DSC291: Machine Learning with Few Labels

Unsupervised Learning
Reinforcement Learning

Zhiting Hu

Lecture 20, May 20, 2024

UC San Diego
HALICIOĞLU DATA SCIENCE INSTITUTE

Plan for the rest of the course

- VAEs (today)
- Reinforcement Learning
- Unified Perspective
- Other topics (if time permits):
  - Diffusion models
  - World models
  - …

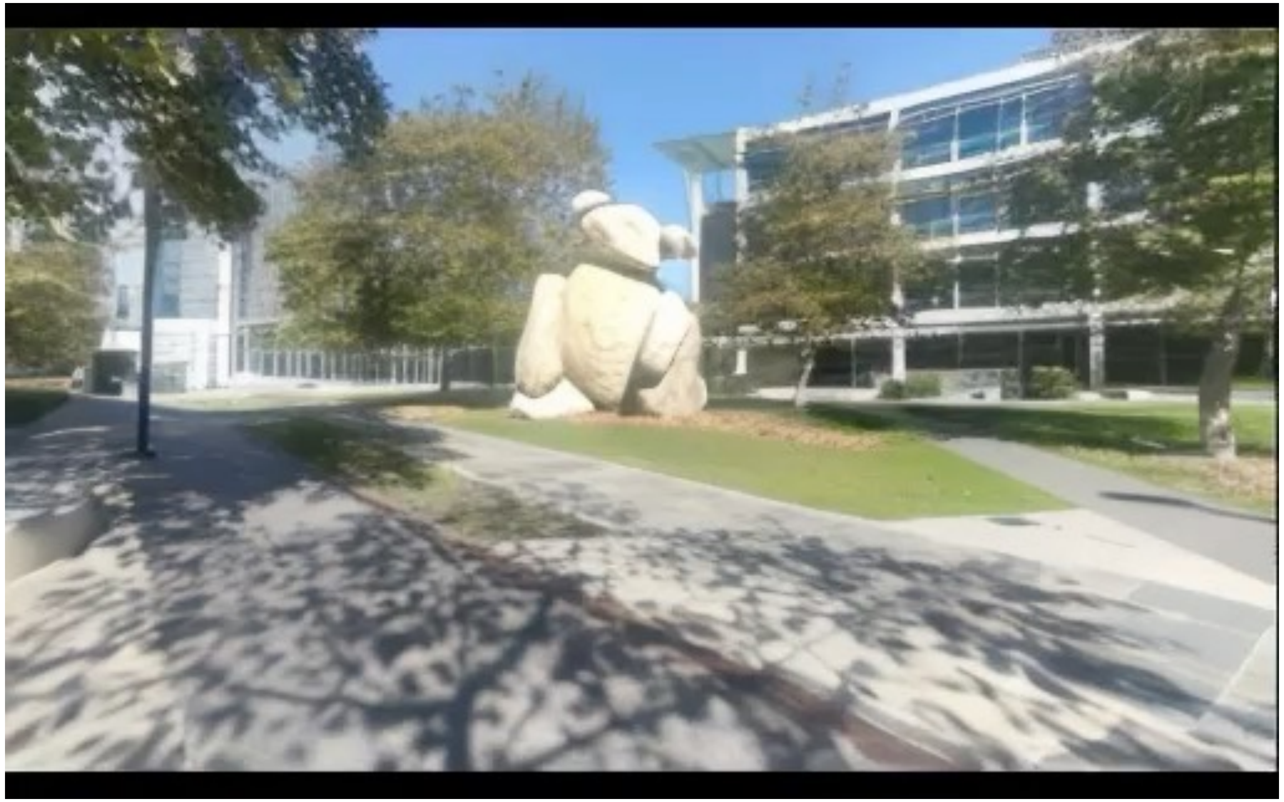
Plan for the rest of the course

- VAEs (today)
- Reinforcement Learning
- Unified Perspective
- Other topics (if time permits):
  - Diffusion models
  - World models
  - …

Real-time control of world state:
Action 1: The red car moves along the path
Action 2: Explosion happens
Action 3: The red car continues to move

Plan for the rest of the course

- VAEs (today)
- Reinforcement Learning
- Unified Perspective
- Other topics (if time permits):
  - Diffusion models
  - World models
  - ...



Action: Turn left

Plan for the rest of the course

- VAEs (
- Reinfo
- Unifie
- Other
  - Diffu
  - Worl
  - ...

Plan for the rest of the course

- VAEs (today)
- Reinforcement Learning
- Unified Perspective
- Other topics (if time permits):
  - Diffusion models
  - World models
  - …

- Homework:
  - To be released on Wed (EM, VI related)
  - Due in the final week

Variational Autoencoders (VAEs)

Variational Auto-Encoders (VAEs)

VAEs are a combination of the following ideas:

- Variational Inference
  - ELBO
- Variational distribution parametrized as neural networks
- Reparameterization trick

[Courtesy: Dhruv, CS 4803]

Variational Auto-Encoders (VAEs)

- Model $p_\theta(\boldsymbol{x}, \boldsymbol{z}) = p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$
  - $p_\theta(\boldsymbol{x}|\boldsymbol{z})$: a.k.a., generative model, generator, (probabilistic) decoder, …
  - $p(\boldsymbol{z})$: prior, e.g., Gaussian
- Assume variational distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$
  - E.g., a Gaussian distribution parameterized as **deep neural networks**
  - a.k.a, recognition model, inference network, (probabilistic) encoder, …
- ELBO:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}, \boldsymbol{z})] + \mathrm{H}(q_\phi(\boldsymbol{z}|\boldsymbol{x}))$$

$$= \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z}))$$

Reconstruction

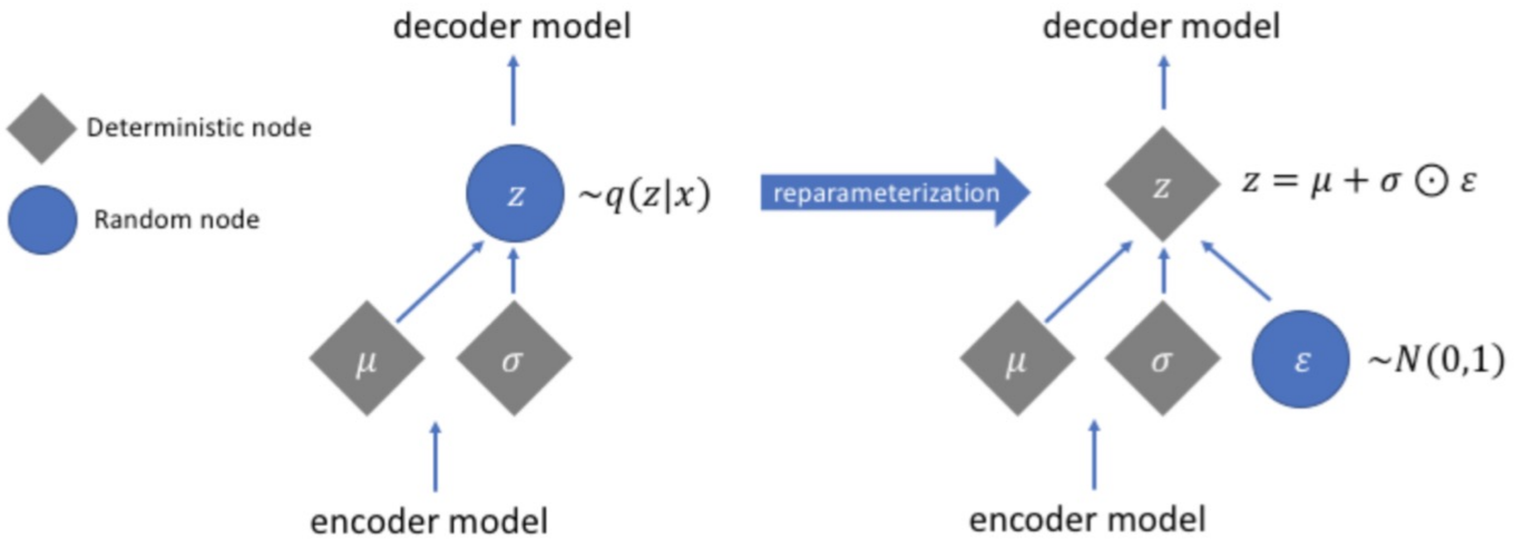
Divergence from prior
(KL divergence between two Guassians
has an analytic form)

Variational Auto-Encoders (VAEs)

- ELBO:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}, \boldsymbol{z})] + \mathrm{H}(q_\phi(\boldsymbol{z}|\boldsymbol{x}))$$
$$= \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z}))$$

- Reparameterization:
  - $[\boldsymbol{\mu}; \boldsymbol{\sigma}] = f_\phi(\boldsymbol{x})$ (a neural network)
  - $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \boldsymbol{N}(\mathbf{0}, \mathbf{1})$



Variational Auto-Encoders (VAEs)

- ELBO:

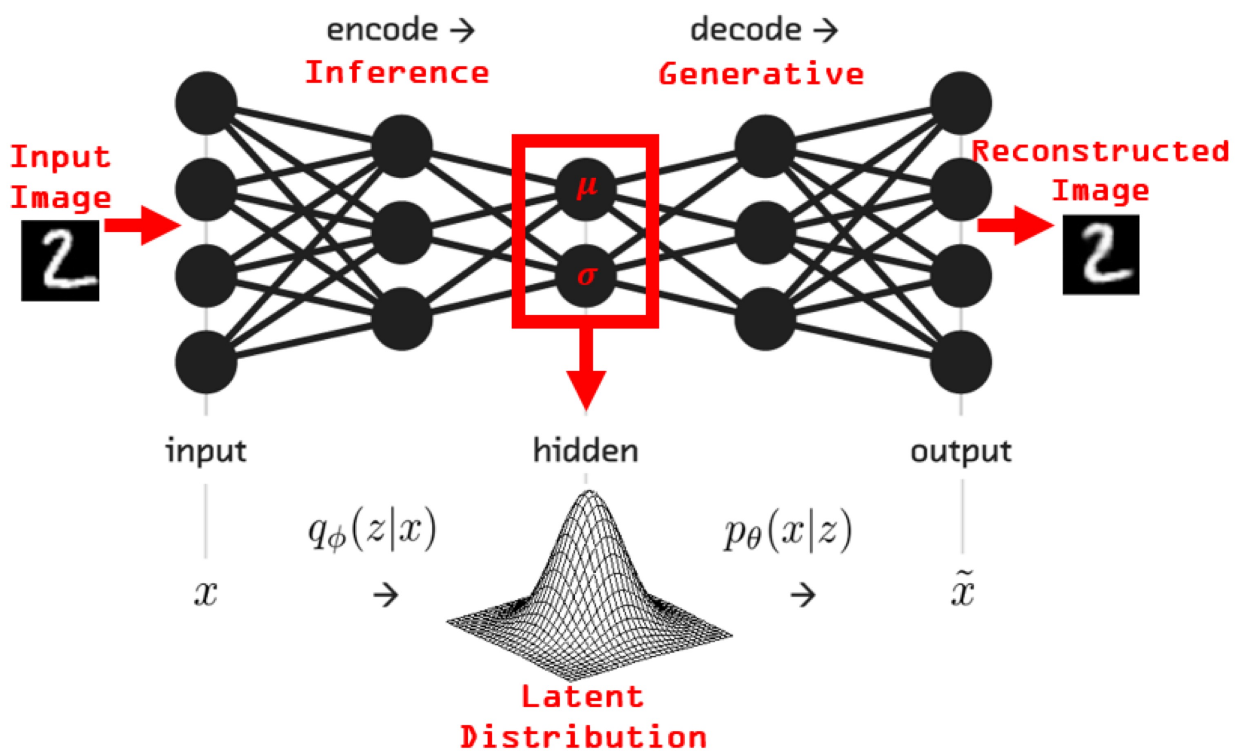
$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}, \boldsymbol{z})] + \mathrm{H}(q_\phi(\boldsymbol{z}|\boldsymbol{x}))$$
$$= \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z}))$$

