CS 4803 / 7643: Deep Learning

Topics:

- (Deep) Reinforcement Learning
- Closing time

Dhruv Batra

Georgia Tech

Administrativia

- Last class today

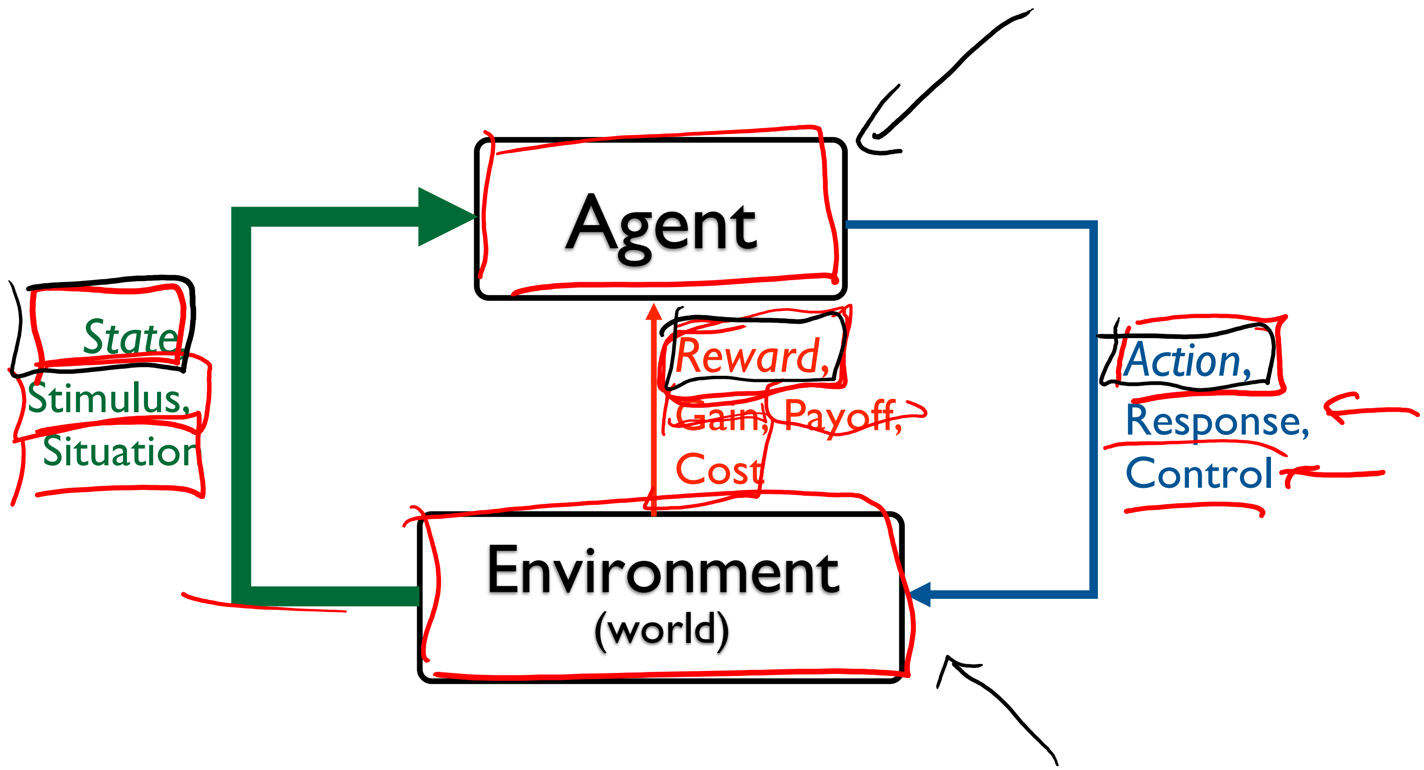
- Project submission
  - Due: 12/04, 11:55pm
  - Last deliverable in the class
  - Can't use late days
  - https://piazza.com/class/jkujs03pgu75cd?cid=225

Recap from last time

Types of Learning

- Supervised learning
  - Learning from a “teacher”
  - Training data includes desired outputs
- Unsupervised learning
  - Discover structure in data
  - Training data does not include desired outputs
- Reinforcement learning
  - Learning to act under evaluative feedback (rewards)

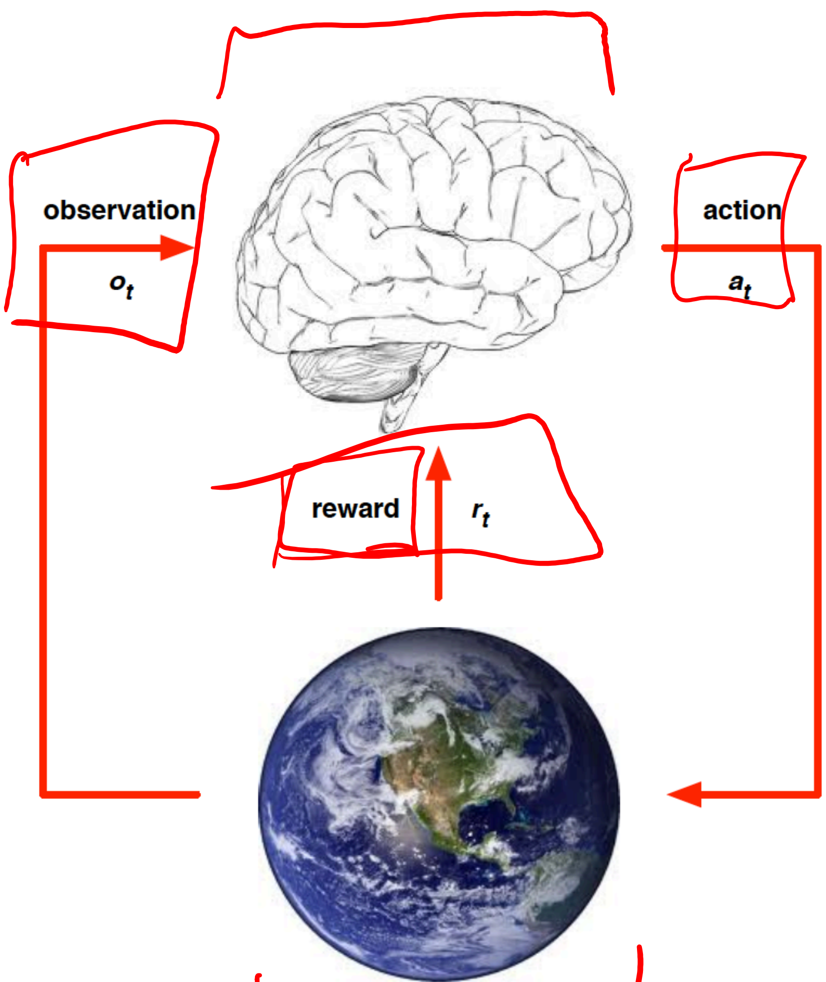
RL API



- Environment may be unknown, nonlinear, stochastic and complex
- Agent learns a policy mapping states to actions
  - Seeking to maximize its cumulative reward in the long run

Slide Credit: Rich Sutton

RL API



- At each step t the agent:
  - Executes action a_t
  - Receives observation o_t
  - Receives scalar reward r_t
- The environment:
  - Receives action a_t
  - Emits observation o_{t+1}
  - Emits scalar reward r_{t+1}

RL

$r_t(s_t, a_t)$

RL		SL
s_t	$\neq$	x_t
a_t	$\neq$	y_t

SL

$\ell(y^*, \hat{y}(x))$

$o_t = f(s_t)$

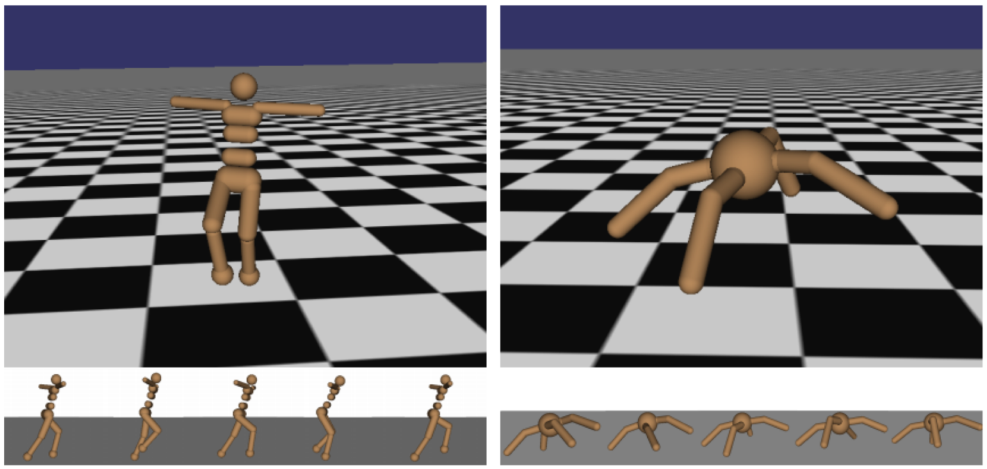
Signature challenges of RL

$-1, +1000$

$s_0, a_1, s_1, \ldots ;$

- Evaluative feedback (reward)
- Sequentiality, delayed consequences
- Need for trial and error, to explore as well as exploit
- Non-stationarity $\quad \boxed{x, y \sim P_{data}(\ )} \quad S \sim P(S_{t+1} \mid S_t, \boxed{a_t})$
- The fleeting nature of time and online data

Robot Locomotion



Objective: Make the robot move forward

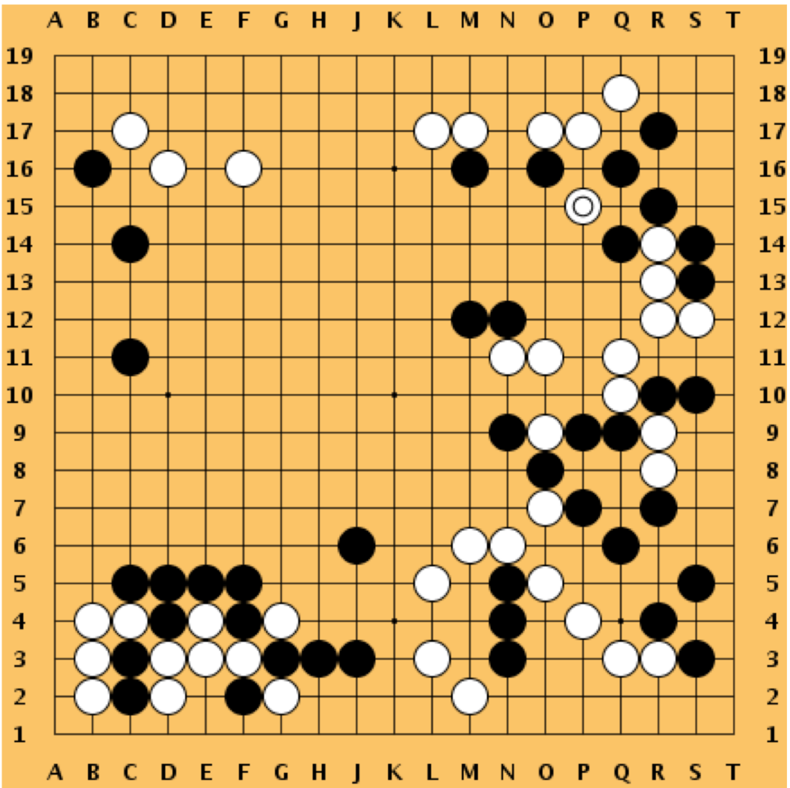
State: Angle and position of the joints
Action: Torques applied on joints
Reward: 1 at each time step upright + forward movement

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!

x y

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

Markov Decision Process

- Mathematical formulation of the RL problem

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

π γ

$\mathcal{S}$: 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

$\rightarrow P(S_{t+1} \mid S_t, a_t)$

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Markov Decision Process

- Mathematical formulation of the RL problem

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

- Life is trajectory: $\ldots \boxed{S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}, S_{t+2},} \ldots$

- **~~Markov property~~**: Current state completely characterizes the state of the world

$$p(r, s'|s, a) = Prob\left[R_{t+1} = r, S_{t+1} = s' \;\middle|\; \underbrace{S_t = s, A_t = a}\right]$$

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Components of an RL Agent

- Policy
  - How does an agent behave?
- Value function
  - How good is each state and/or state-action pair?
- Model
  - Agent’s representation of the environment

Policy

$\pi(\cdot)$

- A policy is how the agent acts

- Formally, map from states to actions

e.g.

State	Action
A	2
B	1

Deterministic policy: $\boxed{a = \pi(s)}$

$\pi_\theta(s)$

Stochastic policy: $\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$

The optimal policy **π***

What's a good policy?

Maximizes current reward? Sum of all future reward?

Discounted future rewards!

$$\sum_{t} r_t + \boxed{\gamma}\, r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} \ldots$$

The optimal policy **π***

What's a good policy?

Maximizes current reward? Sum of all future reward?

Discounted future rewards!

Formally: $\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t\geq 0}^{\infty} \gamma^t r_t | \pi\right]$

with $s_0 \sim p(s_0), a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)$

$a_t = \pi(s_t)$ $\quad s_{t+1} = M(s_t, a_t)$

Components of an RL Agent

- Policy
  - How does an agent behave?
- Value function
  - How good is each state and/or state-action pair?
- Model
  - Agent's representation of the environment

Value Function

- A value function is a prediction of future reward
- "State Value Function" or simply "Value Function" V
  - How good is a state?
  - Am I screwed? Am I winning this game?
- "Action Value Function" or Q-function Q
  - How good is a state action-pair?
  - Should I do this now?

