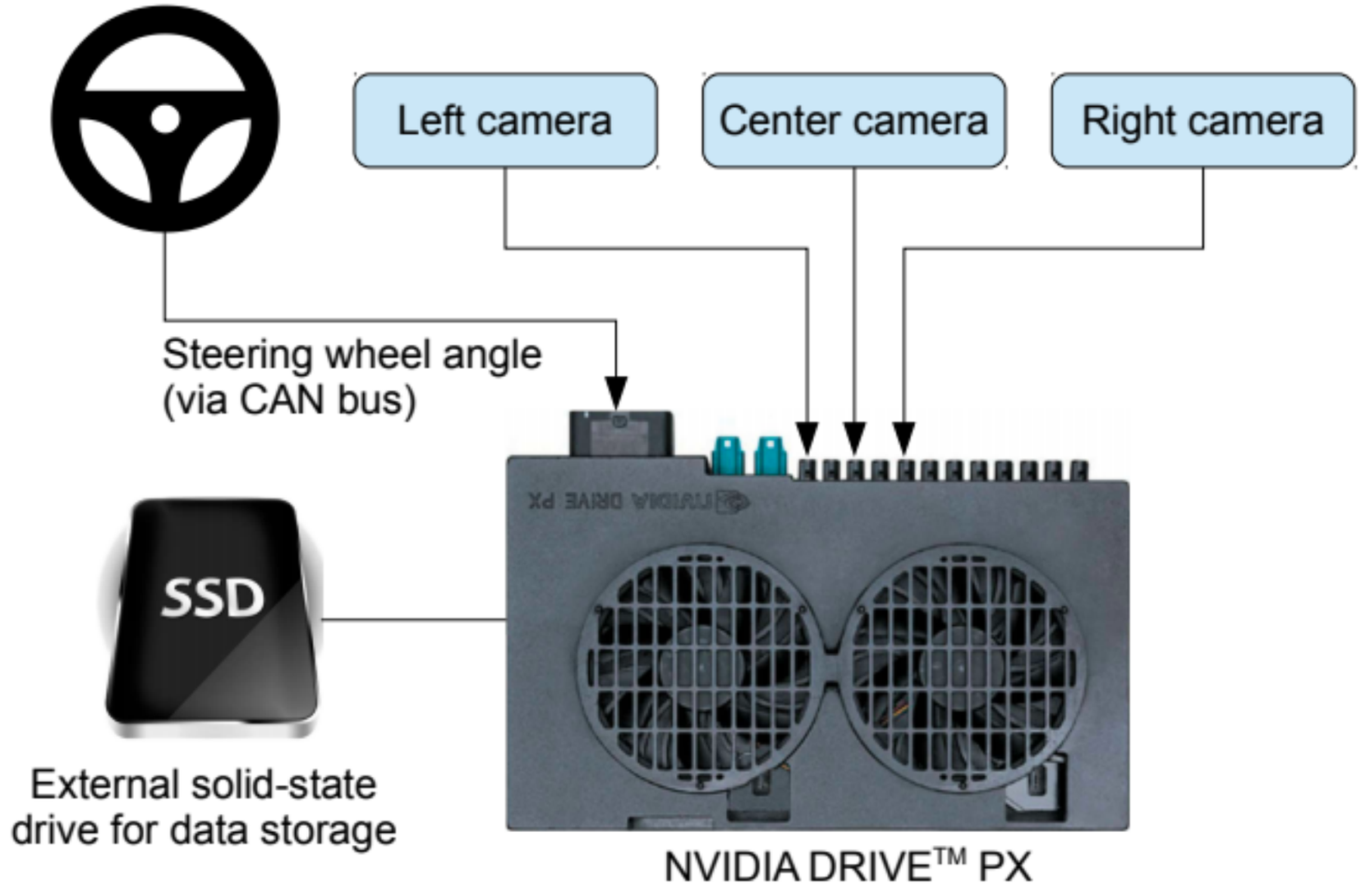
- Use of image simulator to generate images of how the road looks like when the vehicle deviates slightly from its trajectory.
- Simulating the images too longer than training the network

Solution: data augmentations

Change $P_{\pi^*}(\mathbf{o}_t)$ by augmenting the expert demonstration trajectories.
Add examples in expert demonstration trajectories to cover the states/ observations points where the agent will land when trying out its own policy. How?

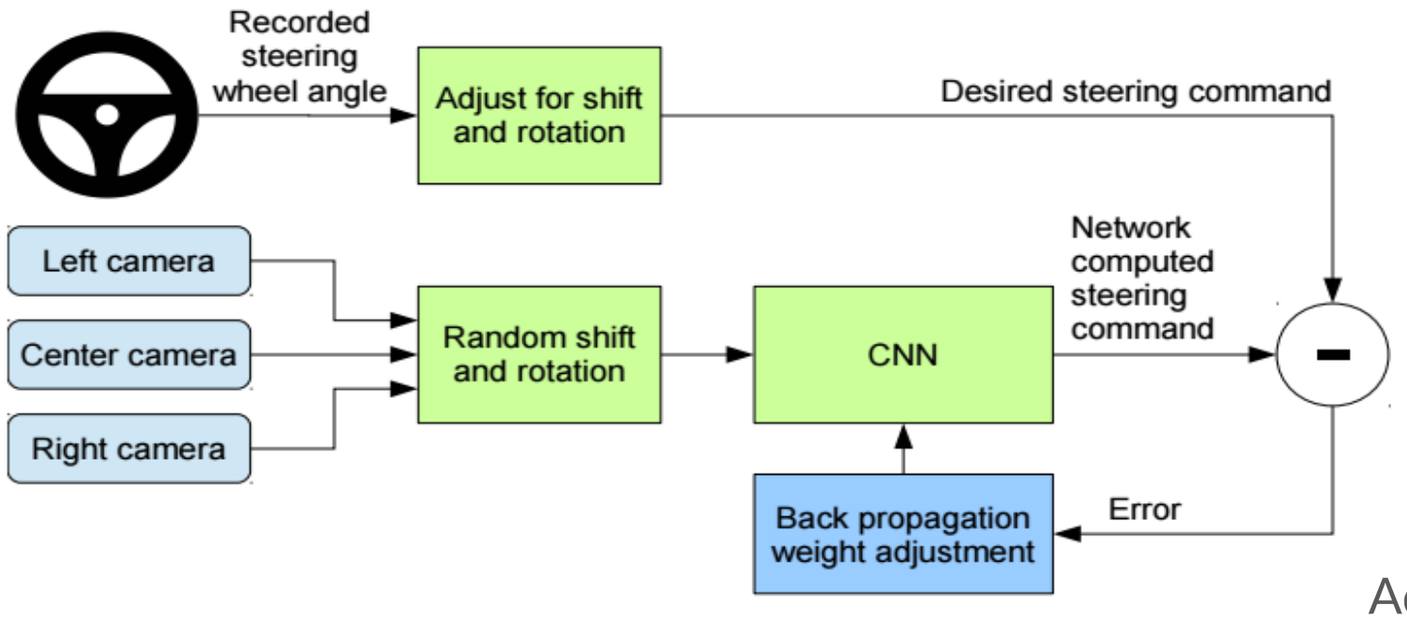
1. By generating synthetic data in simulation
2. By collecting additional data via clever hardware
3. By interactively querying the experts in additional datapoints

Learning to Drive a Car: Supervised Learning

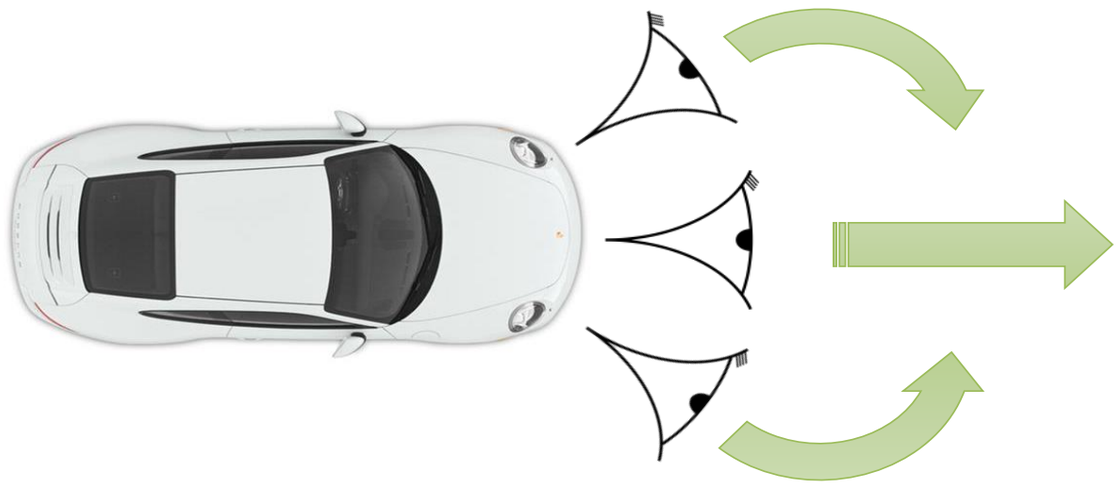


End to End Learning for Self-Driving Cars, NVIDIA, 2016

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

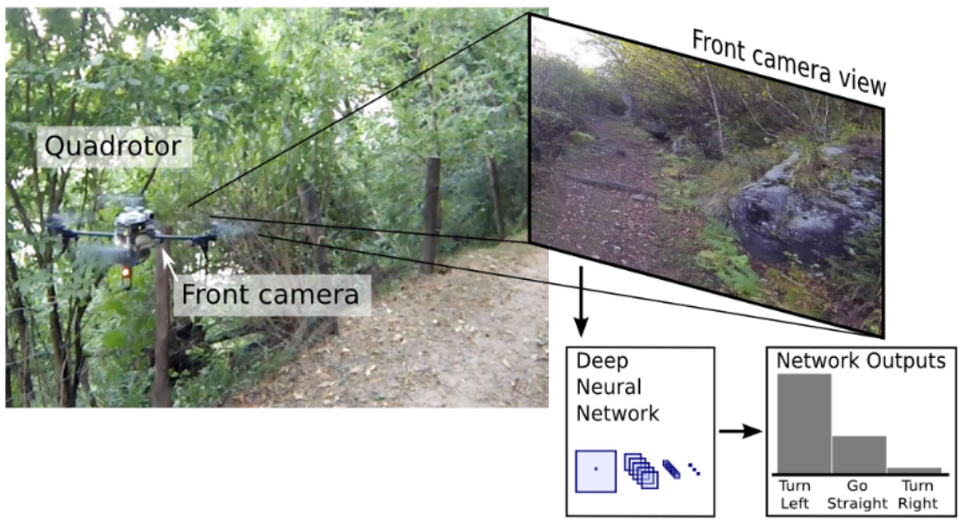
Data Augmentation (2): NVIDIA 2016

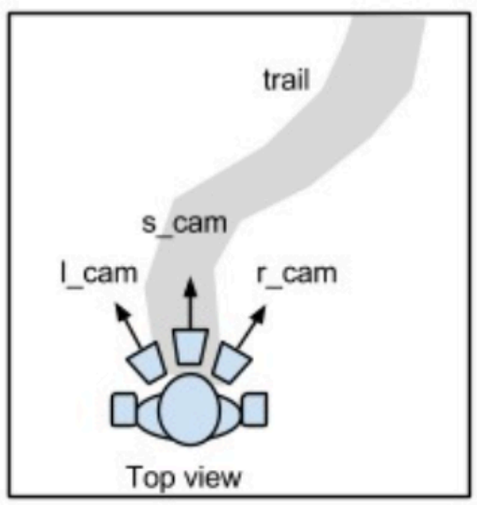
DAVE 2 Driving a Lincoln

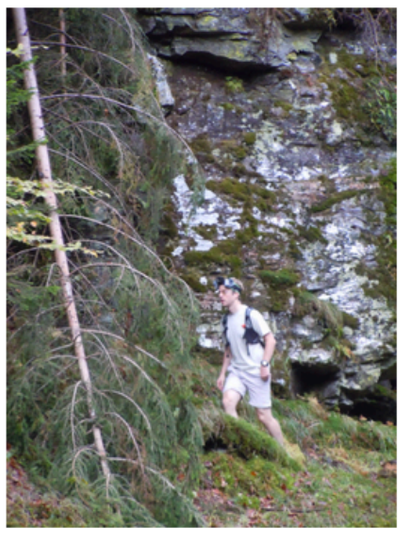
- A convolutional neural network
- Trained by human drivers
- Learns perception, path planning, and control
  "pixel in, action out"
- Front-facing camera is the only sensor

“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

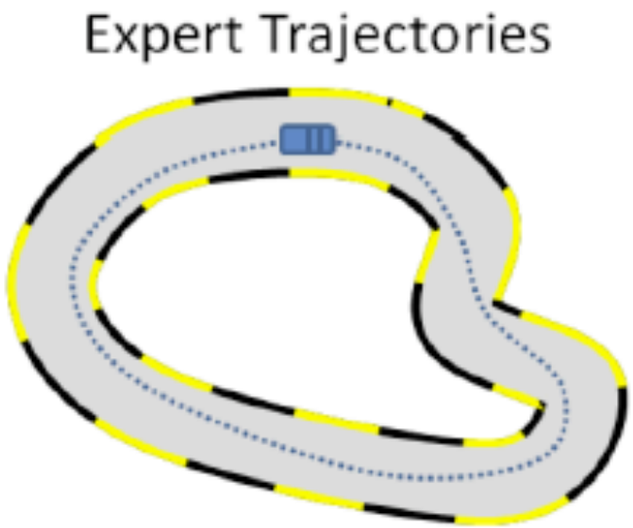
A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots. *Giusti et al.*

Solution: data augmentations

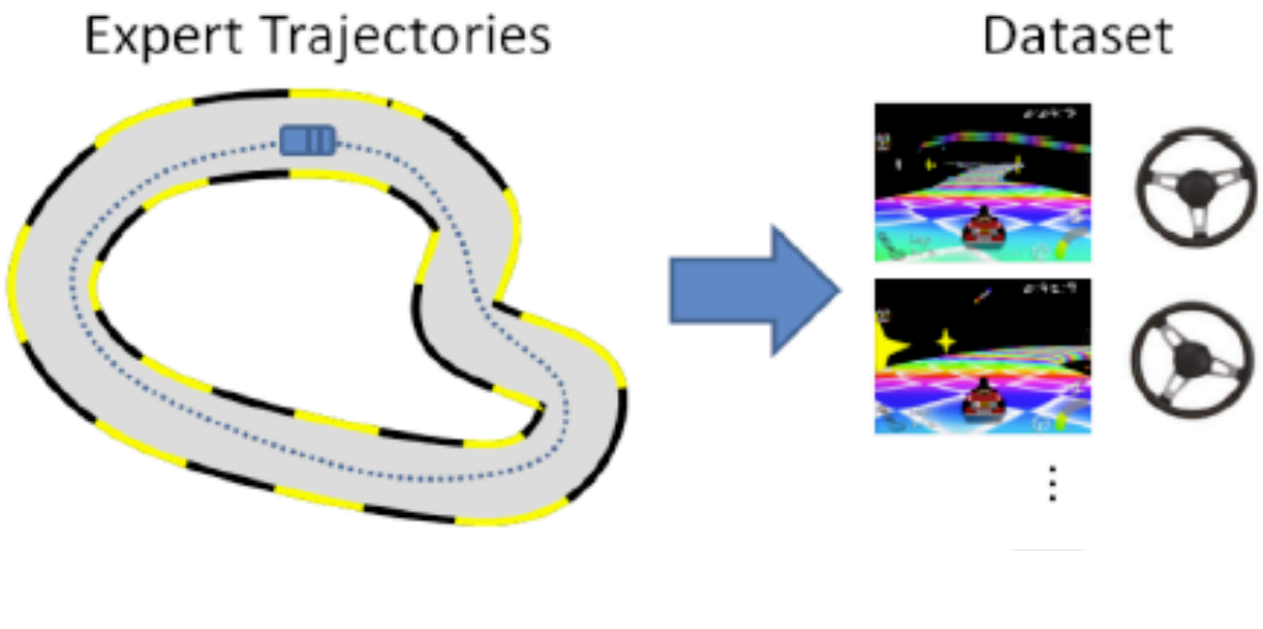
Change $P_{\pi^*}(\mathbf{o}_t)$ by augmenting the expert demonstration trajectories.
Add examples in expert demonstration trajectories to cover the states/ observations points where the agent will land when trying out its own policy. How?

1. By generating synthetic data in simulation
2. By collecting additional data via clever hardware
3. By interactively querying the experts in additional datapoints

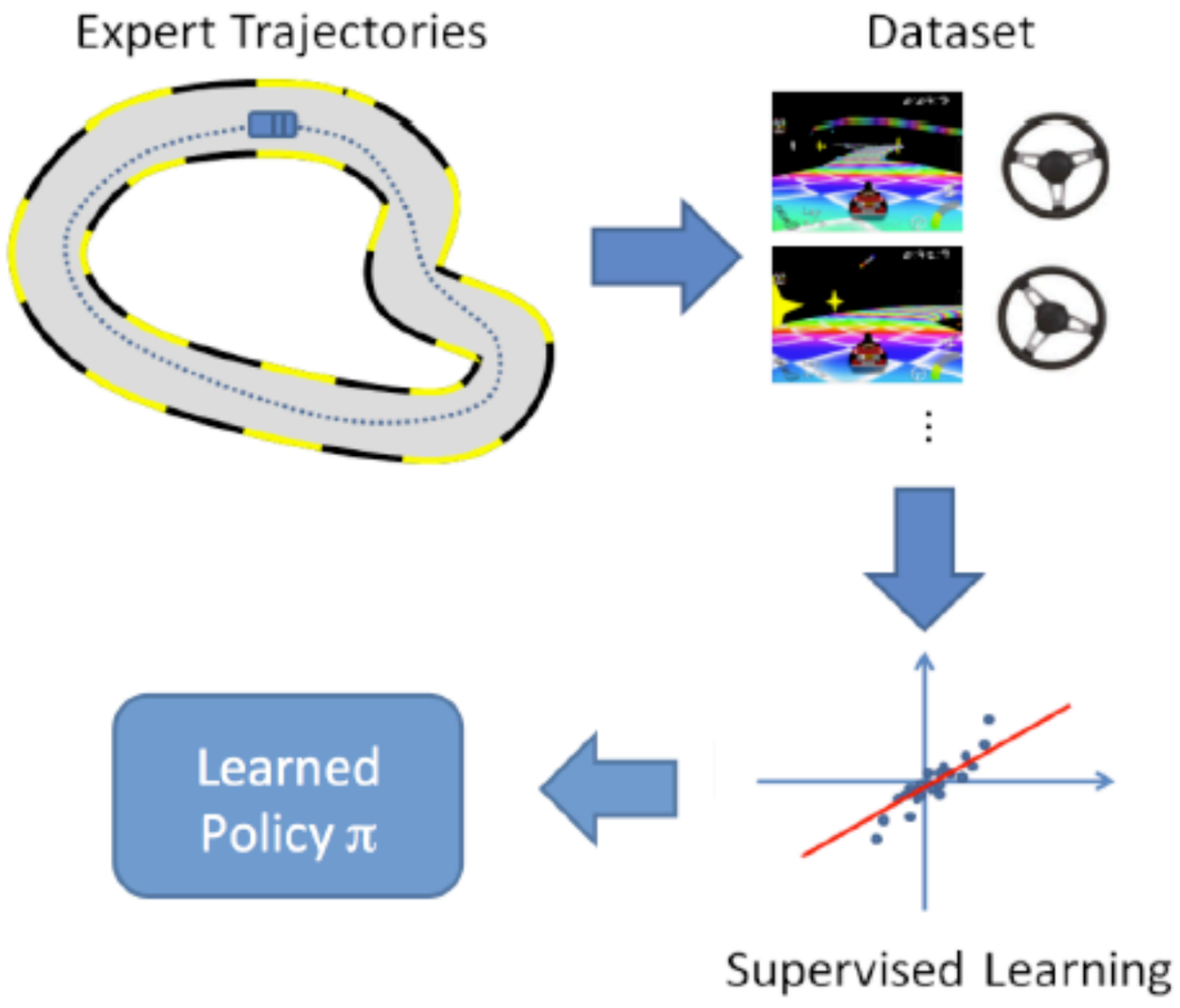
Learning to Drive a Car: Supervised Learning



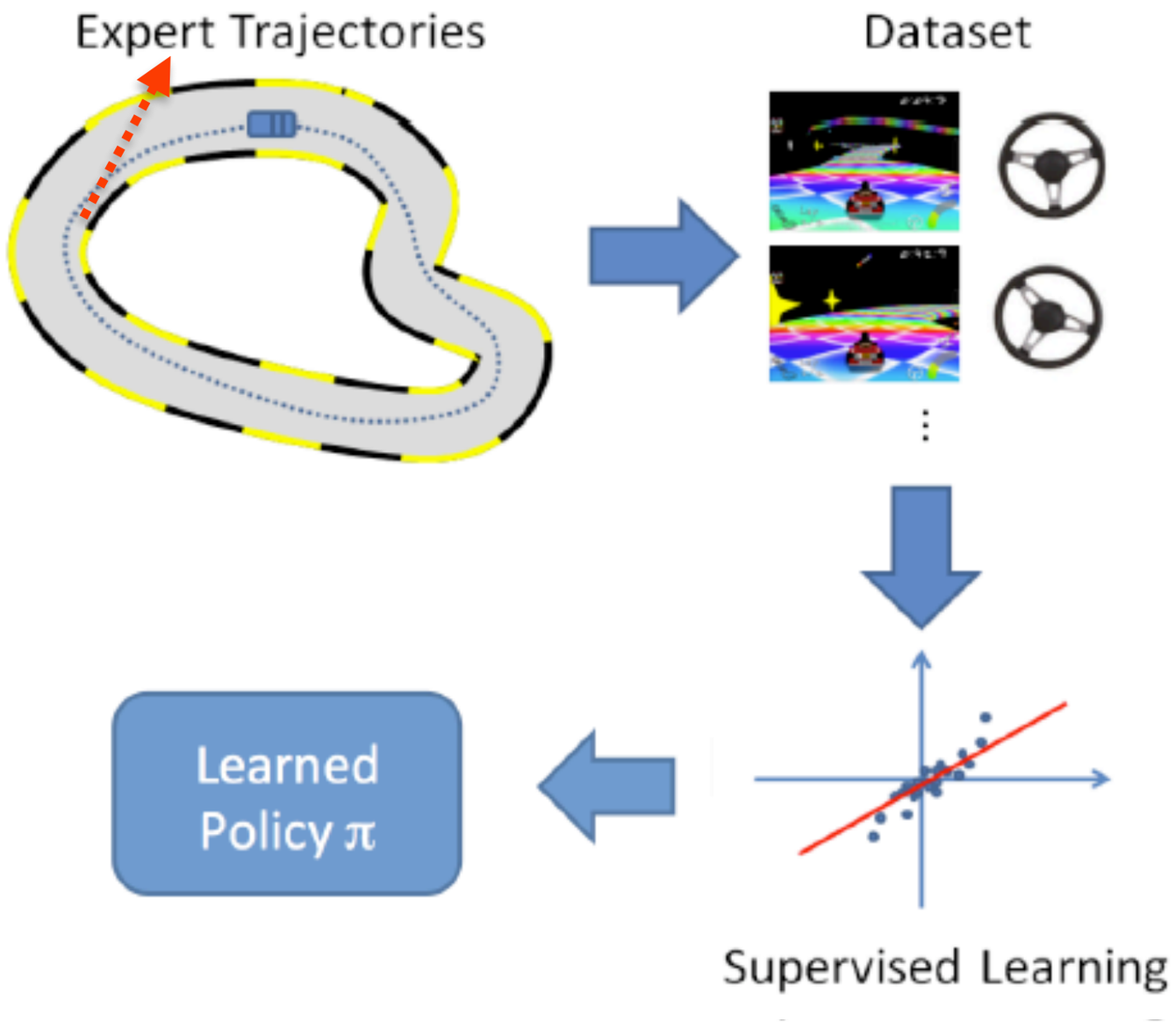
Learning to Drive a Car: Supervised Learning



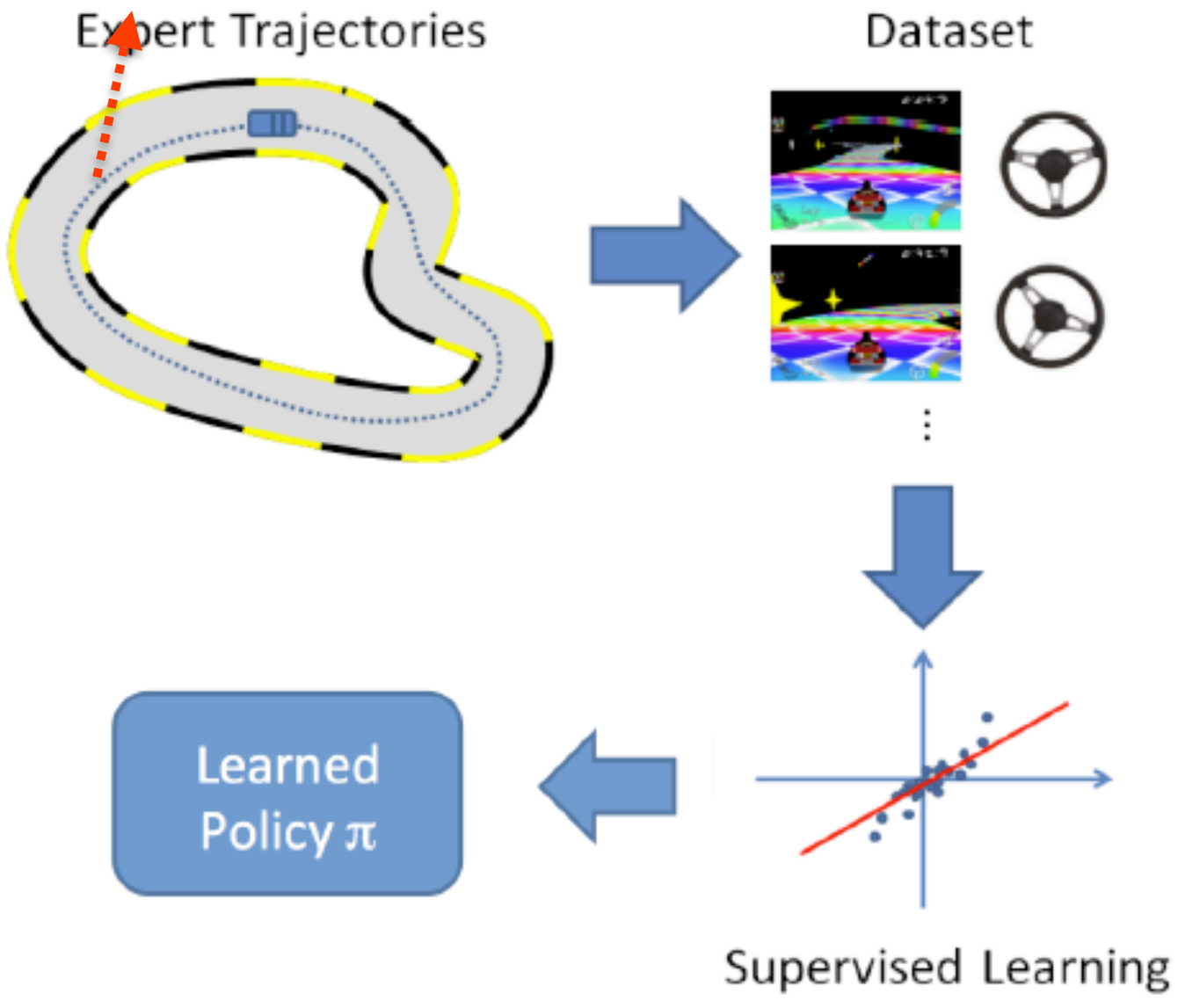
Learning to Drive a Car: Supervised Learning



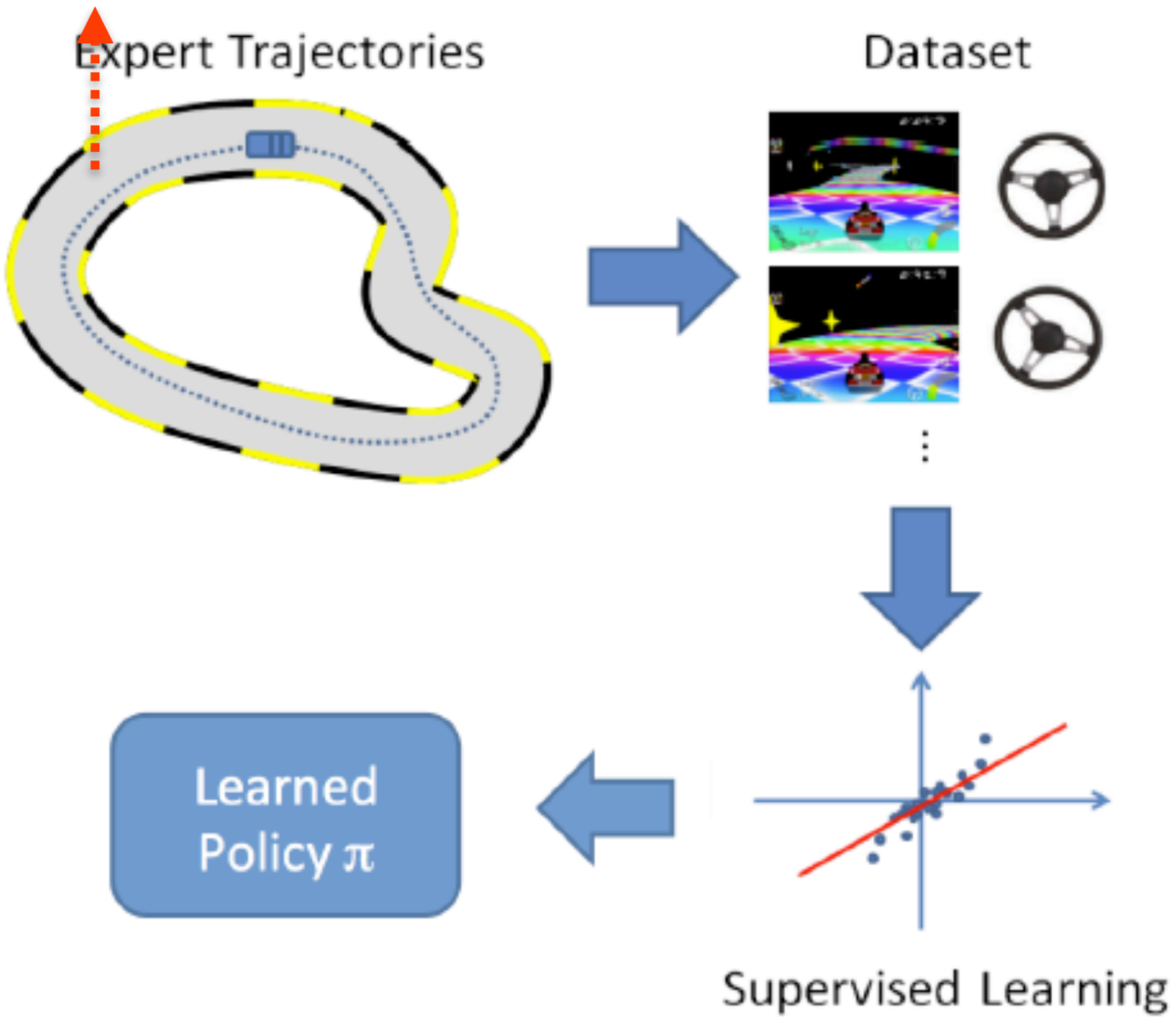
Learning to Drive a Car: Supervised Learning