- Reparameterization:
  - $[\boldsymbol{\mu}; \boldsymbol{\sigma}] = f_\phi(\boldsymbol{x})$ (a neural network)
  - $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \boldsymbol{N}(\mathbf{0}, \mathbf{1})$

$$\nabla_{\boldsymbol{\phi}} \mathcal{L} = \mathrm{E}_{\epsilon \sim N(\mathbf{0},\mathbf{1})}\left[\nabla_{\boldsymbol{z}}\left[\log p_\theta(\boldsymbol{x}, \boldsymbol{z}) - \log q_\phi(\boldsymbol{z}|\boldsymbol{x})\right] \nabla_\phi z(\epsilon, \boldsymbol{\phi})\right]$$

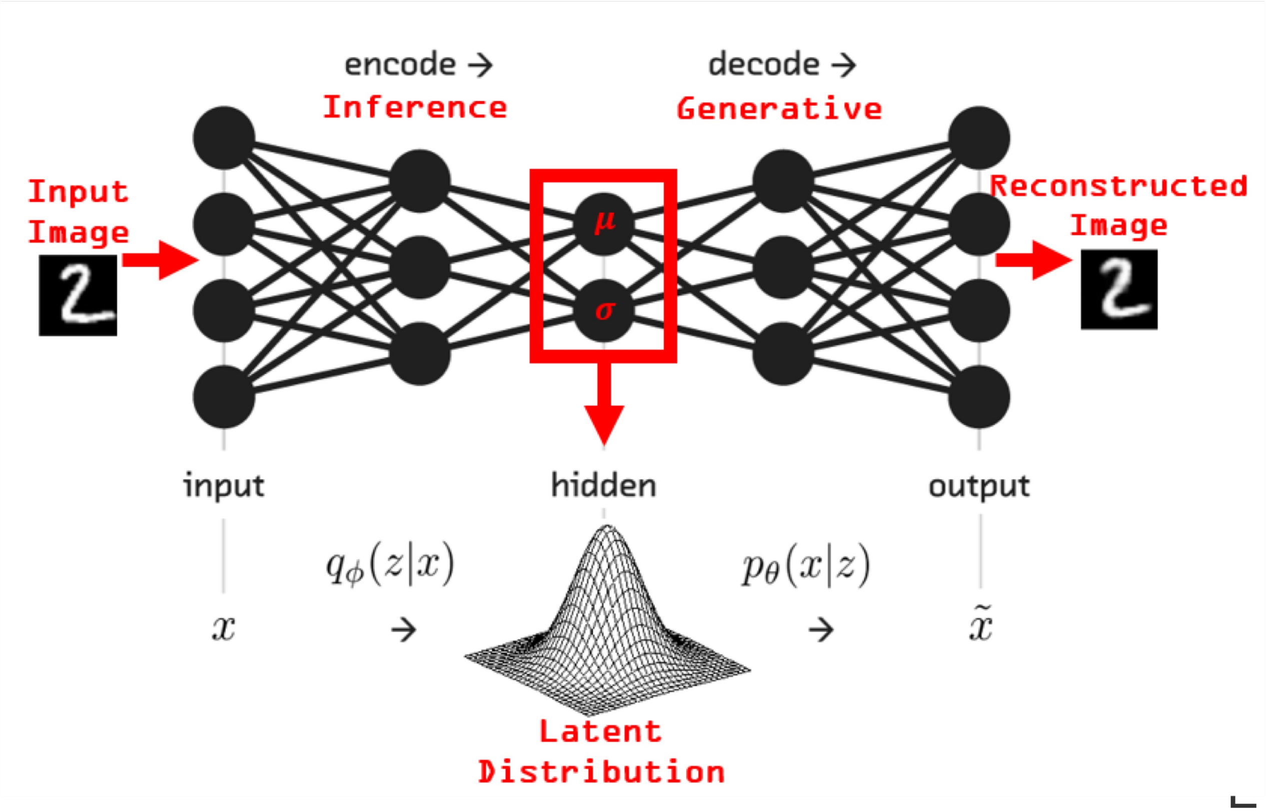
$$\nabla_\theta \mathcal{L} = \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\nabla_\theta \log p_\theta(\boldsymbol{x}, \boldsymbol{z})]$$

Example: VAEs for images



[https://www.kaggle.com/rvislaywade/visualizing-mnist-using-a-variational-autoencoder]

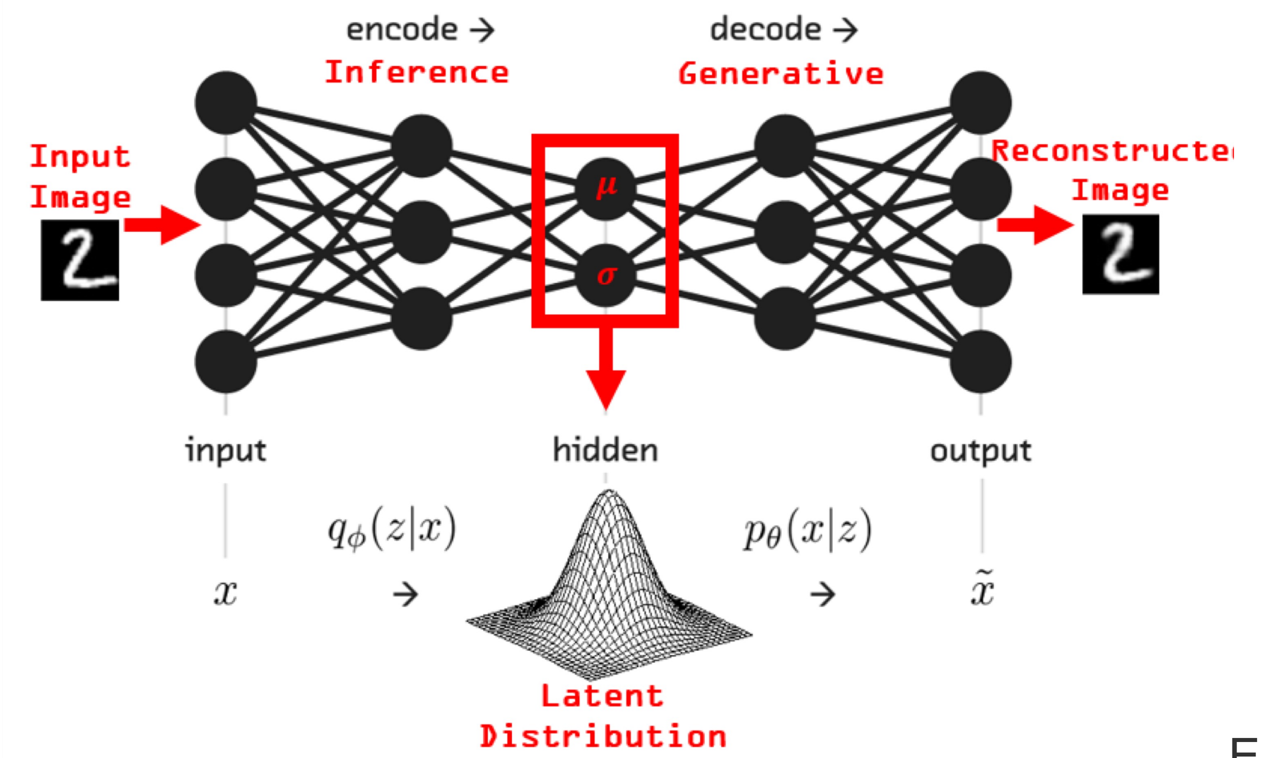
Example: VAEs for images

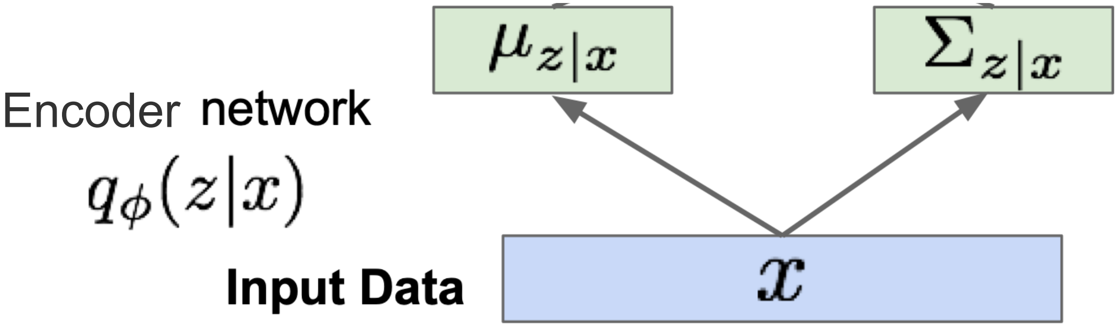


Input Data x

[Courtesy: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n]