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 state s (and following the policy thereafter):

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

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 state s (and following the policy thereafter):

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

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 following the policy thereafter):

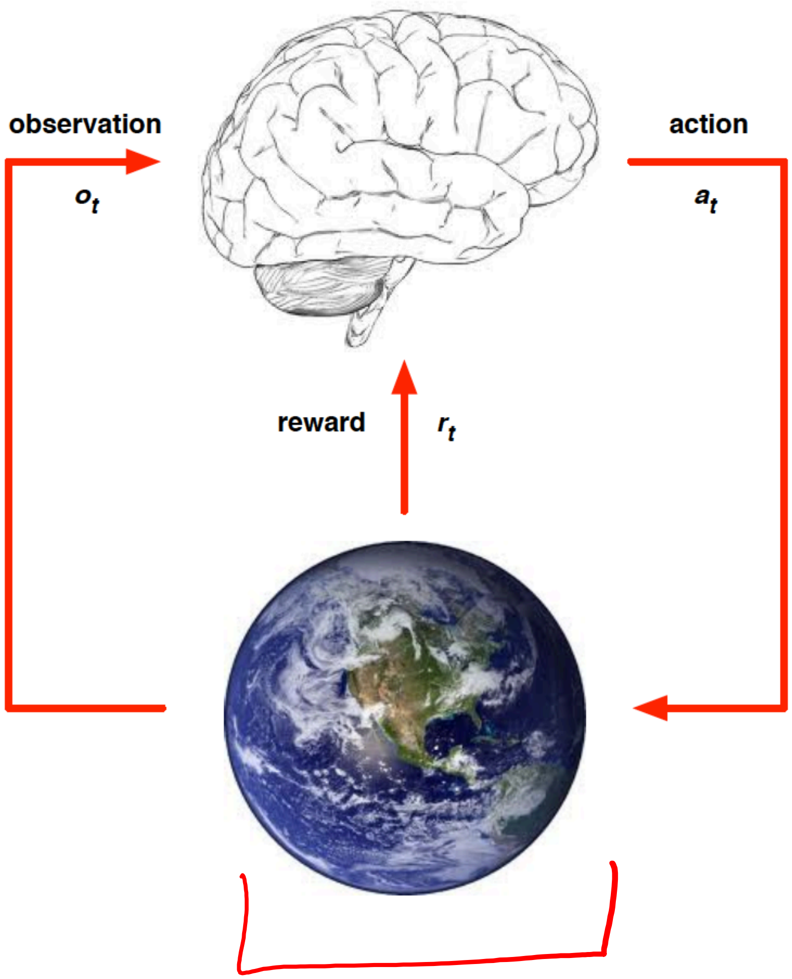
max
a

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

Components of an RL Agent

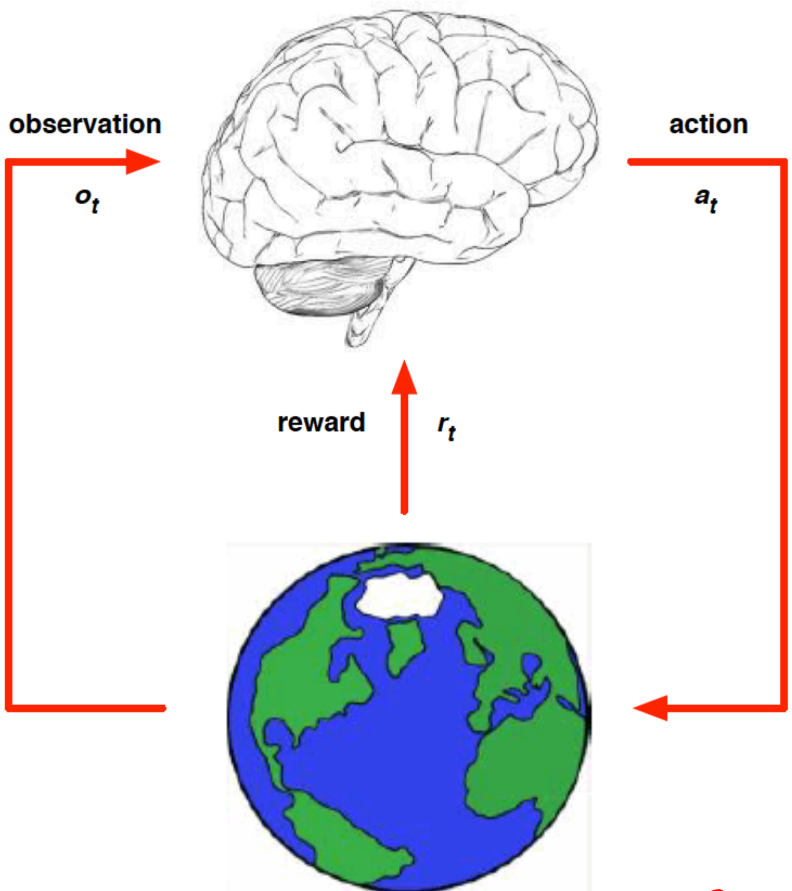
- Policy
  - How does an agent behave?
- Value function
  - How good is each state and/or state-action pair?
- Model
  - Agent's representation of the environment

Model



Model

- Model predicts what the world will do next

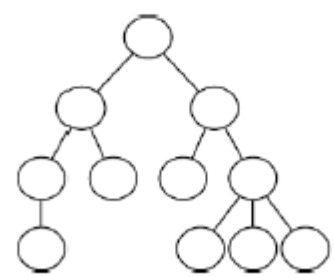


Model is learnt from experience

Acts as proxy for environment

Planner interacts with model

e.g. using lookahead search



$s_t, a_t \longrightarrow s_{t+1}$

$\boxed{P(s_{t+1} | s_t, a_t)}$

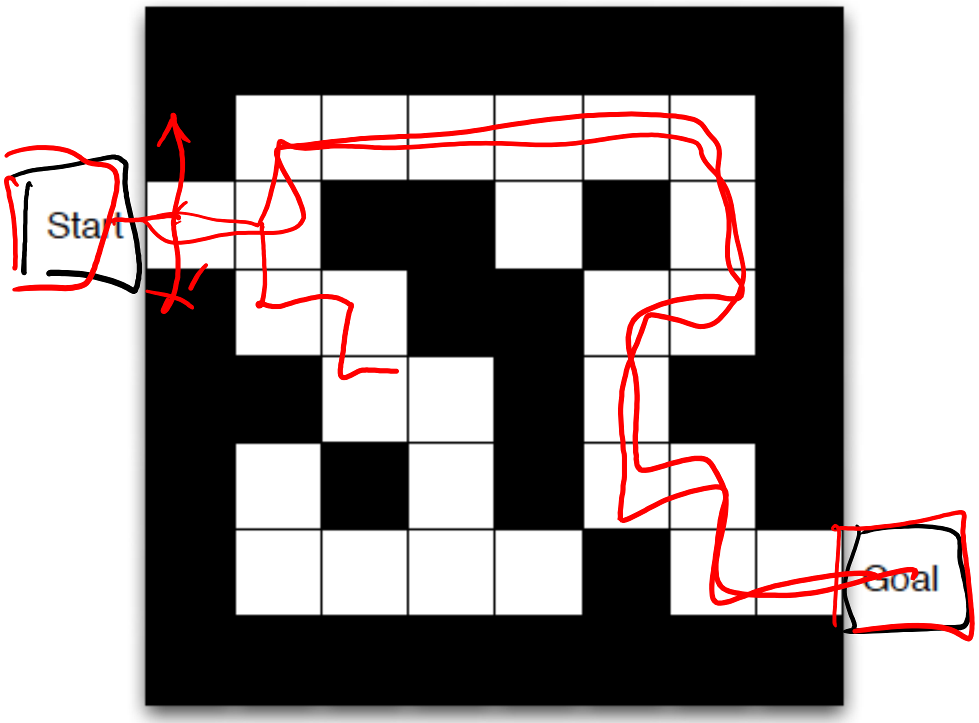
Plan for Today

- (Deep) Reinforcement Learning
  - Policy gradients
- Closing the loop

Components of an RL Agent

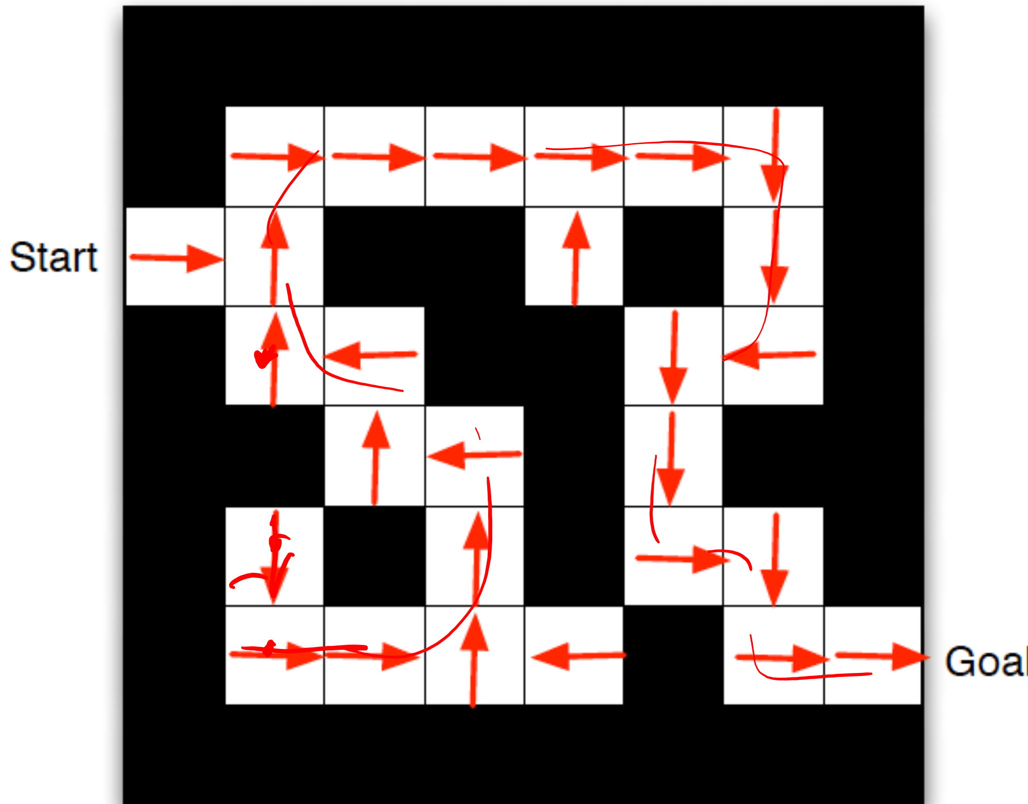
- Policy
  - How does an agent behave?
- Value function
  - How good is each state and/or state-action pair?
- Model
  - Agent's representation of the environment