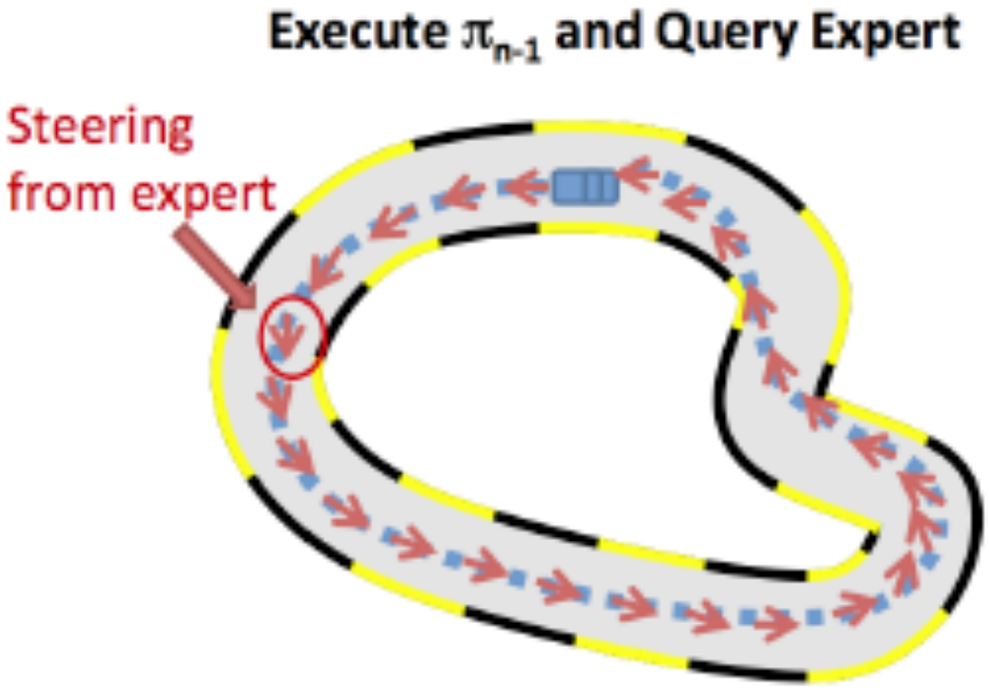
Learning to Drive a Car: Supervised Learning



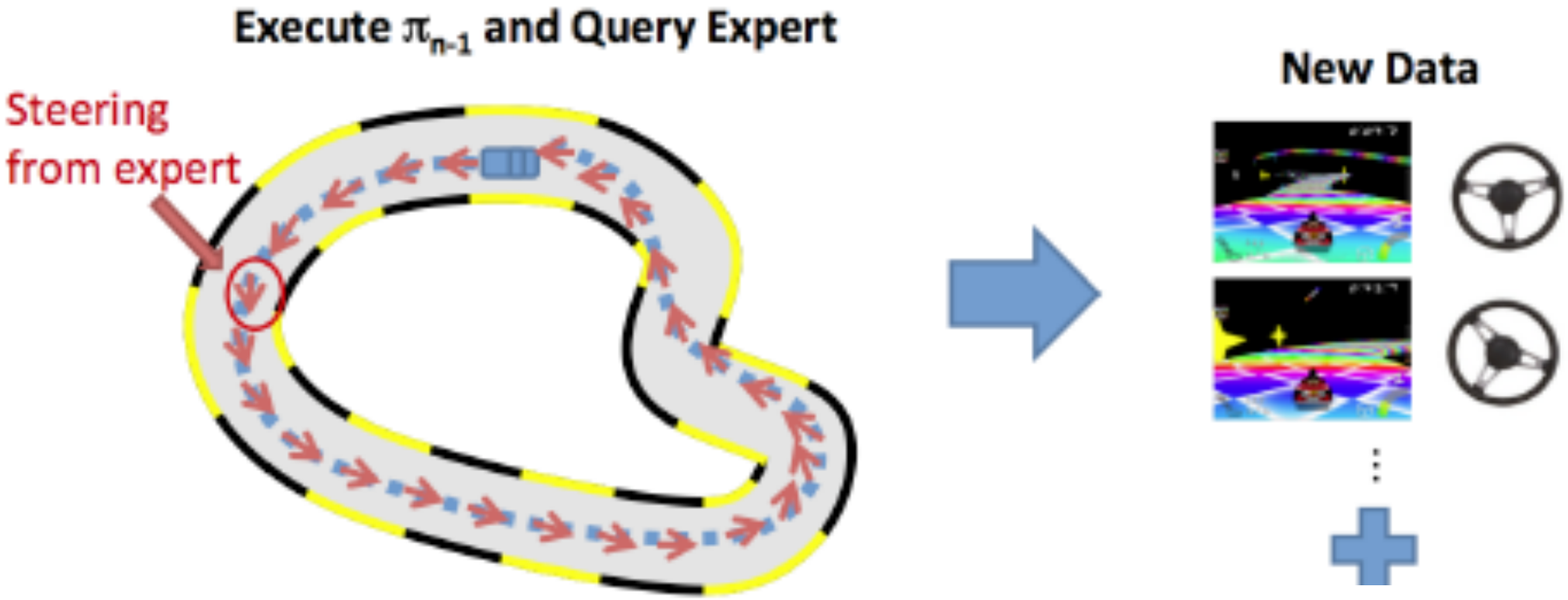
Learning to Drive a Car: Supervised Learning



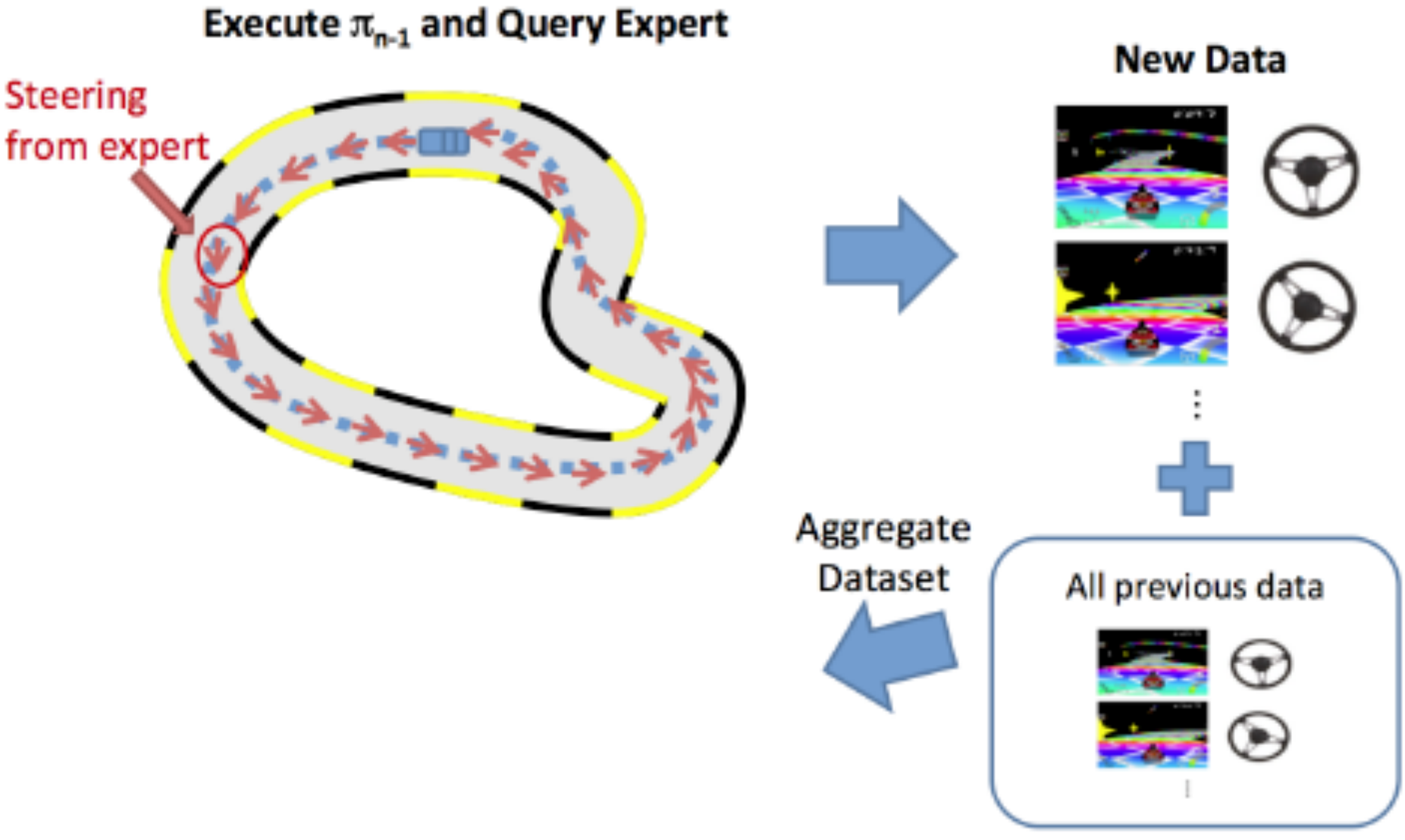
Learning to Race a Car : Interactive learning-DAGGer



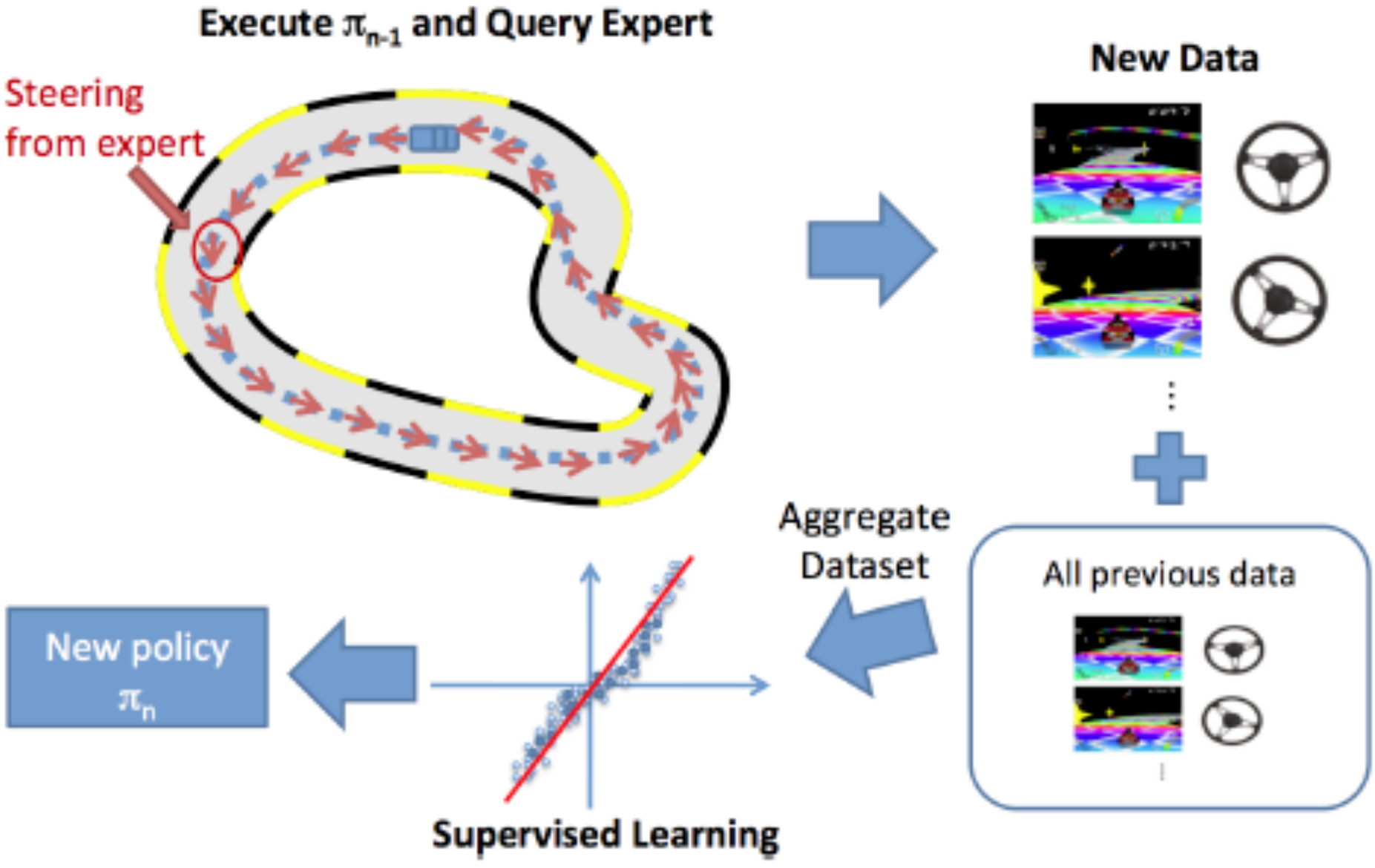
Learning to Race a Car : Interactive learning-DAGGer



Learning to Race a Car : Interactive learning-DAGGer

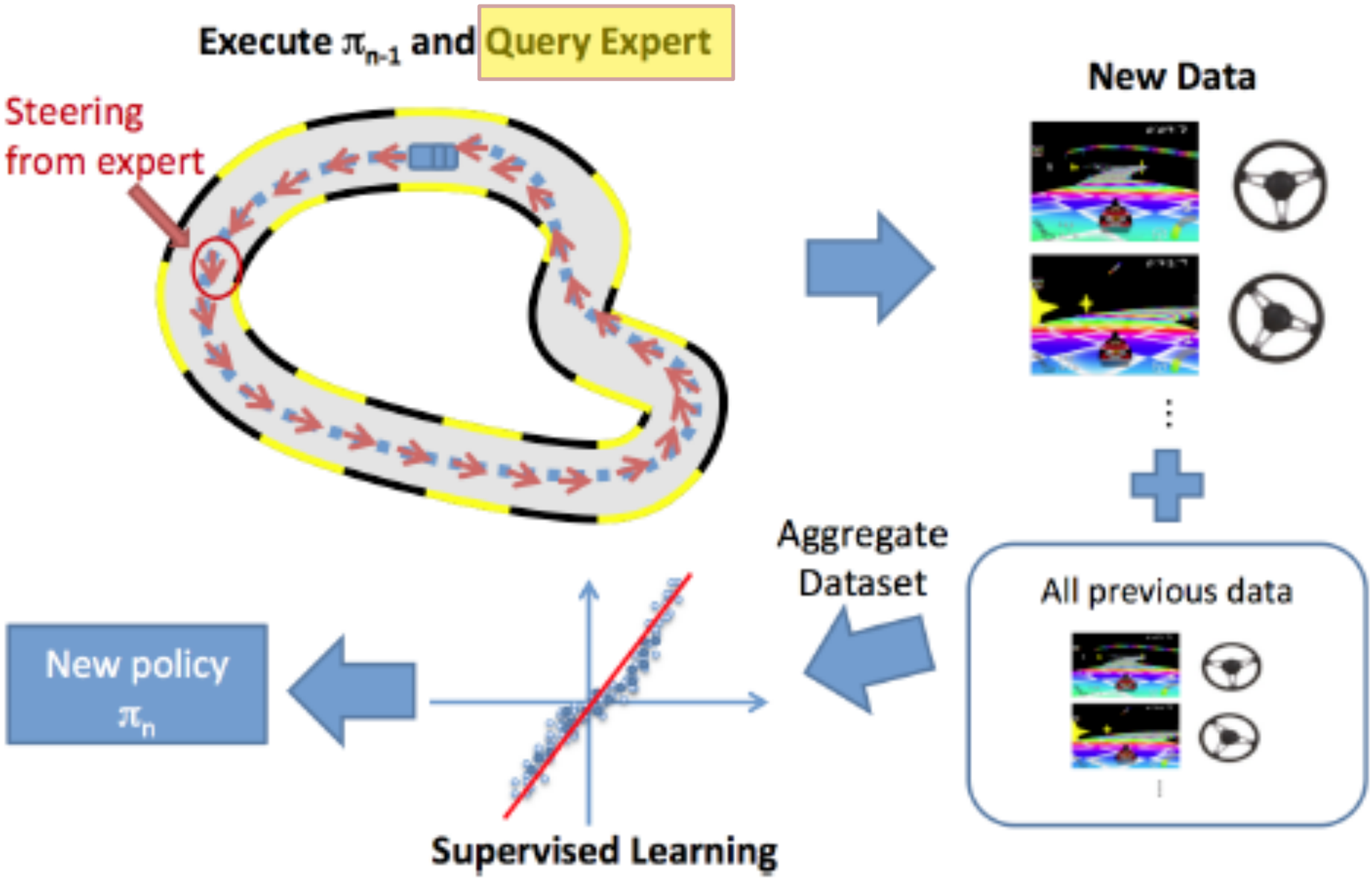


Learning to Race a Car : Interactive learning-DAGGer



Learning to Race a Car : Interactive learning-DAGGer

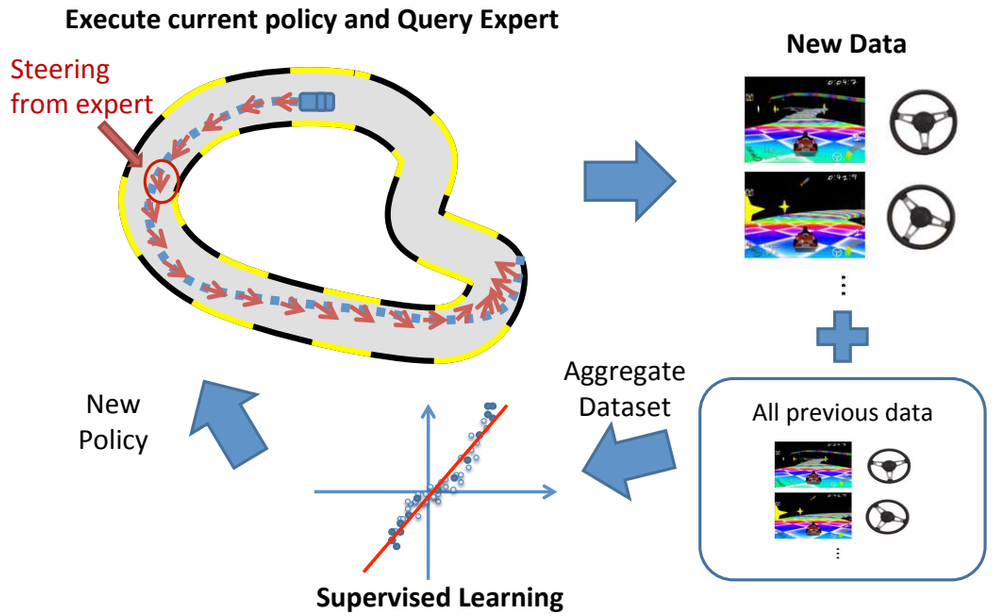
This assumes you can actively access an expert during training!



A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning
Stephane Ross, Geoffrey J. Gordon, J. Andrew Bagnell

DAGGER (in simulation)

Dataset AGGregation: bring learner's and expert's trajectory distributions closer by 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*

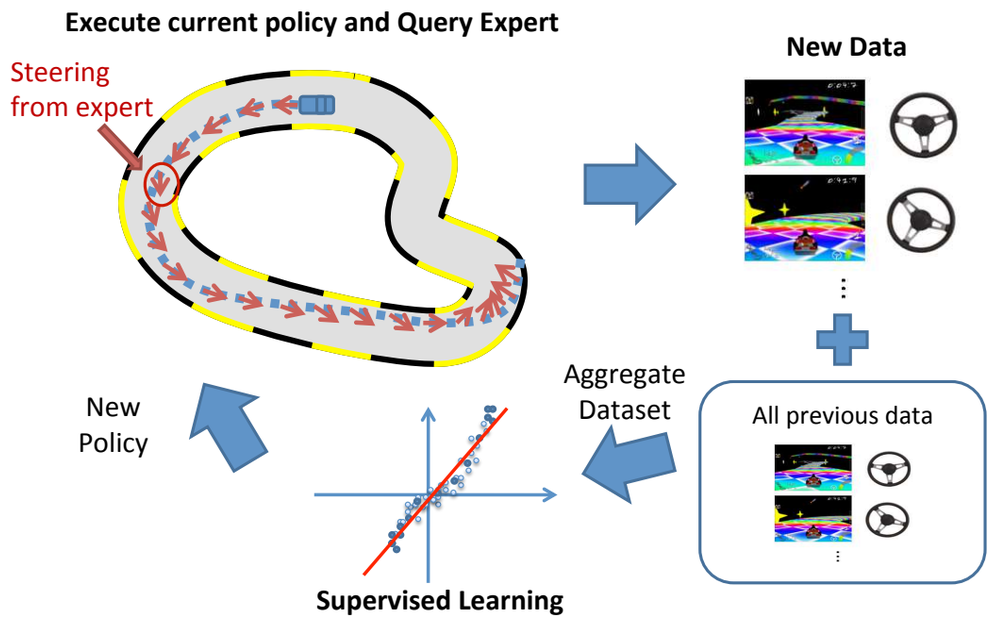
DAGGER (in simulation)

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

1. Train $\pi_\theta(u_t \mid o_t)$ from human data $\mathcal{D}_{\pi^*} = \{o_1, u_1, \ldots o_N, u_N\}$
2. Run $\pi_\theta(u_t \mid o_t)$ to get dataset $\mathcal{D}_\pi = \{o_1, \ldots, o_M\}$
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

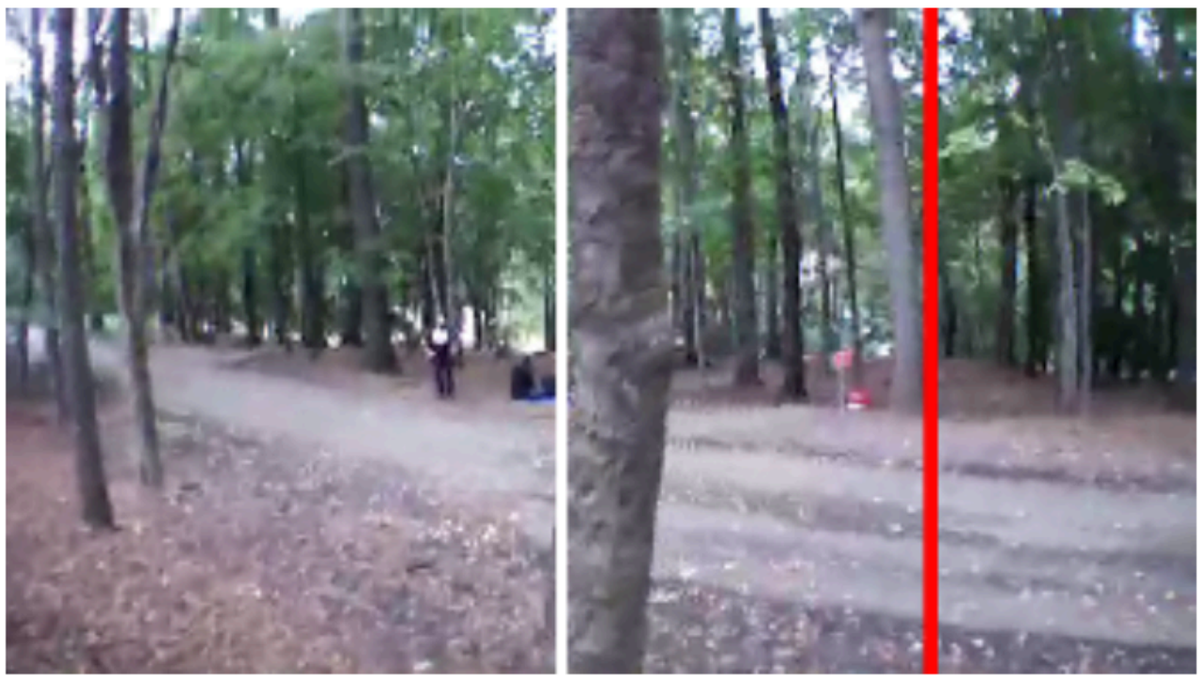
Learning monocular reactive UAV control in cluttered natural environments, Ross et al. 2013

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.

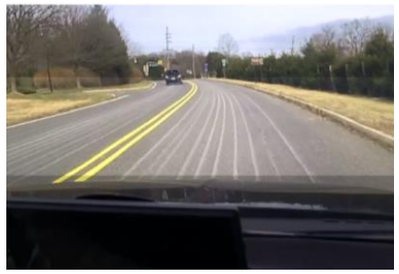
3. Executing an imperfect policy causes accidents, crashes into obstacles. Solution: safety measures which make again the data distribution matching imperfect between train and test, but good enough.

- 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.

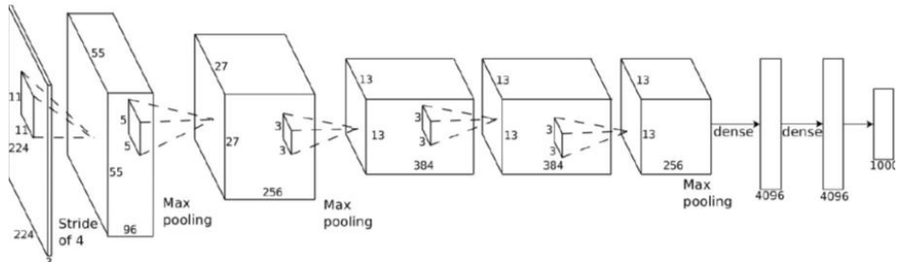
Learning monocular reactive UAV control in cluttered natural environments, Ross et al. 2013

What can go wrong?

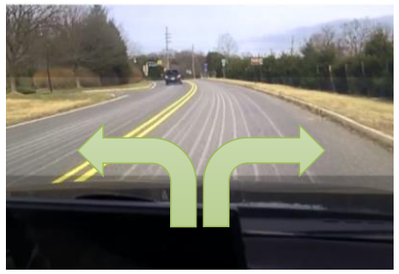
- Compounding errors
  Fix: data augmentation
- Non-markovian observations
  Fix: observation concatenation or recurrent models
- Stochastic expert actions
  Fix: generative modelling (GANs,stochastic latent variable models, action discretisation, gaussian mixture networks)



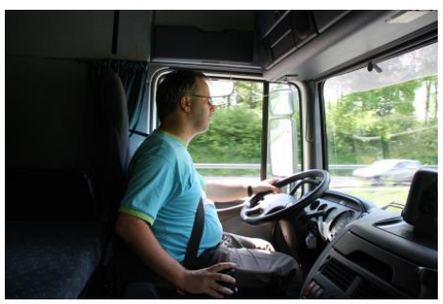
$\mathbf{o}_t$



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



$\mathbf{u}_t$



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

training data

supervised learning

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

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

Markov property

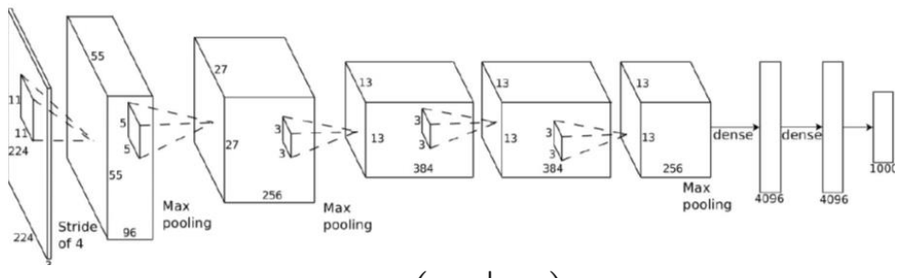
A stochastic process has the Markov property if the conditional probability distribution of future states of the process (conditional on both past and present states) depends only upon the present state, not on the sequence of events that preceded it

$$\mathbb{P}[R_{t+1} = r, S_{t+1} = s' | S_0, A_0, R_1, ..., S_{t-1}, A_{t-1}, R_t, S_t, A_t] = \mathbb{P}[R_{t+1} = r, S_{t+1} = s' | S_t, A_t]$$

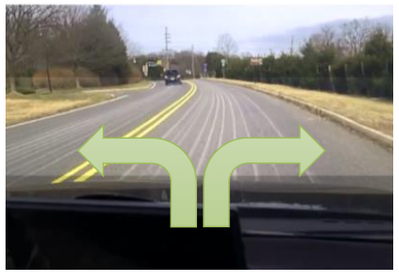
for all $s' \in \mathcal{S}, r \in \mathcal{R}$ and all histories

Non-markovian observations

$\mathbf{o}_t$



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



$\mathbf{u}_t$

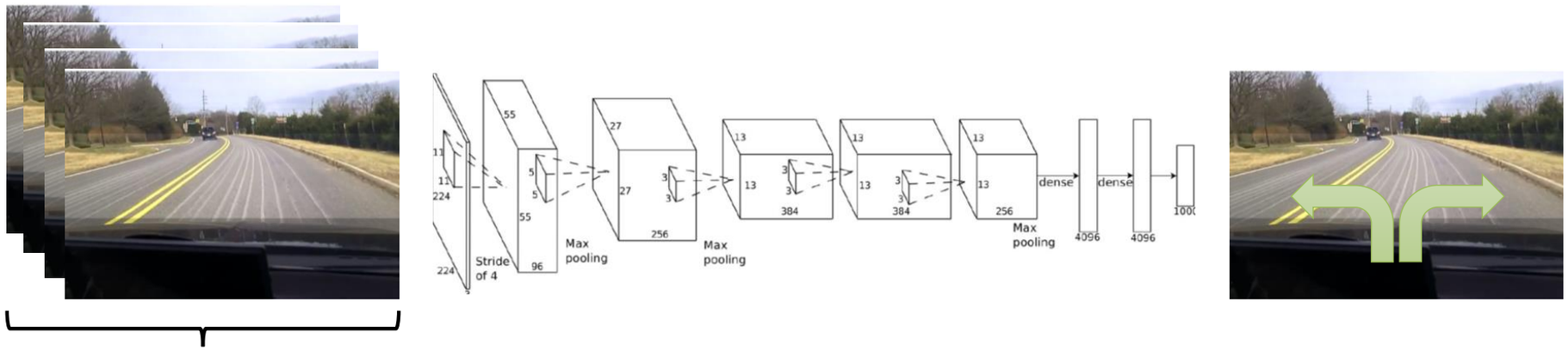
$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$

behavior depends only on current observation

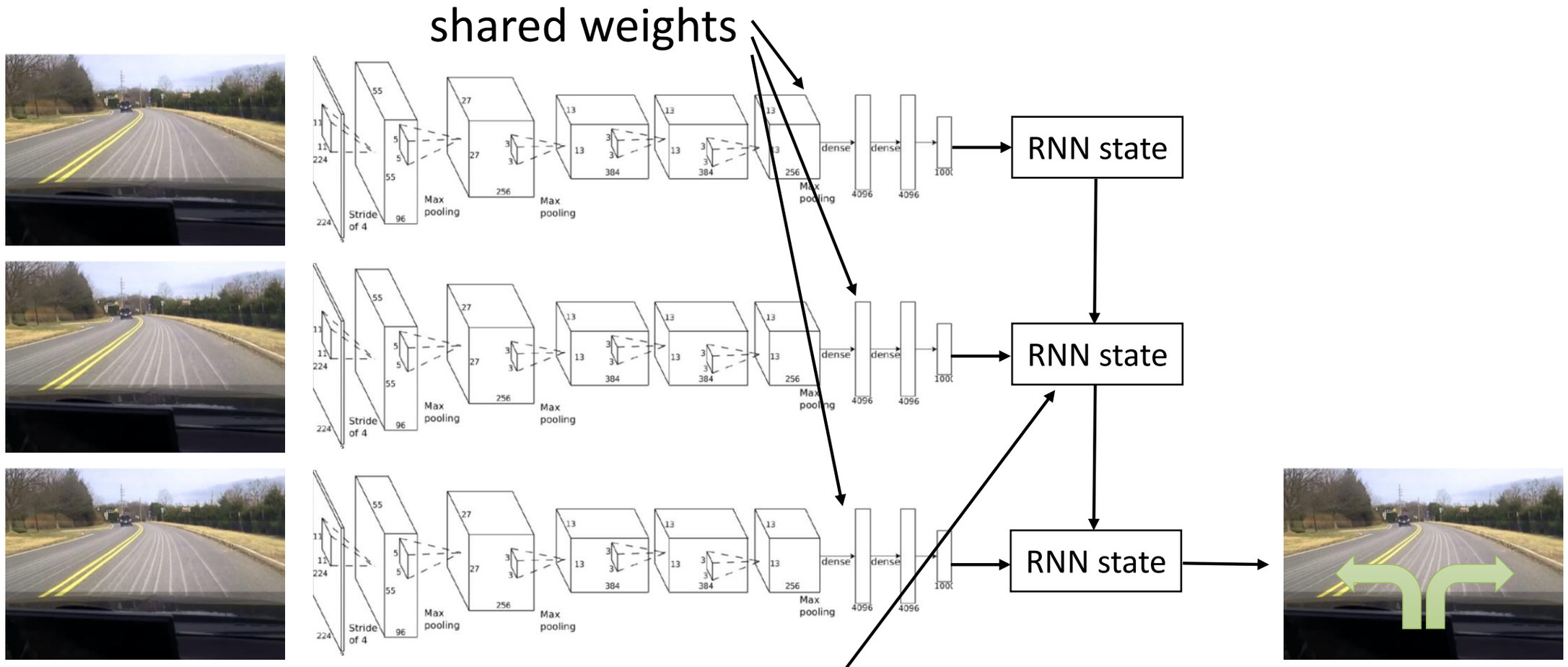
$\pi_\theta(\mathbf{u}_t|\mathbf{o}_1, \ldots, \mathbf{o}_t)$

behavior depends on all past observations

Fix 1: concatenate observations



Fix 2: use (vanilla) 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.
- Later lecture we will explain why vanilla LSTMs