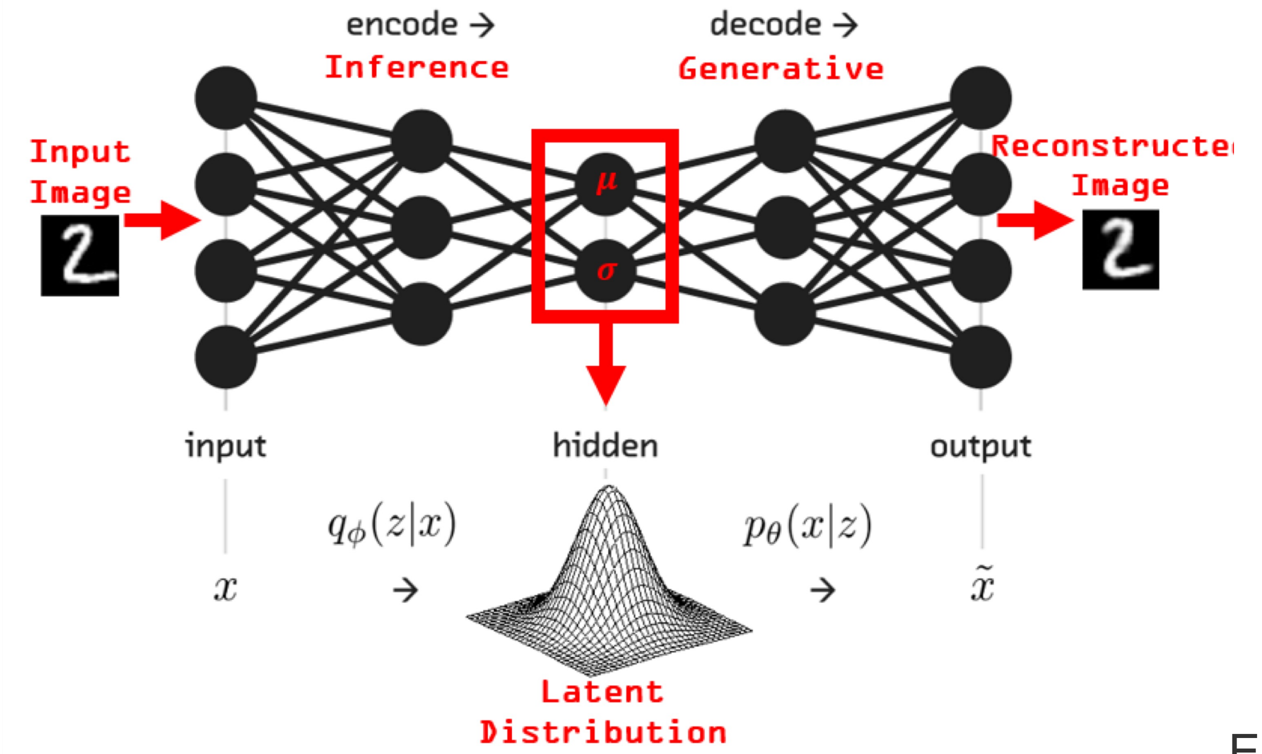
Example: VAEs for images

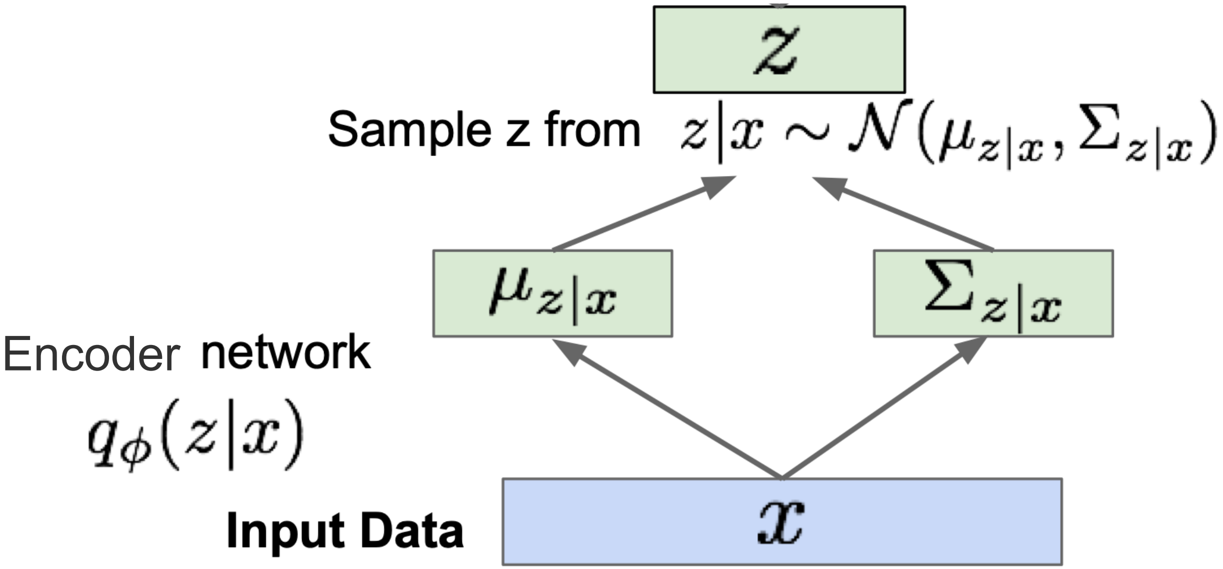




[Courtesy: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n]

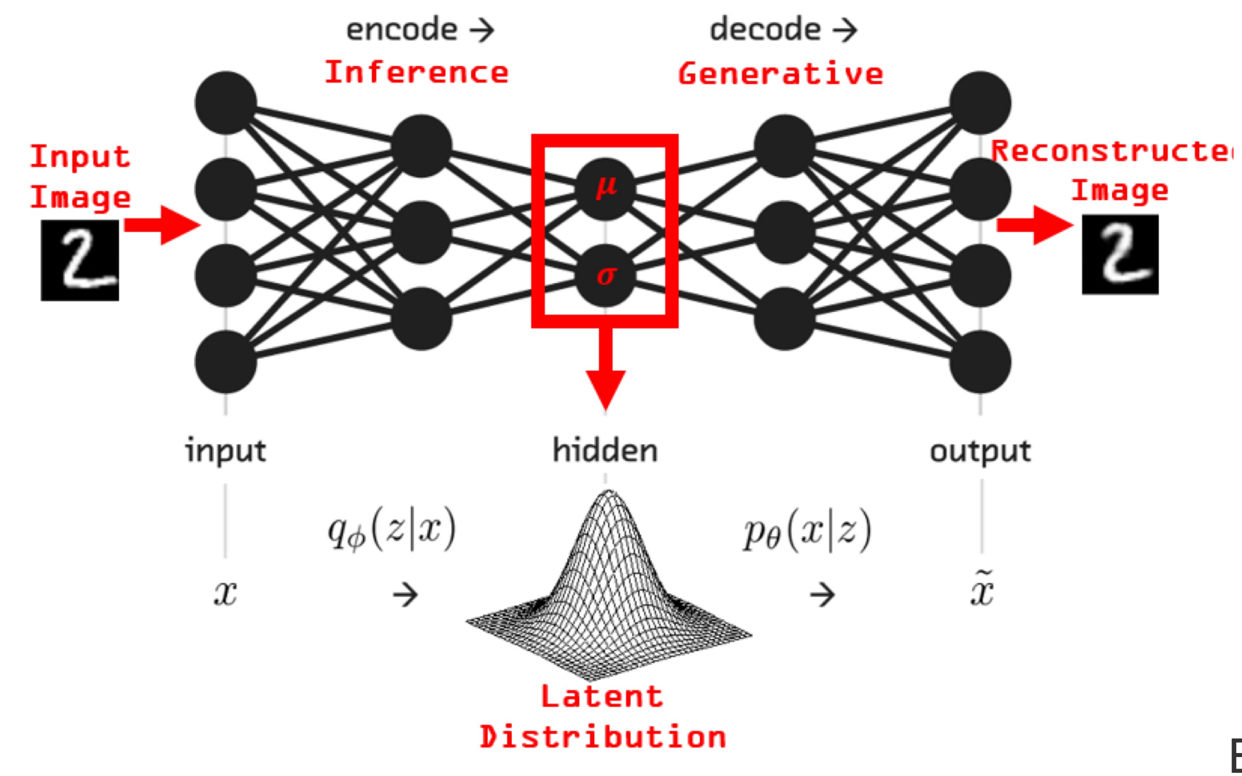
Example: VAEs for images

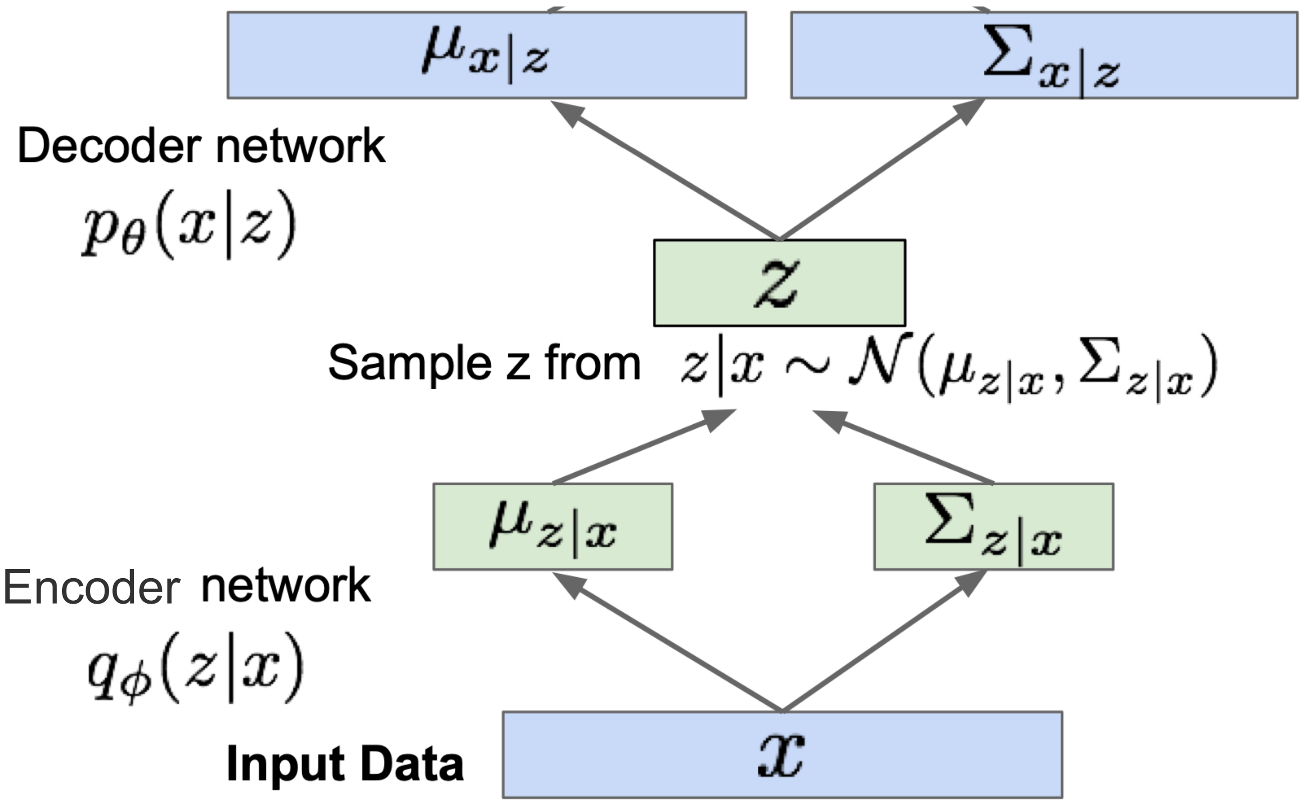




[Courtesy: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n]