Maze Example



- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

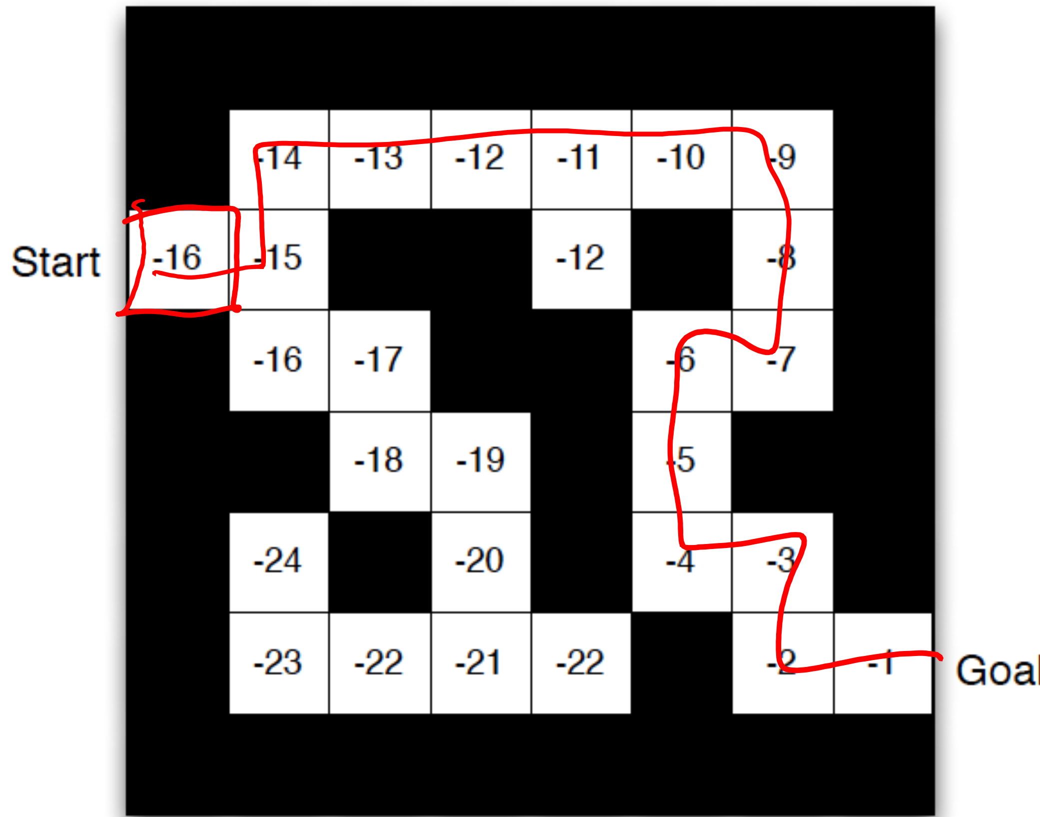
Maze Example: Policy



- Arrows represent policy $\pi(s)$ for each state s

Maze Example: Value

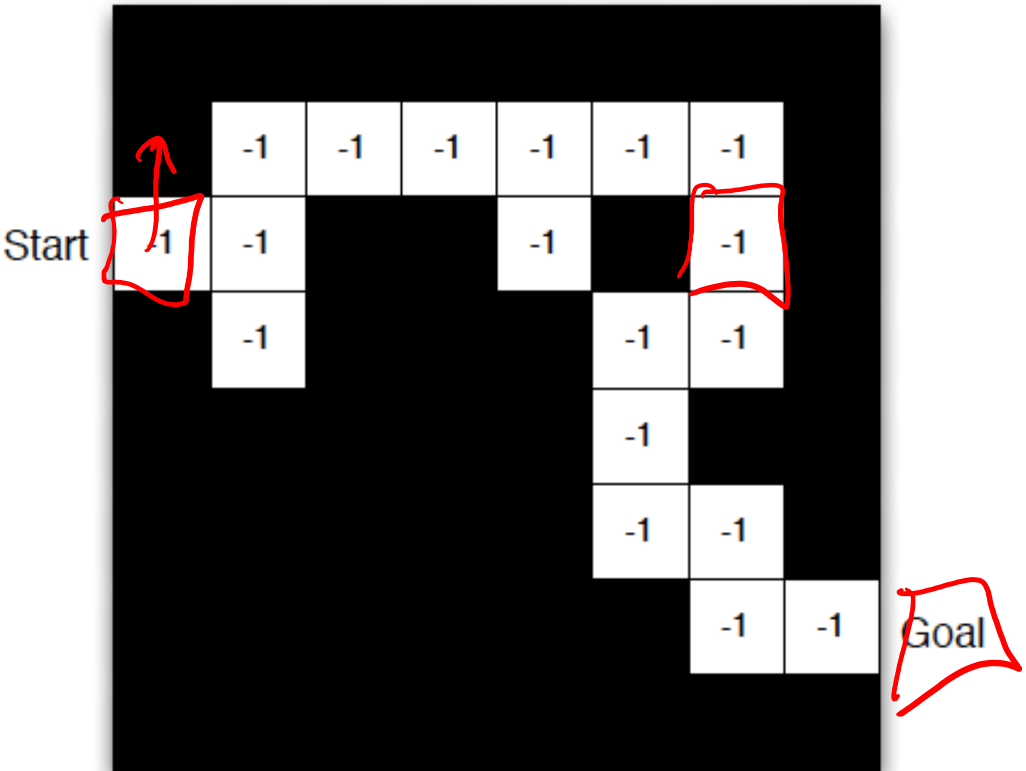
V(s)



- Numbers represent value $v_{\boxed{\pi}}(s)$ of each state s

π^*

Maze Example: Model



- Agent may have an internal model of the environment
- Dynamics: how actions change the state
- Rewards: how much reward from each state
- The model may be imperfect

- Grid layout represents transition model $\boxed{\mathcal{P}^a_{ss'}}$
- Numbers represent immediate reward $\boxed{\mathcal{R}^a_s}$ from each state s (same for all a)

Components of an RL Agent

- Value function
  - How good is each state and/or state-action pair?
- Policy
  - How does an agent behave?
- Model
  - Agent’s representation of the environment

Approaches to RL

- Value-based RL
  - Estimate the optimal action-value function $\boxed{Q^*(s,a)}$
- Policy-based RL
  - Search directly for the optimal policy $\boxed{\pi^*}$
- Model - based RL
  - Build a model of the world
    - State transition, reward probabilities
  - Plan (e.g. by look-ahead) using model

Deep RL

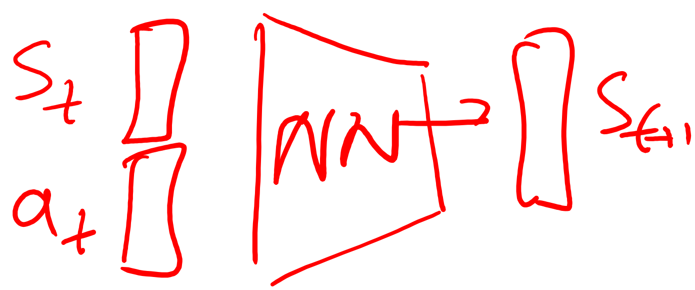
- Value-based RL
  - Use neural nets to represent Q function $\boxed{Q(s, a; \theta)}$

$$\underbrace{Q(s, a; \theta^*)}_{x} \approx \underbrace{\boxed{Q^*(s, a)}}_{y}$$

- Policy-based RL
  - Use neural nets to represent policy $\boxed{\pi_\theta}$

$$\pi_{\theta^*} \approx \pi^*$$

- Model
  - Use neural nets to represent and learn the model



Deep RL

- Value-based RL
  - Use neural nets to represent Q function $Q(s, a; \theta)$

$$Q(s, a; \theta^*) \approx Q^*(s, a)$$

- Policy-based RL
  - Use neural nets to represent policy π_θ

$$\pi_{\theta^*} \approx \pi^*$$

- Model
  - Use neural nets to represent and learn the model

Policy Gradients

REINFORCE

A2C

A3C

Policy Gradients

$s_t \to a_t$

$s_t \to \boxed{\theta} \to a_t$

$\boxed{\pi_\theta}(a_t | s_t)$

Formally, let's define a class of parameterized policies: $\Pi = \{\boxed{\pi_\theta}, \theta \in \mathbb{R}^m\}$

For each policy, define its value:

$$\boxed{J(\theta)} = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | \pi_\theta\right]$$

$$\max_{\theta} \mathbb{E}_{z \sim p_\theta(z)} [f(z)]$$

Policy Gradients

Formally, let's define a class of parameterized policies: $\Pi = \{\pi_\theta, \theta \in \mathbb{R}^m\}$

For each policy, define its value:

$$J(\theta) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | \pi_\theta\right]$$

We want to find the optimal policy $\theta^* = \arg\max_\theta J(\theta)$

How can we do this?

Policy Gradients

Formally, let's define a class of parameterized policies: $\Pi = \{\pi_\theta, \theta \in \mathbb{R}^m\}$

For each policy, define its value:

$$J(\theta) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | \pi_\theta\right]$$

We want to find the optimal policy $\theta^* = \arg\max_\theta J(\theta)$

How can we do this?

Gradient ascent on policy parameters!

REINFORCE algorithm

Mathematically, we can write:

$$J(\theta) = \mathbb{E}_{\boxed{\tau \sim p(\tau;\theta)}}\left[r(\tau)\right]$$
$$= \int_\tau r(\tau) p(\tau;\theta) \mathrm{d}\tau$$

Where r(τ) is the reward of a trajectory $\boxed{\tau} = \overbrace{(s_0, a_0, r_0, s_1, a_1, \ldots \ldots S_{T-1})}^{\text{Trajectory}}$

$$p(\tau;\theta) = p(s_0, a_0, r_0, \ldots \ldots)$$

$$p(s_0) \times \prod_{t=1}^{T} p(s_t, a_t, r_t \mid s_{t-1}, a_{t-1})$$

REINFORCE algorithm

Mathematically, we can write:

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau)\right]$$
$$= \int_{\tau} r(\tau)p(\tau;\theta)\mathrm{d}\tau$$

Where r(τ) is the reward of a trajectory $\tau = (s_0, a_0, r_0, s_1, \ldots)$

World

Policy

$$p(\tau;\theta) = \prod_{t \geq 0} \boxed{p(s_{t+1}|s_t, a_t)} \boxed{\pi_\theta(a_t|s_t)}$$

$$\log p(\tau;\theta) = \boxed{\sum_{t \geq 0} \log p(s_{t+1}|s_t, a_t)} + \boxed{\log \pi_\theta(a_t|s_t)}$$

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

REINFORCE algorithm

Expected reward:
$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau)\right]$$
$$= \int_{\tau} r(\tau)p(\tau;\theta)\mathrm{d}\tau$$

$$\nabla J(\theta) = E\left[\underline{r(\tau)}\ \underline{\nabla_\theta \log p(\tau;\theta)}\right]$$

REINFORCE algorithm

Expected reward: $J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau)\right]$

$$= \int_\tau r(\tau) p(\tau;\theta) \mathrm{d}\tau$$

Now let's differentiate this: $\nabla_\theta J(\theta) = \int_\tau r(\tau) \nabla_\theta p(\tau;\theta) \mathrm{d}\tau$

REINFORCE algorithm

Expected reward: $J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau)\right]$

$$= \int_\tau r(\tau)p(\tau;\theta)\mathrm{d}\tau$$

Now let's differentiate this: $\nabla_\theta J(\theta) = \int_\tau r(\tau)\nabla_\theta p(\tau;\theta)\mathrm{d}\tau$

Intractable! Expectation of gradient is problematic when p depends on θ

REINFORCE algorithm

Expected reward: $$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau)\right]$$
$$= \int_{\tau} r(\tau)p(\tau;\theta)\mathrm{d}\tau$$

Now let's differentiate this: $$\nabla_\theta J(\theta) = \int_{\tau} r(\tau)\nabla_\theta p(\tau;\theta)\mathrm{d}\tau$$

Intractable! Expectation of gradient is problematic when p depends on θ

However, we can use a nice trick: $$\nabla_\theta p(\tau;\theta) = p(\tau;\theta)\frac{\nabla_\theta p(\tau;\theta)}{p(\tau;\theta)} = p(\tau;\theta)\nabla_\theta \log p(\tau;\theta)$$

REINFORCE algorithm

Expected reward: $$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau)\right] = \int_\tau r(\tau)p(\tau;\theta)\mathrm{d}\tau$$

Now let's differentiate this: $$\nabla_\theta J(\theta) = \int_\tau r(\tau)\nabla_\theta p(\tau;\theta)\mathrm{d}\tau$$

Intractable! Expectation of gradient is problematic when p depends on θ

However, we can use a nice trick: $$\nabla_\theta p(\tau;\theta) = p(\tau;\theta)\frac{\nabla_\theta p(\tau;\theta)}{p(\tau;\theta)} = p(\tau;\theta)\nabla_\theta \log p(\tau;\theta)$$

If we inject this back:

$$\nabla_\theta J(\theta) = \int_\tau \left(r(\tau)\nabla_\theta \log p(\tau;\theta)\right)p(\tau;\theta)\mathrm{d}\tau$$
$$= \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau)\nabla_\theta \log p(\tau;\theta)\right]$$

Can estimate with Monte Carlo sampling

REINFORCE algorithm

Can we compute those quantities without knowing the transition probabilities?

policy

We have: $p(\tau;\theta) = \prod_{t\geq 0} p(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$

REINFORCE algorithm

Can we compute those quantities without knowing the transition probabilities?

We have: $p(\tau;\theta) = \prod_{t\geq 0} p(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$

$\frac{\partial}{\partial\theta}$ Thus: $\underline{\log p(\tau;\theta)} = \sum_{t\geq 0} \boxed{\cancel{\log p(s_{t+1}|s_t, a_t)}} + \boxed{\log \pi_\theta(a_t|s_t)}$

REINFORCE algorithm

Can we compute those quantities without knowing the transition probabilities?

We have: $p(\tau;\theta) = \prod_{t\geq 0} p(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$

Thus: $\log p(\tau;\theta) = \sum_{t\geq 0} \log p(s_{t+1}|s_t, a_t) + \log \pi_\theta(a_t|s_t)$

And when differentiating: $\nabla_\theta \log p(\tau;\theta) = \sum_{t\geq 0} \nabla_\theta \log \boxed{\pi_\theta}(a_t|s_t)$

Doesn't depend on transition probabilities!

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

REINFORCE algorithm

$$\nabla_\theta J(\theta) = \int_\tau \left(r(\tau)\nabla_\theta \log p(\tau;\theta)\right) p(\tau;\theta)\mathrm{d}\tau$$
$$= \mathbb{E}_{\tau\sim p(\tau;\theta)}\left[r(\tau)\nabla_\theta \log p(\tau;\theta)\right]$$

Can we compute those quantities without knowing the transition probabilities?

We have: $p(\tau;\theta) = \prod_{t\geq 0} p(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$

Thus: $\log p(\tau;\theta) = \sum_{t\geq 0} \log p(s_{t+1}|s_t, a_t) + \log \pi_\theta(a_t|s_t)$

And when differentiating: $\nabla_\theta \log p(\tau;\theta) = \sum_{t\geq 0} \nabla_\theta \log \pi_\theta(a_t|s_t)$

Doesn't depend on transition probabilities!

Therefore when sampling a trajectory τ, we can estimate J(θ) with

$$\boxed{\nabla_\theta \boxed{J(\theta)} \approx \boxed{\sum_{t\geq 0} r(\tau)\boxed{\nabla_\theta \log \pi_\theta(a_t|s_t)}}}$$

Intuition

(+) ↑
(−) ↓

Gradient estimator: $$\nabla_\theta J(\theta) \approx \sum_{t\geq 0} \boxed{r(\tau)} \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Interpretation:

- If r(τ) is high, push up the probabilities of the actions seen
- If r(τ) is low, push down the probabilities of the actions seen

Intuition

Gradient estimator: $$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \boxed{r(\tau)} \boxed{\nabla_\theta} \boxed{\log \pi_\theta(a_t|s_t)}$$

Interpretation:

- If r(τ) is high, push up the probabilities of the actions seen
- If r(τ) is low, push down the probabilities of the actions seen

Might seem simplistic to say that if a trajectory is good then all its actions were good. But in expectation, it averages out!