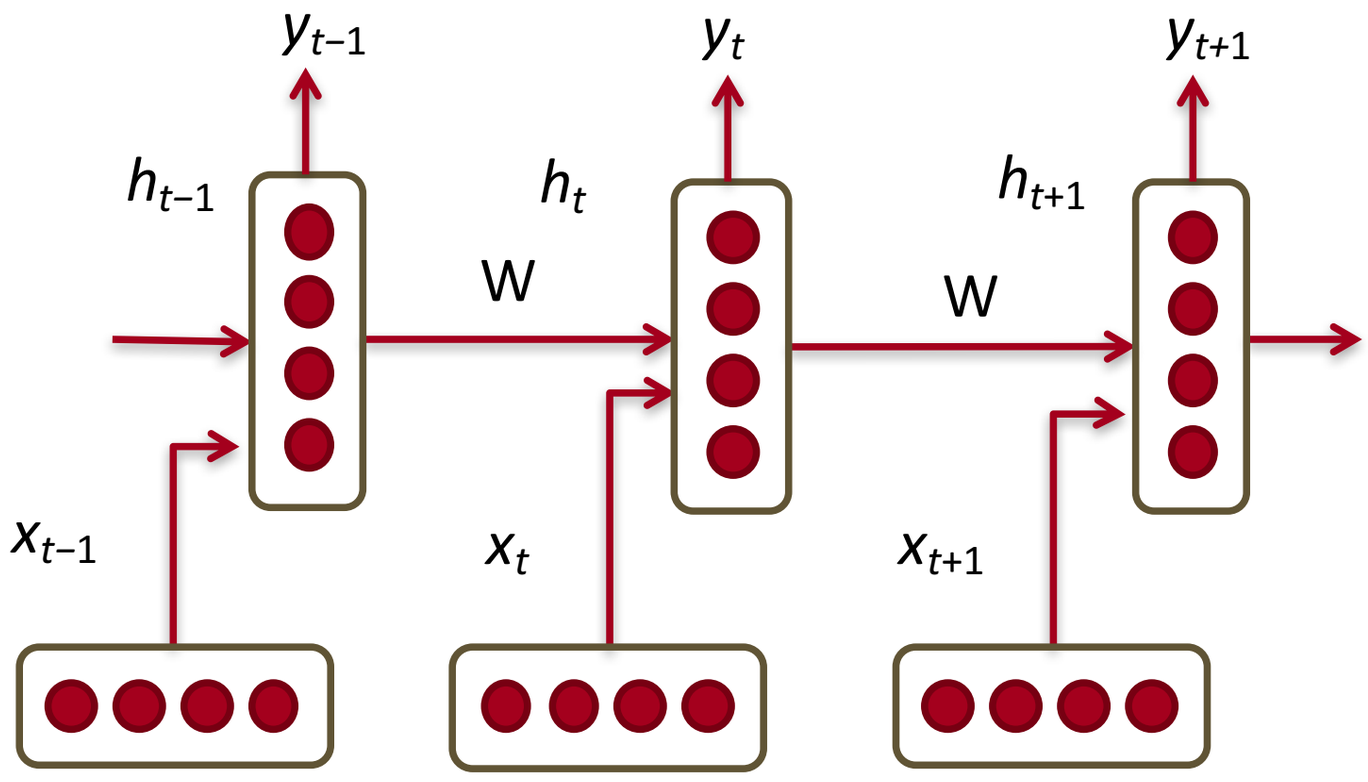


Diagram from Richard Socher

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)

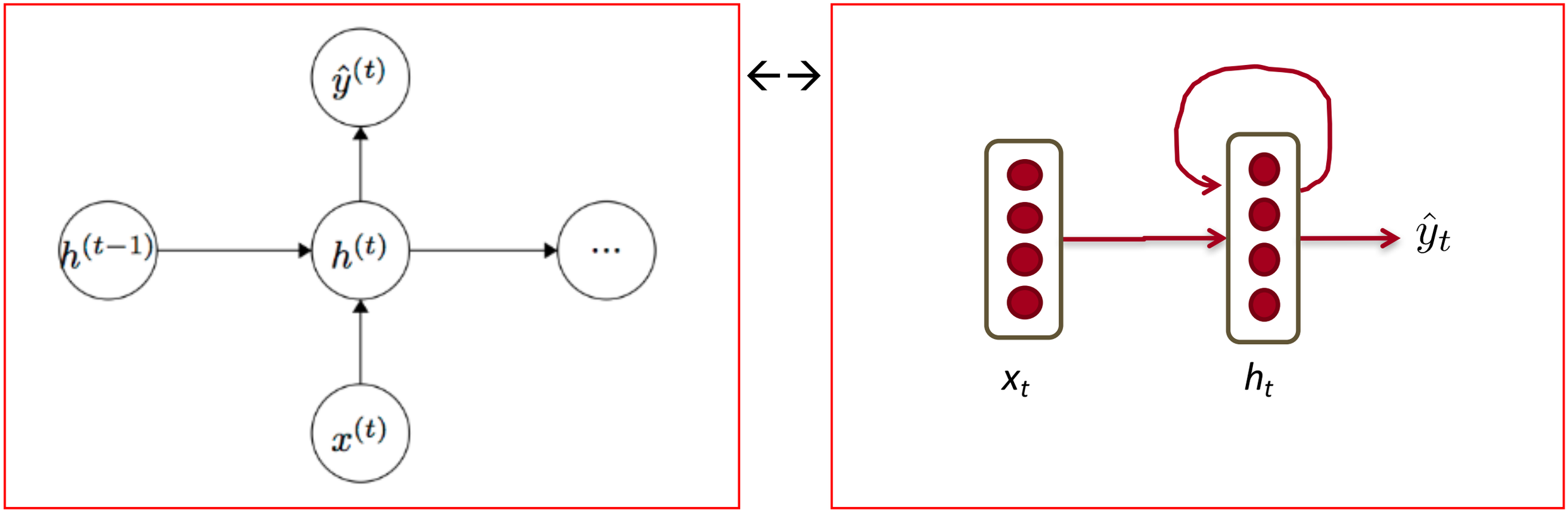


Diagram from Richard Socher

Fix 2: use recurrent networks

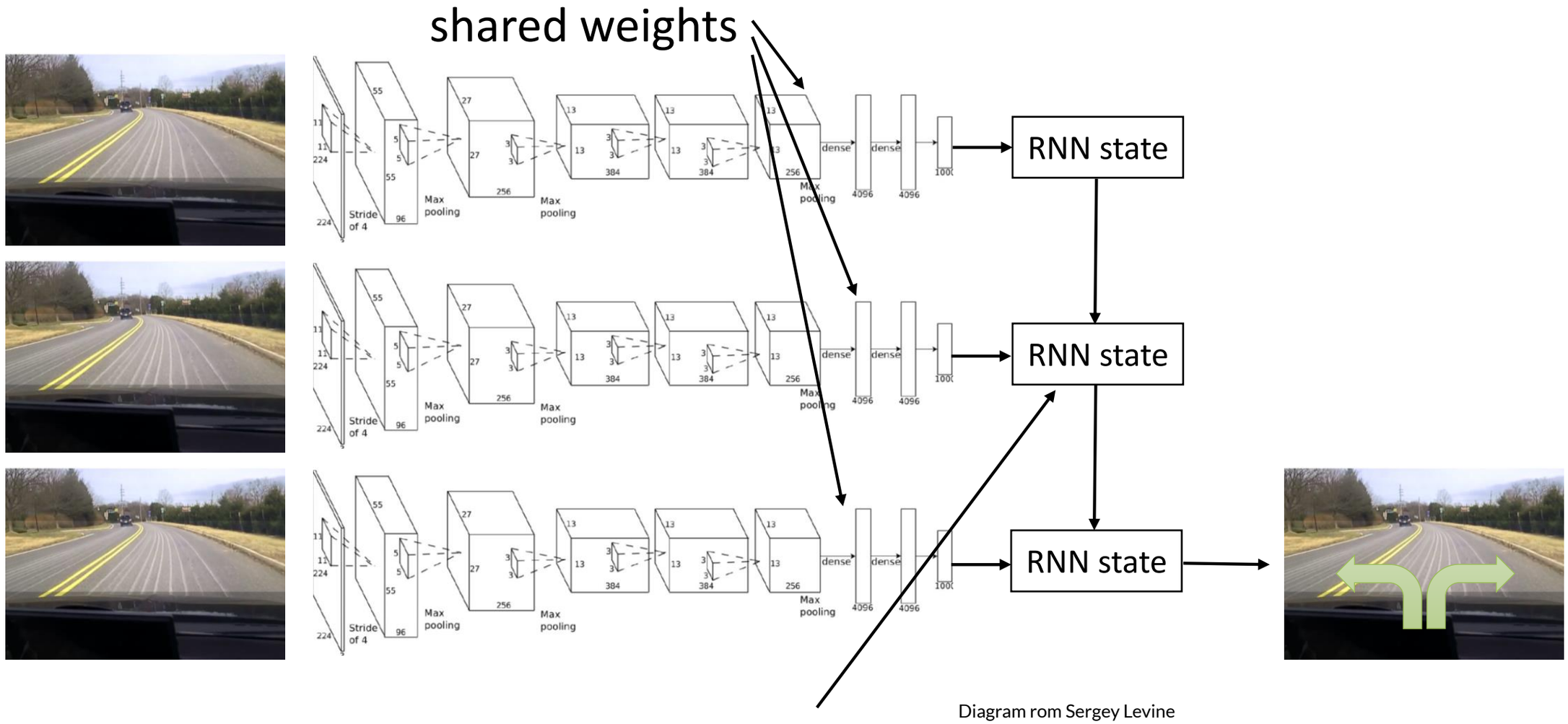


Diagram rom Sergey Levine

Fix 2: use recurrent networks

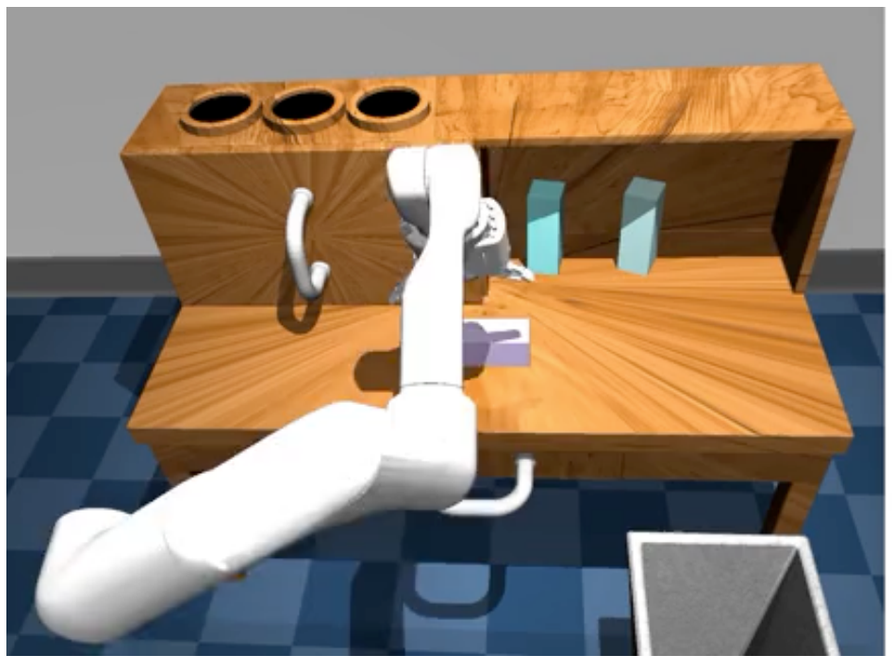
- Usually much more structure is needed in the latent state than a vanilla LSTM can provide, e.g., detections and trackless of objects.
- We will discuss later in the course structured recurrent neural networks for video perception.

Generalized policies

- Often times we want to achieve many related goals instead of one goal.
- For example push object A to location $(10,10,10)$ and to location $(10,12,10)$.
- The two pushing policies should have many things in common.
- Training such policies jointly is beneficial!
- This is just an instantiation of multi-task learning: training deep representations for multiple tasks, is often beneficial for every task.

$$\pi(s;\theta) \quad \Rightarrow \quad \pi(s, g;\theta) \quad s, g \in \mathcal{S}$$

Learning generalized policies from imitation



2.5x
speedup

Teleoperated trajectories:

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

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

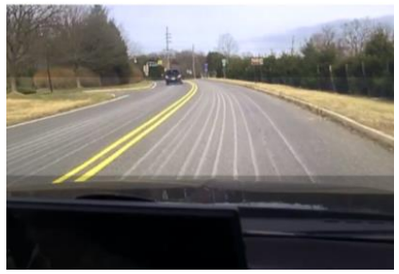
$o_1^3, u_1^3, o_2^3, u_2^3, o_3^3, u_3^3, \ldots.$

Imagine I collect lots of teleoperated data and I want to learn a set of generalized policies from them.

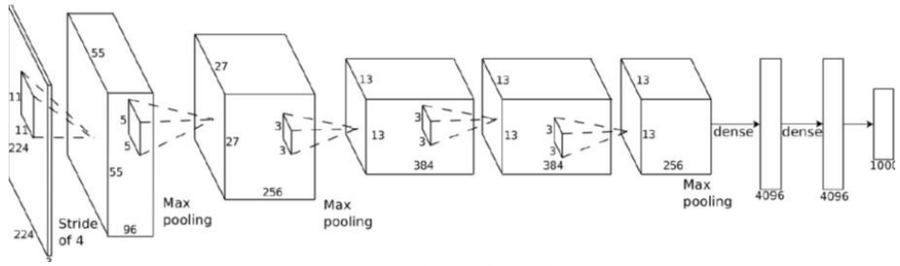
I take the collected trajectories and pick some as state s_{t_1} and s_{t_1+k} as goal.

What can go wrong?

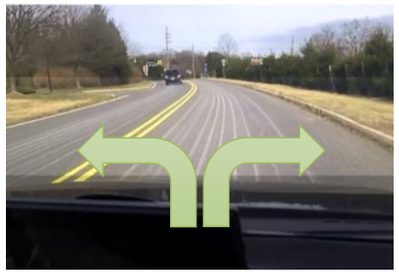
- Compounding errors
  Fix: data augmentation
- Non-markovian observations
  Fix: observation concatenation or recurrent models
- Multimodality of expert actions
  Fix: action discretization, Gaussian mixture networks, latent variable models



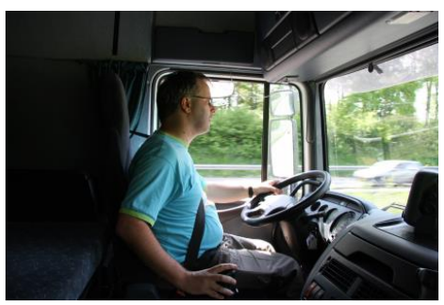
$\mathbf{o}_t$



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

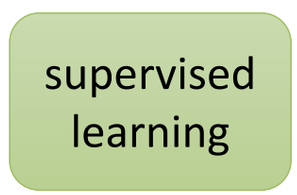


$\mathbf{u}_t$



$\mathbf{o}_t$

$\mathbf{u}_t$



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

Multimodality in expert actions

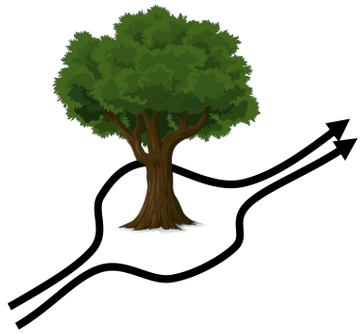
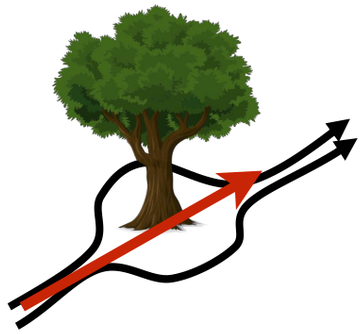


Diagram from Sergey Levine

Regression fails under multimodality

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



predicted streering angles

Multimodality in expert actions: Fixes

Going beyond regression:

- Discretize the action space and train a classifier that predicts a categorical distribution over the discrete action space.

- 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)

- Variational autoencoders

- Generative adversarial networks

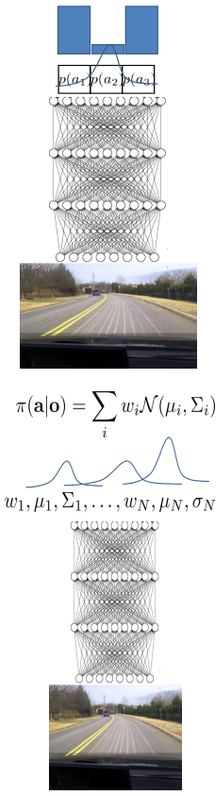


Diagram from Sergey Levine

Discretizing the action space

Problem with brute force discretization:

- Often times the action space contains many degrees of freedom, e.g. rotation and translation of the gripper in all 3 dimensions (6DoF).
- Discretizing the 6 DoF action space and producing a single categorical distribution causes training example sparsity.
- Consider two training examples,
  $\{(\mathbf{s}_1, \mathbf{a}_1 = (5,3,7,10^o,30^o,40^o)), (\mathbf{s}_2, \mathbf{a}_2 = (5,3,7,10^o,30^o,50^o))\}$
  Though the two rotations share the same value in the first 5 dinensions, our discretization causes the two action values to fall in two different bins. Binning causes the distance between the continuous values to be lost.
- What if we discretize each dimension instead, and sample from 6 categorical distributions to produce our action sample?
- Independent sampling of related random variables, such as our action dimensions, will result in wrong action samples.
- Solutions?

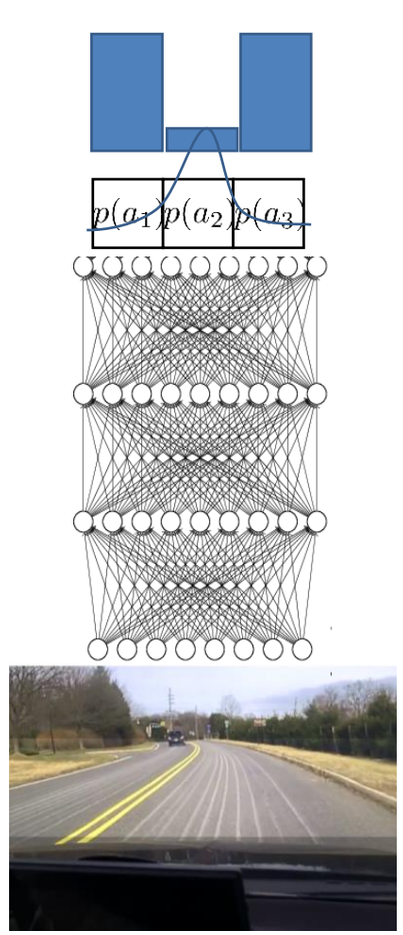


Diagram from Sergey Levine

Autoregressive action sampling

Problem with brute force discretization:

- Often times the action space contains many degrees of freedom, e.g. rotation and translation of the gripper in all 3 dimensions (6DoF).
- Discretizing the 6 DoF action space and producing a single categorical distribution causes training example sparsity.
- Consider two training examples,
  $\{(\mathbf{s}_1, \mathbf{a}_1 = (5,3,7,10^o,30^o,40^o)), (\mathbf{s}_2, \mathbf{a}_2 = (5,3,7,10^o,30^o,50^o))\}$
  Though the two rotations share the same value in the first 5 dinensions, our discretization causes the two action values to fall in two different bins. Binning causes the distance between the continuous values to be lost.
- What if we discretize each dimension instead, and sample from 6 categorical distributions to produce our action sample?
- Independent sampling of related random variables, such as our action dimensions, will result in wrong action samples.
- Solutions?

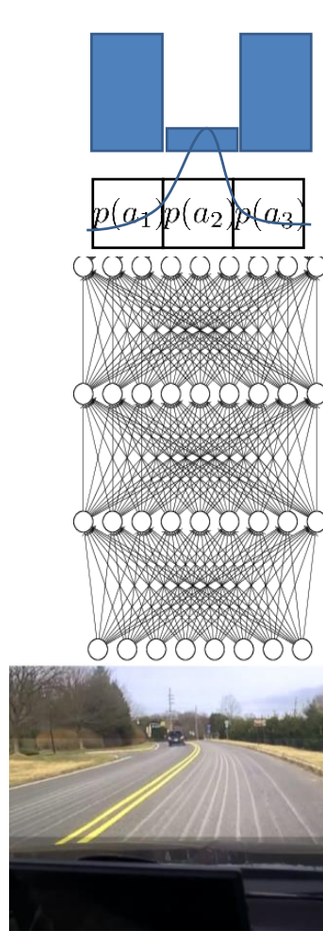


Diagram from Sergey Levine

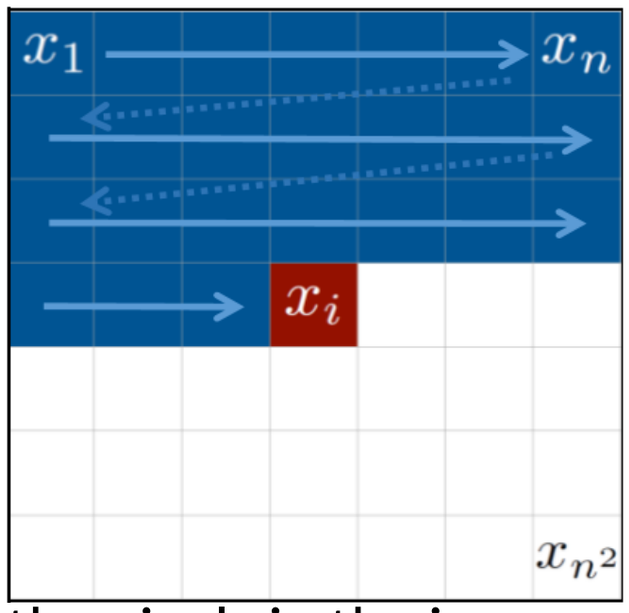
Autoregressive action sampling

Intuition: $$P(\mathbf{a_t}) = P(a_t^1, a_t^2, a_t^3, a_t^4, a_t^5, a_t^6)$$

Bayes theorem: $$P(\mathbf{a}_t) = \prod_{i=1}^{6} P(a_t^i \mid a_t^1 \ldots a_t^{i-1})$$

Define an ordering of action dimensions and sample them sequenctionally while conditioning on earlier sampled action dimensions.

For imitation learning: $$P(\mathbf{a}_t \mid s_t; \theta) = \prod_{i=1}^{6} P(a_t^i \mid a_t^1 \ldots a_t^{i-1}, s_t; \theta)$$



Pixel RNN: a neural networks that sequentially predicts the pixels in the image

Multinomial Distribution for Pixel Value

- Treat pixels as discrete variables:
  - To estimate a pixel value, do classification in every channel (256 classes indicating pixel values 0-255)
  - Implemented with a final softmax layer

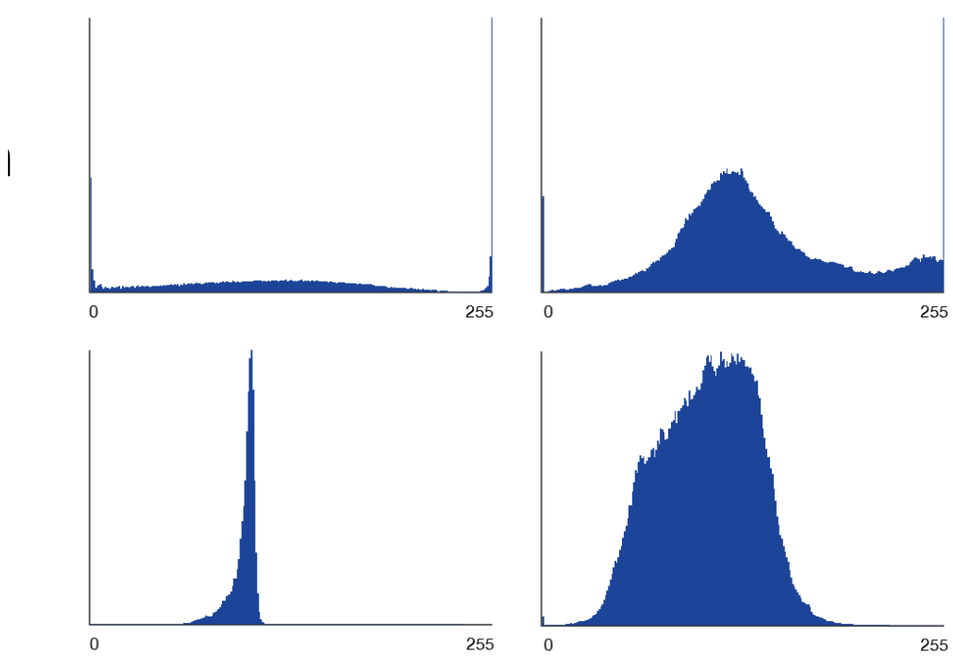
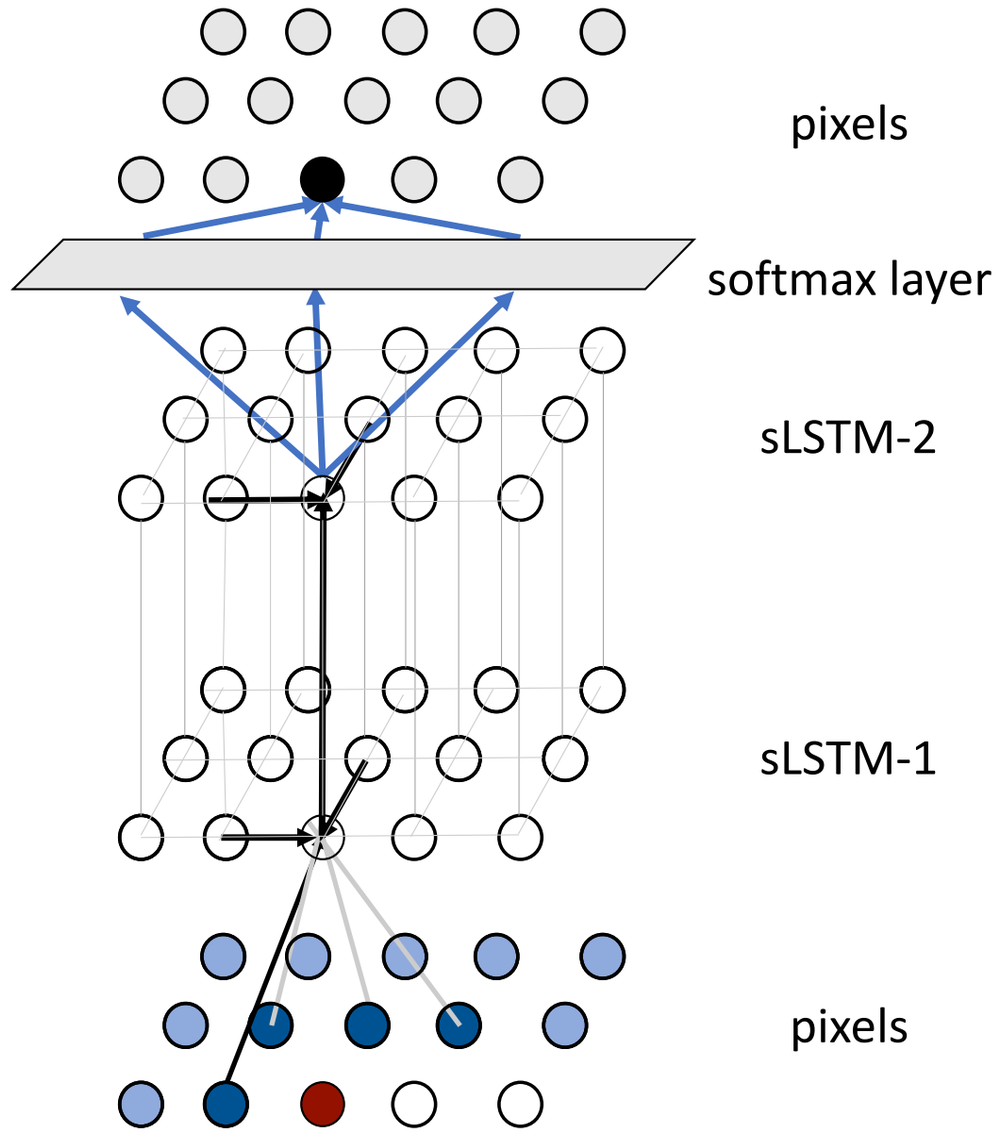


Figure: Example softmax outputs in the final layer, representing probability distribution over 256 classes.

Spatial LSTM



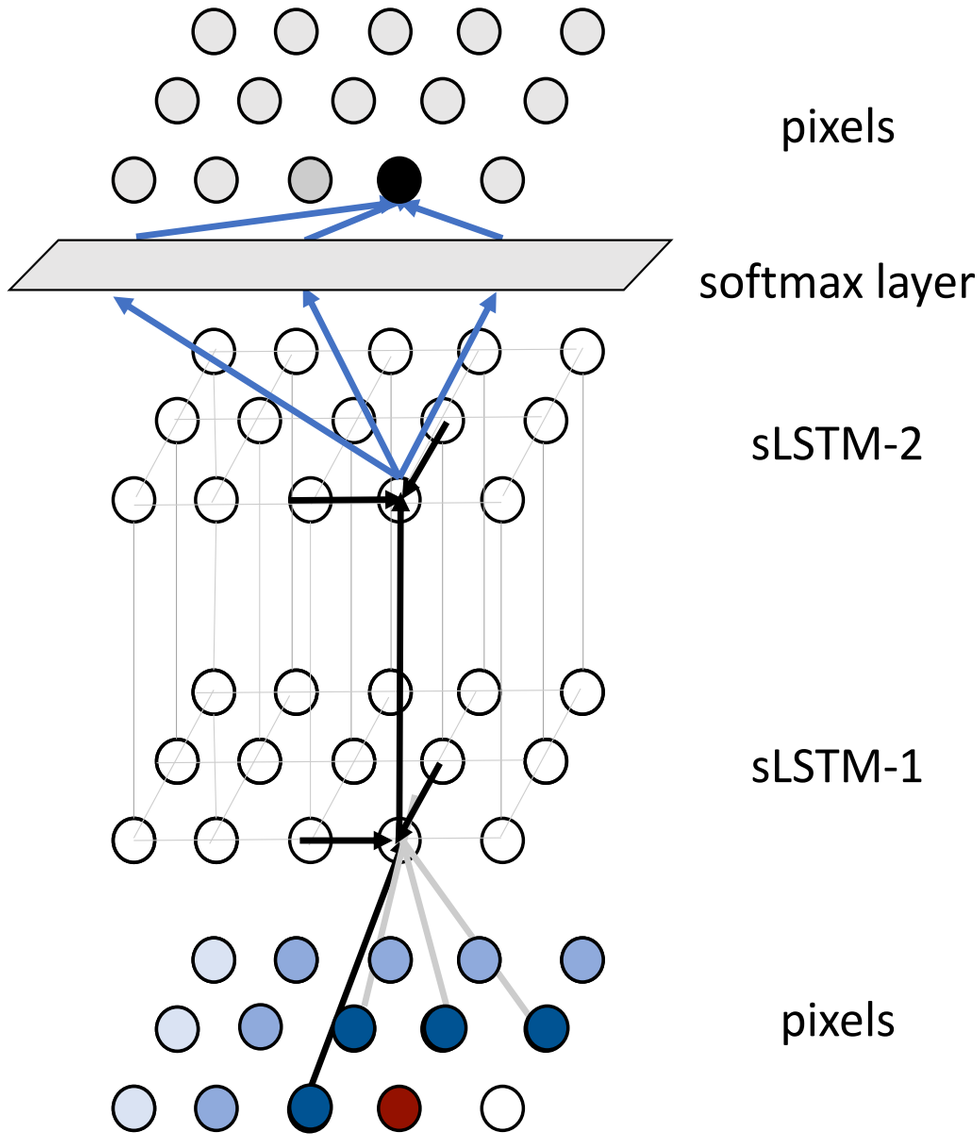
Adapted from: Generative image modeling using spatial LSTM. Theis & Bethge, 2015

x_{ij} the pixel i am estimating the value for

$\mathbf{x}_{<ij}$ the pixel that have already been predicted, and on which our LSTM is conditioning

Spatial LSTM

Too slow, no parallelization: I update the pixels one by one.