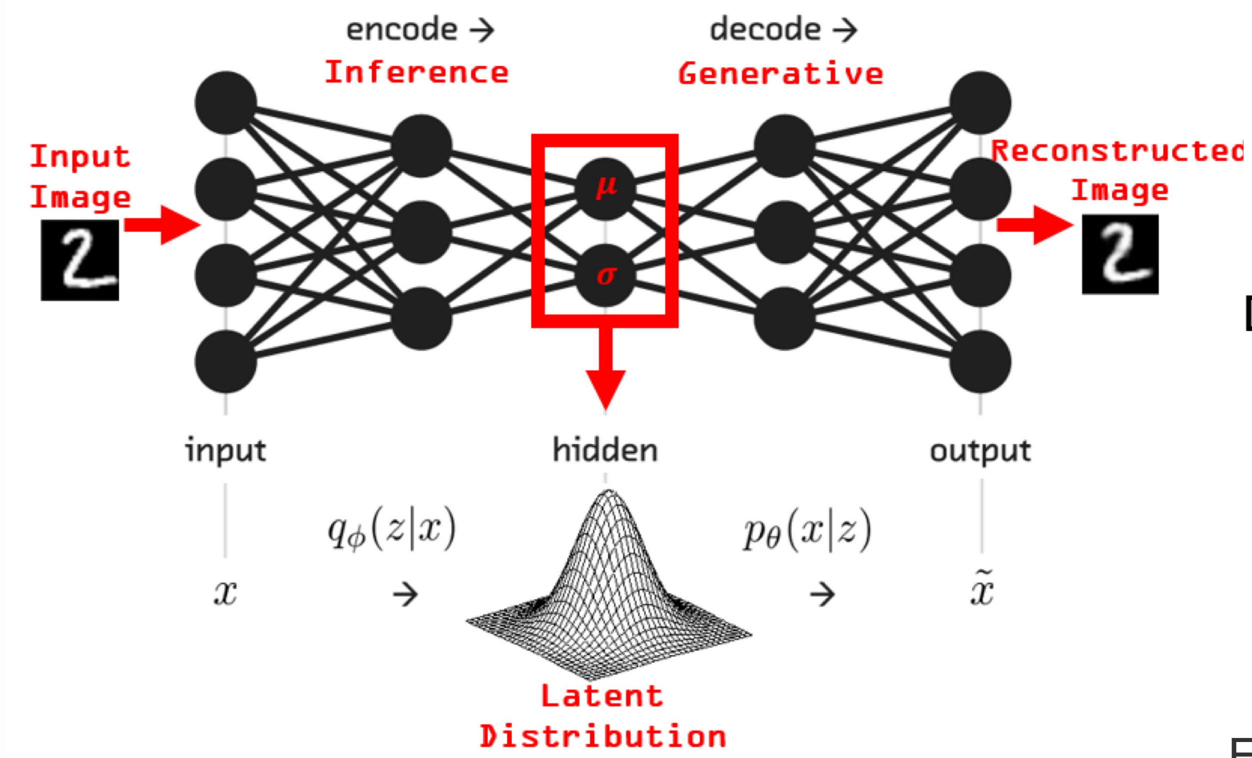
Example: VAEs for images

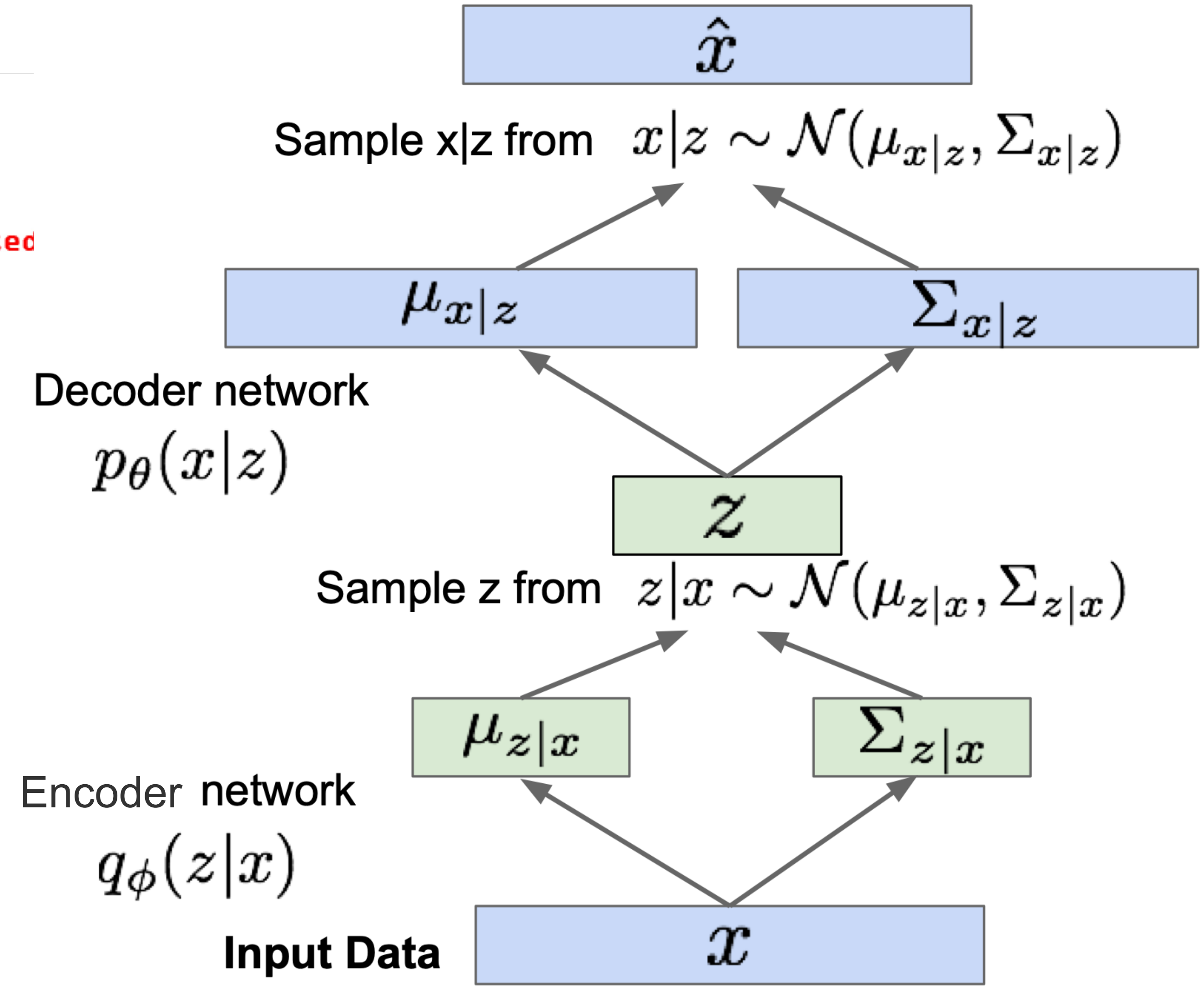




[Courtesy: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n]

Example: VAEs for images



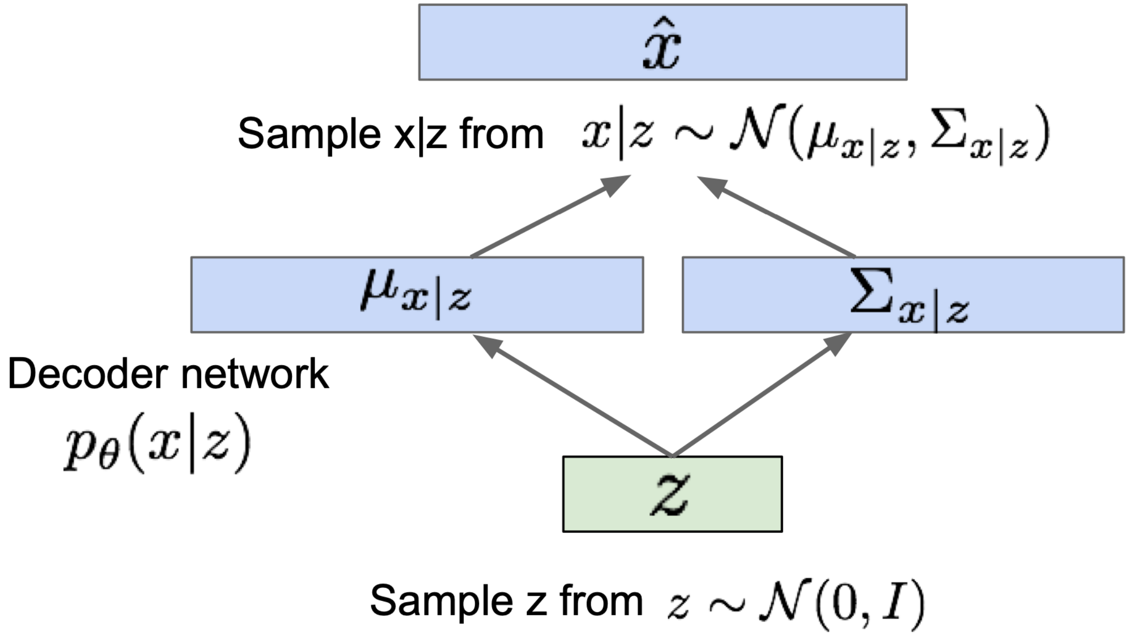


[Courtesy: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n]