Pong from pixels

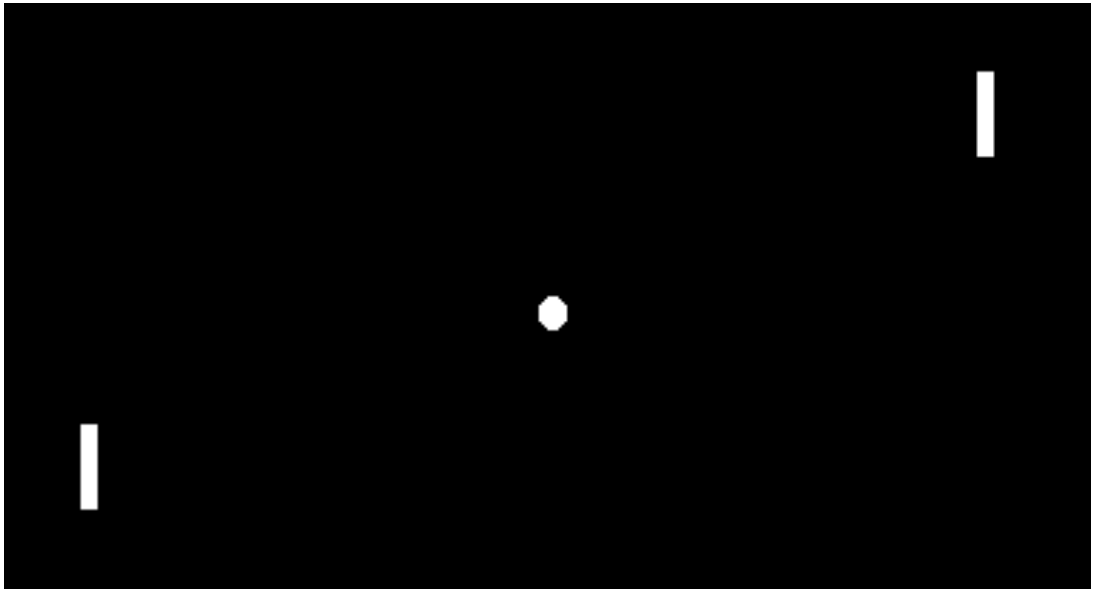
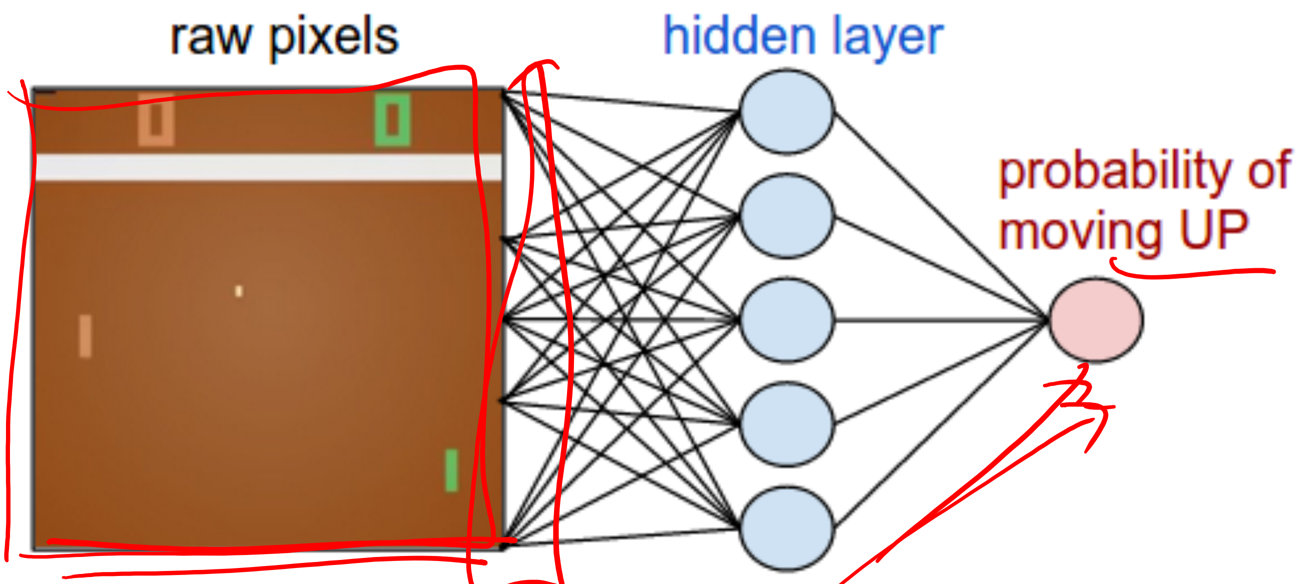


Image Credit: http://karpathy.github.io/2016/05/31/rl/

Pong from pixels



s

O_t

$\pi_{\theta}(a_t \mid s_t / O_t)$

Image Credit: http://karpathy.github.io/2016/05/31/rl/

Pong from pixels

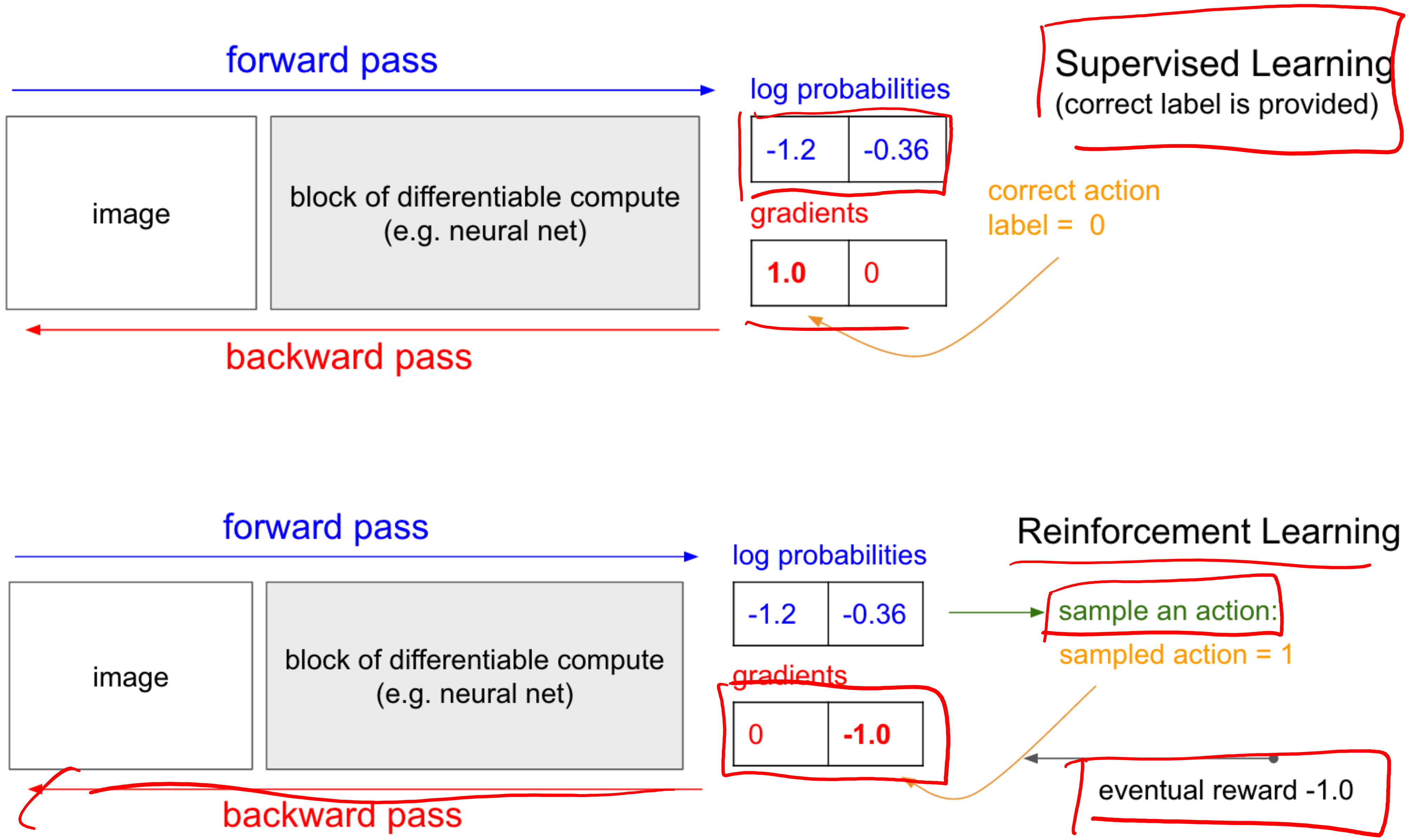
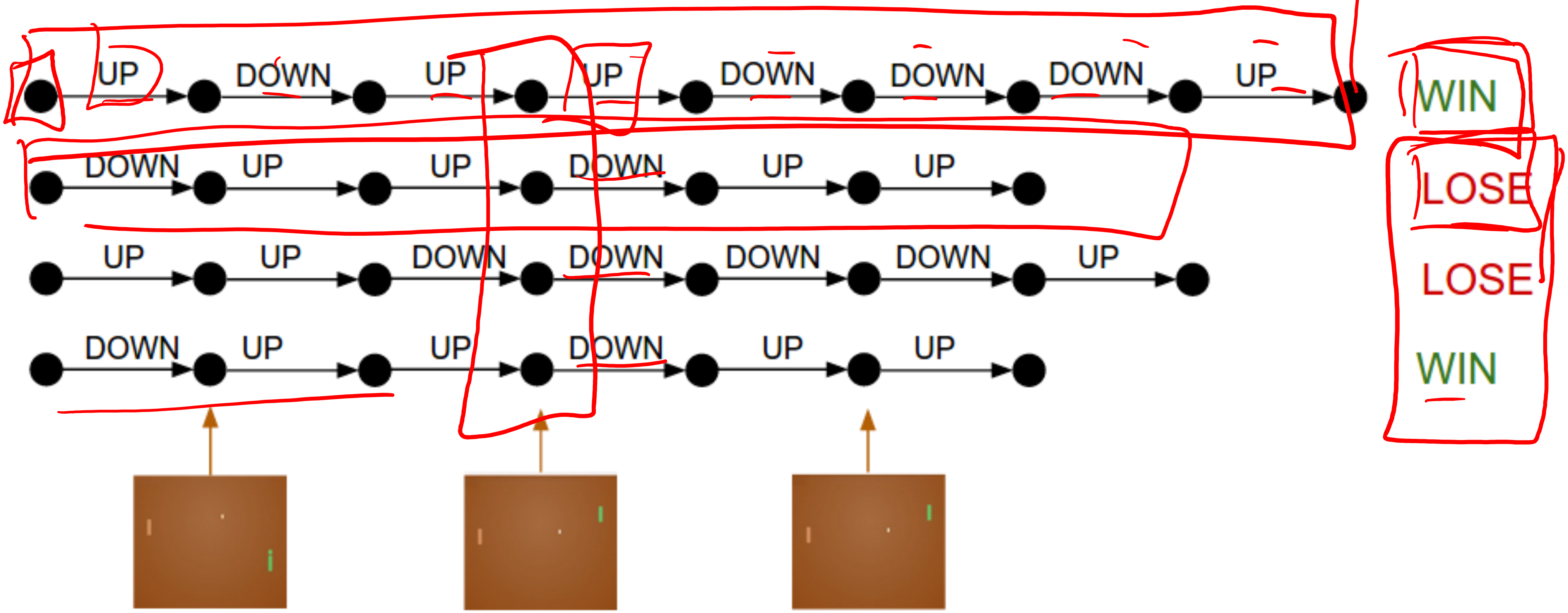


Image Credit: http://karpathy.github.io/2016/05/31/rl/

Intuition


UP DOWN UP UP DOWN DOWN DOWN UP
WIN
DOWN UP UP DOWN UP UP
LOSE
UP UP DOWN DOWN DOWN DOWN UP
LOSE
DOWN UP UP DOWN UP UP
WIN

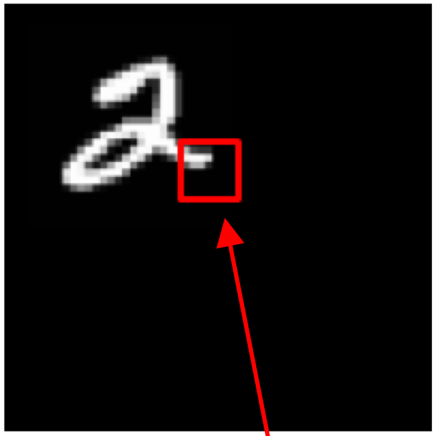
REINFORCE in action: Recurrent Attention Model (RAM)

Objective: Image Classification

Take a sequence of "glimpses" selectively focusing on regions of the image, to predict class

- Inspiration from human perception and eye movements
- Saves computational resources => scalability
- Able to ignore clutter / irrelevant parts of image

State: Glimpses seen so far
Action: (x,y) coordinates (center of glimpse) of where to look next in image
Reward: 1 at the final timestep if image correctly classified, 0 otherwise


glimpse

[Mnih et al. 2014]