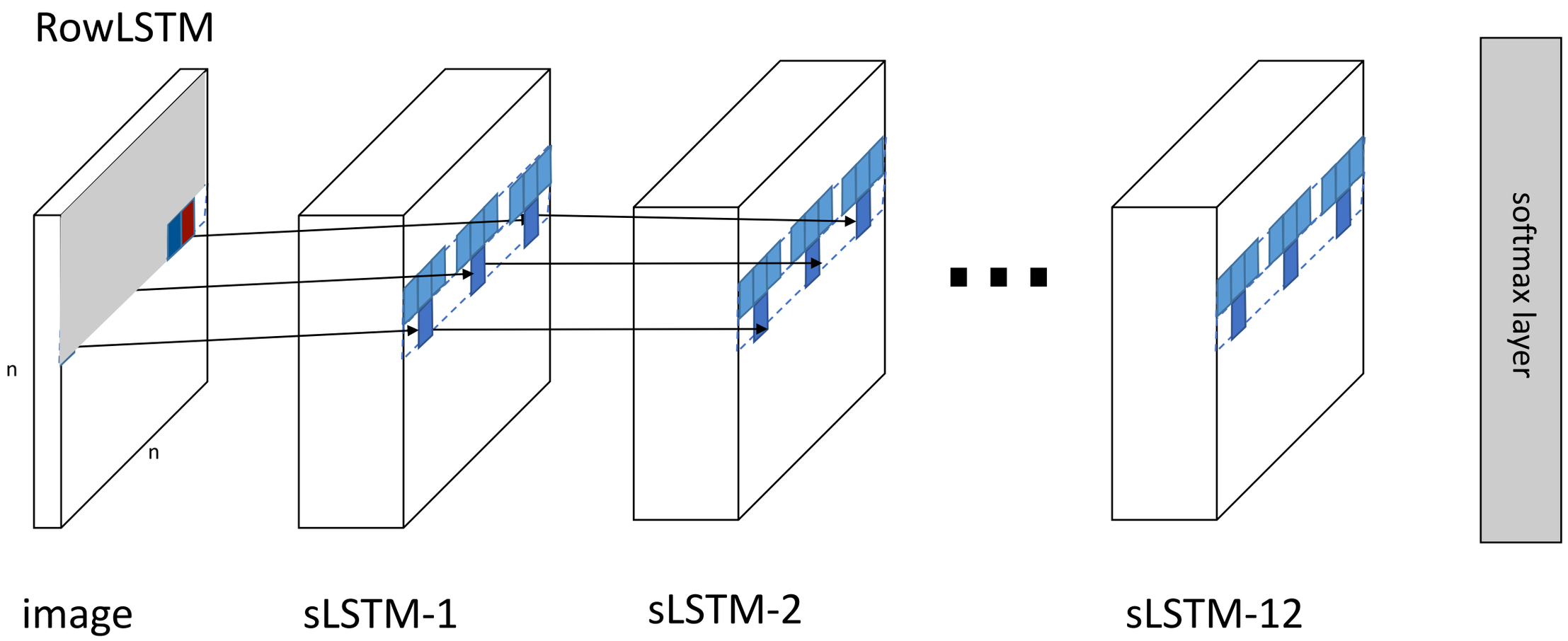


Adapted from: Generative image modeling using spatial LSTM. Theis & Bethge, 2015

x_{ij} the pixel i am estimating the value for

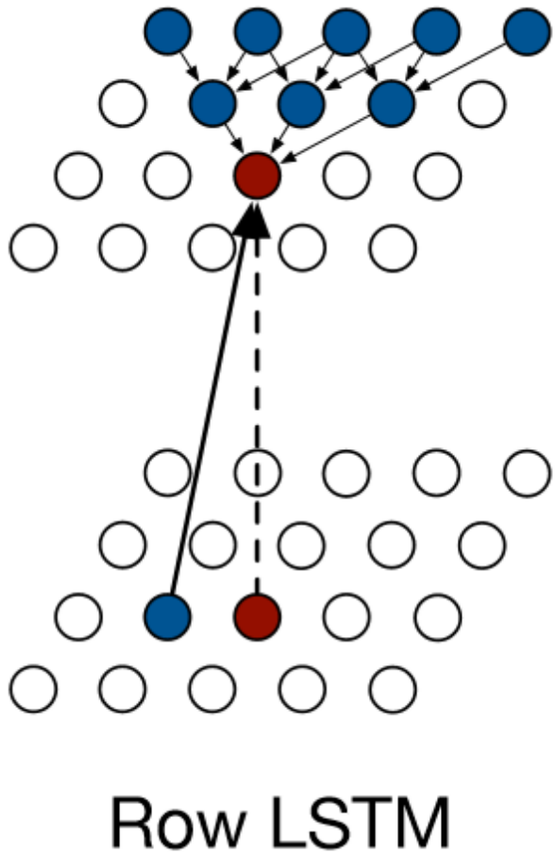
$\mathbf{x}_{<ij}$ the pixel that have already been predicted, and on which our LSTM is conditioning

Pixel RNN



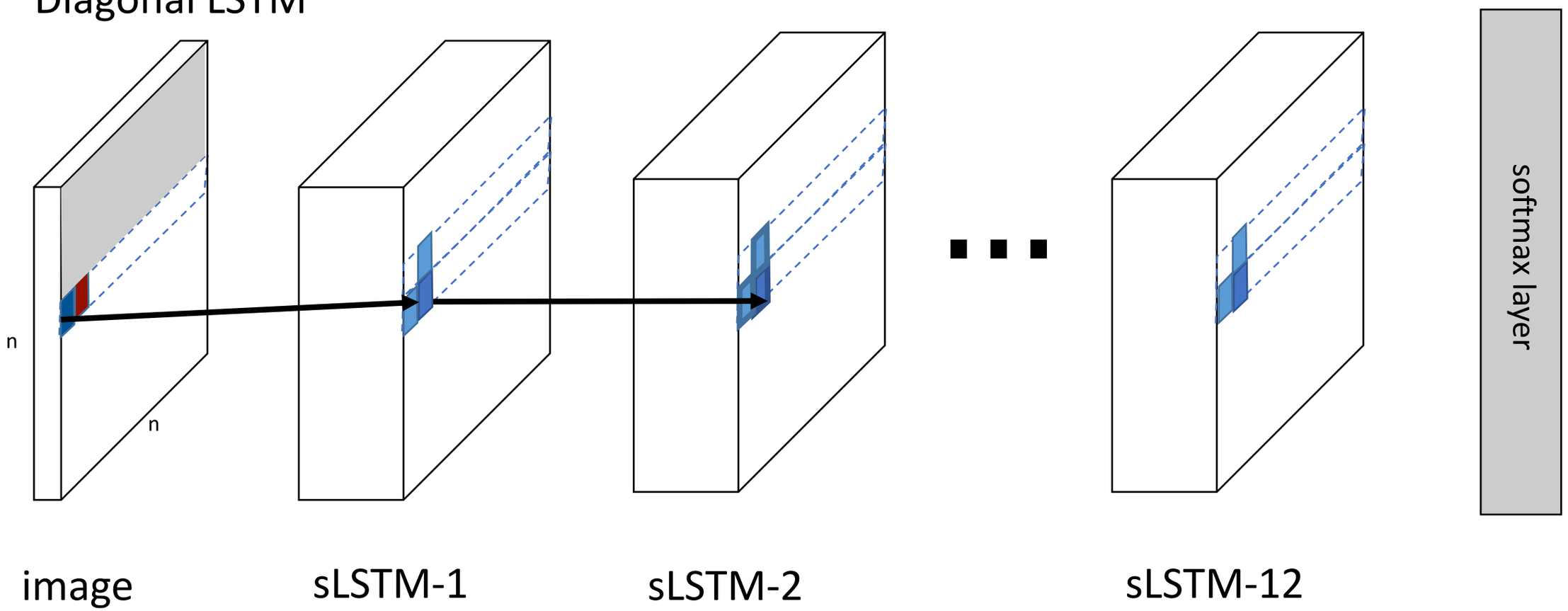
Pixel recurrent neural networks, ICML 2016

Row LSTM



Pixel recurrent neural networks, ICML 2016

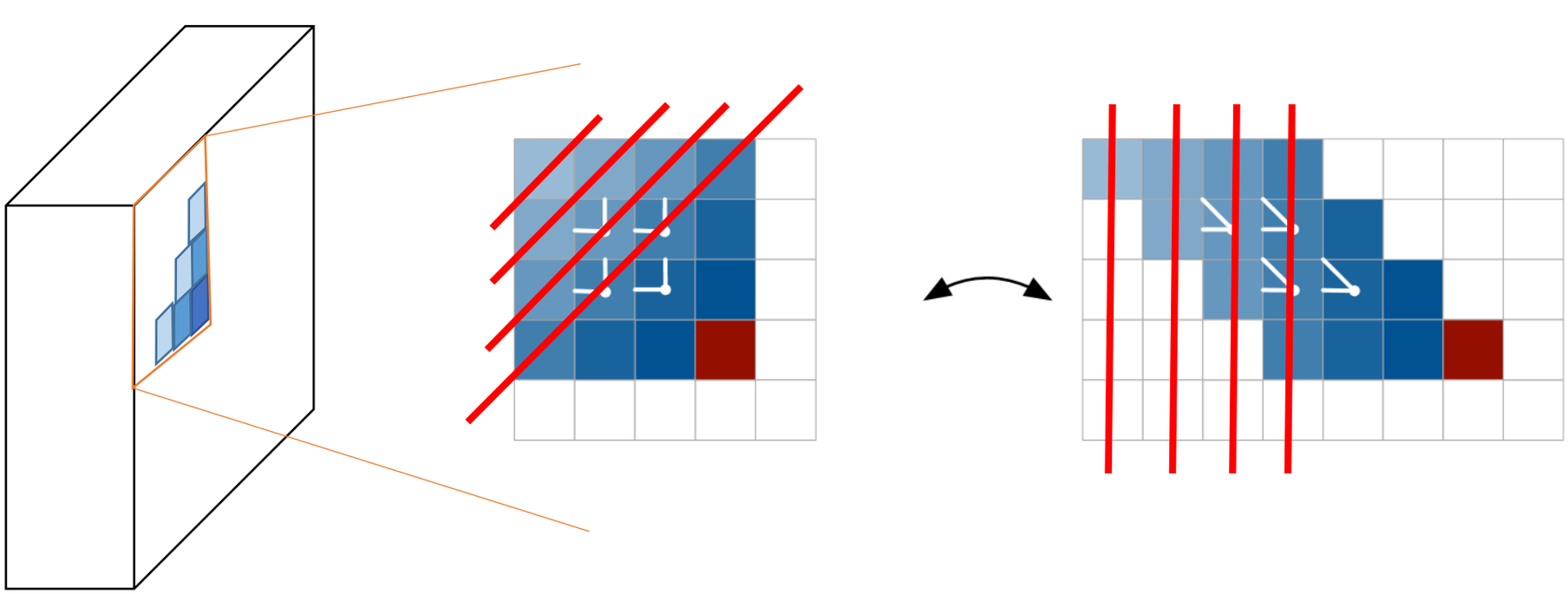
Pixel RNN



Pixel recurrent neural networks, ICML 2016

Diagonal LSTM

- To optimize, we skew the feature maps so it can be parallelized



Pixel recurrent neural networks, ICML 2016

Multimodality in expert actions: Fixes

Going beyond regression:

- Discretize the action space and train a classifier that predicts a categorical distribution over the discrete action space.

- 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)

- Variational autoencoders

- Generative adversarial networks

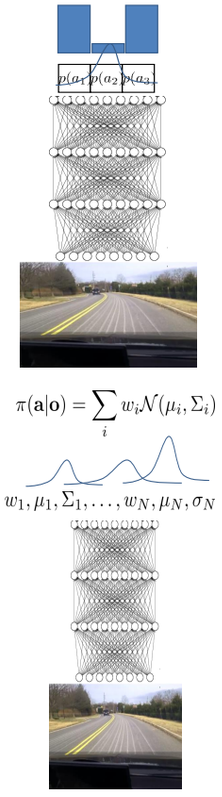
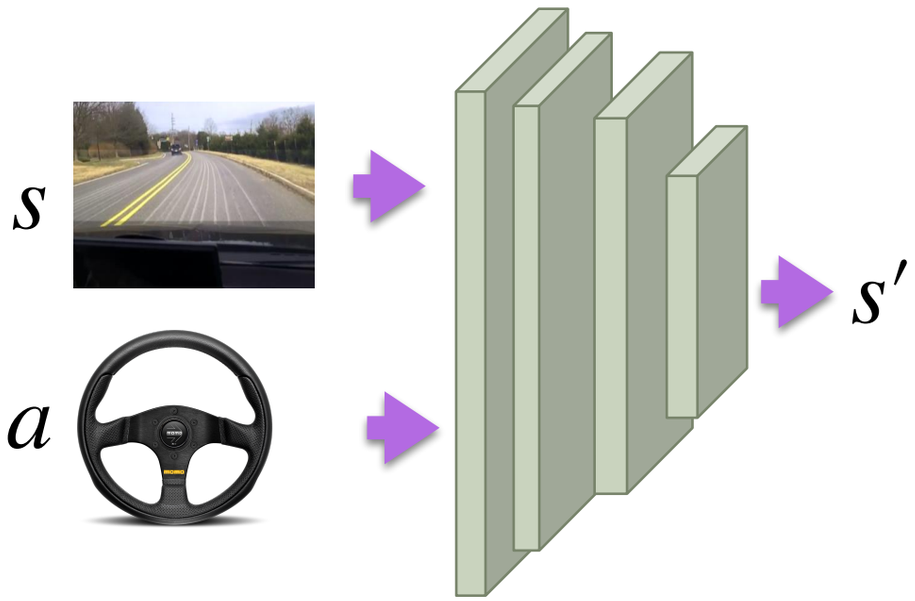


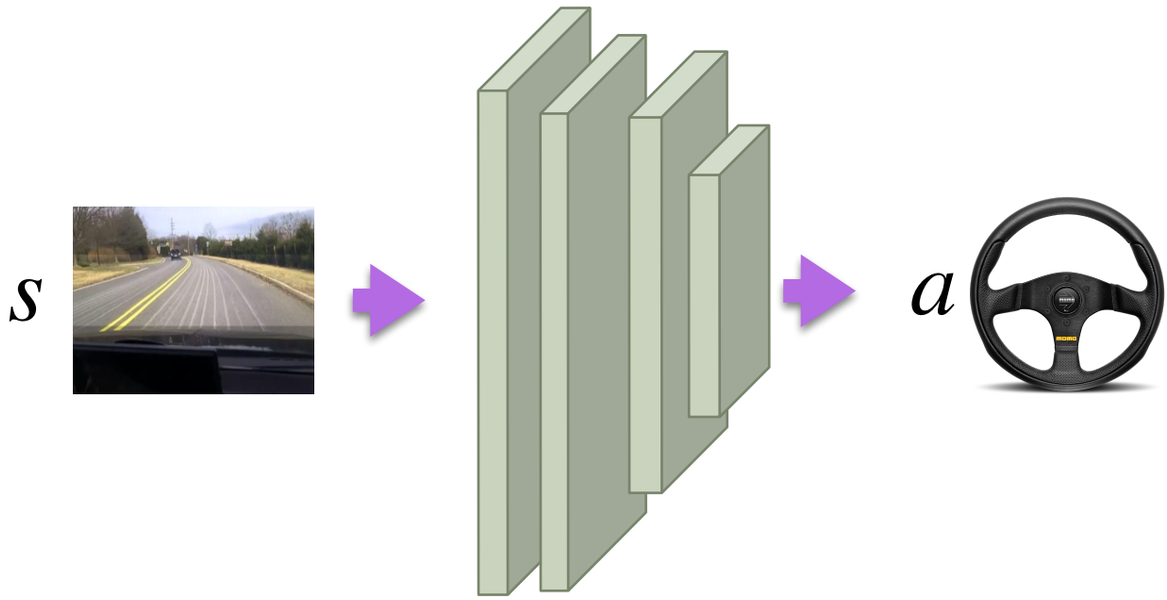
Diagram from Sergey Levine

Predicting continuous distributions

for dynamics

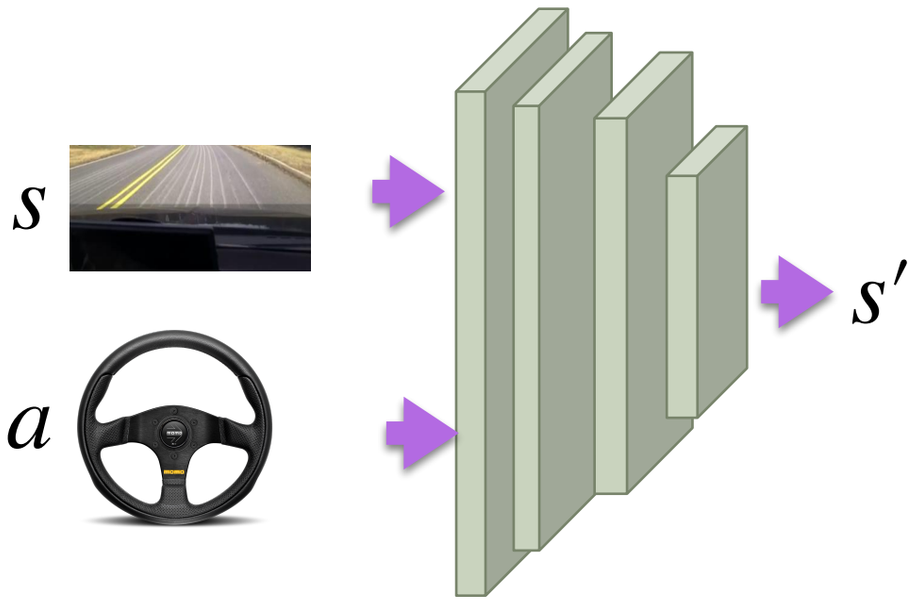


for policies

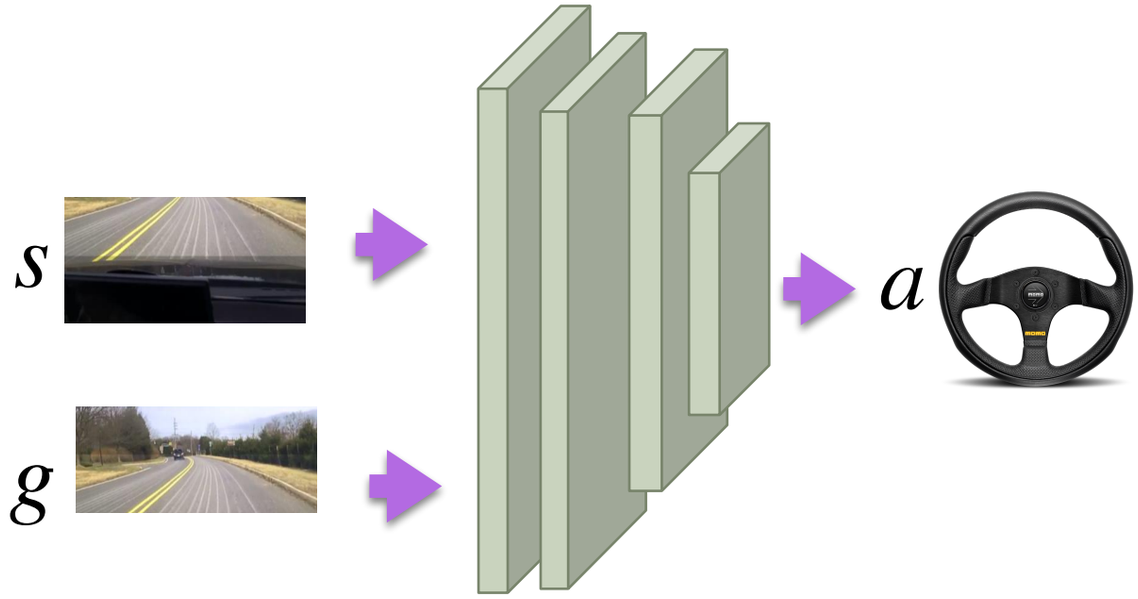


Predicting continuous distributions

for dynamics

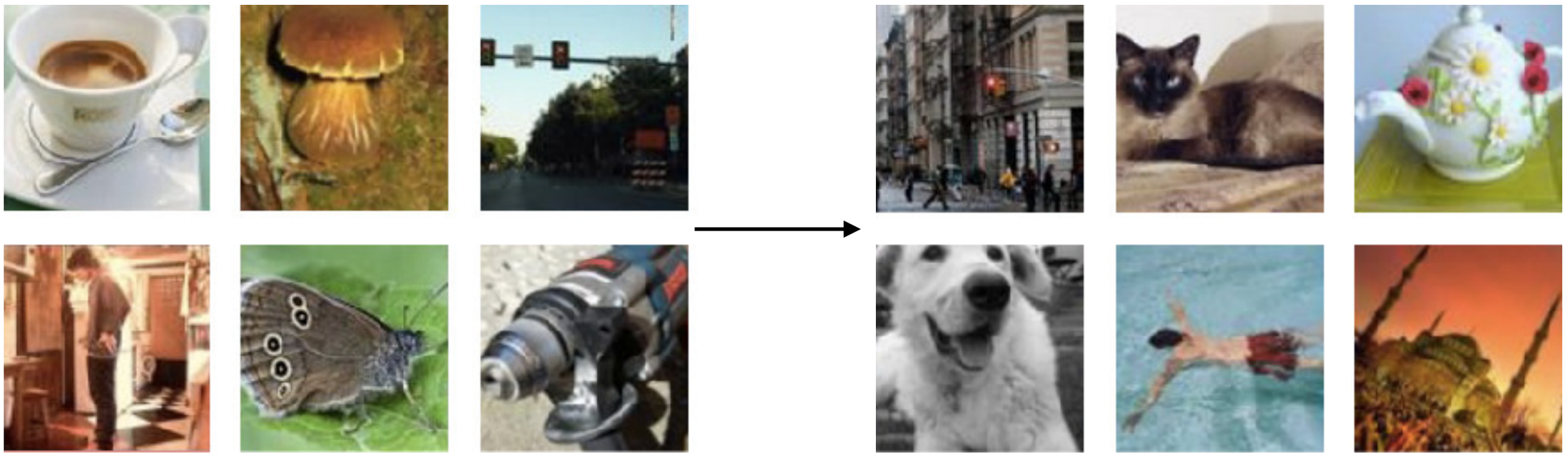


for generalized policies



Generative modeling

- Sample generation



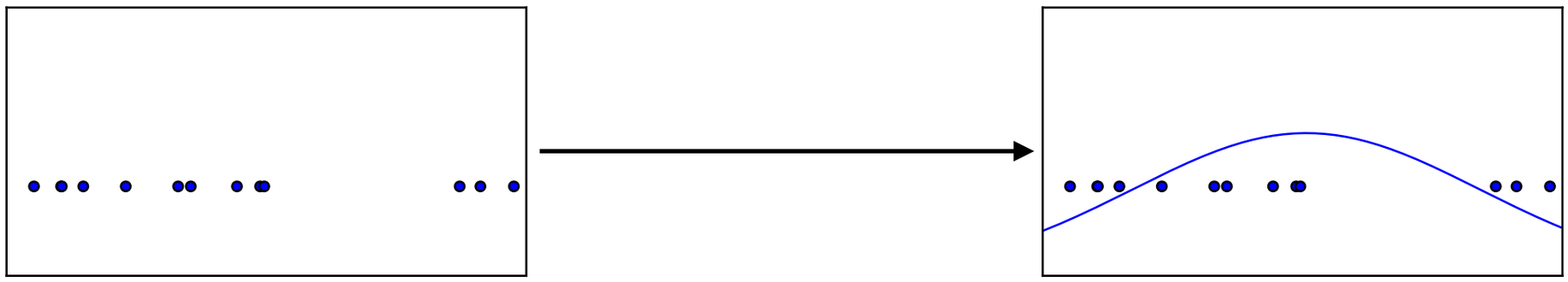
Training examples → Model samples

(Goodfellow 2016)

Generating samples that follow the statistics of the domain we are trying to model.

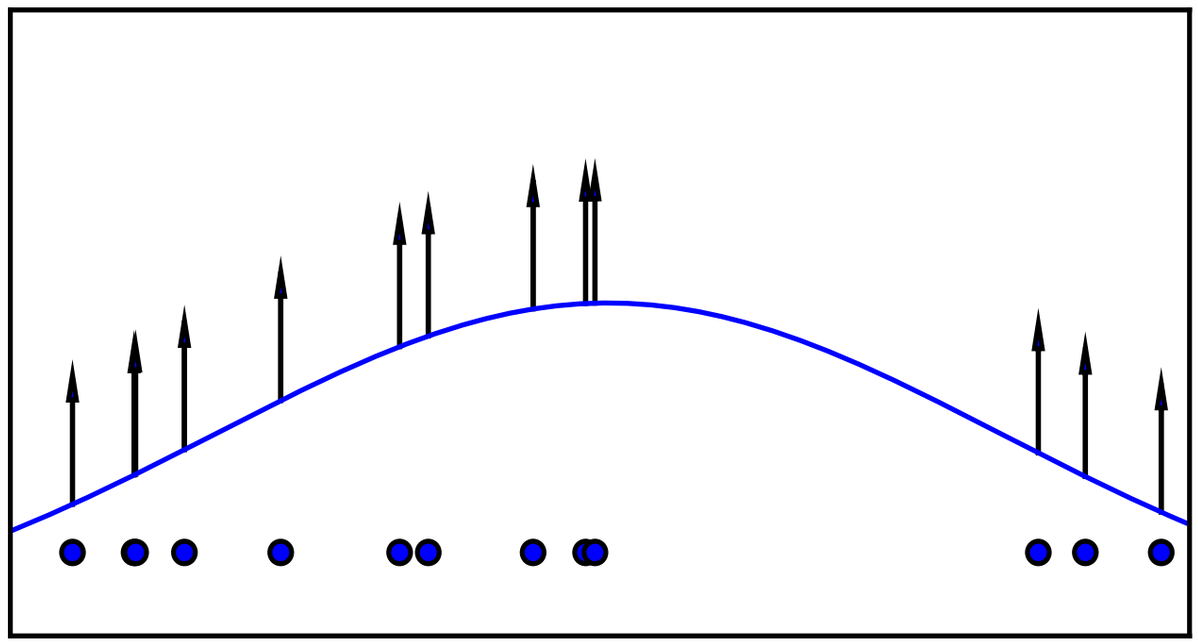
Generative modeling

- Density estimation



Assigning probabilities to objects of the domain we are trying to model

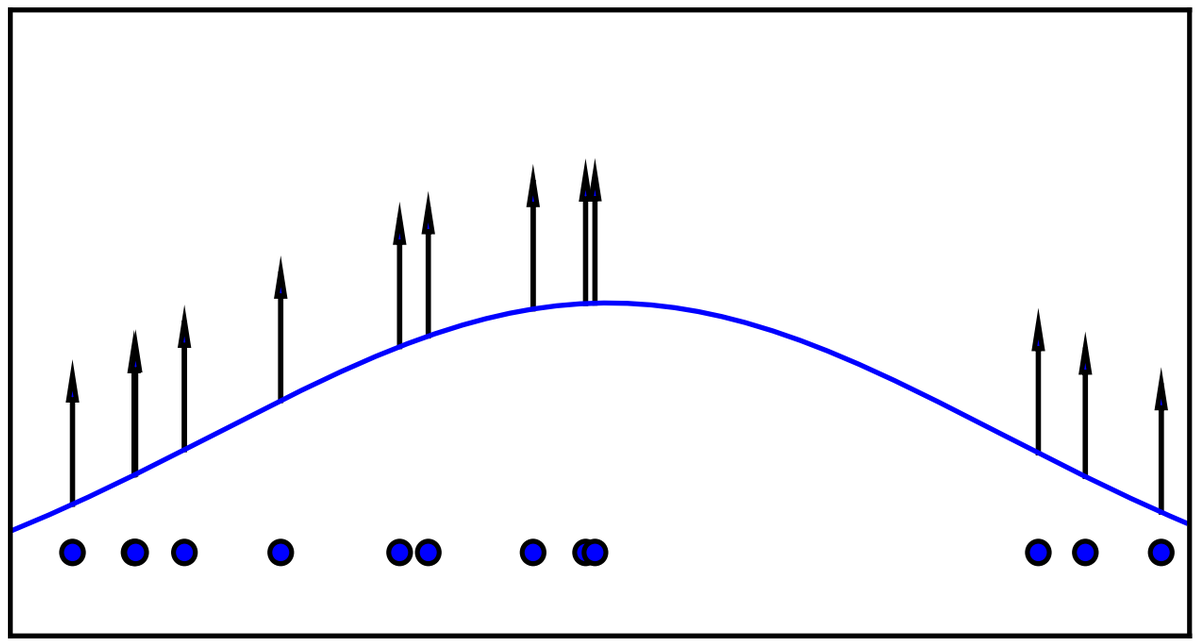
Maximum Likelihood



$$\max_{\theta} . \quad \mathbb{E}_{x \sim p_{\text{data}}} \log p_{\text{model}}(\mathbf{x} \mid \theta)$$

$$\max_{\theta} . \quad \sum_{i=j}^{N} \log p_{\text{model}}(\mathbf{x}_j \mid \theta)$$