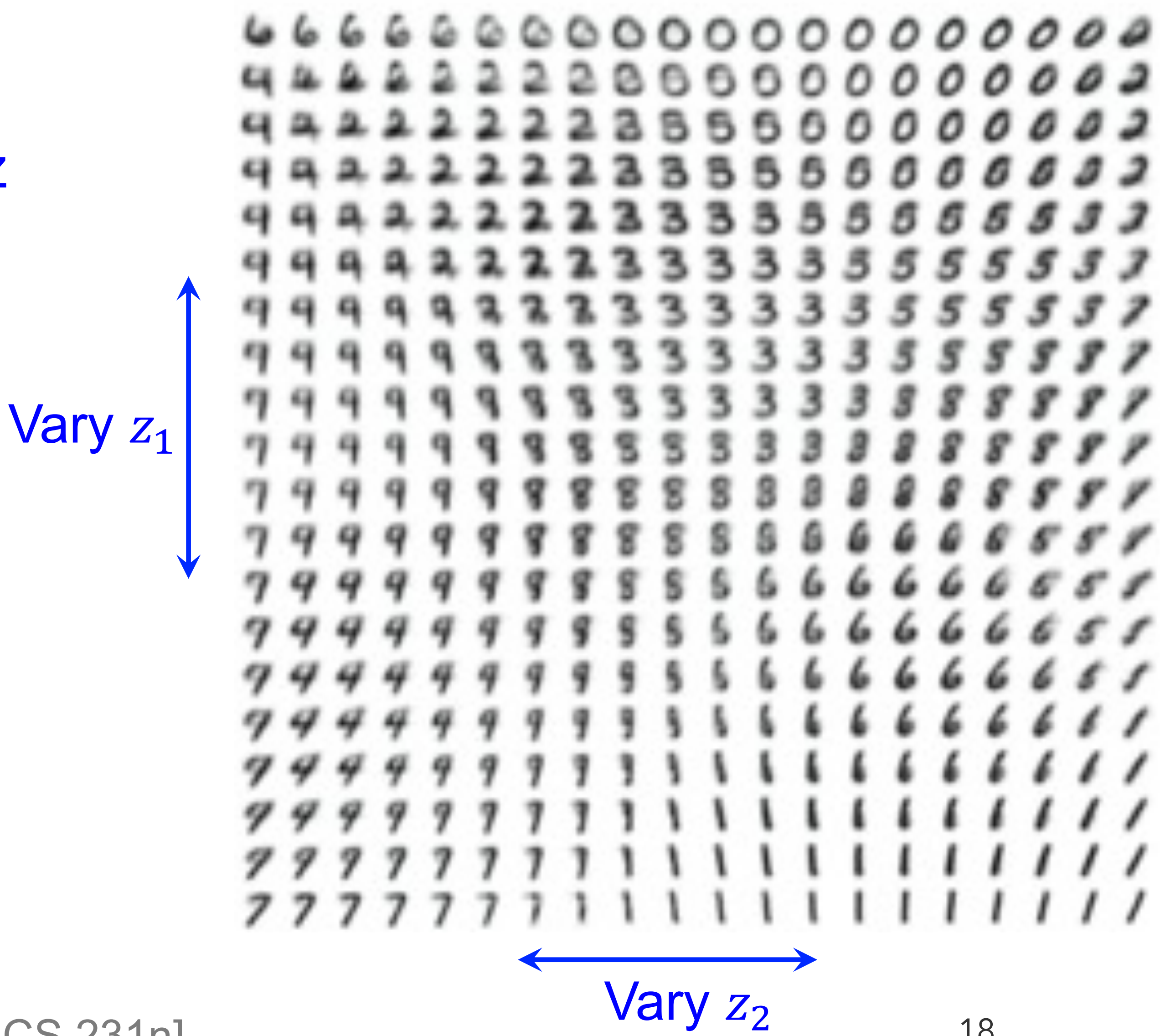
Example: VAEs for images

Generating samples:

- Use decoder network. Now sample z from prior!



Data manifold for 2-d **z**

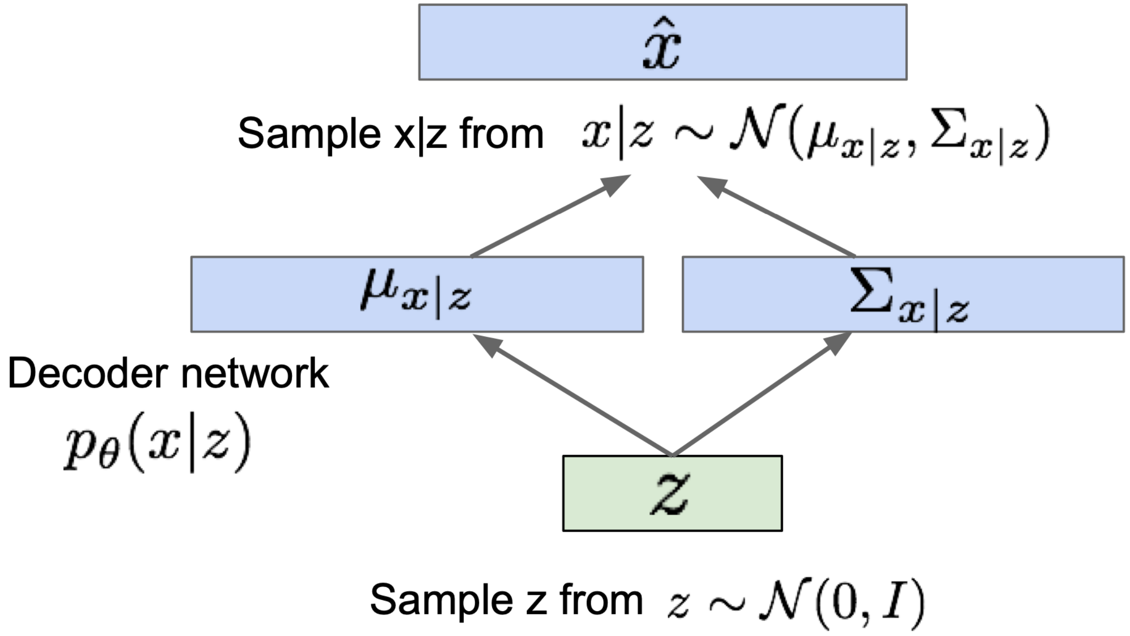


[Courtesy: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n]

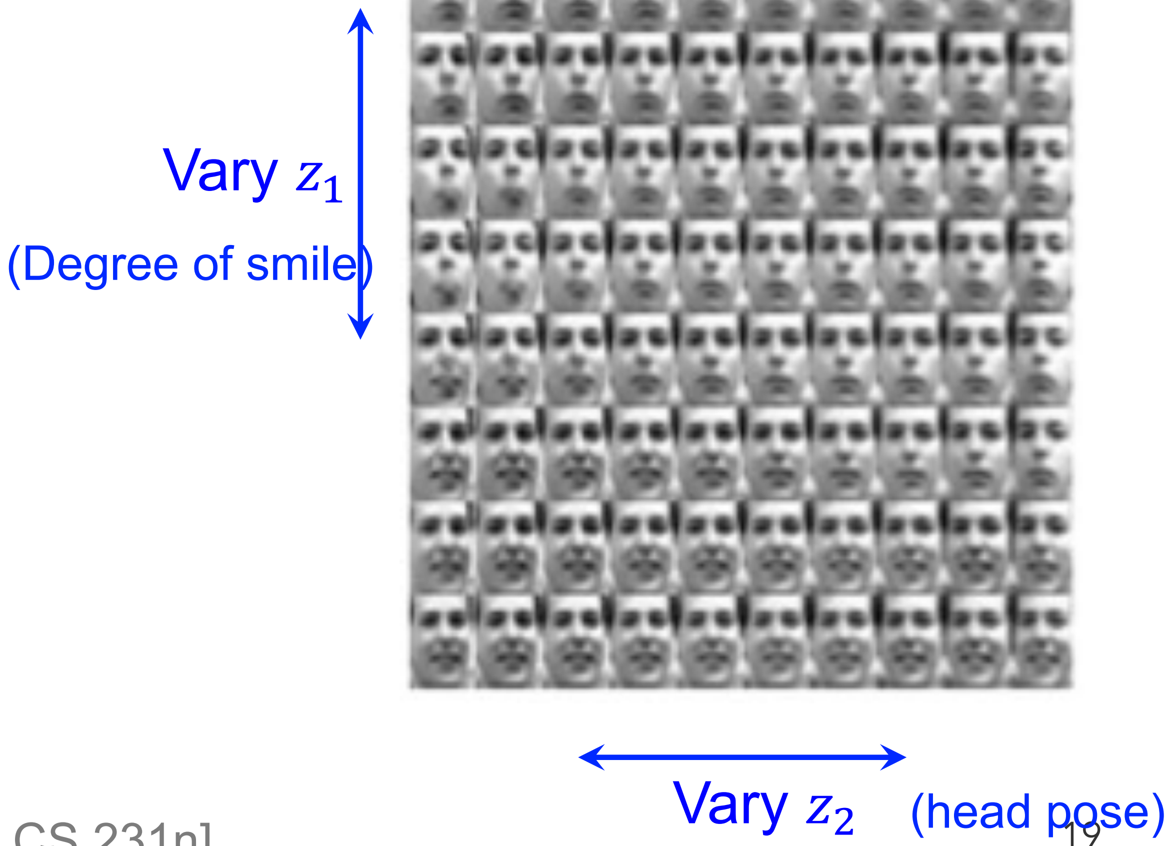
Example: VAEs for images

Generating samples:

- Use decoder network. Now sample z from prior!



Data manifold for 2-d **z**



[Courtesy: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n]

Example: VAEs for text

- Latent code interpolation and sentences generation from VAEs [Bowman et al., 2015].

“ i want to talk to you . ”
“i want to be with you . ”
“i do n’t want to be with you . ”
i do n’t want to be with you .
she did n’t want to be with him .

Note: Amortized Variational Inference

- Variational distribution as an inference model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ with parameters $\boldsymbol{\phi}$ (which was traditionally factored over samples)
- Amortize the cost of inference by learning a **single** data-dependent inference model
- The trained inference model can be used for quick inference on new data

Variational Auto-encoders: Summary

- A combination of the following ideas:
  - Variational Inference: ELBO
  - Variational distribution parametrized as neural networks
  - Reparameterization trick

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = [\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z}))$$

Reconstruction — Divergence from prior

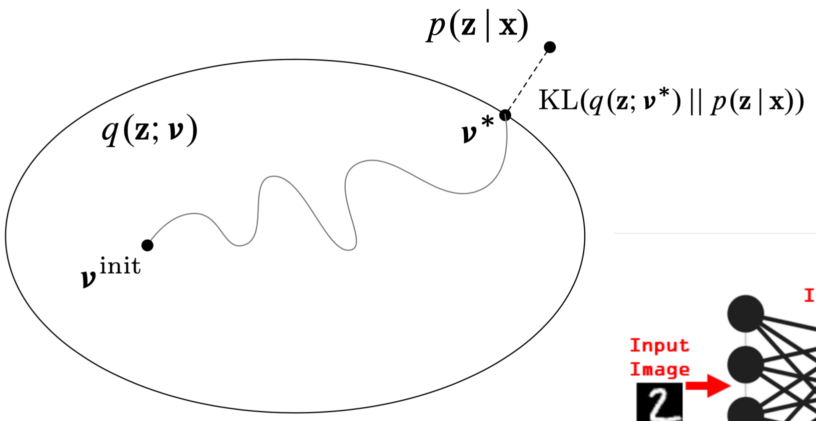
(Razavi et al., 2019)