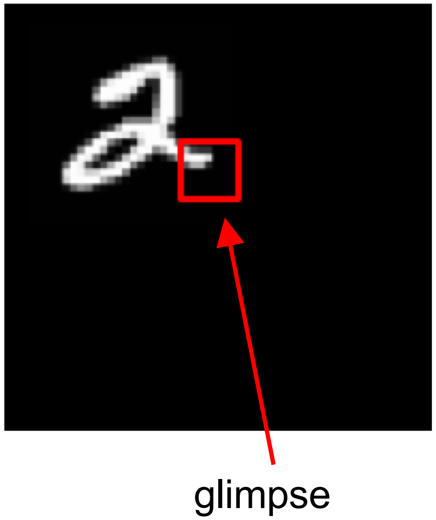
REINFORCE in action: Recurrent Attention Model (RAM)

Objective: Image Classification

Take a sequence of “glimpses” selectively focusing on regions of the image, to predict class

- Inspiration from human perception and eye movements
- Saves computational resources => scalability
- Able to ignore clutter / irrelevant parts of image

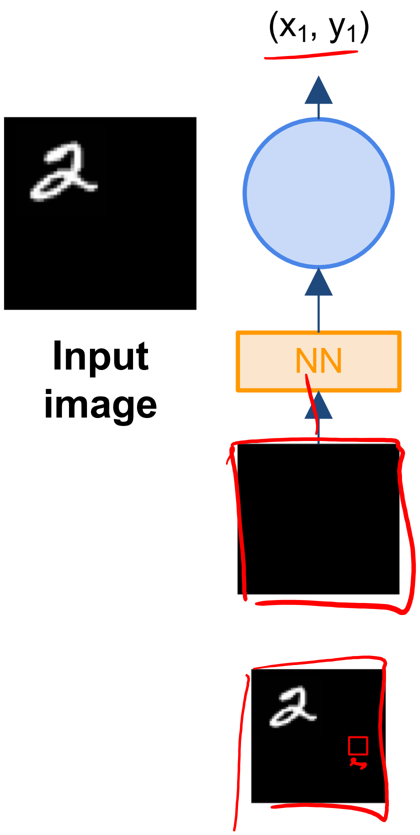
State: Glimpses seen so far
Action: (x,y) coordinates (center of glimpse) of where to look next in image
Reward: 1 at the final timestep if image correctly classified, 0 otherwise



Glimpsing is a non-differentiable operation => learn policy for how to take glimpse actions using REINFORCE
Given state of glimpses seen so far, use RNN to model the state and output next action

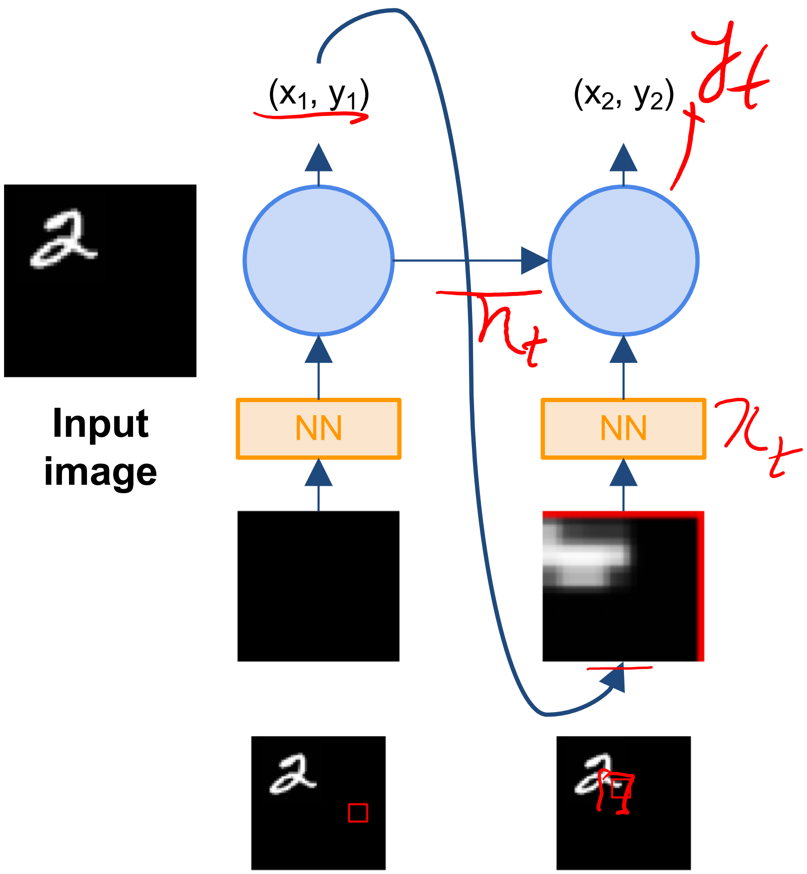
[Mnih et al. 2014]

REINFORCE in action: Recurrent Attention Model (RAM)



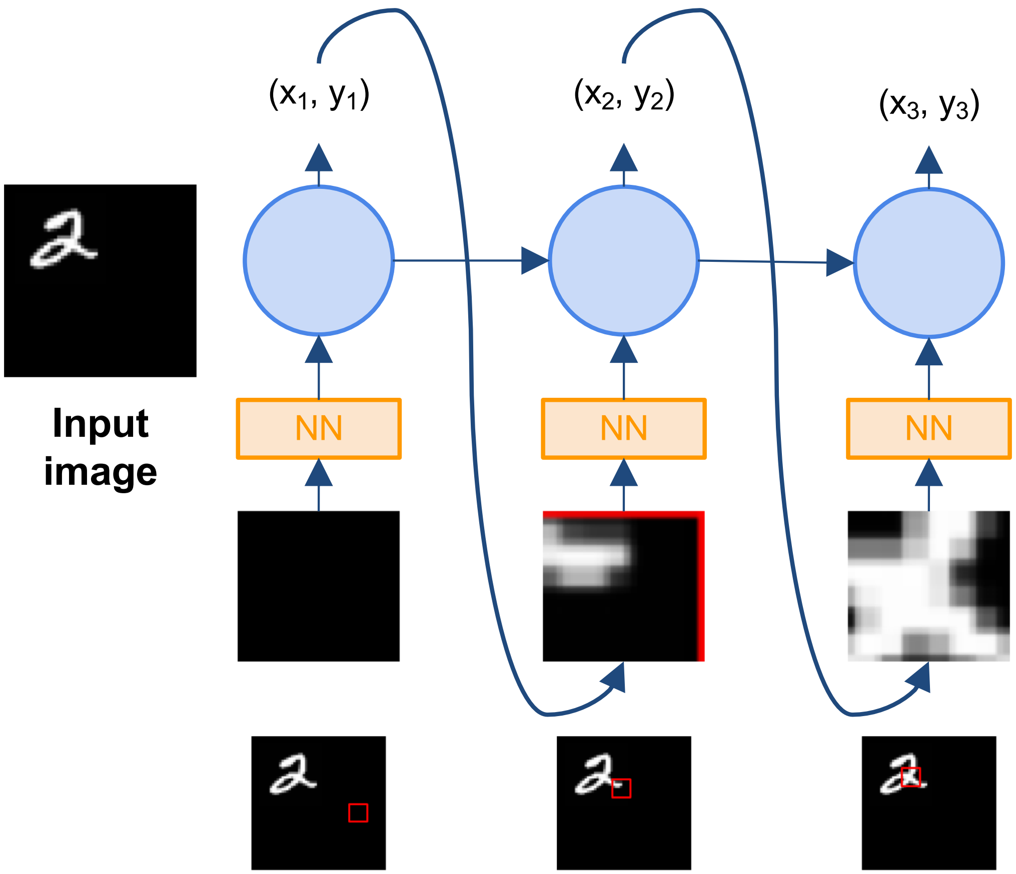
[Mnih et al. 2014]

REINFORCE in action: Recurrent Attention Model (RAM)



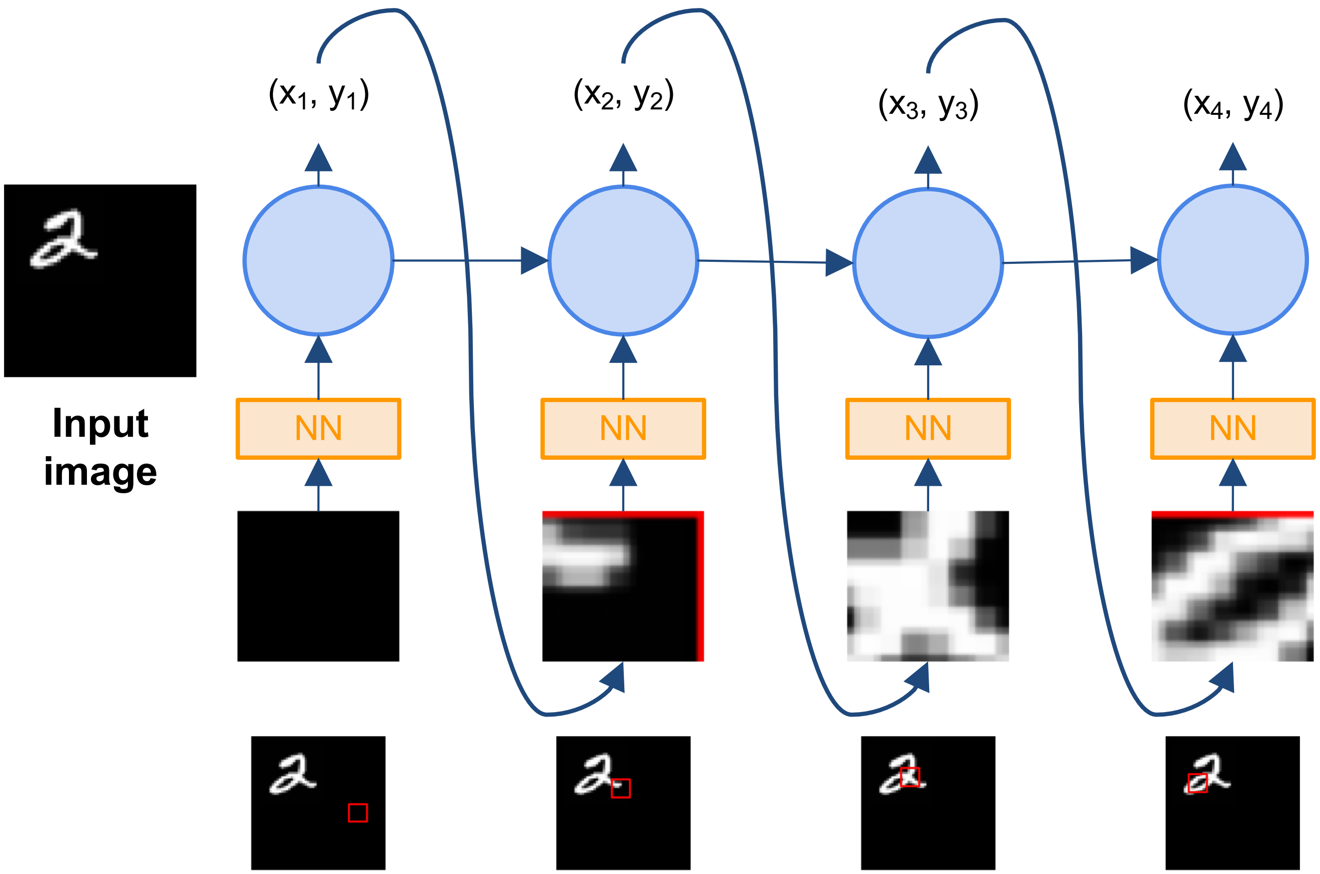
[Mnih et al. 2014]

REINFORCE in action: Recurrent Attention Model (RAM)