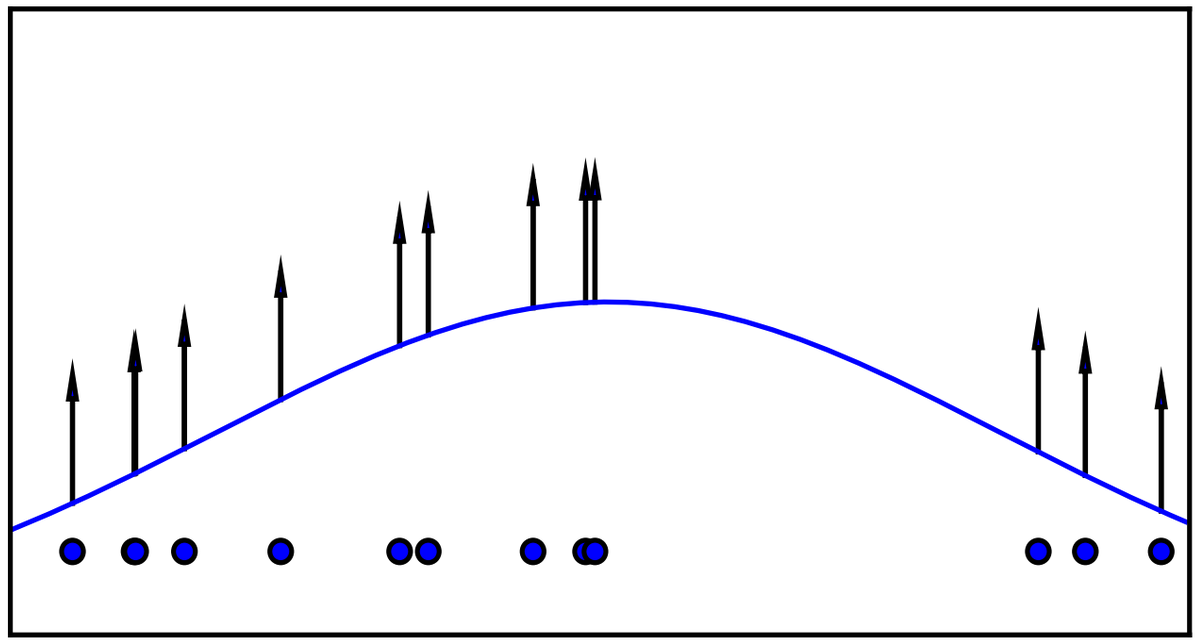
Maximum Likelihood



$$\max_{\theta} . \quad \mathbb{E}_{x \sim p_{\text{data}}} \log p_{\text{model}}(\mathbf{x} \mid \theta)$$

explicit density

Maximum Conditional Likelihood



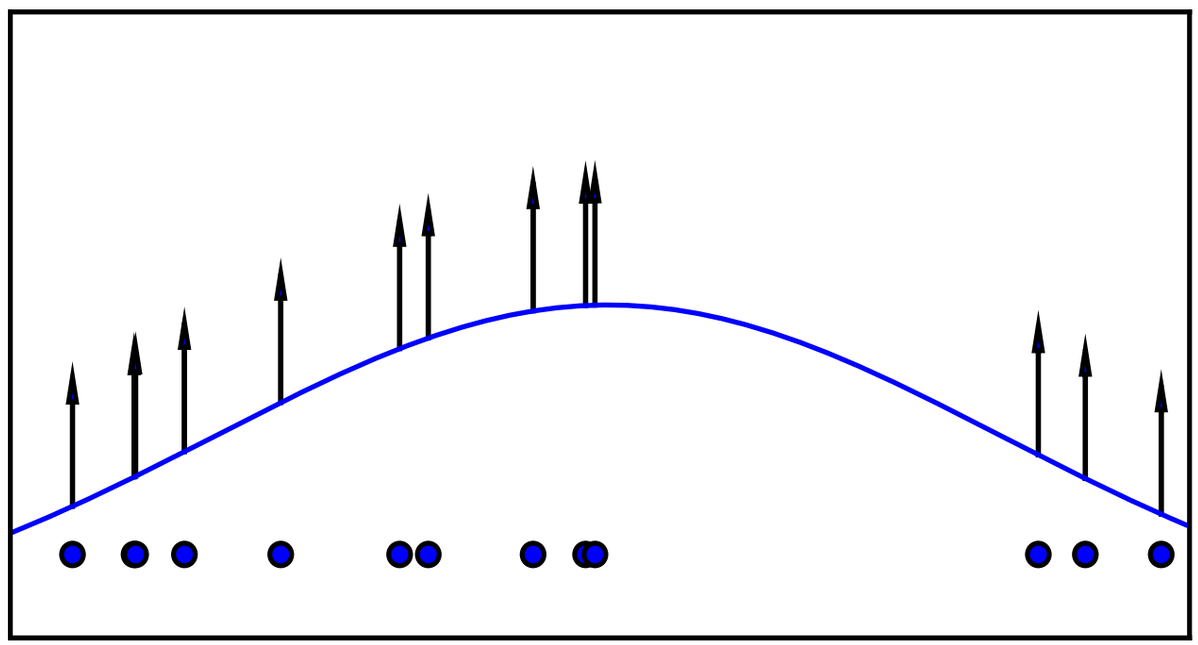
$$\max_{\theta} . \quad \mathbb{E}_{x \sim p_{\text{data}}} \log p_{\text{model}}(\mathbf{x} \mid \theta, c)$$

explicit density

extra conditioning information

Maximum Conditional Likelihood

$$\mathrm{D}_{\mathrm{KL}}(P\|Q) = -\sum_{x\in X} P(x)\log\left(\frac{Q(x)}{P(x)}\right)$$



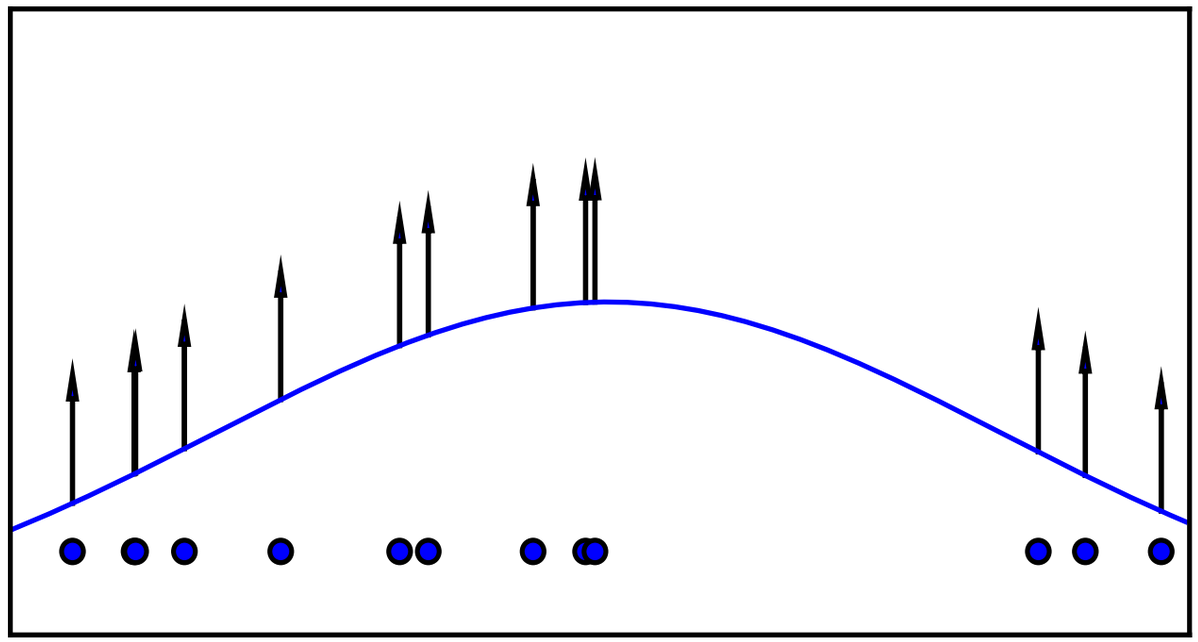
$$\max_{\theta} . \quad \mathbb{E}_{x\sim p_{\text{data}}} \log p_{\text{model}}(\mathbf{x}\,|\,\theta, c)$$

equiv. to

$$\theta^* = \arg\min_{\theta} \mathrm{D}_{\mathrm{KL}}\left(p_{\text{data}}\|p_{\text{model}}(\mathbf{x}\,|\,\theta, c)\right)$$

$\mathbb{E}_{x\sim p_{\text{data}}} \log p_{\text{data}}(\mathbf{x}\,|\,\theta)$ does not depend on θ

Maximum Likelihood-Gaussian with fixed covariance

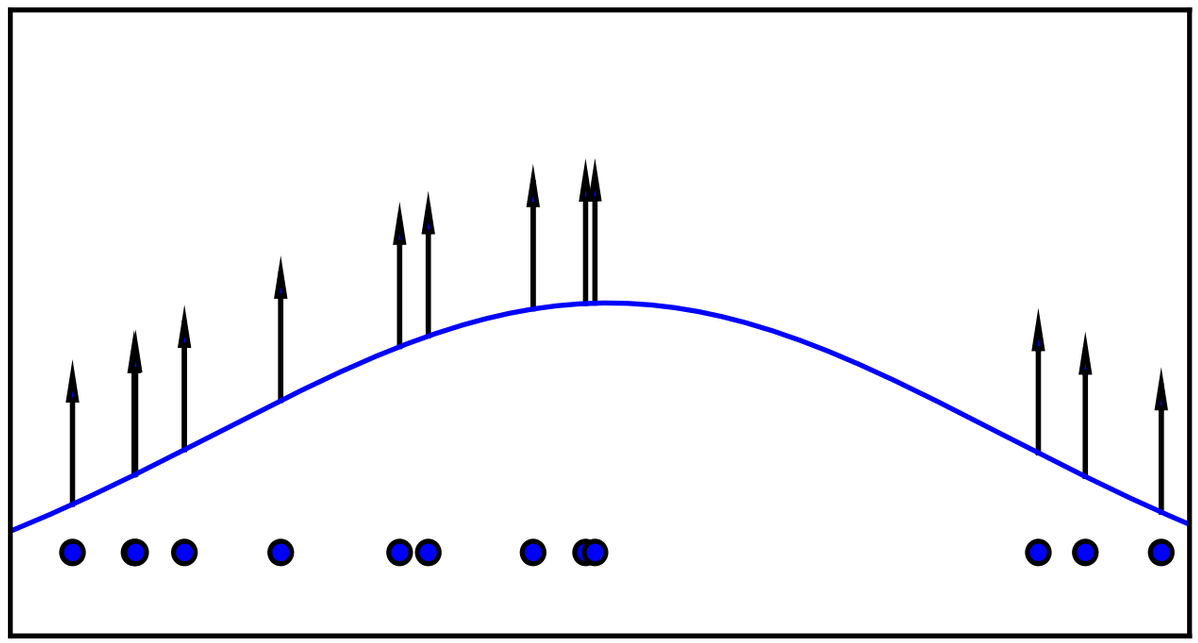


$$\theta^* = \arg\max_{\theta} \mathbb{E}_{x \sim p_{\text{data}}} \log p_{\text{model}}(\mathbf{x} \mid \theta, c)$$

$$p_{\text{model}}(\mathbf{x} \mid \theta, c) = \frac{1}{(2\pi)^{-\frac{k}{2}} \det(\Sigma)^{-\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu(\theta, \mathbf{c}))^{\top} \Sigma^{-1} (\mathbf{x} - \mu(\theta, \mathbf{c}))\right), \text{where } \Sigma = \mathrm{I}$$

Maximum Likelihood-Gaussian with fixed covariance

$$p_{\text{model}}(\mathbf{x}\,|\,\theta, c) = \frac{1}{(2\pi)^{-\frac{k}{2}}\det(\Sigma)^{-\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu(\theta,\mathbf{c}))^{\top}\Sigma^{-1}(\mathbf{x}-\mu(\theta,\mathbf{c}))\right), \text{where } \Sigma = \mathbf{I}$$



$$\max_{\theta}. \quad \mathbb{E}_{x\sim p_{\text{data}}} \log p_{\text{model}}(\mathbf{x}\,|\,\theta, c)$$

equiv. to $$\min_{\theta}. \quad \mathbb{E}_{x\sim p_{\text{data}}} \|\mathbf{x}-\mu(\theta,\mathbf{c})\|_2^2$$

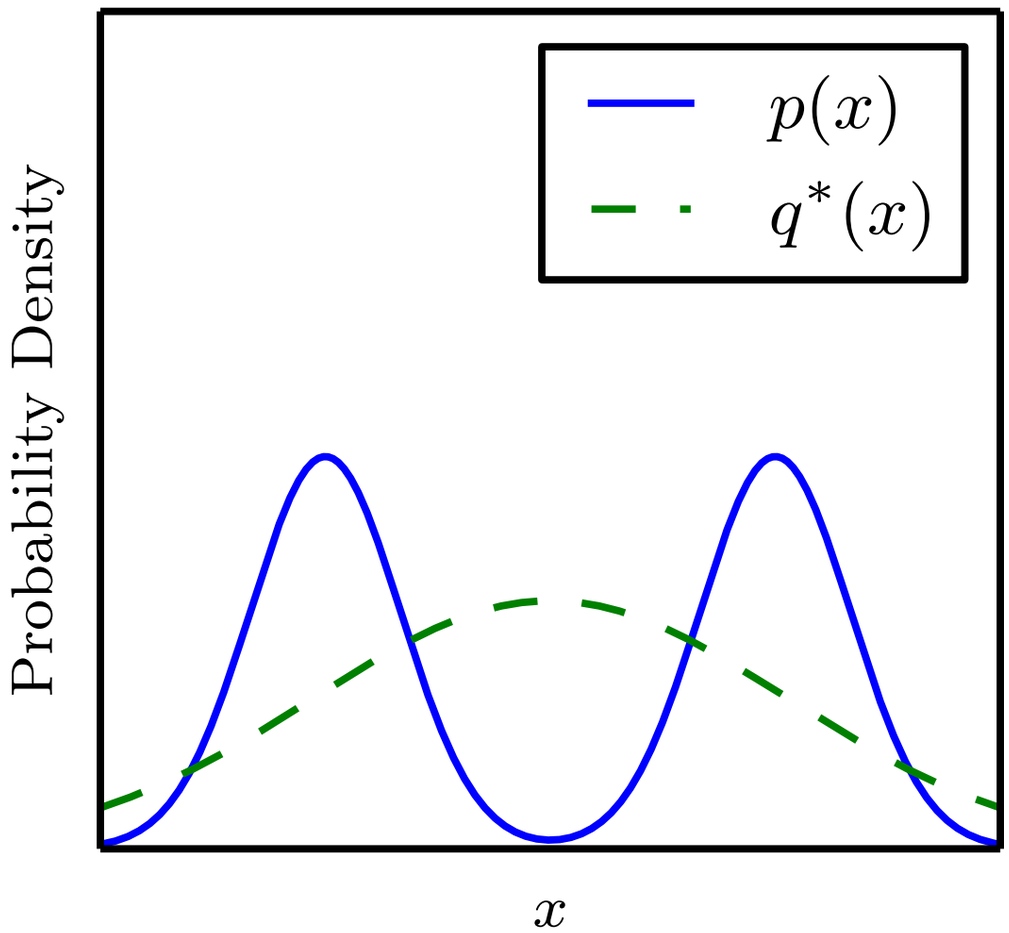
e.g., behavior cloning with continuous actions

Behaviour cloning

$$\mathrm{D}_{\mathrm{KL}}(P\|Q) = -\sum_{x\in X} P(x)\log\left(\frac{Q(x)}{P(x)}\right)$$

Non realistic action samples, due to non expressive policy (plain regressor)

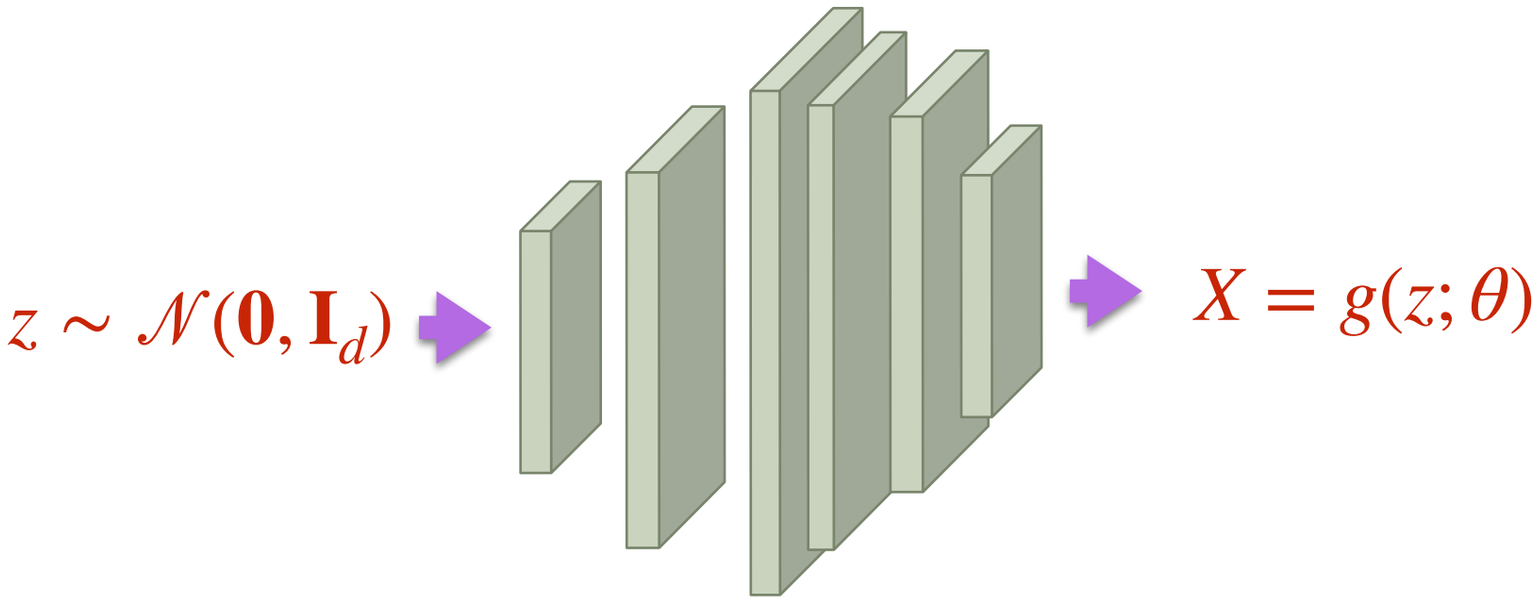
$$q^* = \mathrm{argmin}_q D_{\mathrm{KL}}(p\|q)$$

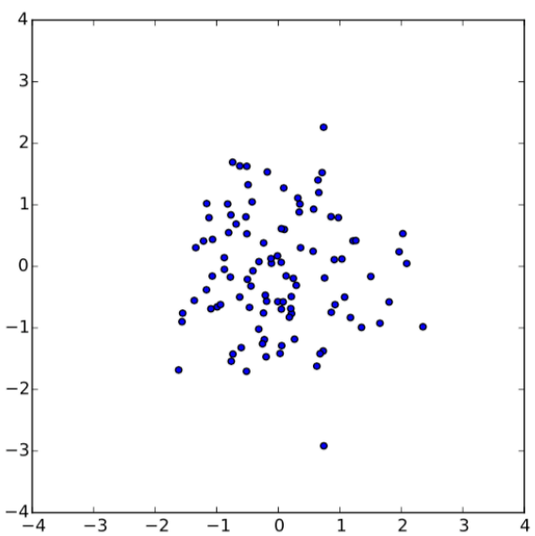


Maximum likelihood

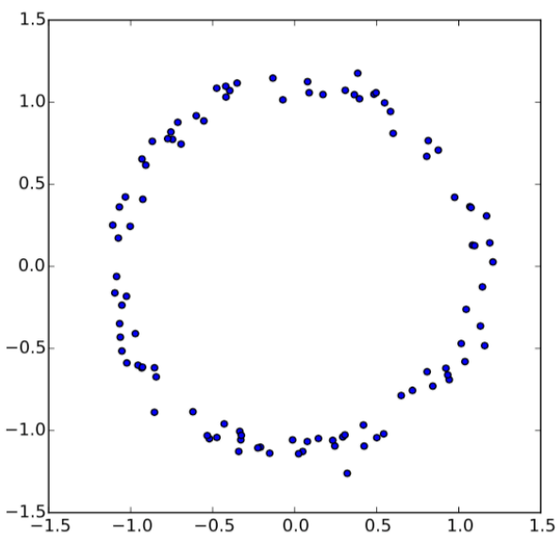
Latent variable models

Transform samples z from a fixed Gaussian distribution to samples from an arbitrary output distribution by feeding samples z through a deterministic function g.





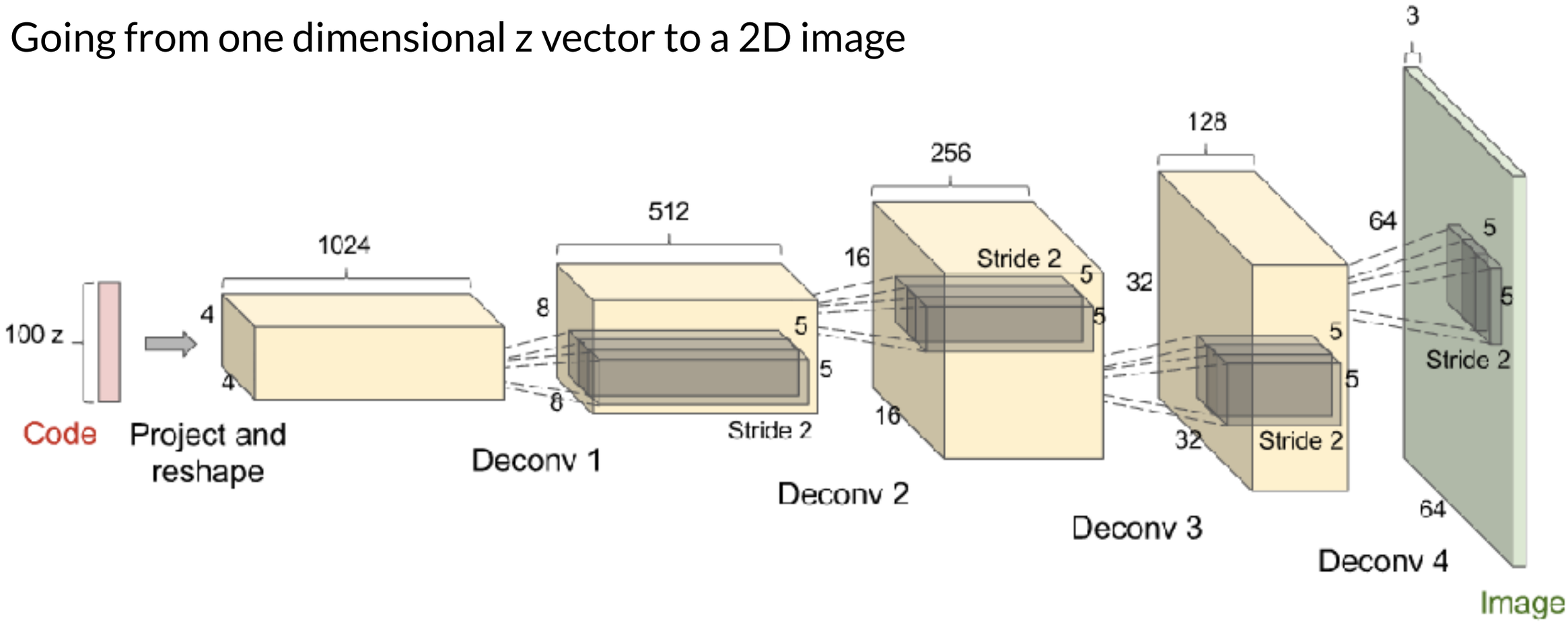
$z \sim \mathcal{N}(\mathbf{0}, I)$



$$g(z) = \frac{z}{10} + \frac{z}{\|z\|}$$

Latent variable models

Going from one dimensional z vector to a 2D image

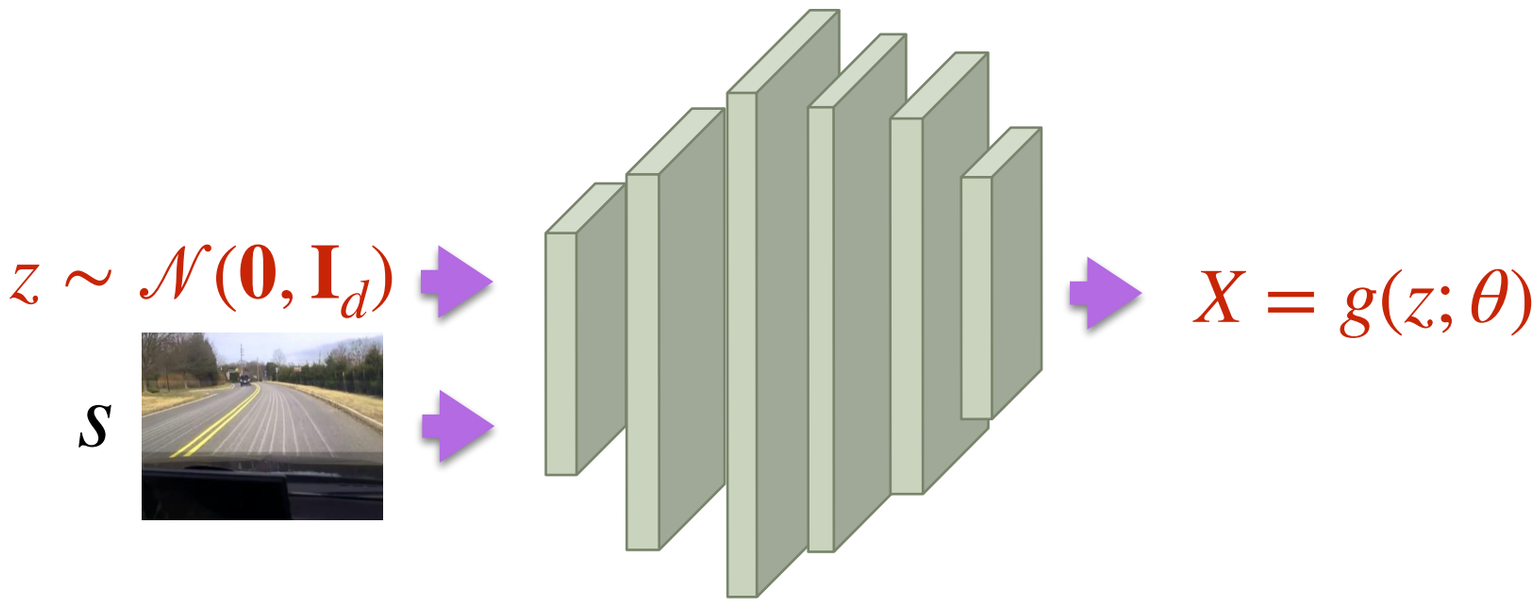


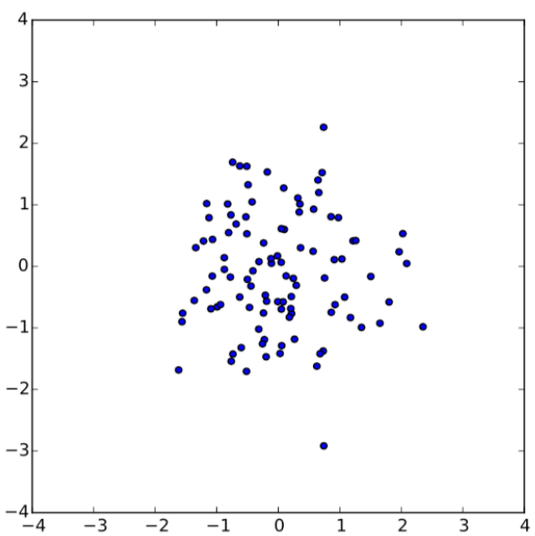
(Radford et al 2015)

Conditional latent variable models

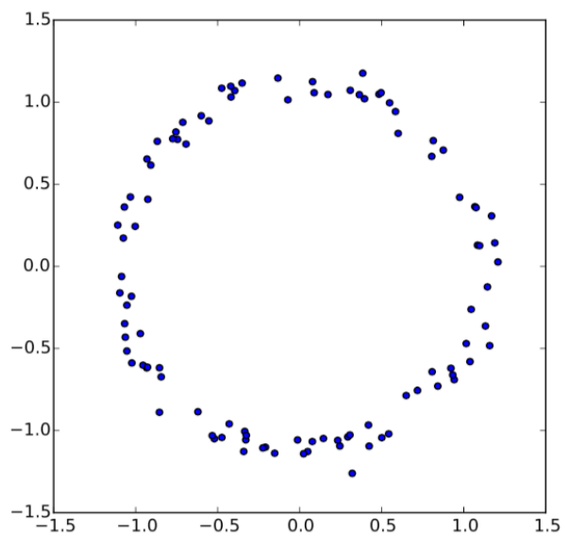
In few slides!

Transform samples z from a fixed Gaussian distribution to samples from an arbitrary output distribution by feeding samples z through a deterministic function g.





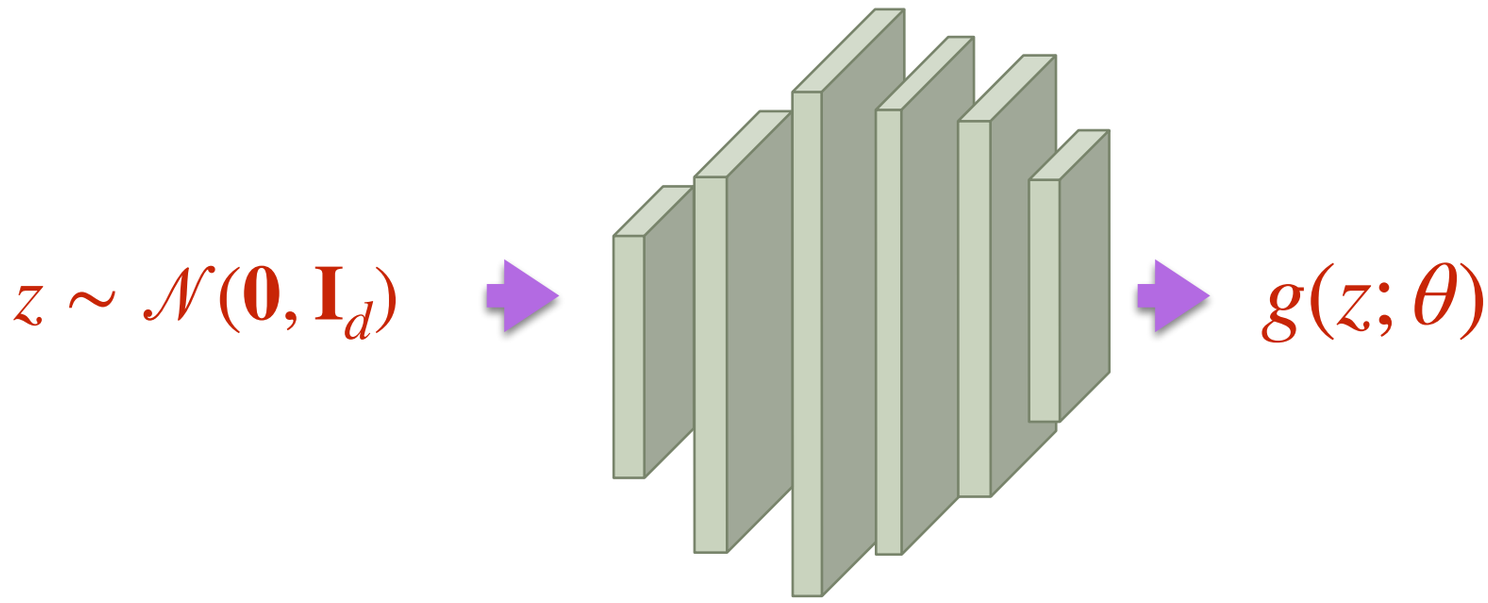
$z \sim \mathcal{N}(\mathbf{0}, I)$



$$g(z) = \frac{z}{10} + \frac{z}{\|z\|}$$

Tutorial on variational Autoencoders, Doersch 2016

Latent variable models



The probability of an example X given a sample z is usually set to be a Gaussian:

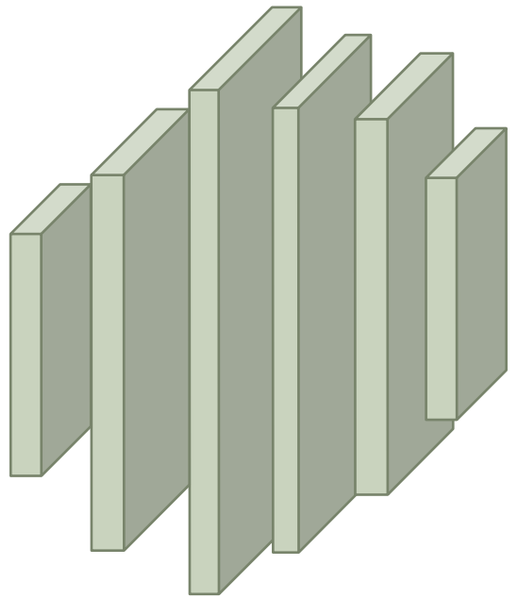
$$X \sim \mathcal{N}(X \mid g(z;\theta), \sigma^2 \cdot \mathbf{I}_D)$$

$$P(X \mid z;\theta) = \frac{\exp\left(-\frac{1}{2}(X - g(z;\theta))^\top (\sigma^2 \cdot \mathbf{I}_D)^{-1}(X - g(z;\theta)\right)}{\sqrt{(2\pi)^D \det(\sigma^2 \cdot \mathbf{I}_D)}}$$

$$P(X \mid z;\theta) = \frac{\exp\left(-\frac{1}{2}(X - g(z;\theta))^\top (\sigma^{-2} \cdot \mathbf{I}_D)(X - g(z;\theta)\right)}{\sqrt{(2\pi)^D (\sigma^2)^D}}$$

Latent variable models

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



$g(z;\theta)$

The probability of an example X given a sample z is usually set to be a Gaussian:

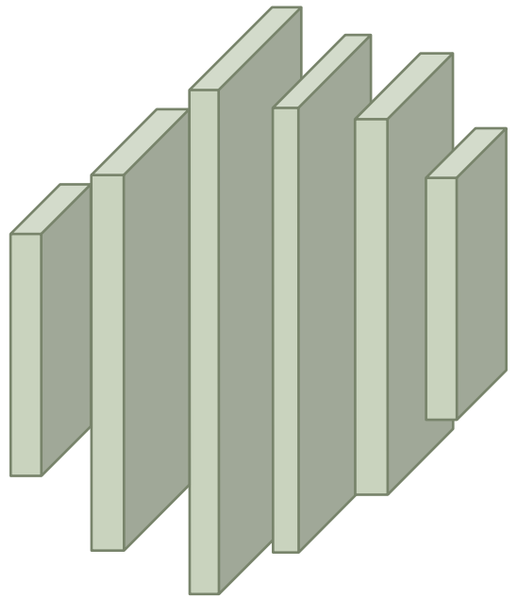
$$P(X \mid z;\theta) = \mathcal{N}(X \mid g(z;\theta), \sigma^2 \cdot \mathbf{I}_d)$$

Given a dataset of N examples, we will use maximum likelihood to estimate parameters θ of the deterministic function g. This means, we want to maximize the probability of each of the training samples, under our generative model:

$$\min_{\theta} . \quad -\frac{1}{N}\sum_{j=1}^{N} \log P(X_j) \quad \text{where} \quad P(X) = \int P(X \mid z)P(z)dz$$

Tutorial on variational Autoencoders, Doersch 2016

Latent variable models

$z \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$  $g(z;\theta)$ $X \sim \mathcal{N}(X \mid g(z;\theta), \sigma^2 \cdot \mathbf{I}_D)$

Let's approximate the intractable integral $P(X) = \int P(X \mid z)P(z)dz$

with a finite set of samples: $P(X) \approx \frac{1}{n}\sum_{i=1}^{n} P(X \mid z_i;\theta)$

$$\min_{\theta} . \quad -\frac{1}{N}\sum_{j=1}^{N} \log P(X_j) = -\frac{1}{N}\sum_{j=1}^{N} \log\left(\frac{1}{n}\sum_{i=1}^{n} P(X_j \mid z_i;\theta)\right)$$

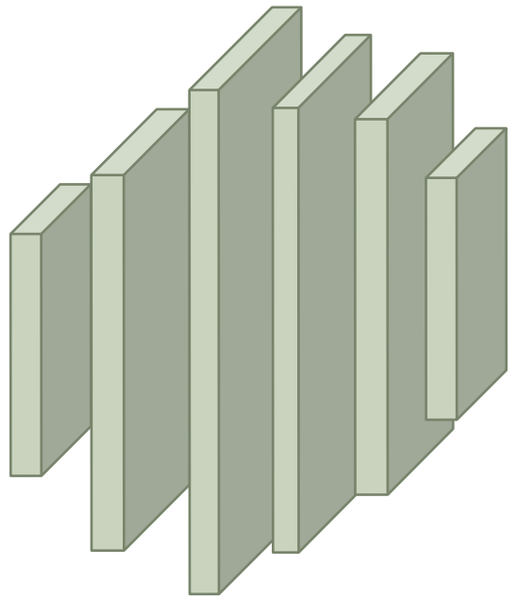
Q: Can we backpropagate here?