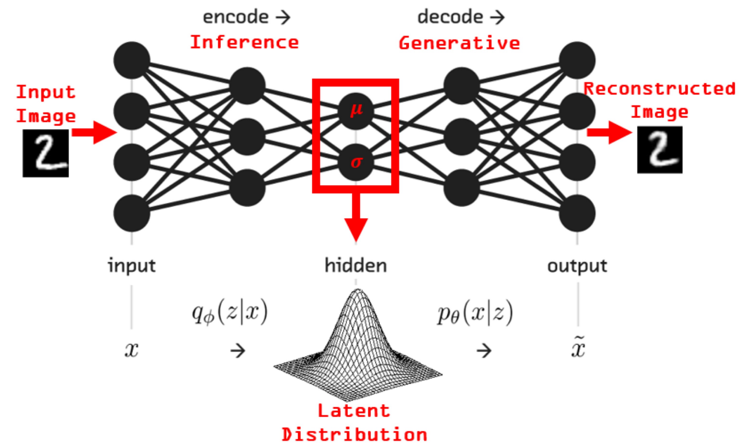
- Pros:
  - Principled approach to generative models
  - Allows inference of $q(z|x)$, can be useful feature representation for other tasks
- Cons:
  - Samples blurrier and lower quality compared to GANs
  - Tend to collapse on text data

Summary: Supervised / Unsupervised Learning

- Supervised Learning
  - Maximum likelihood estimation (MLE)

- Unsupervised learning
  - Maximum likelihood estimation (MLE) with latent variables
    - Marginal log-likelihood
  - EM algorithm for MLE
    - ELBO / Variational free energy
  - Variational Inference
    - ELBO / Variational free energy
    - Variational distributions
      - Factorized (mean-field VI)
      - Mixture of Gaussians (Black-box VI)
      - Neural-based (VAEs)

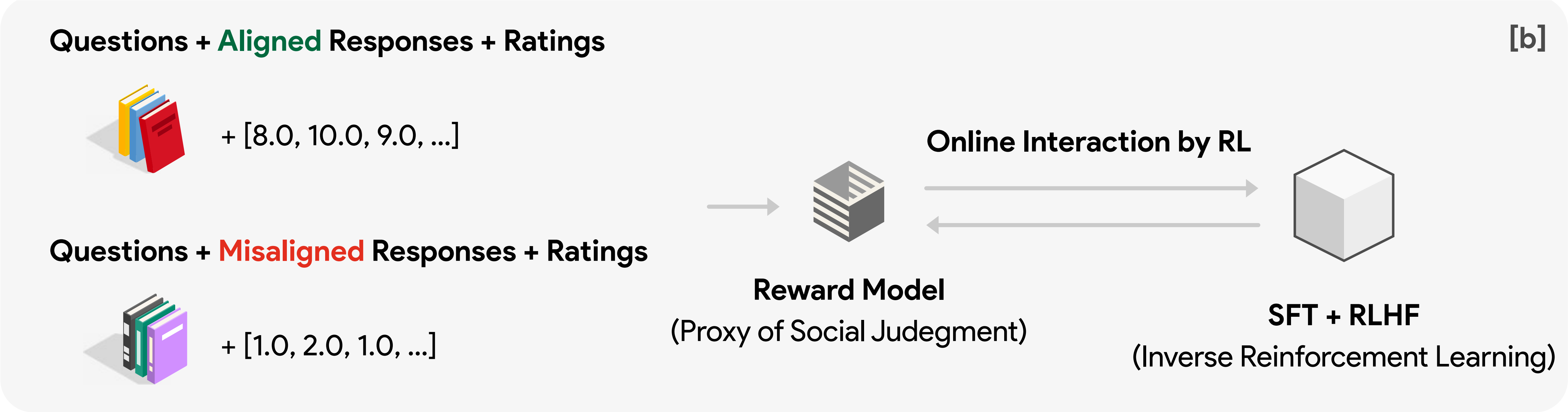




Reinforcement Learning (RL)

Recall: RL for LLMs

- RLHF: Reinforcement Learning with Human Feedback

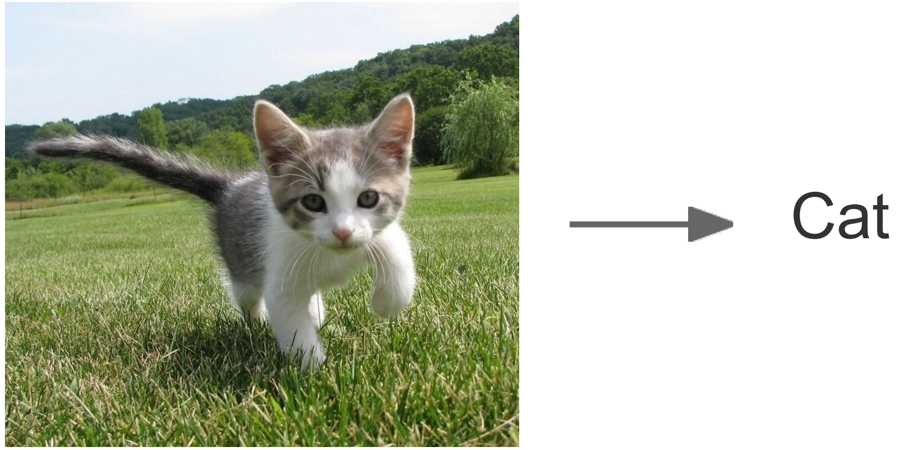


So far… Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.



Classification

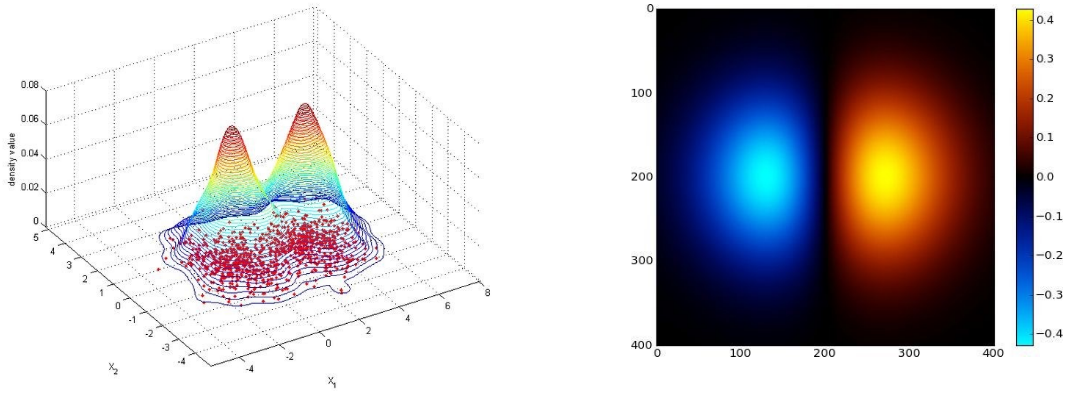
So far… Unsupervised Learning

Data: x
no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

1-d density estimation

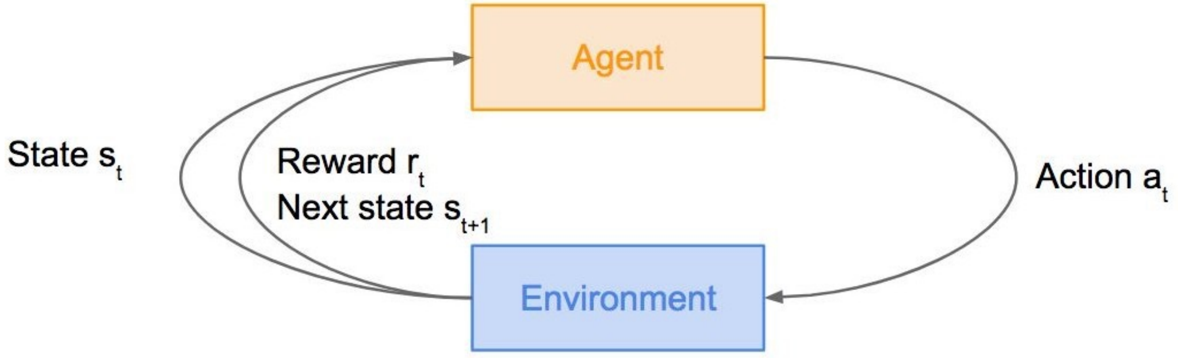


2-d density estimation

Today: Reinforcement Learning

Problems involving an **agent** interacting with an **environment**, which provides numeric **reward** signals