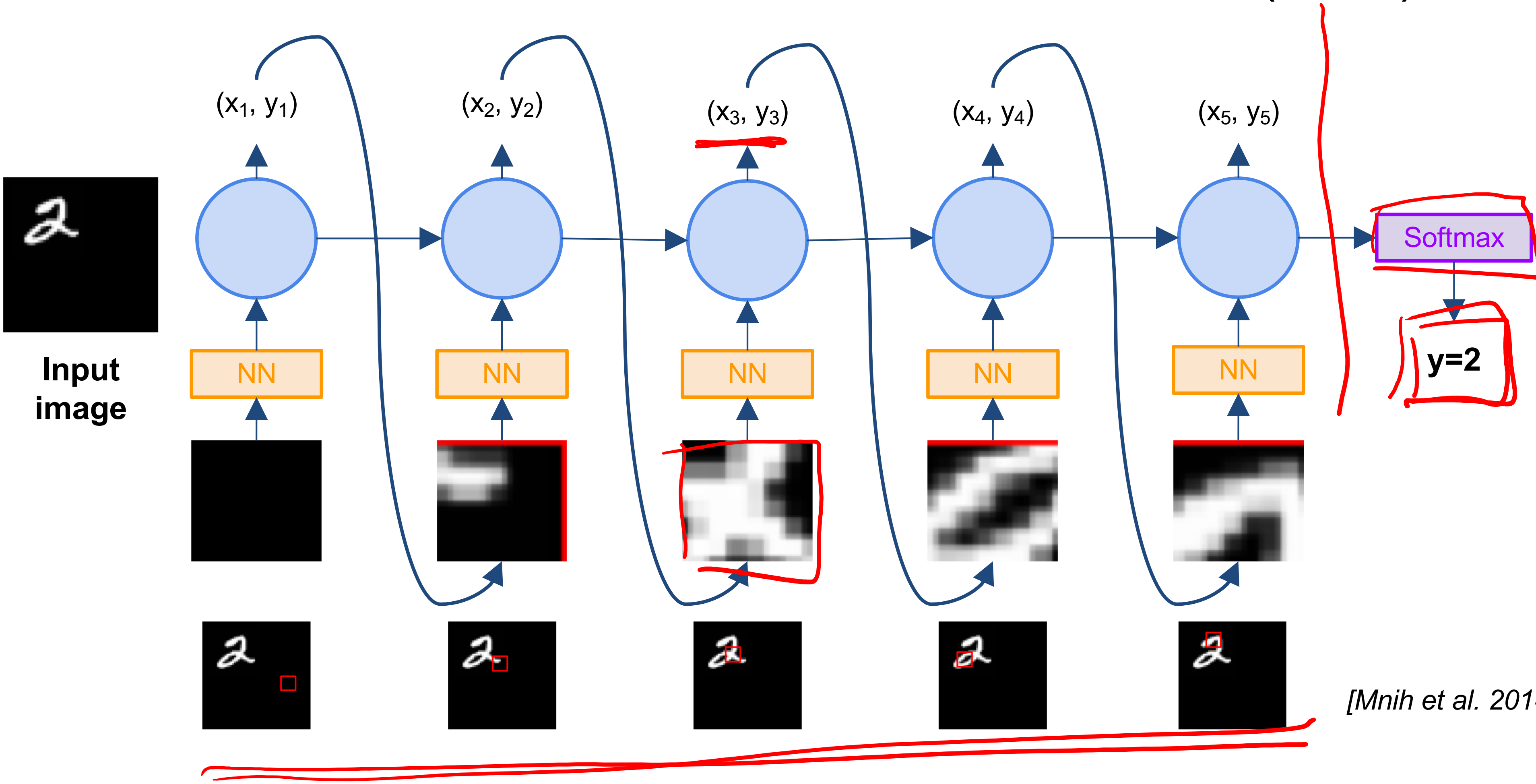
[Mnih et al. 2014]

REINFORCE in action: Recurrent Attention Model (RAM)



[Mnih et al. 2014]

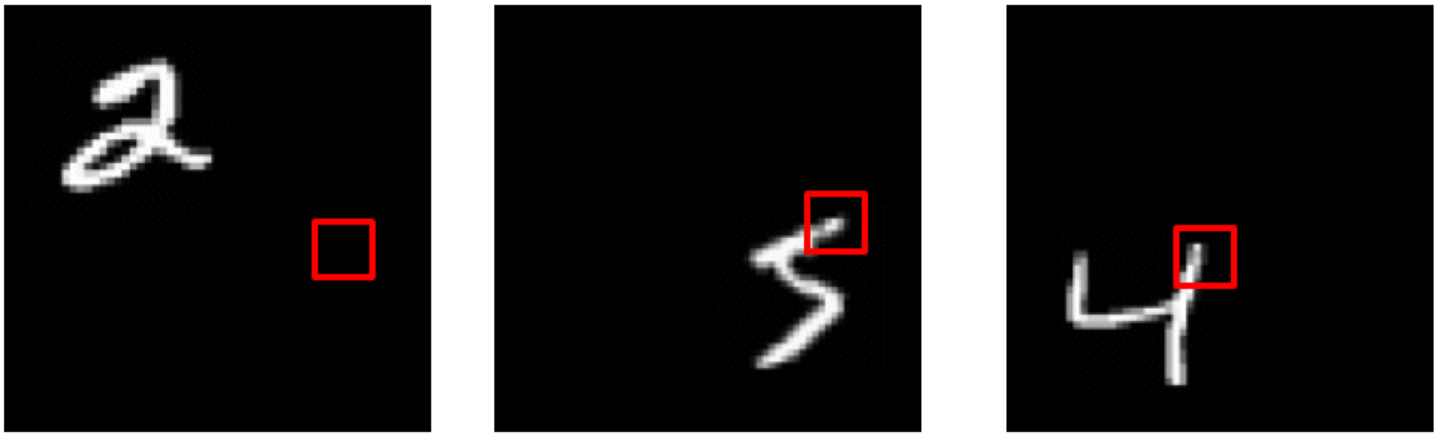
REINFORCE in action: Recurrent Attention Model (RAM)



[Mnih et al. 2014]

$\pi_\theta(a_t \mid o_t)$

REINFORCE in action: Recurrent Attention Model (RAM)



Has also been used in many other tasks including fine-grained image recognition, image captioning, and visual question-answering!

[Mnih et al. 2014]

Visual Dialog
A cat drinking water out of a coffee mug.
What color is the mug?
White and red
Are there any pictures on it?
No, something is there can't tell what it is
Is the mug and cat on a table?
Yes, they are
Are there other items on the table?
Yes, magazines, books, toaster and basket, and a plate
Start typing question here ...

Learning Cooperative Visual Dialog Agents with Deep Reinforcement Learning

[ICCV ‘17]

Abhishek Das*
(Georgia Tech)

Satwik Kottur*
(CMU)

José Moura
(CMU)

Stefan Lee
(Virginia Tech)

Dhruv Batra
(Georgia Tech)

Image Guessing Game

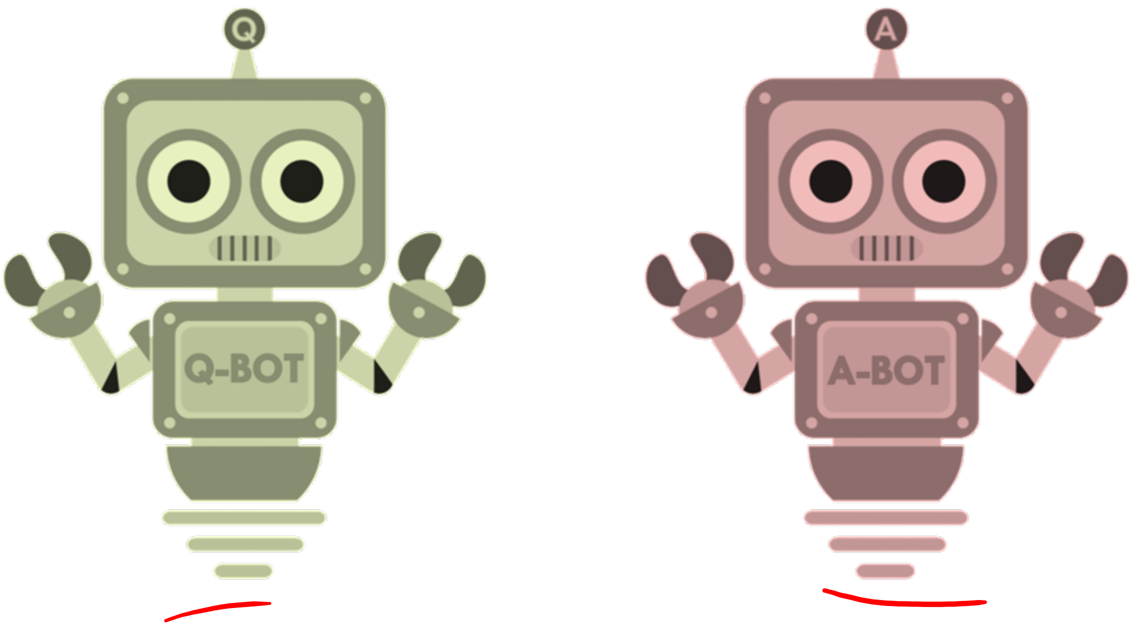
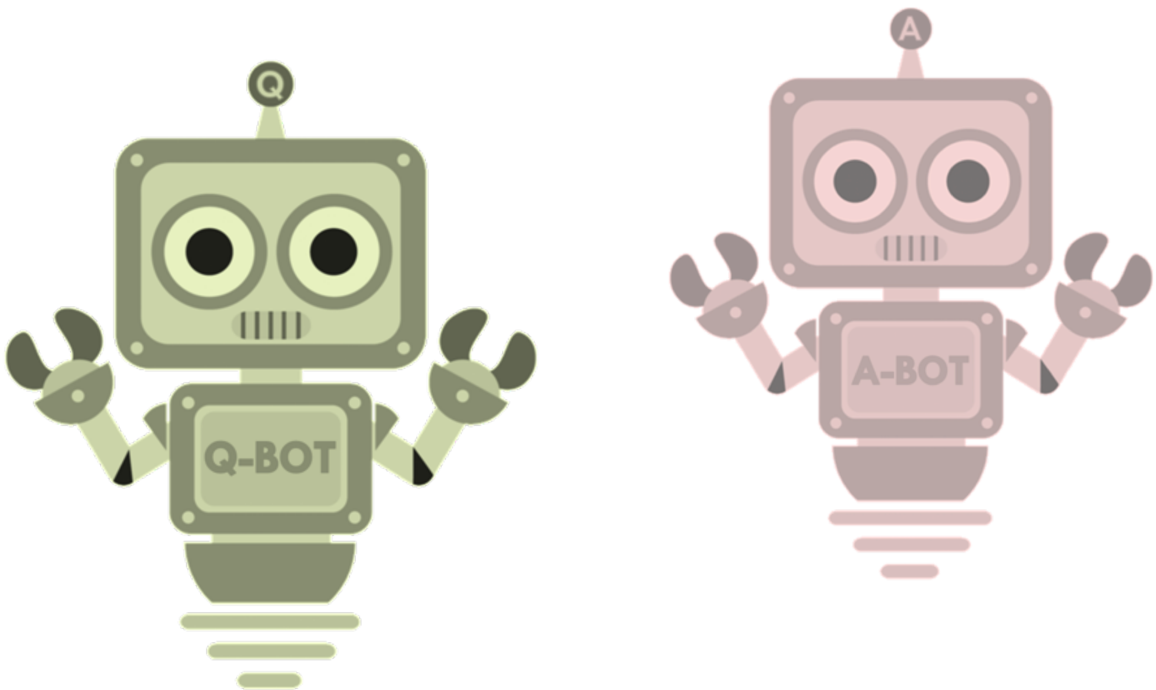
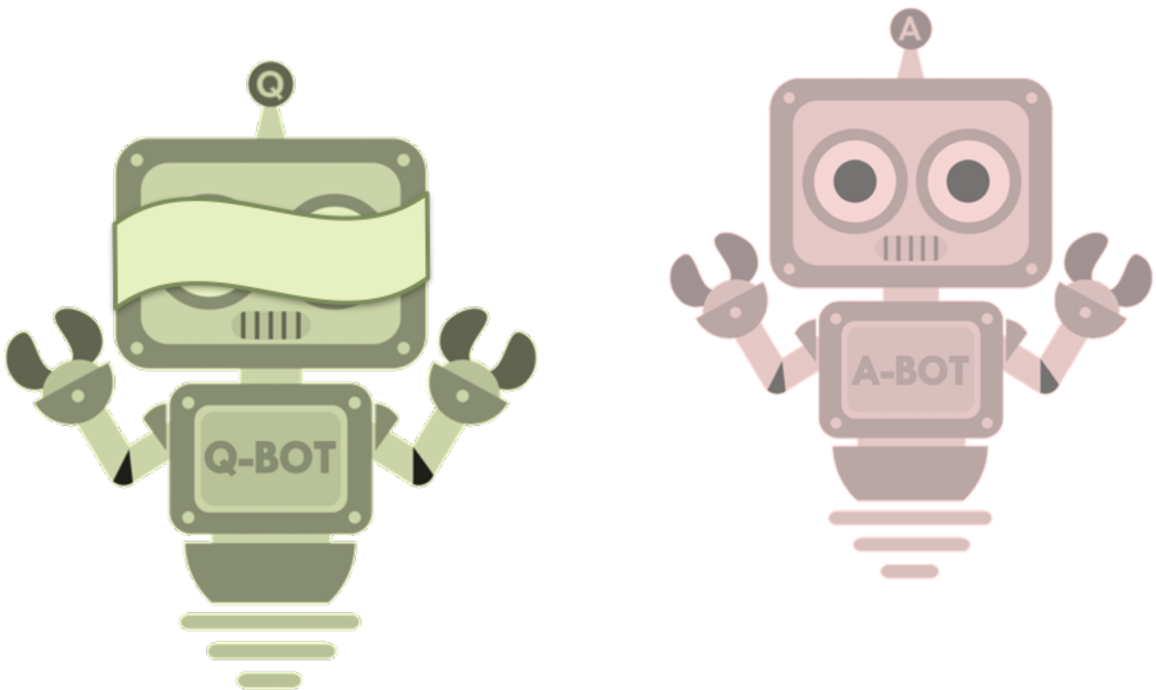