A: Yes, we can. Everything is differentiable. Yet, this is not a good approximation.

Tutorial on variational Autoencoders, Doersch 2016

n best loss

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



$$g(z;\theta) \qquad X \sim \mathcal{N}(X \mid g(z;\theta), \sigma^2 \cdot \mathbf{I}_D)$$

Let's approximate the intractable integral $P(X) = \int P(X \mid z)P(z)dz$

with a finite set of samples: $P(X) \approx \frac{1}{n}\sum_{i=1}^{n} P(X \mid z_i;\theta)$

Some works consider penalizing only the best out of n samples:

$$\max_{\theta} . \quad \frac{1}{N}\sum_{j=1}^{N} \max_{i \in \{1 \ldots n\}} \left(\log P(X_j \mid z_i;\theta) \right) \qquad \text{equiv. to} \qquad \min_{\theta} . \quad \frac{1}{N}\sum_{j=1}^{N} \min_{i \in \{1 \ldots n\}} \left(\|g(z;\theta) - X_j\| \right)$$

Implicit maximum likelihood estimation, Li and Malik
Motion Prediction Under Multimodality with Conditional Stochastic Networks, Google

Variational Autoencoders

- Let's consider sampling z from an alternative distribution $Q(z)$ (as opposed to the prior $\mathcal{N}(0,\mathbf{I}_d)$) and minimize the KL between this $Q(z)$ and the posterior distribution $P(z \mid X)$.
- I can pick any distribution Q I like, I will also condition it on X (I will cheat!) to help inform the sampling!

$$D_{KL}(Q(z \mid X), P(z \mid X))$$

Tutorial on variational Autoencoders, Doersch 2016

Variational Autoencoders

- Let's consider sampling z from an alternative distribution $Q(z)$ (as opposed to the prior $\mathcal{N}(0,\mathbf{I}_d)$) and minimize the KL between this $Q(z)$ and the posterior distribution $P(z|X)$.
- I can pick any distribution Q I like, I will also condition it on X (I will cheat!) to help inform the sampling!

$$D_{KL}(Q(z|X), P(z|X)) = \int Q(z|X) \log \frac{Q(z|X)}{P(z|X)} dz$$

Tutorial on variational Autoencoders, Doersch 2016

Variational Autoencoders

- Let's consider sampling z from an alternative distribution $Q(z)$ (as opposed to the prior $\mathcal{N}(0,\mathbf{I}_d)$) and minimize the KL between this $Q(z)$ and the posterior distribution $P(z|X)$.
- I can pick any distribution Q I like, I will also condition it on X (I will cheat!) to help inform the sampling!

$$D_{KL}(Q(z|X), P(z|X)) = \int Q(z|X) \log \frac{Q(z|X)}{P(z|X)} dz$$
$$= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log P(z|X)$$

Tutorial on variational Autoencoders, Doersch 2016

Variational Autoencoders

- Let's consider sampling z from an alternative distribution $Q(z)$ (as opposed to the prior $\mathcal{N}(0,\mathbf{I}_d)$) and minimize the KL between this $Q(z)$ and the posterior distribution $P(z|X)$.
- I can pick any distribution Q I like, I will also condition it on X (I will cheat!) to help inform the sampling!

$$
\begin{aligned}
D_{KL}(Q(z|X), P(z|X)) &= \int Q(z|X) \log \frac{Q(z|X)}{P(z|X)} dz \\
&= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log P(z|X) \\
&= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log \frac{P(X|z)P(z)}{P(X)}
\end{aligned}
$$

Tutorial on variational Autoencoders, Doersch 2016

Variational Autoencoders

- Let's consider sampling z from an alternative distribution $Q(z)$ (as opposed to the prior $\mathcal{N}(0,\mathbf{I}_d)$) and minimize the KL between this $Q(z)$ and the posterior distribution $P(z|X)$.
- I can pick any distribution Q I like, I will also condition it on X (I will cheat!) to help inform the sampling!

$$
\begin{aligned}
D_{KL}(Q(z|X), P(z|X)) &= \int Q(z|X) \log \frac{Q(z|X)}{P(z|X)} dz \\
&= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log P(z|X) \\
&= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log \frac{P(X|z)P(z)}{P(X)} \\
&= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log P(X|z) - \mathbb{E}_Q \log P(z) + \log P(X)
\end{aligned}
$$

Tutorial on variational Autoencoders, Doersch 2016

Variational Autoencoders

- Let's consider sampling z from an alternative distribution $Q(z)$ (as opposed to the prior $\mathcal{N}(0,\mathbf{I}_d)$) and minimize the KL between this $Q(z)$ and the posterior distribution $P(z|X)$.
- I can pick any distribution Q I like, I will also condition it on X (I will cheat!) to help inform the sampling!

$$
\begin{aligned}
D_{KL}(Q(z|X), P(z|X)) &= \int Q(z|X) \log \frac{Q(z|X)}{P(z|X)} dz \\
&= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log P(z|X) \\
&= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log \frac{P(X|z)P(z)}{P(X)} \\
&= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log P(X|z) - \mathbb{E}_Q \log P(z) + \log P(X) \\
&= D_{KL}(Q(z|X)|P(z)) - \mathbb{E}_Q \log P(X|z) + \log P(X)
\end{aligned}
$$

Q will be a parametrized Gaussian distribution!

Tutorial on variational Autoencoders, Doersch 2016

Variational Autoencoders

- Let's consider sampling z from an alternative distribution $Q(z)$ (as opposed to the prior $\mathcal{N}(0,\mathbf{I}_d)$) and minimize the KL between this $Q(z)$ and the posterior distribution $P(z|X)$.
- I can pick any distribution Q I like, I will also condition it on X (I will cheat!) to help inform the sampling!

$$
\begin{aligned}
D_{KL}(Q(z|X), P(z|X)) &= \int Q(z|X) \log \frac{Q(z|X)}{P(z|X)} dz \\
&= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log P(z|X) \\
&= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log \frac{P(X|z)P(z)}{P(X)} \\
&= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log P(X|z) - \mathbb{E}_Q \log P(z) + \log P(X) \\
&= D_{KL}(Q(z|X), P(z)) - \mathbb{E}_Q \log P(X|z) + \log P(X)
\end{aligned}
$$

$$
D_{KL}(Q(z|X;\phi), P(z)) - \mathbb{E}_Q \log P(X|z;\theta)
$$

Tutorial on variational Autoencoders, Doersch 2016

Variational Autoencoders

- Let's consider sampling z from an alternative distribution $Q(z)$ (as opposed to the prior $\mathcal{N}(0,\mathbf{I}_d)$) and minimize the KL between this $Q(z)$ and the posterior distribution $P(z|X)$.
- I can pick any distribution Q I like, I will also condition it on X (I will cheat!) to help inform the sampling!

$$
\begin{aligned}
D_{KL}(Q(z|X), P(z|X)) &= \int Q(z|X) \log \frac{Q(z|X)}{P(z|X)} dz \\
&= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log P(z|X) \\
&= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log \frac{P(X|z)P(z)}{P(X)} \\
&= \mathbb{E}_Q \log Q(z|X) - \mathbb{E}_Q \log P(X|z) - \mathbb{E}_Q \log P(z) + \log P(X) \\
&= D_{KL}(Q(z|X)|P(z)) - \mathbb{E}_Q \log P(X|z) + \log P(X)
\end{aligned}
$$

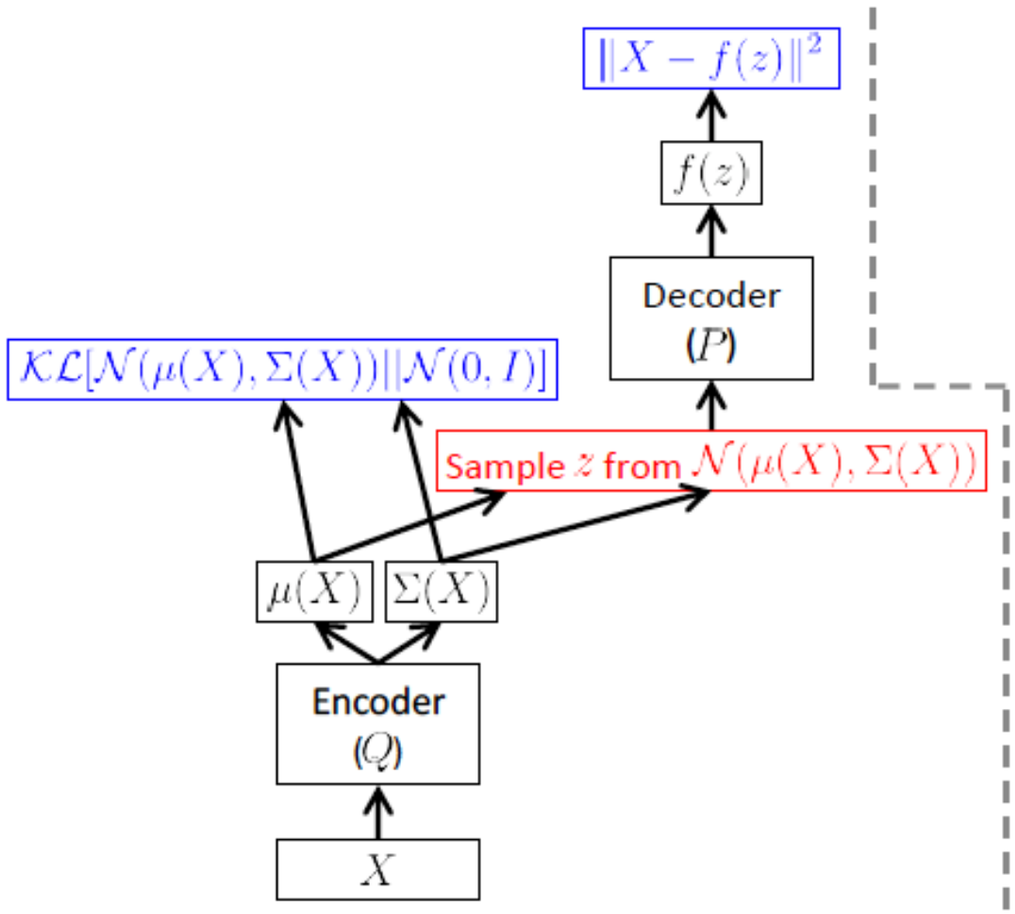
$$
\min_{\phi,\theta} . \quad D_{KL}(Q(z|X;\phi), P(z)) - \mathbb{E}_Q \log P(X|z;\theta)
$$

encoder

decoder

Tutorial on variational Autoencoders, Doersch 2016

Variational Autoencoder



$$\min_{\phi,\theta} . \quad D_{KL}(Q(z \mid X; \phi), P(z)) - \mathbb{E}_Q \log P(X \mid z; \theta)$$

Q: Can we backpropagate?

Tutorial on variational Autoencoders, Doersch 2016

Remember: Sampling from a Gaussian distribution

$$x_i \sim \mathcal{N}(0,1)$$

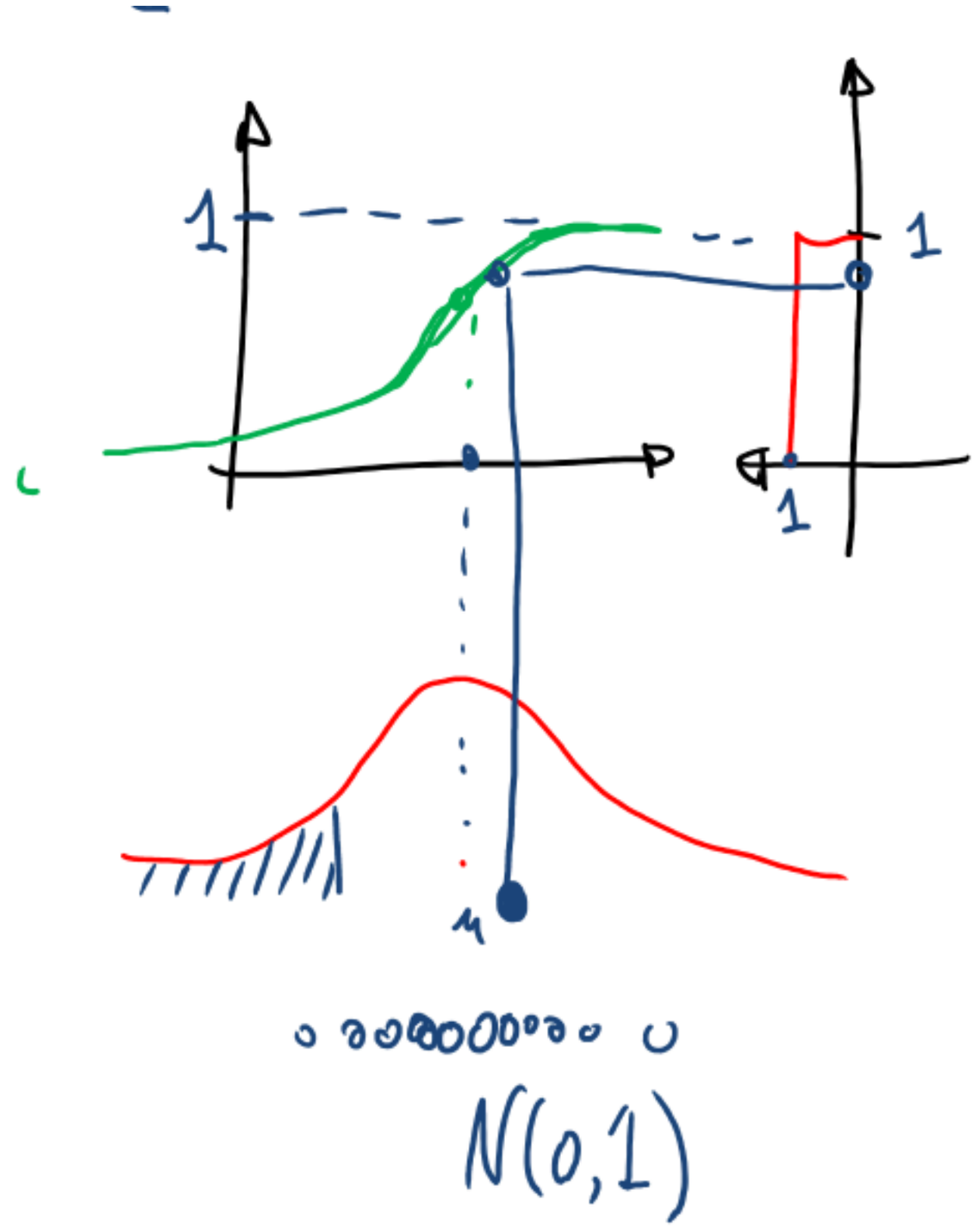
$$x_i \sim \mathcal{N}(\mu, \sigma^2)$$
$$\sim \mu + \sigma\mathcal{N}(0,1)$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_i \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right)$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_i \sim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \right)$$

$\mathbf{x}$ μ Σ

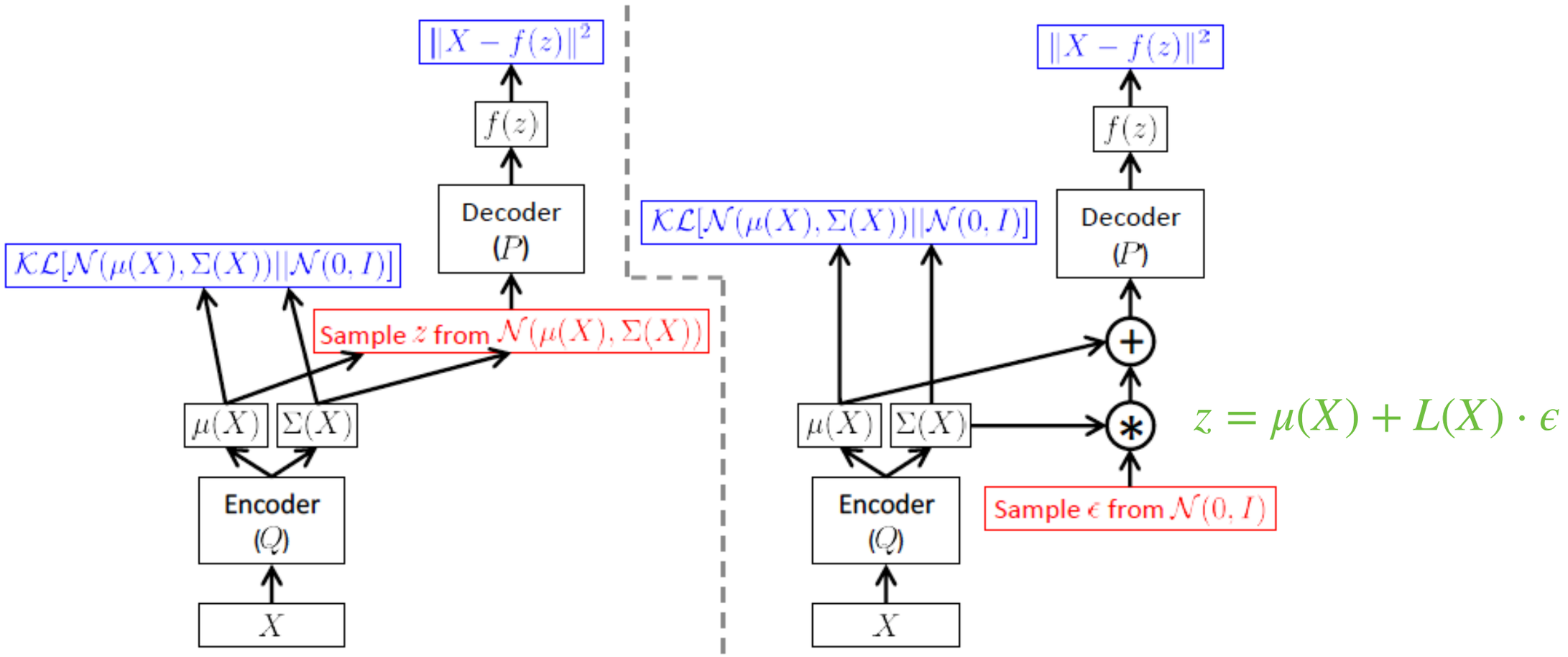
$$\mathbf{x} \sim \mathcal{N}(\mu, \Sigma) \qquad x \sim \mu + L\mathcal{N}(0,I)$$



Cholesky decomposition $\Sigma = LL^T$

Variational Autoencoder

From left to right: re-parametrization trick!

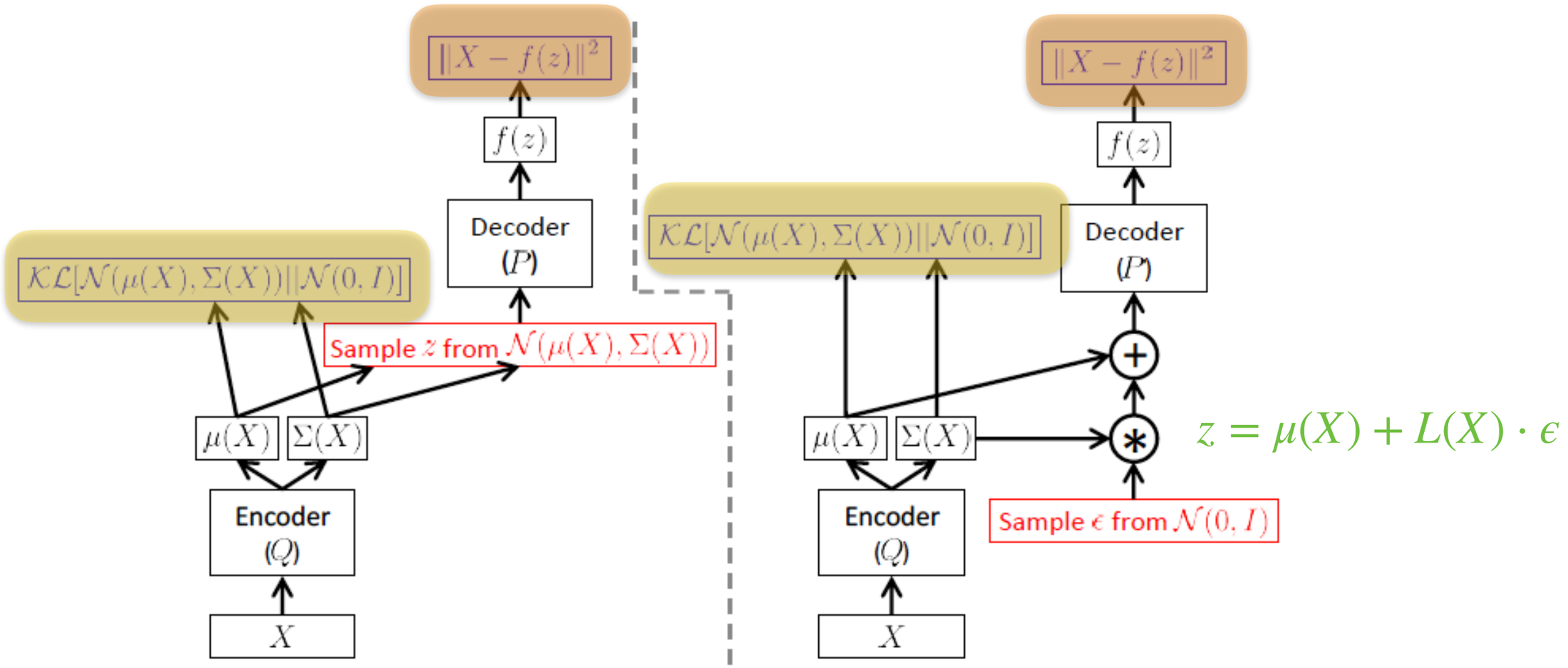


$$\min_{\phi,\theta} . \quad D_{KL}(Q(z|X;\phi)||P(z)) - \mathbb{E}_Q \log P(X|z;\theta)$$

Tutorial on variational Autoencoders, Doersch 2016

Variational Autoencoder

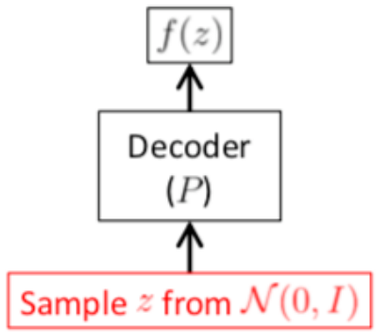
From left to right: re-parametrization trick!



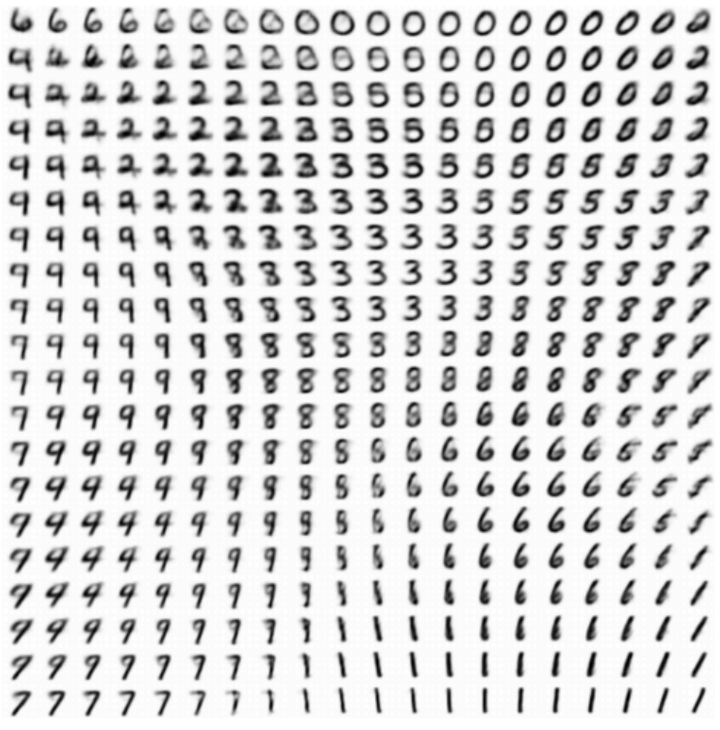
$$\min_{\phi,\theta} . \quad D_{KL}(Q(z|X;\phi)||P(z)) - \mathbb{E}_Q \log P(X|z;\theta)$$

Tutorial on variational Autoencoders, Doersch 2016

Variational Autoencoder - Test time



(a) Learned Frey Face manifold



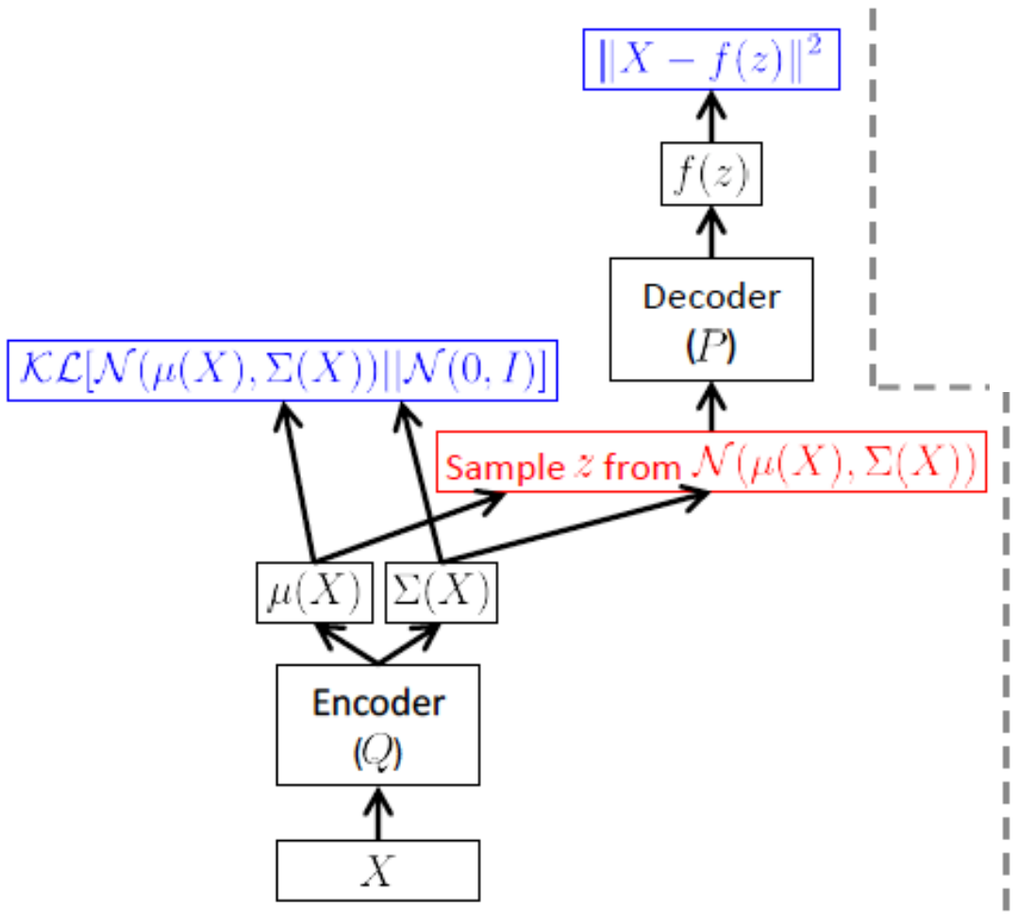
(b) Learned MNIST manifold

To predict actions for my state, shouldn't I take the state into account?