Goal: Learn how to take actions in order to maximize reward

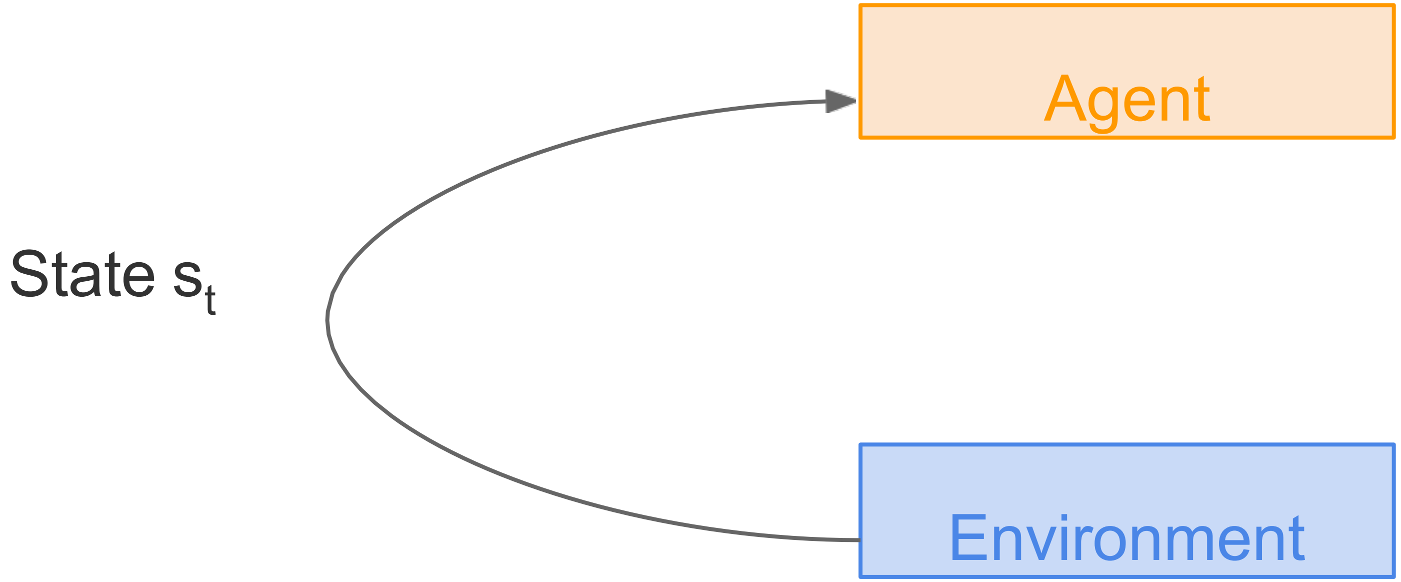


Overview

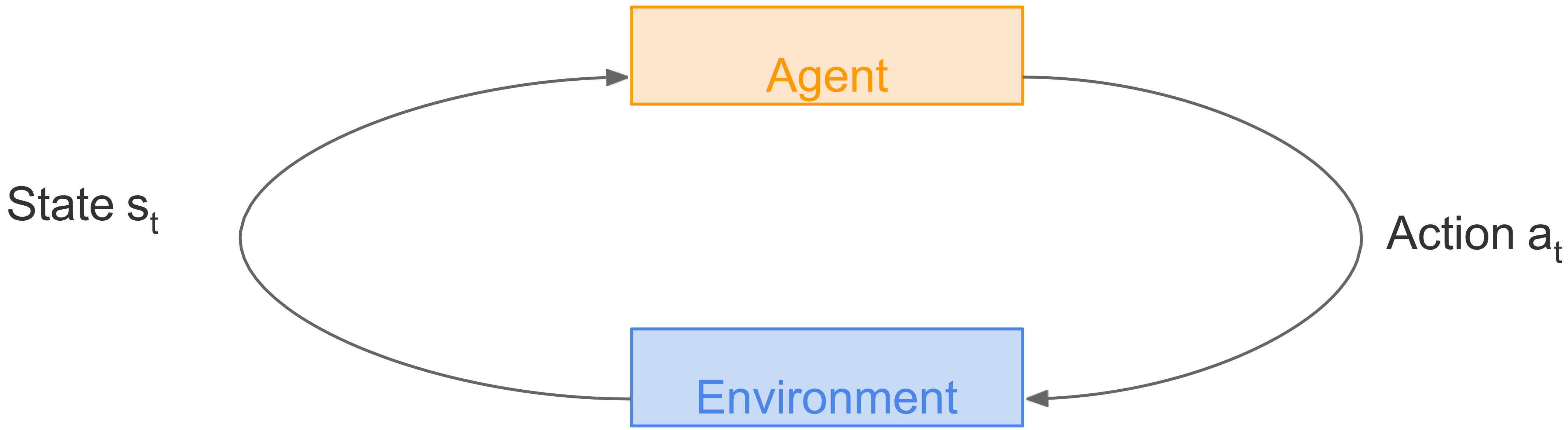
- What is Reinforcement Learning?
- Markov Decision Processes
- Q-Learning
- Policy Gradients

Reinforcement Learning

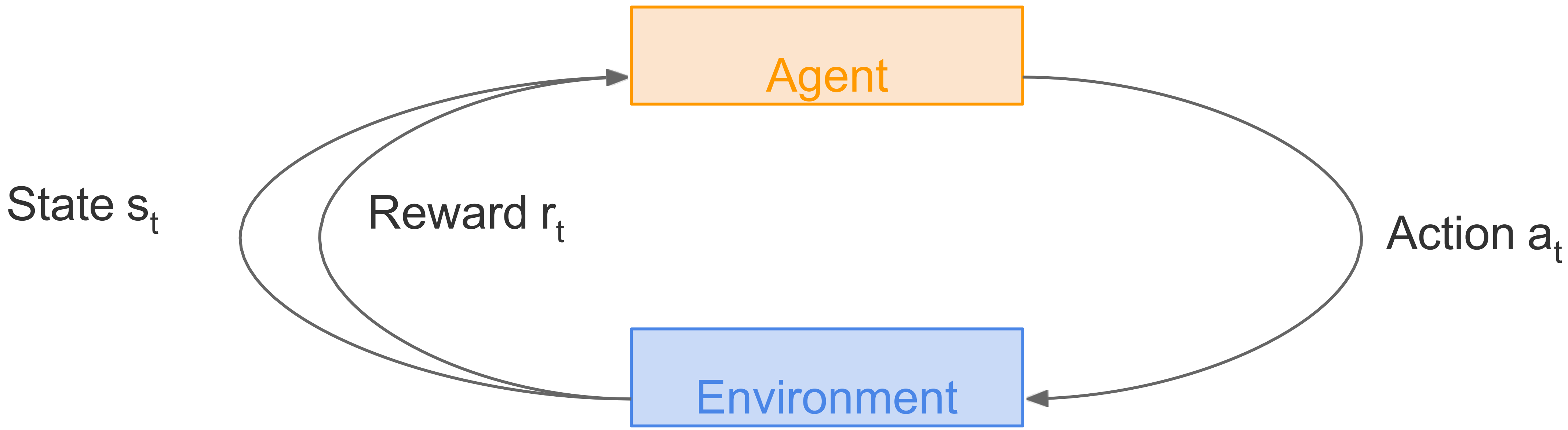
Reinforcement Learning



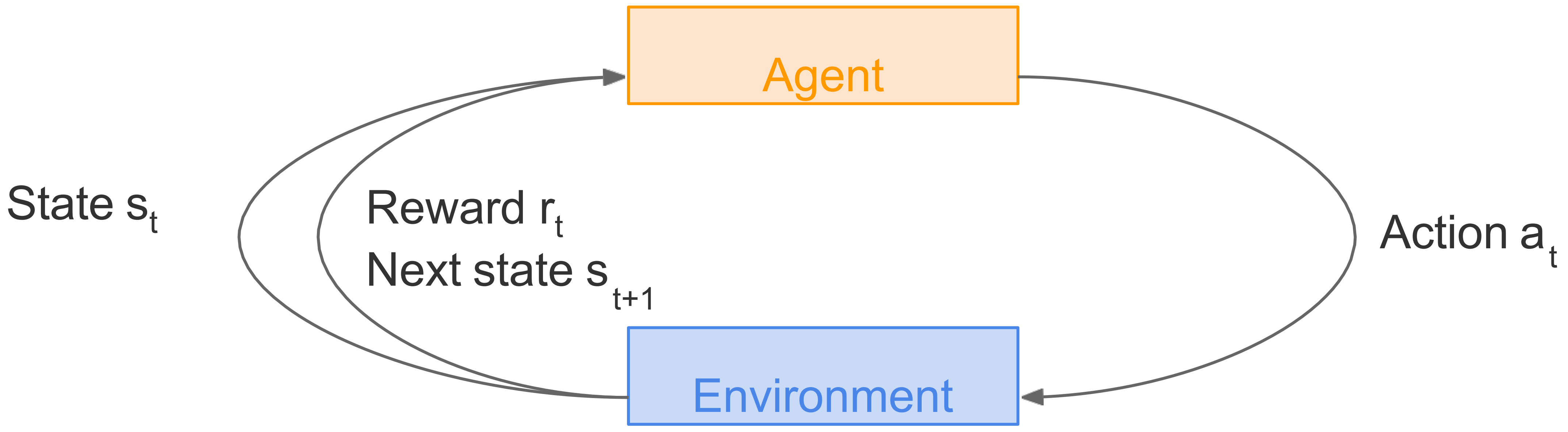
Reinforcement Learning



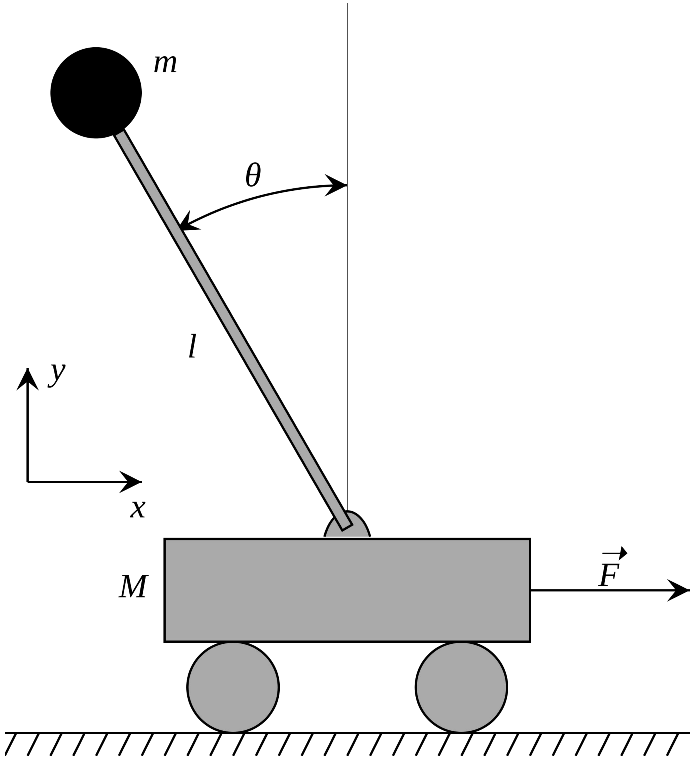
Reinforcement Learning



Reinforcement Learning



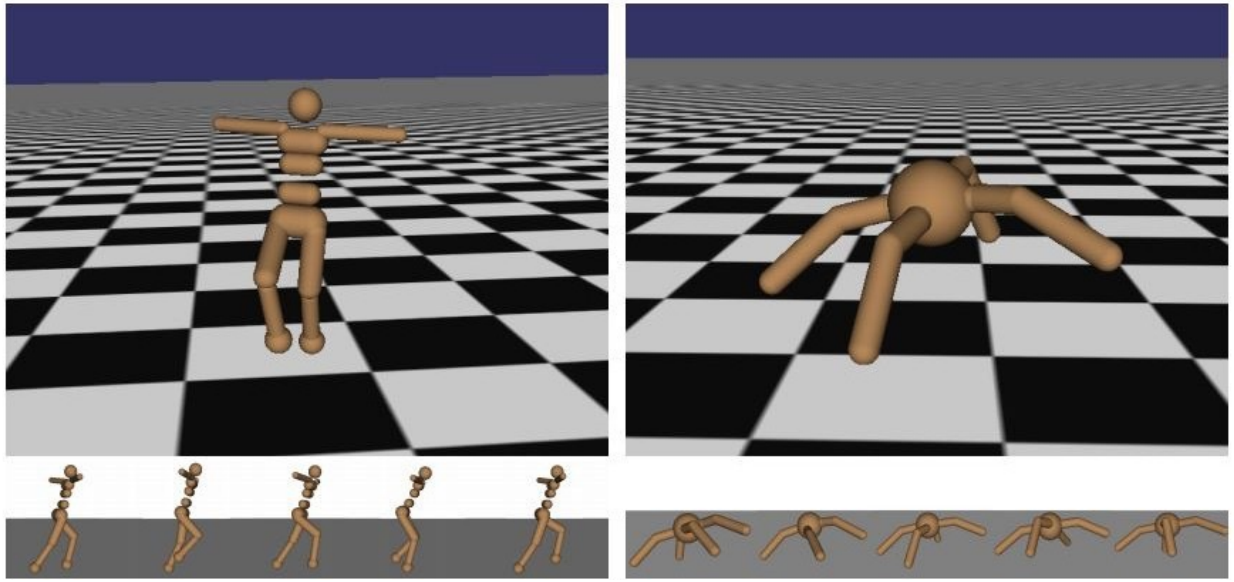
Cart-Pole Problem



Objective: Balance a pole on top of a movable cart

State: angle, angular speed, position, horizontal velocity
Action: horizontal force applied on the cart
Reward: 1 at each time step if the pole is upright