Image Guessing Game



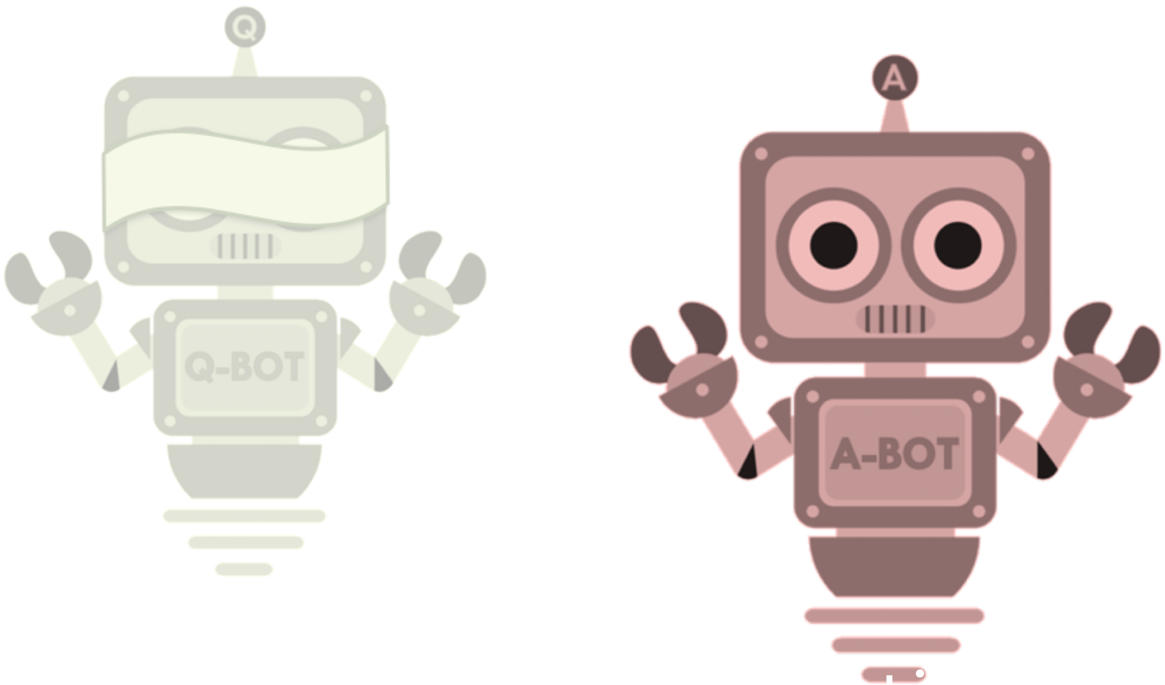
Q-Bot asks questions

Image Guessing Game



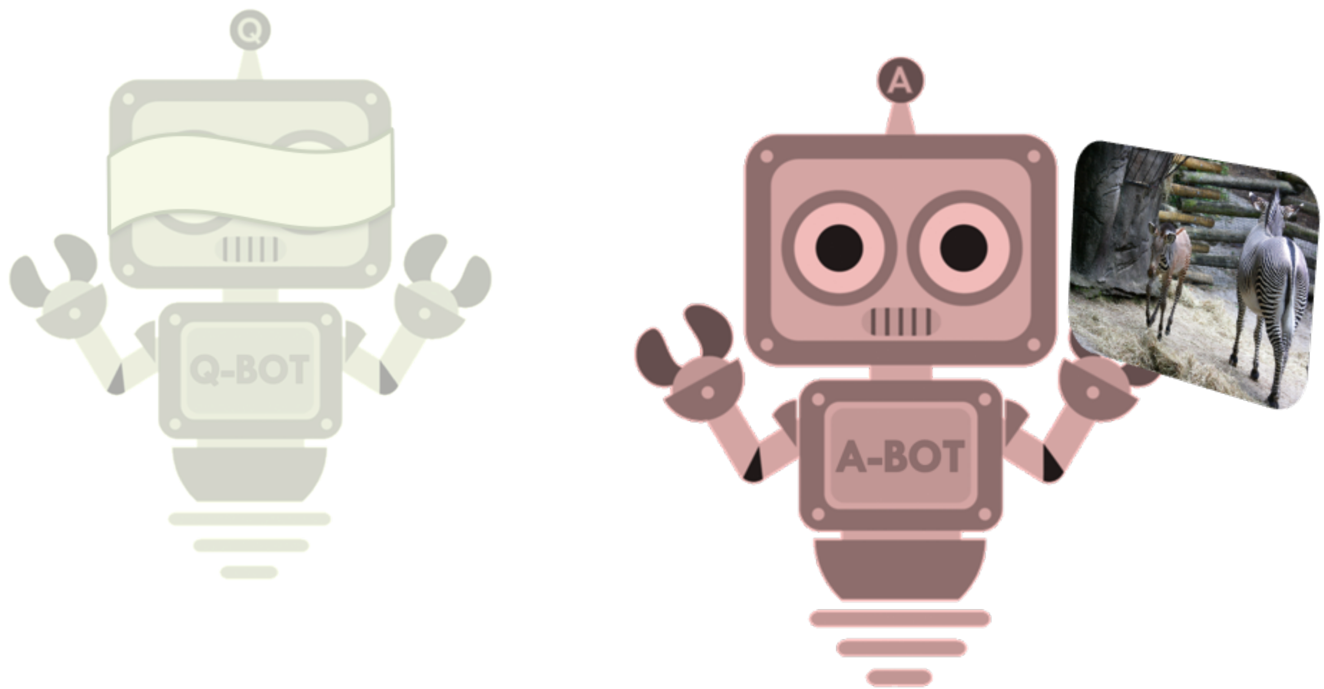
Q-Bot is blindfolded

Image Guessing Game



A-Bot answers questions

Image Guessing Game



A-Bot sees an image

Image Guessing Game

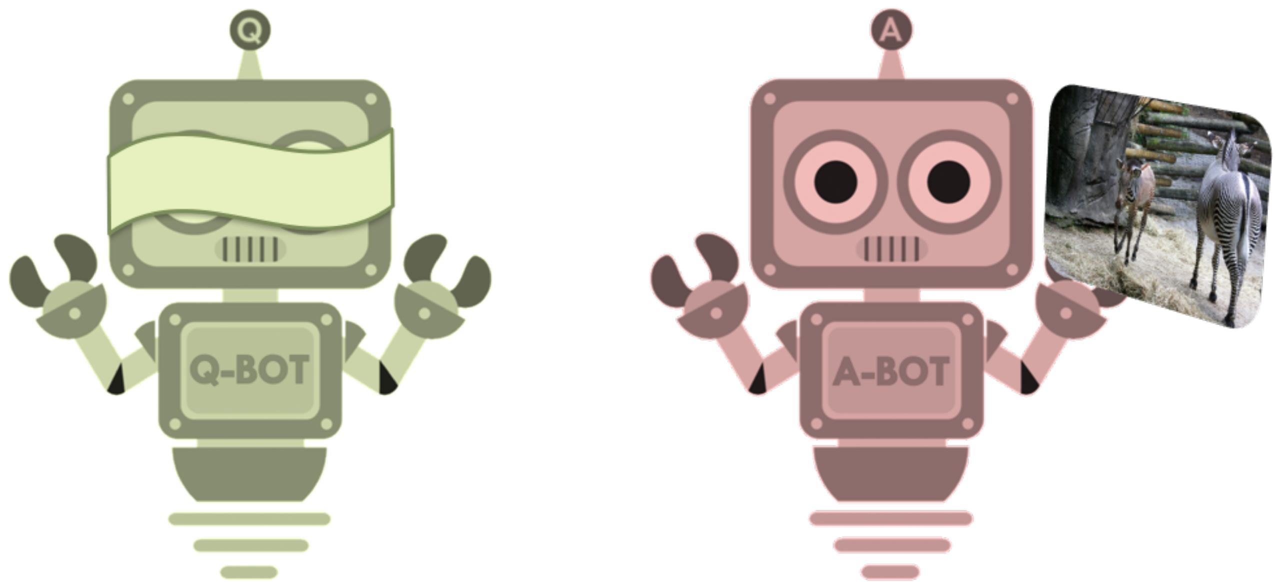


Image Guessing Game

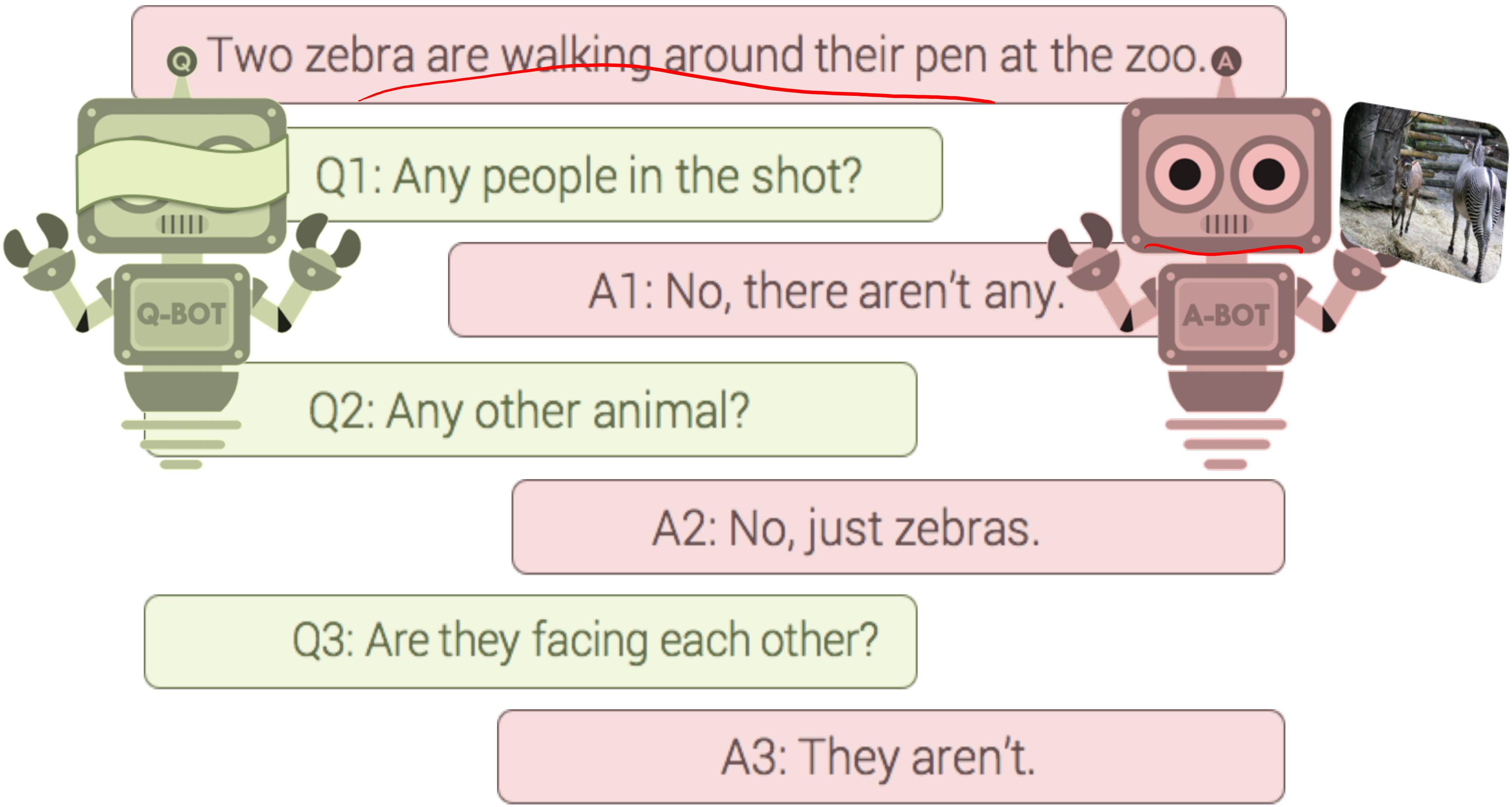
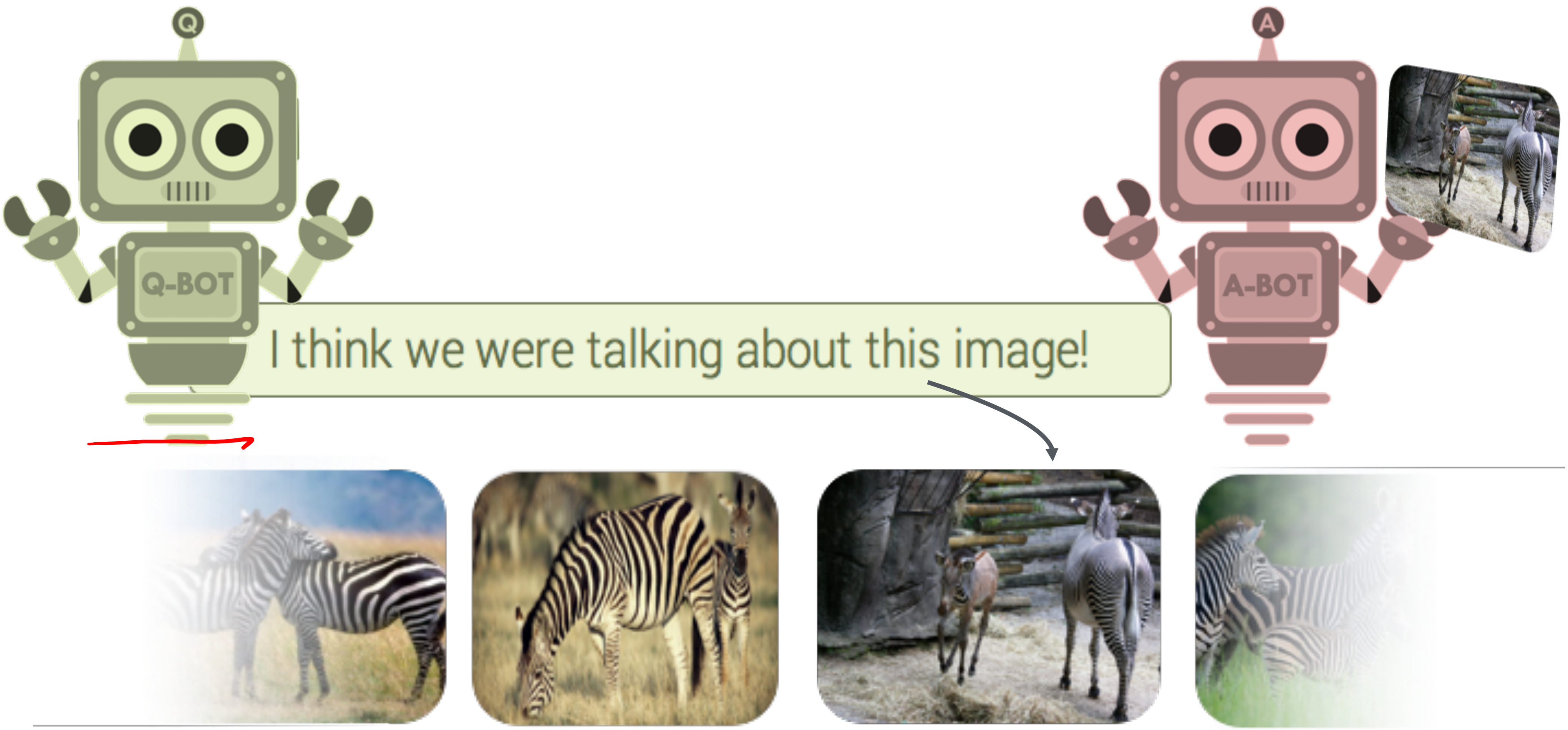
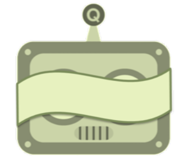


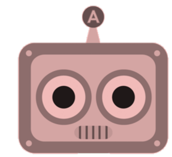
Image Guessing Game

A3: They aren't.