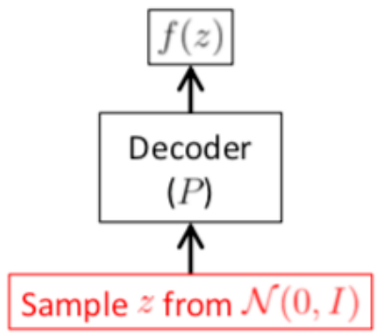
Tutorial on variational Autoencoders, Doersch 2016

(Unconditional) VAE

training time



test time

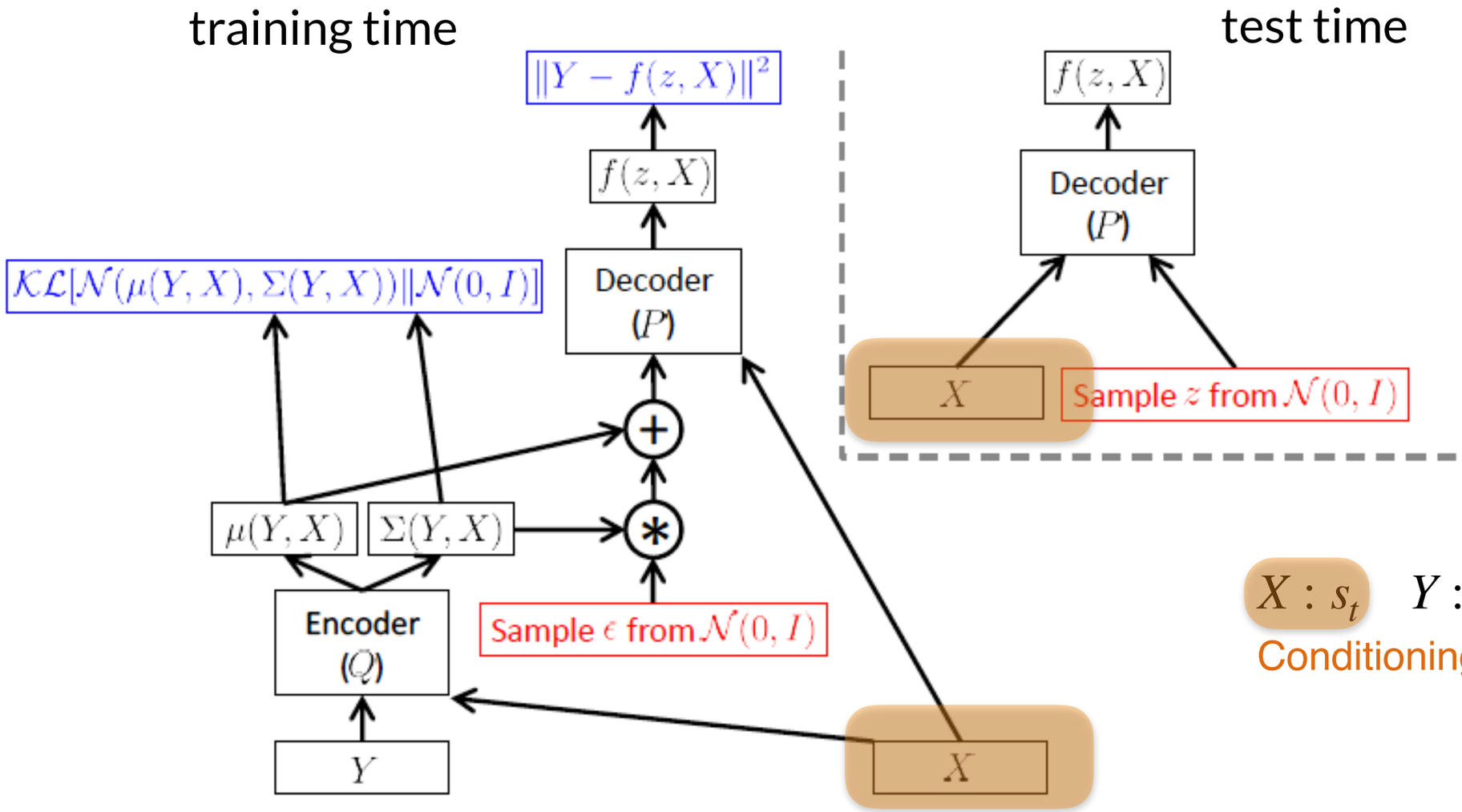


$$\min_{\phi,\theta} . \quad D_{KL}(Q(z|X;\phi), P(z)) - \mathbb{E}_Q \log P(X|z;\theta)$$

Tutorial on variational Autoencoders, Doersch 2016

Conditional VAE

I condition on X to generate Y



$X : s_t$ $Y : a_t$

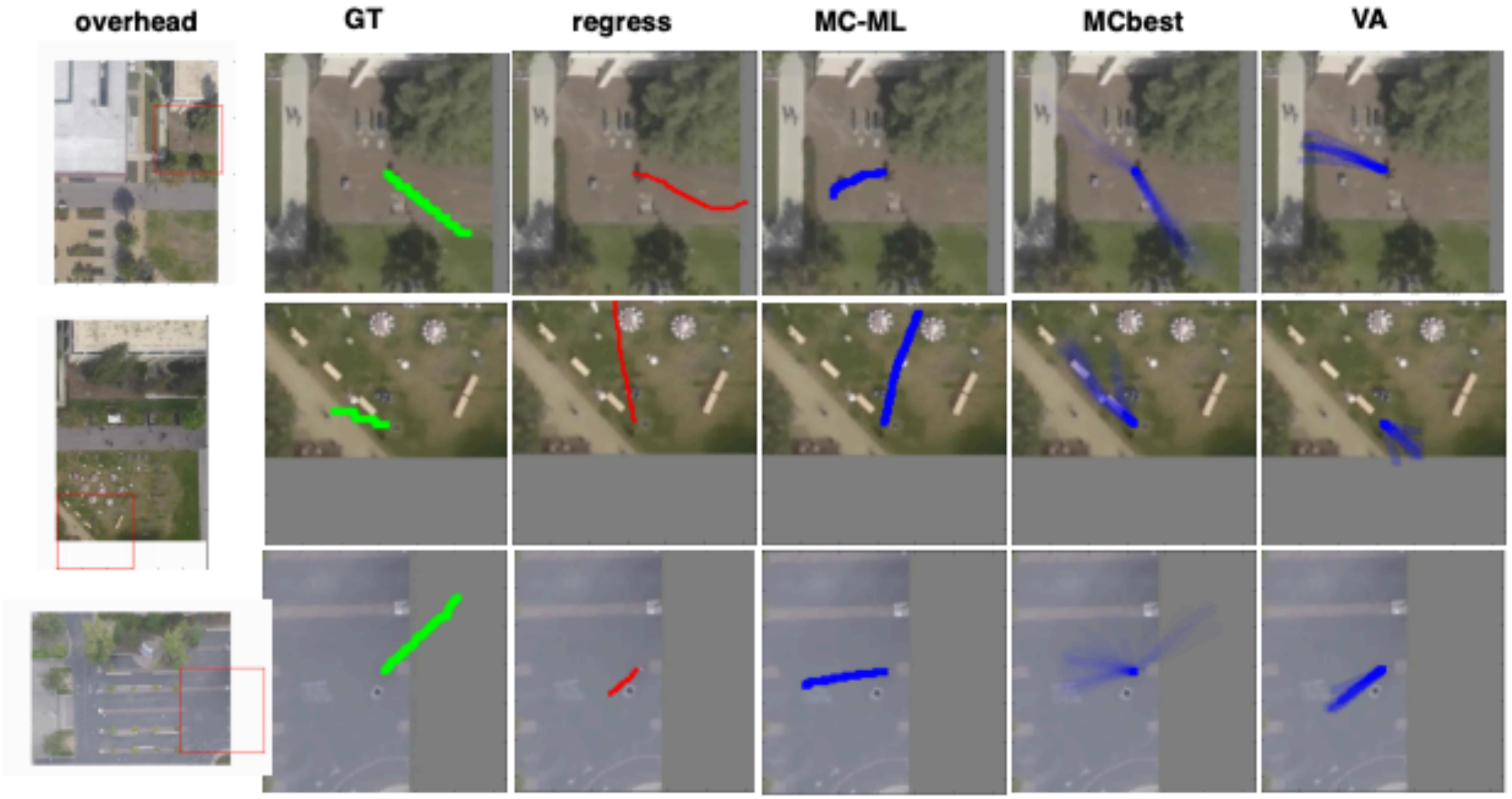
Conditioning

$$\min_{\phi,\theta} . \quad D_{KL}(Q(z|X, Y), P(z)) - \mathbb{E}_Q \log P(Y|X, z)$$

Tutorial on variational Autoencoders, Doersch 2016

Conditional VAE

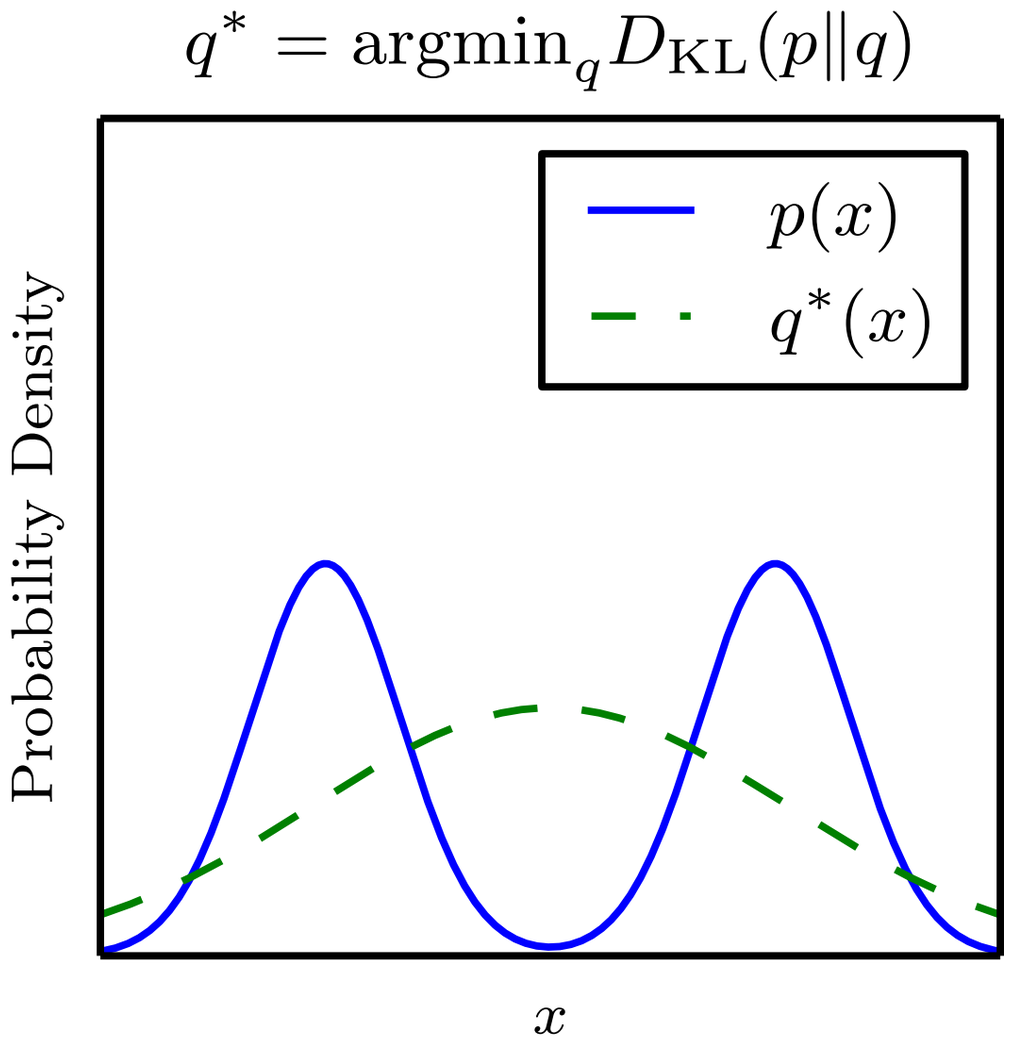
For future trajectory and frame prediction



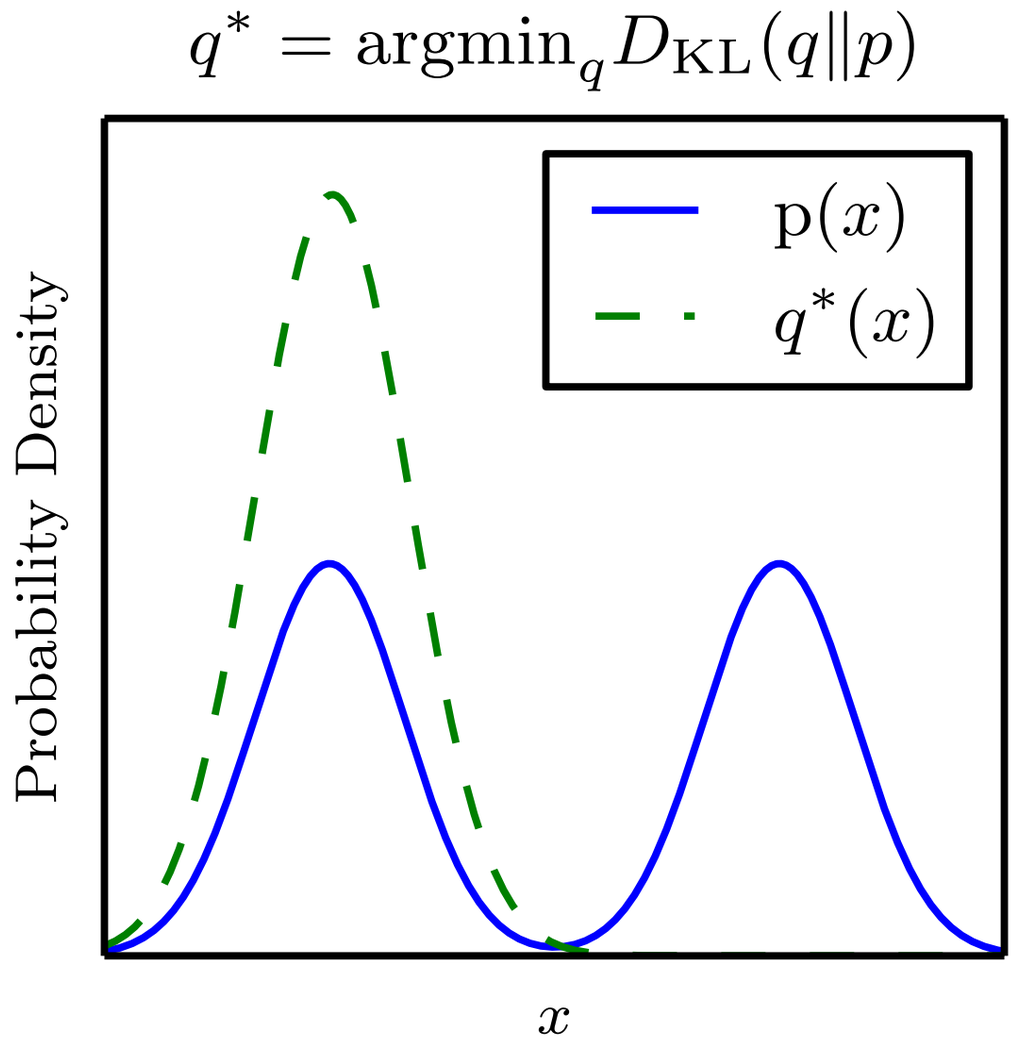
Motion Prediction Under Multimodality with Conditional Stochastic Networks, Google

GANs to the rescue

$$\mathrm{D}_{\mathrm{KL}}(P\|Q) = -\sum_{x\in X} P(x)\log\left(\frac{Q(x)}{P(x)}\right)$$

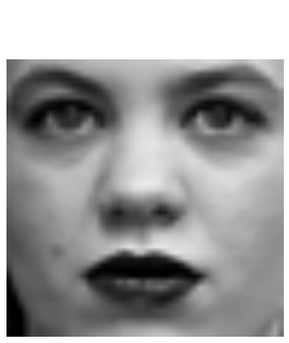


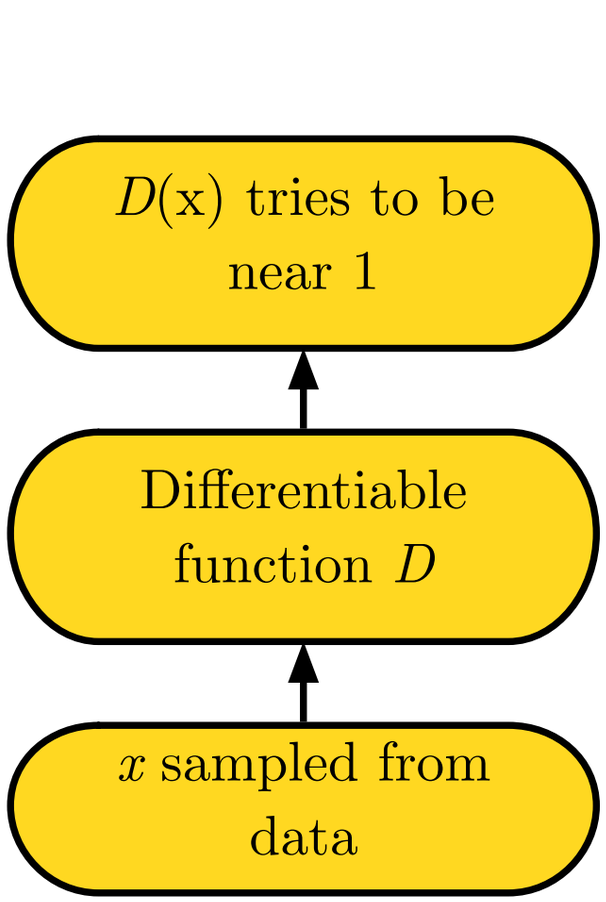
Maximum likelihood

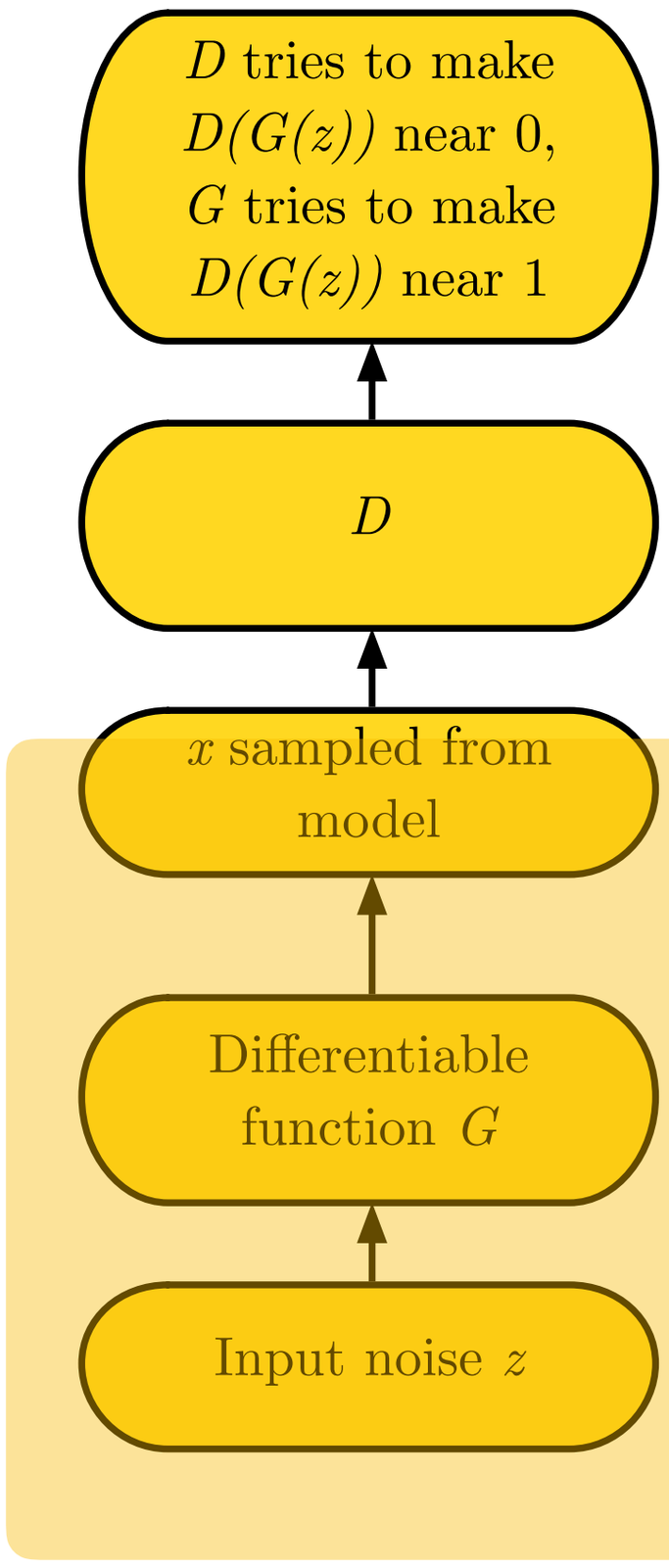


Reverse KL

Adversarial Nets Framework



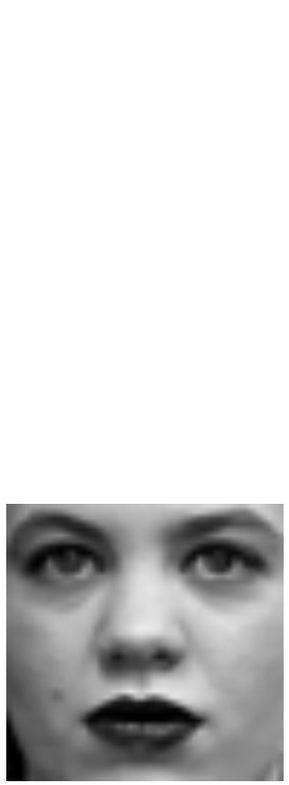


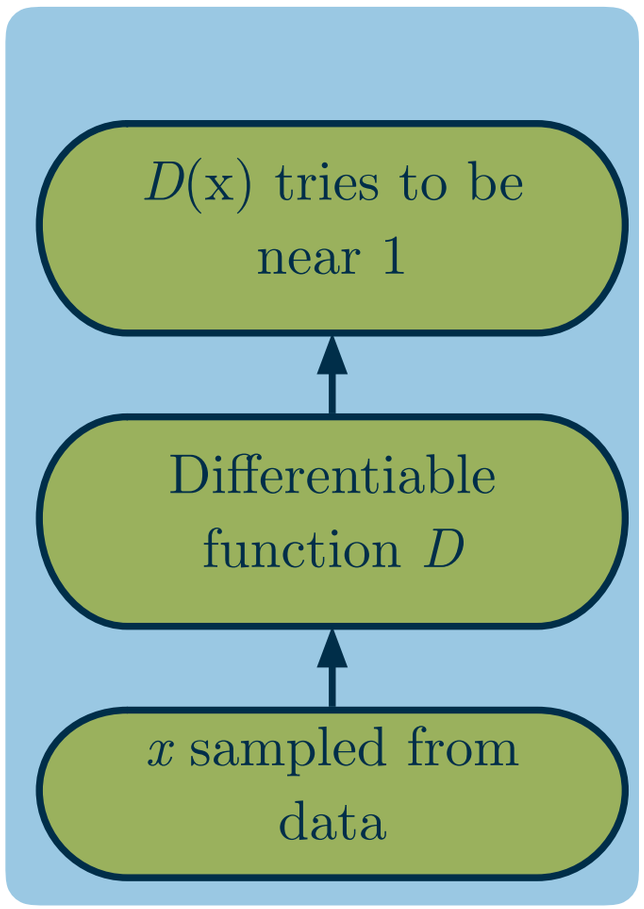


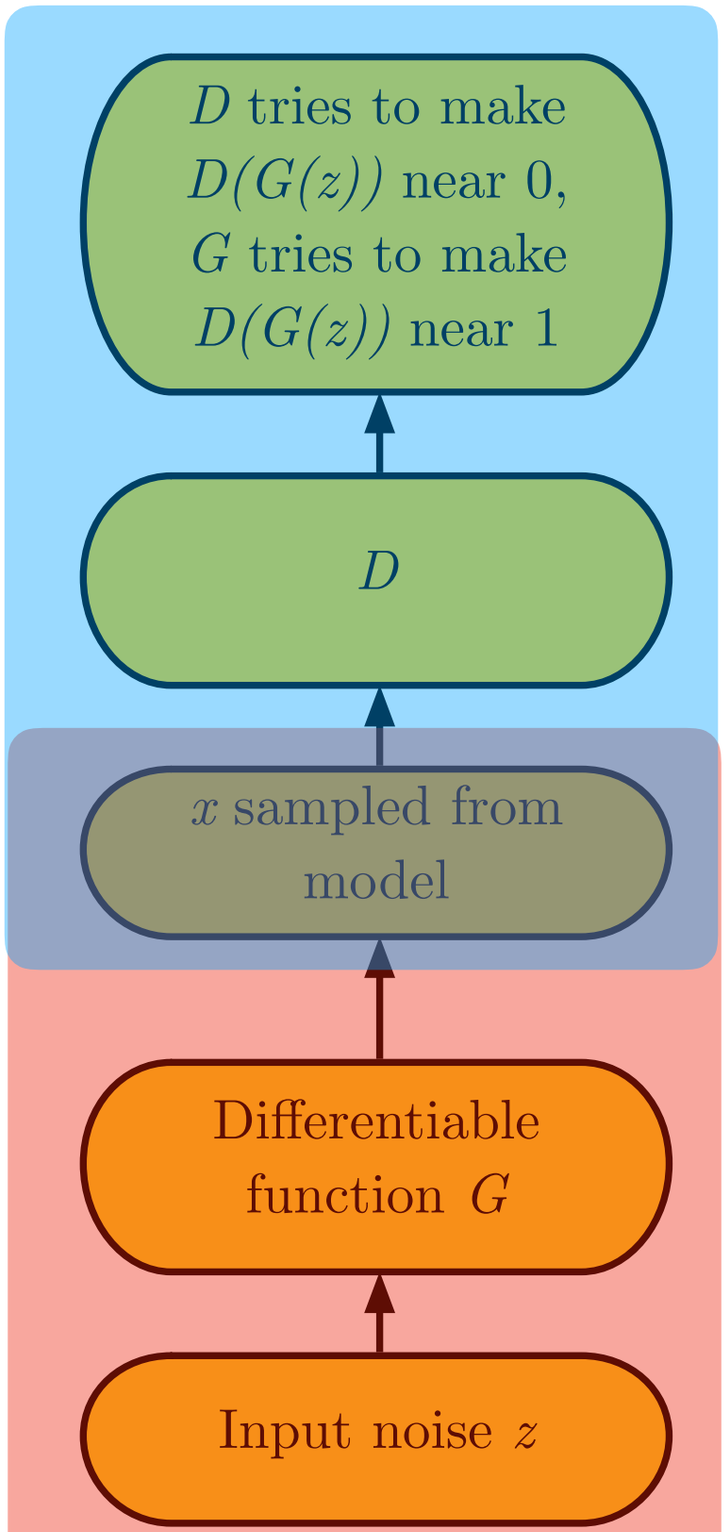
Generator G

(Goodfellow 2016)

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$





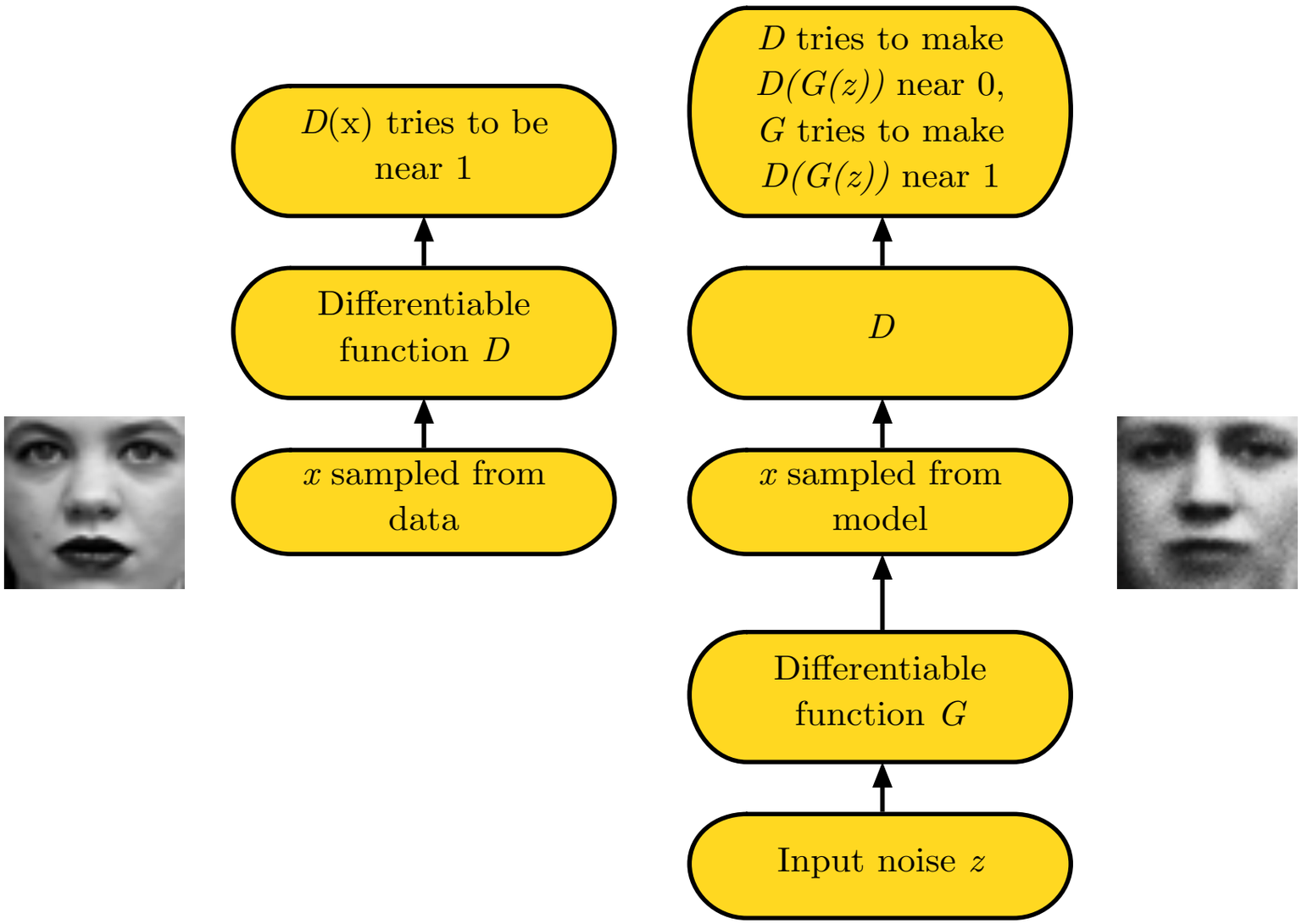


Discriminator D

Generator G

Training Procedure

- Use SGD-like algorithm of choice (Adam) on two minibatches simultaneously:
  - A minibatch of training examples
  - A minibatch of generated samples
- Optional: run k steps of one player for every step of the other player.



(Goodfellow 2016)

- What if the generator maps all noise vectors to a single super photorealistic image?
- What if we train the discriminator till convergence (it is just a supervised classifier…) and becomes perfect in distinguishing real from generated images?

A minimax game

$$\min_{G} \max_{D} V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

Optimal discriminator strategy

$$\min_{G} \max_{D} V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

Optimal discriminator strategy

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$

$$V(D,G) = \int_x p_{\text{data}}(x)\log D(x)dx + \int_z p_z(z)\log(1-D(G(z)))dz$$

Optimal discriminator strategy

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$

$$V(D,G) = \int_x p_{\text{data}}(x)\log D(x)dx + \int_z p_z(z)\log(1-D(G(z)))dz$$

$$\int_x p_{\text{data}}(x)\log D(x)dx + \int_x p_G(x)\log(1-D(x))dx$$

Optimal discriminator strategy

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$

$$V(D,G) = \int_x p_{\text{data}}(x)\log D(x)dx + \int_z p_z(z)\log(1-D(G(z)))dz$$

$$= \int_x p_{\text{data}}(x)\log D(x)dx + \int_x p_G(x)\log(1-D(x))dx$$

$$= \int_x p_{\text{data}}(x)\log D(x) + p_G(x)\log(1-D(x))dx$$

Optimal discriminator strategy

$$V(D,G) = \int_x p_{\text{data}}(x)\log D(x) + p_G(x)\log(1-D(x))dx$$

The discriminator assigns values D(x) to each image x. Let's take the derivative to see where the optimum is attained.