Robot Locomotion



Objective: Make the robot move forward

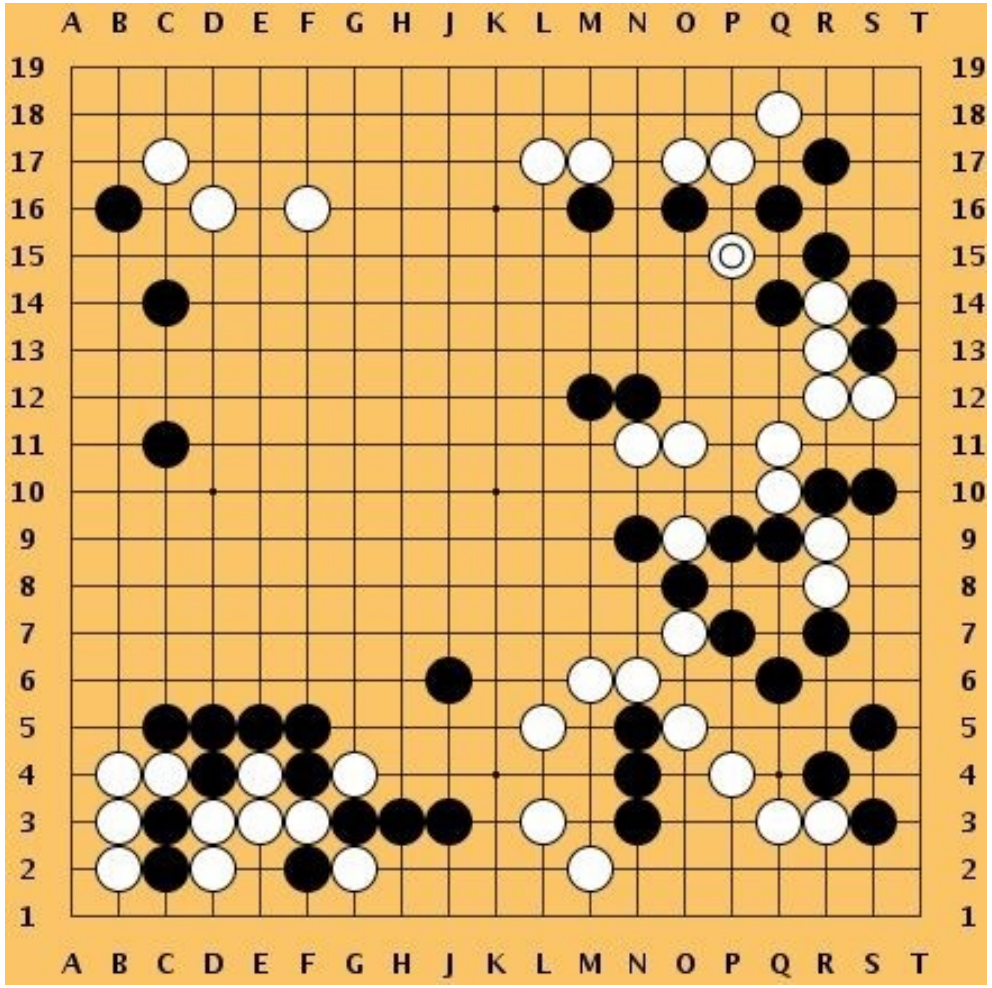
State: Angle and position of the joints
Action: Torques applied on joints
Reward: 1 at each time step upright + forward movement

Atari Games

Objective: Complete the game with the highest score

State: Raw pixel inputs of the game state
Action: Game controls e.g. Left, Right, Up, Down
Reward: Score increase/decrease at each time step

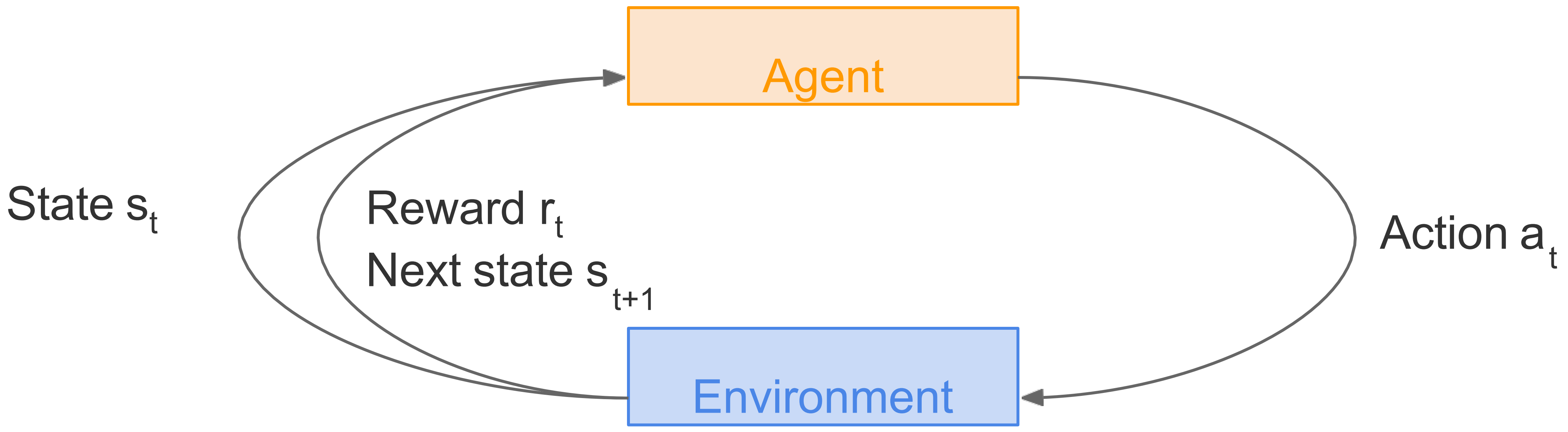
Go



Objective: Win the game!

State: Position of all pieces
Action: Where to put the next piece down
Reward: 1 if win at the end of the game, 0 otherwise

How can we mathematically formalize the RL problem?



Markov Decision Process

- Mathematical formulation of the RL problem
- **Markov property**: Current state completely characterises the state of the world

Defined by: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{P}, \gamma)$

$\mathcal{S}$: set of possible states
$\mathcal{A}$: set of possible actions
$\mathcal{R}$: distribution of reward given (state, action) pair
$\mathbb{P}$: transition probability i.e. distribution over next state given (state, action) pair
γ : discount factor

Markov Decision Process

- At time step t=0, environment samples initial state $s_0 \sim p(s_0)$
- Then, for t=0 until done:
  - Agent selects action a_t
  - Environment samples reward $r_t \sim R(. | s_t, a_t)$
  - Environment samples next state $s_{t+1} \sim P(. | s_t, a_t)$
  - Agent receives reward r_t and next state s_{t+1}

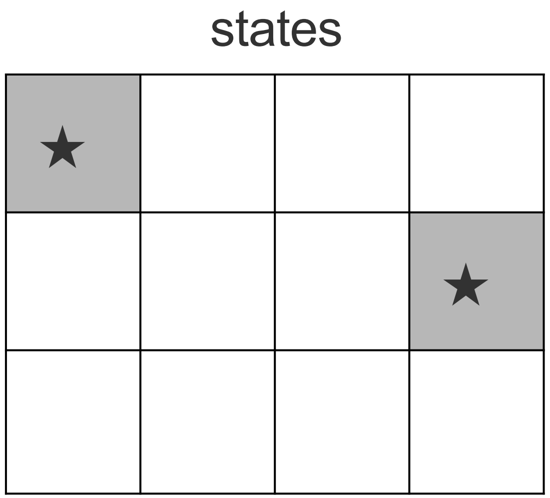
- A policy π is a function from S to A that specifies what action to take in each state
- **Objective**: find policy π* that maximizes cumulative discounted reward: $\sum_{t \geq 0} \gamma^t r_t$

A simple MDP: Grid World

actions = {

1. right
2. left
3. up
4. down

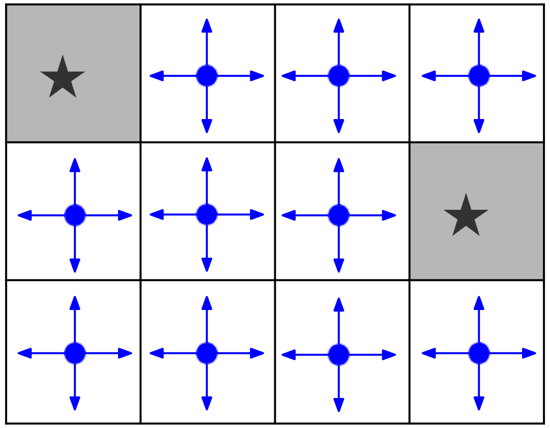
}



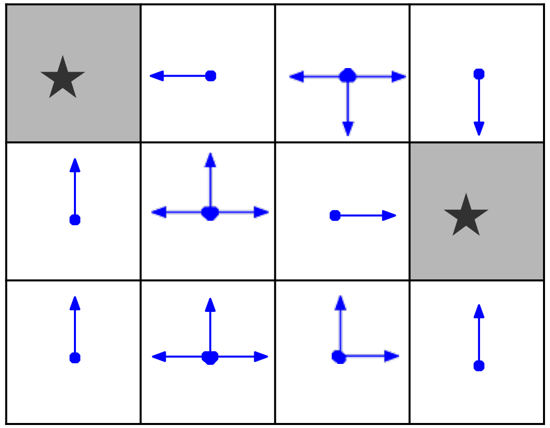
Set a negative "reward" for each transition (e.g. $r = -1$)

Objective: reach one of terminal states (greyed out) in least number of actions

A simple MDP: Grid World



Random Policy



Optimal Policy

The optimal policy π*

We want to find optimal policy π* that maximizes the sum of rewards.

How do we handle the randomness (initial state, transition probability…)?

The optimal policy π*

We want to find optimal policy π* that maximizes the sum of rewards.

How do we handle the randomness (initial state, transition probability…)?
Maximize the **expected sum of rewards!**

Formally: $\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | \pi\right]$ with $s_0 \sim p(s_0), a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)$

Definitions: Value function and Q-value function

Following a policy produces sample trajectories (or paths) s_0, a_0, r_0, s_1, a_1, r_1, …

Definitions: Value function and Q-value function

Following a policy produces sample trajectories (or paths) s_0, a_0, r_0, s_1, a_1, r_1, …

How good is a state?

The **value function** at state s, is the expected cumulative reward from following the policy from state s:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, \pi\right]$$

Definitions: Value function and Q-value function

Following a policy produces sample trajectories (or paths) $s_0, a_0, r_0, s_1, a_1, r_1, \ldots$

How good is a state?
The **value function** at state s, is the expected cumulative reward from following the policy from state s:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, \pi\right]$$

How good is a state-action pair?
The **Q-value function** at state s and action a, is the expected cumulative reward from taking action a in state s and then following the policy:

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right]$$

Questions?