Optimal discriminator strategy

$$V(D,G) = \int_x p_{\text{data}}(x)\log D(x) + p_G(x)\log(1-D(x))dx$$

$$\frac{d}{dD(x)}\left(p_{\text{data}}(x)\log D(x) + p_G(x)\log(1-D(x)\right) = 0$$

Optimal discriminator strategy

$$V(D,G) = \int_x p_{\text{data}}(x)\log D(x) + p_G(x)\log(1-D(x))dx$$

$$\frac{d}{dD(x)}\left(p_{\text{data}}(x)\log D(x) + p_G(x)\log(1-D(x)\right) = 0$$

$$\Leftrightarrow p_{\text{data}}(x)\frac{1}{D(x)} - p_G(x)\frac{1}{1-D(x)} = 0$$

Optimal discriminator strategy

$$V(D,G) = \int_x p_{\text{data}}(x)\log D(x) + p_G(x)\log(1-D(x))dx$$

$$\frac{d}{dD(x)}\left(p_{\text{data}}(x)\log D(x) + p_G(x)\log(1-D(x)\right) = 0$$

$$\Leftrightarrow p_{\text{data}}(x)\frac{1}{D(x)} - p_G(x)\frac{1}{1-D(x)} = 0$$

$$\Leftrightarrow p_{\text{data}}(x)\frac{1}{D(x)} = p_G(x)\frac{1}{1-D(x)}$$

Optimal discriminator strategy

$$V(D,G) = \int_x p_{\text{data}}(x)\log D(x) + p_G(x)\log(1-D(x))dx$$

$$\frac{d}{dD(x)}\left(p_{\text{data}}(x)\log D(x) + p_G(x)\log(1-D(x)\right) = 0$$

$$\Leftrightarrow p_{\text{data}}(x)\frac{1}{D(x)} - p_G(x)\frac{1}{1-D(x)} = 0$$

$$\Leftrightarrow p_{\text{data}}(x)\frac{1}{D(x)} = p_G(x)\frac{1}{1-D(x)}$$

$$\Leftrightarrow p_{\text{data}}(x)(1-D(x)) = p_G(x)D(x)$$

Optimal discriminator strategy

$$V(D,G) = \int_x p_{\text{data}}(x)\log D(x) + p_G(x)\log(1-D(x))dx$$

$$\frac{d}{dD(x)}\left(p_{\text{data}}(x)\log D(x) + p_G(x)\log(1-D(x)\right) = 0$$

$$\Leftrightarrow p_{\text{data}}(x)\frac{1}{D(x)} - p_G(x)\frac{1}{1-D(x)} = 0$$

$$\Leftrightarrow p_{\text{data}}(x)\frac{1}{D(x)} = p_G(x)\frac{1}{1-D(x)}$$

$$\Leftrightarrow p_{\text{data}}(x)(1-D(x)) = p_G(x)D(x)$$

$$\Leftrightarrow D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}$$

Optimal generator strategy

$$C(G) = \max_D V(G, D)$$

Optimal generator strategy

$$
\begin{aligned}
C(G) &= \max_{D} V(G, D) \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D_G^*(G(z))]
\end{aligned}
$$

Optimal generator strategy

$$
\begin{aligned}
C(G) &= \max_{D} V(G, D) \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D_G^*(G(z))] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - D_G^*(x)]
\end{aligned}
$$

Optimal generator strategy

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D_G^*(G(z))] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - D_G^*(x)] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)})]
\end{aligned}
$$

Optimal generator strategy

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D_G^*(G(z))] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - D_G^*(x)] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)})] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})]
\end{aligned}
$$

Optimal generator strategy

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D_G^*(G(z))] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - D_G^*(x)] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)})] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] - \log 4 + \log 4
\end{aligned}
$$

Optimal generator strategy

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D_G^*(G(z))] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - D_G^*(x)] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)})] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] - \log 4 + \log 4 \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{2p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{2p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] - \log 4
\end{aligned}
$$

Optimal generator strategy

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D_G^*(G(z))] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - D_G^*(x)] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)})] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] - \log 4 + \log 4 \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{2p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{2p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] - \log 4 \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{\frac{p_{\text{data}}(x) + p_G(x)}{2}}] + \mathbb{E}_{x \sim p_G(x)}[\log \frac{p_{\text{G}}(x)}{\frac{p_{\text{data}}(x) + p_G(x)}{2}}] - \log 4
\end{aligned}
$$

Optimal generator strategy

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D_G^*(G(z))] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - D_G^*(x)] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(1 - \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)})] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] - \log 4 + \log 4 \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{2p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{2p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] - \log 4 \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{\frac{p_{\text{data}}(x) + p_G(x)}{2}}] + \mathbb{E}_{x \sim p_G(x)}[\log \frac{p_{\text{G}}(x)}{\frac{p_{\text{data}}(x) + p_G(x)}{2}}] - \log 4 \\
&= \mathrm{D}_{\mathrm{KL}}\left(p_{data}(x) || \frac{p_{\text{data}}(x) + p_G(x)}{2}\right) + \mathrm{D}_{\mathrm{KL}}\left(p_G(x) \| \frac{p_{\text{data}}(x) + p_G(x)}{2}\right) - \log 4
\end{aligned}
$$

Optimal generator strategy

$$
\begin{aligned}
C(G) &= \max_D V(G,D) \\
&= \mathbb{E}_{x\sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D_G^*(G(z))] \\
&= \mathbb{E}_{x\sim p_{data}(x)}[\log D_G^*(x)] + \mathbb{E}_{x\sim p_G(x)}[\log(1-D_G^*(x)] \\
&= \mathbb{E}_{x\sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_G(x)}] + \mathbb{E}_{x\sim p_G(x)}[\log(1-\frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_G(x)})] \\
&= \mathbb{E}_{x\sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_G(x)}] + \mathbb{E}_{x\sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x)+p_G(x)})] \\
&= \mathbb{E}_{x\sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_G(x)}] + \mathbb{E}_{x\sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x)+p_G(x)})] - \log 4 + \log 4 \\
&= \mathbb{E}_{x\sim p_{data}(x)}[\log \frac{2p_{\text{data}}(x)}{p_{\text{data}}(x)+p_G(x)}] + \mathbb{E}_{x\sim p_G(x)}[\log(\frac{2p_{\text{G}}(x)}{p_{\text{data}}(x)+p_G(x)})] - \log 4 \\
&= \mathbb{E}_{x\sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{\frac{p_{\text{data}}(x)+p_G(x)}{2}}] + \mathbb{E}_{x\sim p_G(x)}[\log \frac{p_{\text{G}}(x)}{\frac{p_{\text{data}}(x)+p_G(x)}{2}}] - \log 4 \\
&= \mathrm{D}_{\mathrm{KL}}\left(p_{data}(x)||\frac{p_{\text{data}}(x)+p_G(x)}{2}\right) + \mathrm{D}_{\mathrm{KL}}\left(p_G(x)\|\frac{p_{\text{data}}(x)+p_G(x)}{2}\right) - \log 4 \\
&= 2\mathrm{D}_{\mathrm{JSD}}\left(p_{data}(x)||p_G(x)\right) - \log 4
\end{aligned}
$$

Optimal generator strategy

$$
\begin{aligned}
C(G) &= \max_{D} V(G, D) \\
&= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}] + \mathbb{E}_{x \sim p_G(x)}[\log(\frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_G(x)})] \\
&= 2\mathrm{D}_{\mathrm{JSD}}\left(p_{data}(x) \,||\, p_G(x)\right) - \log 4
\end{aligned}
$$

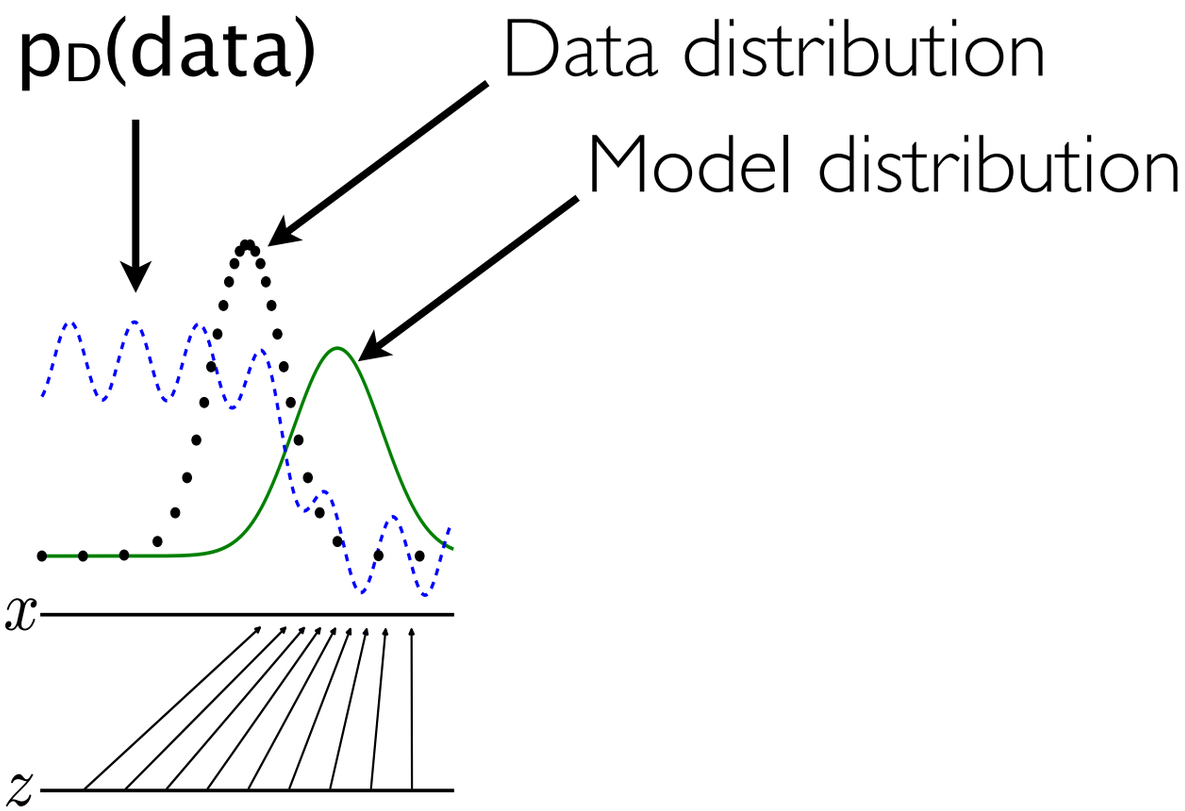
Since $\mathrm{D}_{\mathrm{JSD}} \geq 0, \quad C(G) \geq -\log 4$

By setting $P_G(x) = \mathrm{p}_{\text{data}}(\mathrm{x})$ in the equation above, we get:

$$
C(G) = \mathbb{E}_{x \sim \mathrm{p}_{\text{data}}(x)} \log \frac{1}{2} + \mathbb{E}_{x \sim p_G(x)} \log \frac{1}{2} = -\log 4
$$

Thus generator achieves the optimum when $P_G(x) = \mathrm{p}_{\text{data}}(\mathrm{x})$.

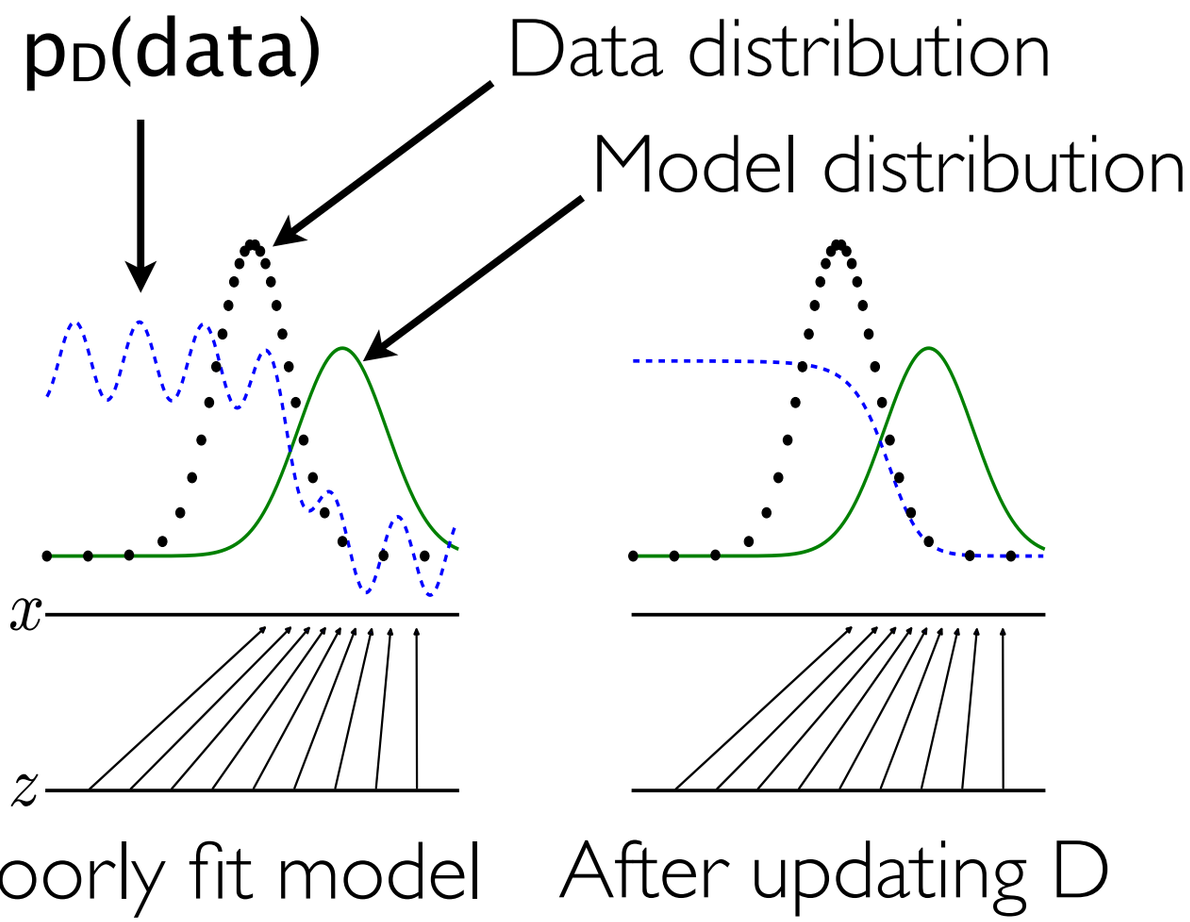
Learning process



Poorly fit model

Diagram from Ian Goodfellow

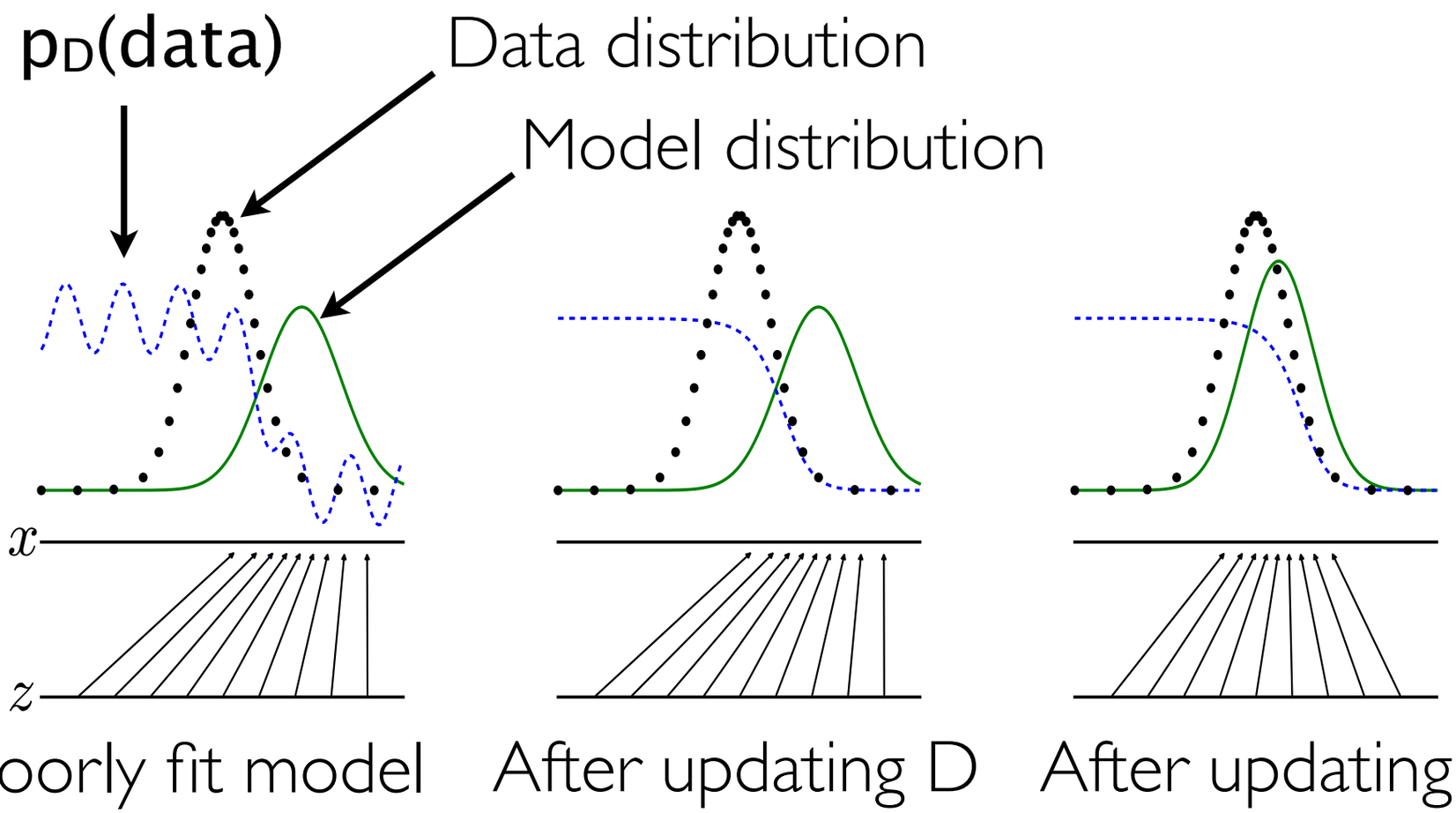
Learning process



Poorly fit model After updating D

Diagram from Ian Goodfellow

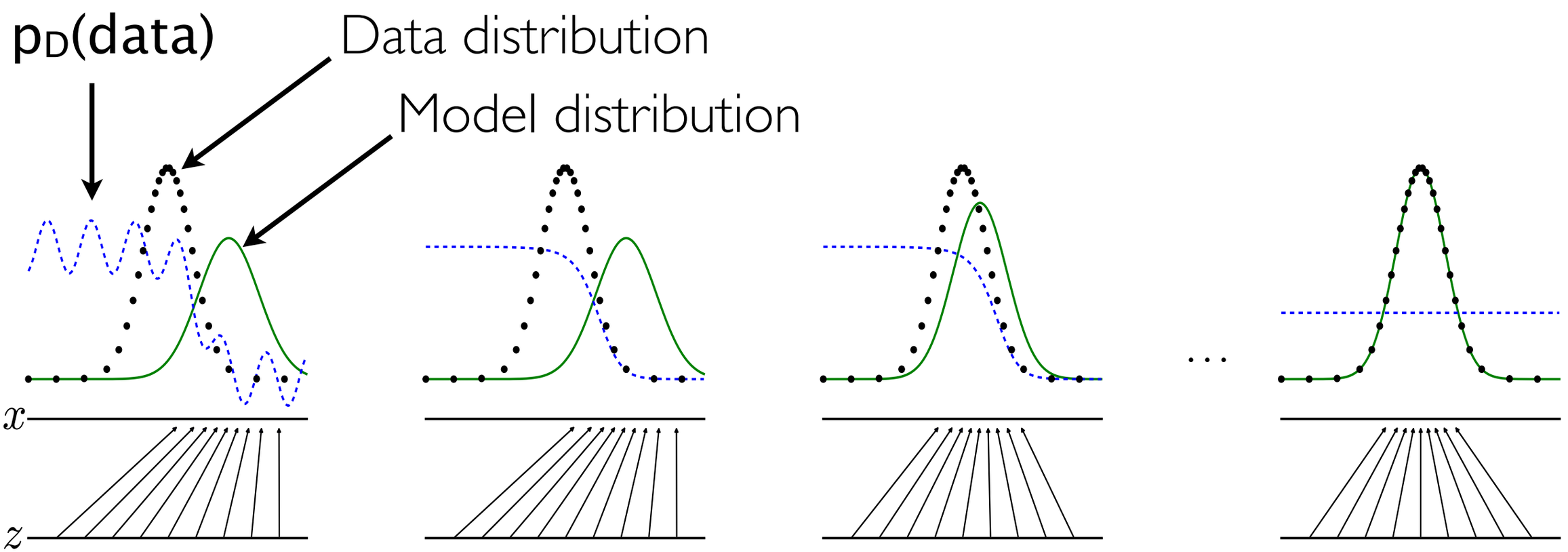
Learning process



Poorly fit model After updating D After updating G

Diagram from Ian Goodfellow

Learning process



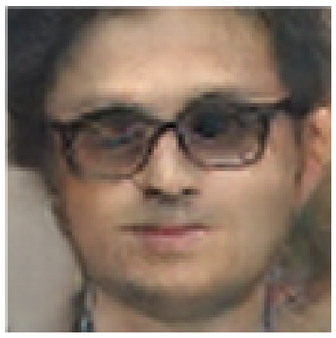
Poorly fit model | After updating D | After updating G | Mixed strategy equilibrium

Diagram from Ian Goodfellow

DCGANs for LSUN Bedrooms

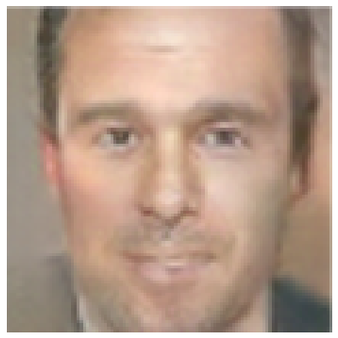
(Radford et al 2015)

Vector Space Arithmetic



Man with glasses

-



Man

+

Woman

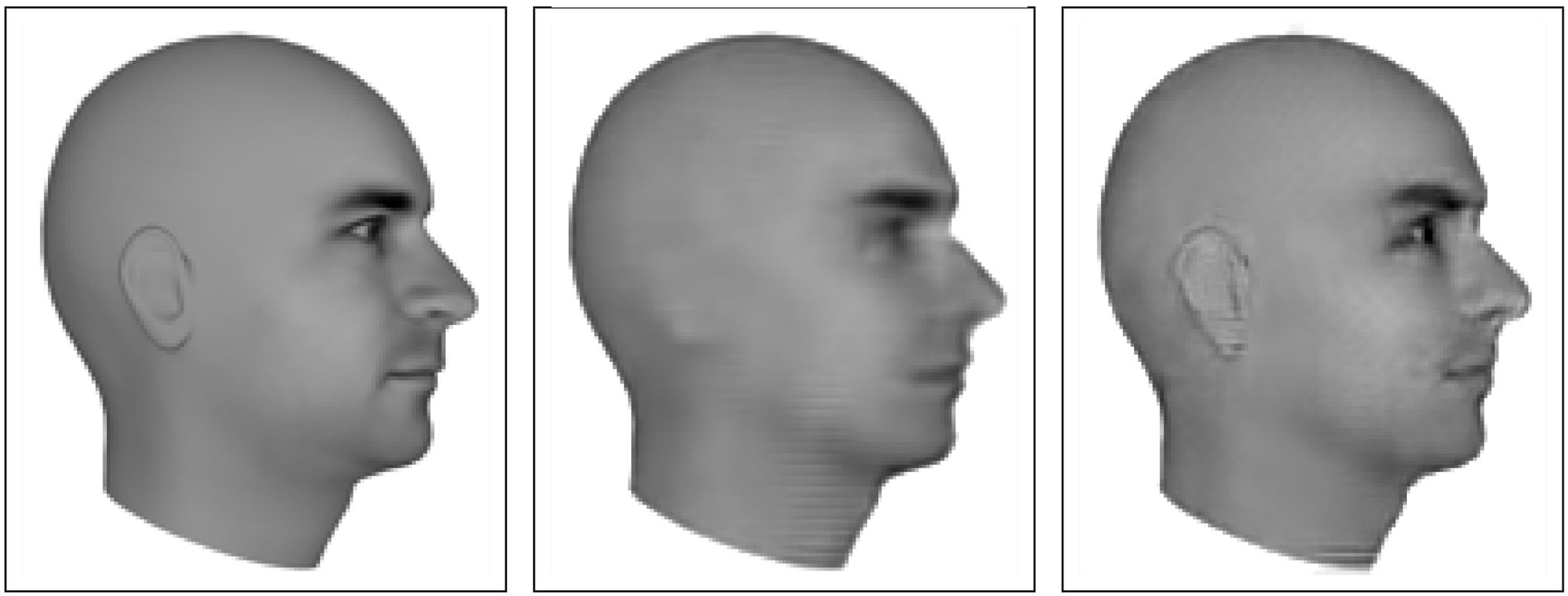
=

Woman with Glasses

(Radford et al, 2015)

Next Video Frame Prediction

Groundtruth | Max. Likelihood | Adversarial



(Lotter et al 2016)

Generative Adversarial Imitation learning

The policy network will be our generator, that conditions on the state:

$$\pi_\theta(s) \rightarrow a$$

Generative Adversarial Imitation learning

Find a policy π_θ that makes it impossible for a discriminator network to distinguish between state-actions from the expert demonstrations and state-actions visited by the learnt policy π_θ:

$$\min_{\pi_\theta} \max_{D} V(D, G) = \mathbb{E}_{(s,a)\sim \text{Demo}}[\log D(s, a)] + \mathbb{E}_{(s,a)\sim \pi_\theta}[\log(1 - D(s, a))]$$

Generative Adversarial Imitation learning

Find a policy π_θ that makes it impossible for a discriminator network to distinguish between state-actions from the expert demonstrations and state-actions visited by the learnt policy π_θ:

$$\min_{\pi_\theta} \max_{D} V(D, G) = \mathbb{E}_{(s,a)\sim \text{Demo}}[\log D(s, a)] + \mathbb{E}_{(s,a)\sim \pi_\theta}[\log(1 - D(s, a))]$$

The reward for the policy optimization is how well I matched the demonstrator's trajectory distribution, else, how well I confused the discriminator.

$$r(s, a) = \log D(s, a), (s, a) \sim \pi_\theta$$

(D outputs 1 if states come from the expert policy)

Generative Adversarial Imitation learning

Find a policy π_θ that makes it impossible for a discriminator network to distinguish between state-actions from the expert demonstrations and state-actions visited by the learnt policy π_θ:

$$\min_{\pi_\theta} \max_{D} V(D,G) = \mathbb{E}_{(s,a)\sim \text{Demo}}[\log D(s,a)] + \mathbb{E}_{(s,a)\sim \pi_\theta}[\log(1-D(s,a))]$$

The reward for the policy optimization is how well I matched the demonstrator's trajectory distribution, else, how well I confused the discriminator.

$$r(s,a) = \log D(s,a), (s,a) \sim \pi_\theta$$

(D outputs 1 if states come from the expert policy)

$$\min_{G} \max_{D} V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$

Generative Adversarial Imitation learning

Find a policy π_θ that makes it impossible for a discriminator network to distinguish between state-actions from the expert demonstrations and state-actions visited by the learnt policy π_θ

$$\min_{\pi_\theta} \max_{D} V(D,G) = \mathbb{E}_{(s,a)\sim \text{Demo}}[\log D(s,a)] + \mathbb{E}_{(s,a)\sim \pi_\theta}[\log(1-D(s,a))]$$

The reward for the policy optimization is how well I matched the demonstrator's trajectory distribution, else, how well I confused the discriminator.

$$r(s,a) = \log D(s,a), (s,a) \sim \pi_\theta$$

Q: how would we change the above if the action space between expert and imitator are different?

$$\min_{G} \max_{D} V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$

GAIL-action spaces differ between teacher and learner

Find a policy π_θ that makes it impossible for a discriminator network to distinguish between states from the expert demonstrations and states visited by the learnt policy π_θ

$$\min_{\pi_\theta} \max_{D} V(D,G) = \mathbb{E}_{(s)\sim \text{Demo}}[\log D(s)] + \mathbb{E}_{(a,s)\sim \pi_\theta}[\log(1-D(s))]$$

Reward for the policy optimization is how well I matched the demo trajectory distribution, else, how well I confused the discriminator.

$$r(s) = \log D(s)$$

$$\min_{G} \max_{D} V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$

Generative Adversarial Imitation Learning

Jonathan Ho
Stanford University
hoj@cs.stanford.edu

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

NIPS 2016

Algorithm 1 Generative adversarial imitation learning

Input: Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminator parameters θ_0, w_0
for $i = 0, 1, 2, \ldots$ **do**
  Sample trajectories $\tau_i \sim \pi_{\theta_i}$
  Update the discriminator parameters from w_i to w_{i+1} with the gradient

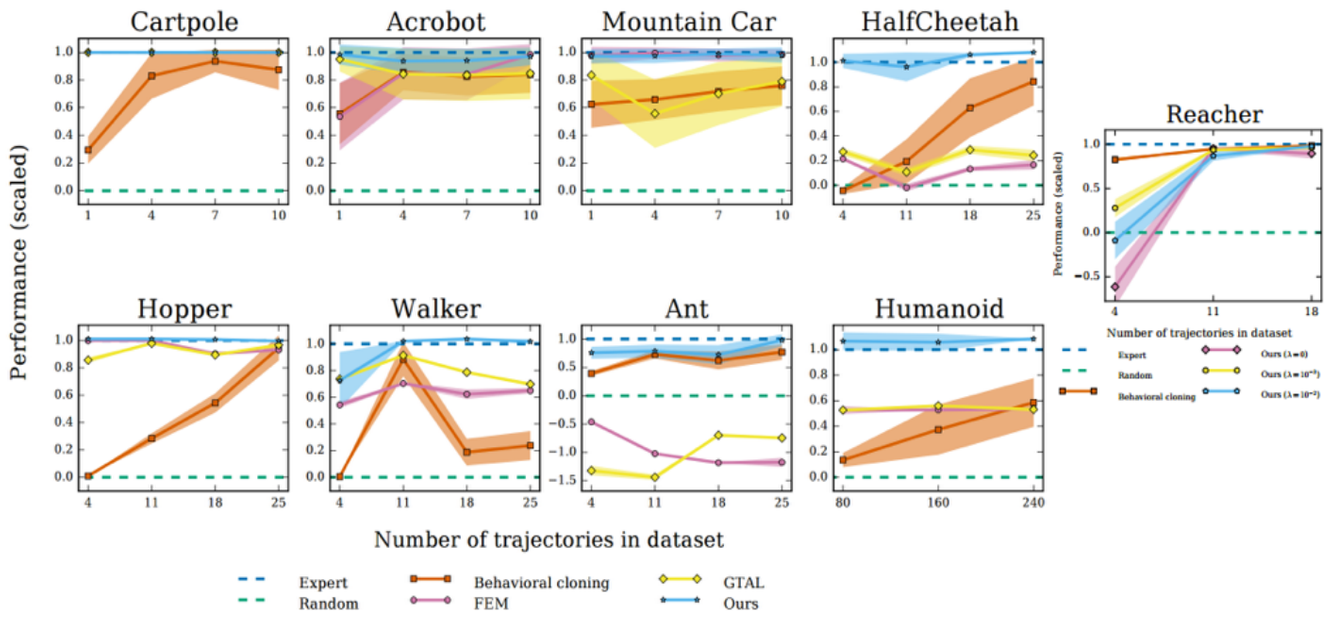
$$\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s,a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s,a))] \tag{17}$$

  Take a policy step from θ_i to θ_{i+1}, using the TRPO rule with cost function $\log(D_{w_{i+1}}(s,a))$. Specifically, take a KL-constrained natural gradient step with

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

end for

Generative Adversarial Imitation learning



- GAIL: a reinforcement learning method with a reward based on trajectory distribution matching between the agent and an expert. This does seem to be the right objective.

- BC: reduces imitation learning to supervised learning for individual actions. Not the right cost function: the training objective is open loop action prediction; the test objective is closed loop control, that is why we suffer from distribution shift.

- GAIL performs better than behaviour cloning but it requires MORE interactions with the environment.