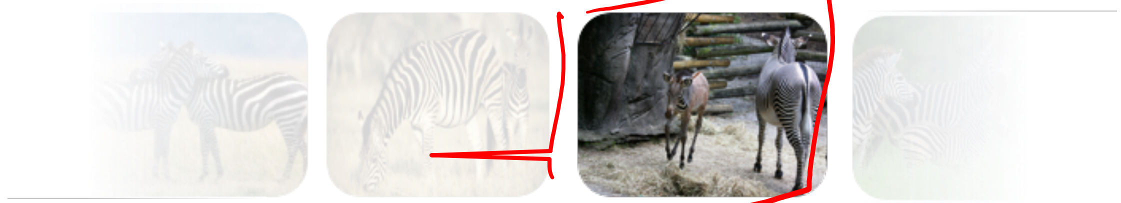


RL for Cooperative Dialog Agents



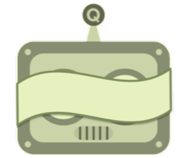


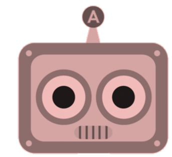
- Agents: (Q-bot, A-bot)
- Environment: Image



- Action:
  - Q-bot: question (symbol sequence) — **q_t** *Any people in the shot?*
  - A-bot: answer (symbol sequence) — **a_t** *No, there aren't any.*
  - Q-bot: image regression — $\hat{y}_t \in \mathbb{R}^{4096}$
- State
  - Q-bot: $s_t^Q = [c, q_1, a_1, \ldots, q_{t-1}, a_{t-1}]$
  - A-bot: $s_t^A = [I, c, q_1, a_1, \ldots, q_{t-1}, a_{t-1}, q_t]$

RL for Cooperative Dialog Agents

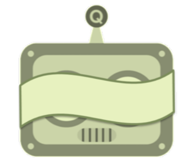
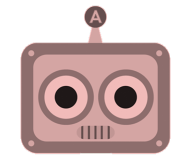




- Action:
  - Q-bot: question (symbol sequence) q_t *Any people in the shot?*
  - A-bot: answer (symbol sequence) a_t *No, there aren't any.*
  - Q-bot: image regression $\hat{y}_t \in \mathbb{R}^{4096}$

- State
  - Q-bot: $s_t^Q = [c, q_1, a_1, \ldots, q_{t-1}, a_{t-1}]$
  - A-bot: $s_t^A = [I, c, q_1, a_1, \ldots, q_{t-1}, a_{t-1}, q_t]$

RL for Cooperative Dialog Agents

- Action:
  - Q-bot: question (symbol sequence)
  - A-bot: answer (symbol sequence)
  - Q-bot: image regression

q_t *Any people in the shot?*

a_t *No, there aren't any.*

$\hat{y}_t \in \mathbb{R}^{4096}$

- State
  - Q-bot: $s_t^Q = [c, q_1, a_1, \ldots, q_{t-1}, a_{t-1}]$
  - A-bot: $s_t^A = [I, c, q_1, a_1, \ldots, q_{t-1}, a_{t-1}, q_t]$

- Policy

Q-bot: $\pi_Q(q_t | S_{t-1}^Q)$

A-bot: $\pi_A(a_t | S_{t-1}^A)$

- Reward

$$r_t\Big(\underbrace{s_t^Q}_{\text{state}}, \underbrace{(q_t, a_t, y_t)}_{\text{action}}\Big) = \underbrace{\ell\left(\hat{y}_{t-1}, y^{gt}\right)}_{\text{distance at } t\text{-}1} - \underbrace{\ell\left(\hat{y}_t, y^{gt}\right)}_{\text{distance at } t}$$

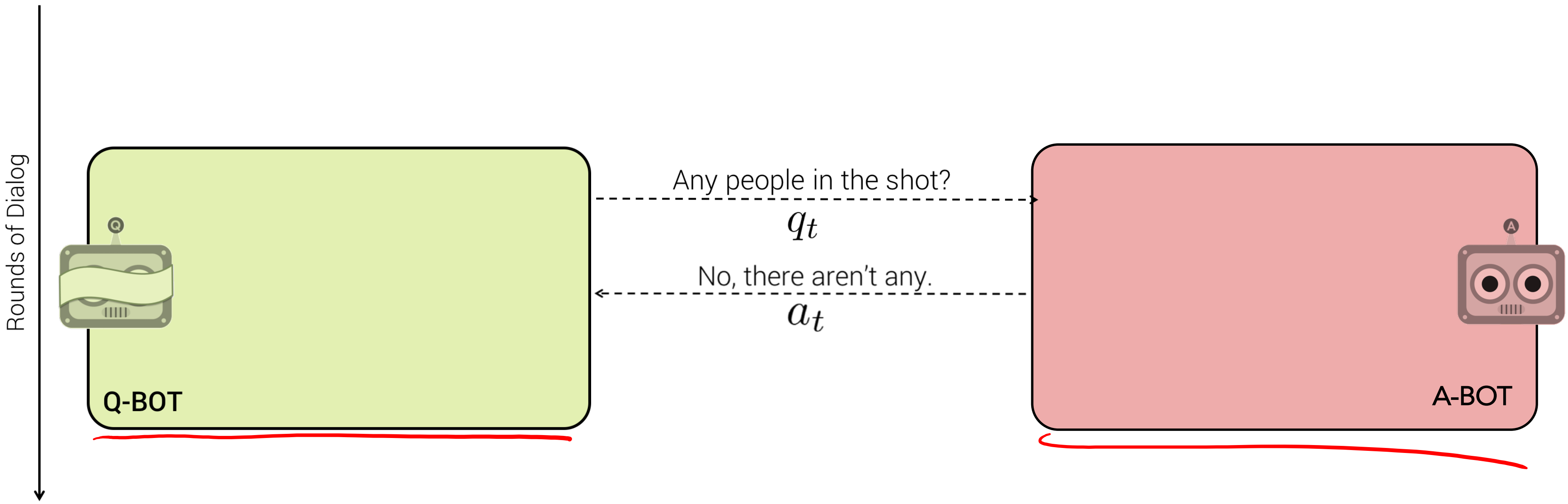
Policy Networks

Q-Bot

$\pi_Q(q_t|S^Q_{t-1})$

A-Bot

$\pi_A(a_t|S^A_{t-1})$



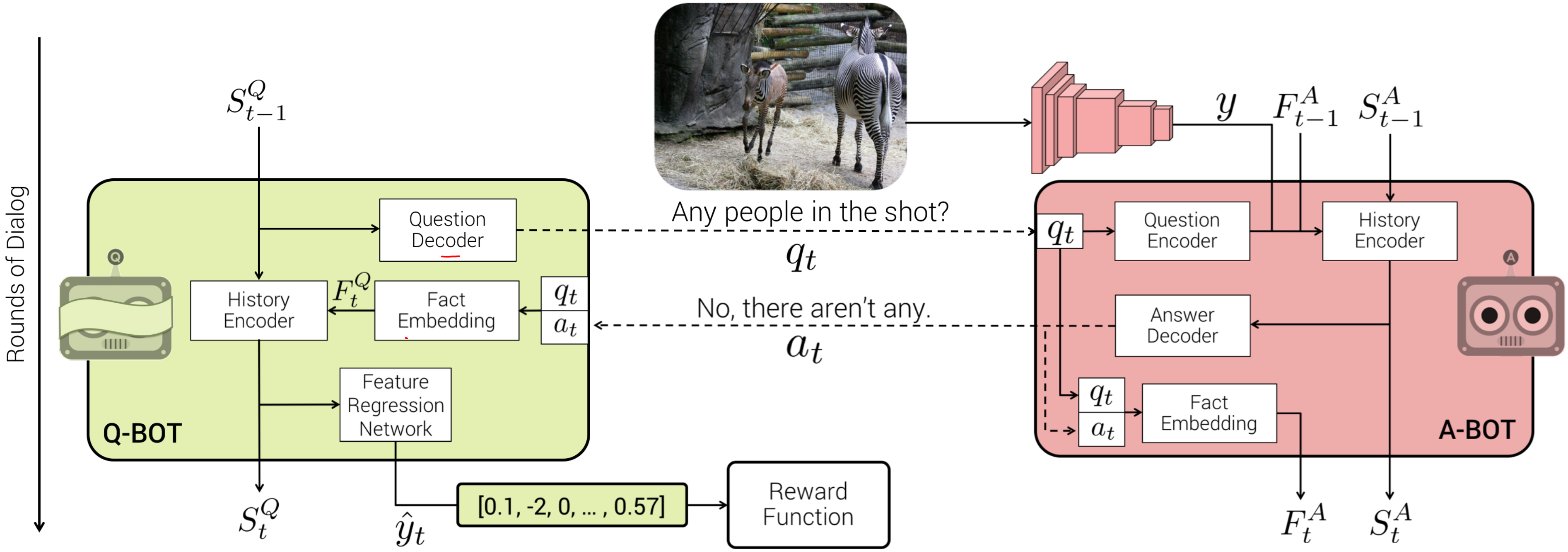
Policy Networks

Q-Bot

$\pi_Q(q_t|S^Q_{t-1})$

A-Bot

$\pi_A(a_t|S^A_{t-1})$

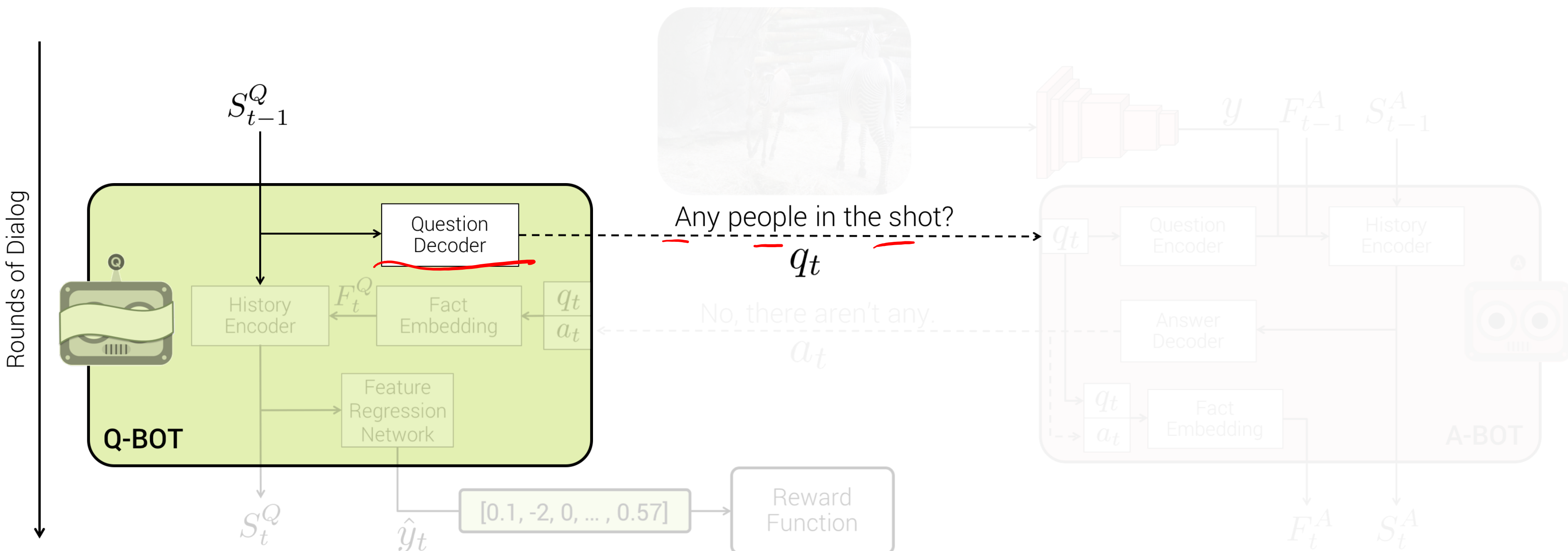


Slide Credit: Abhishek Das

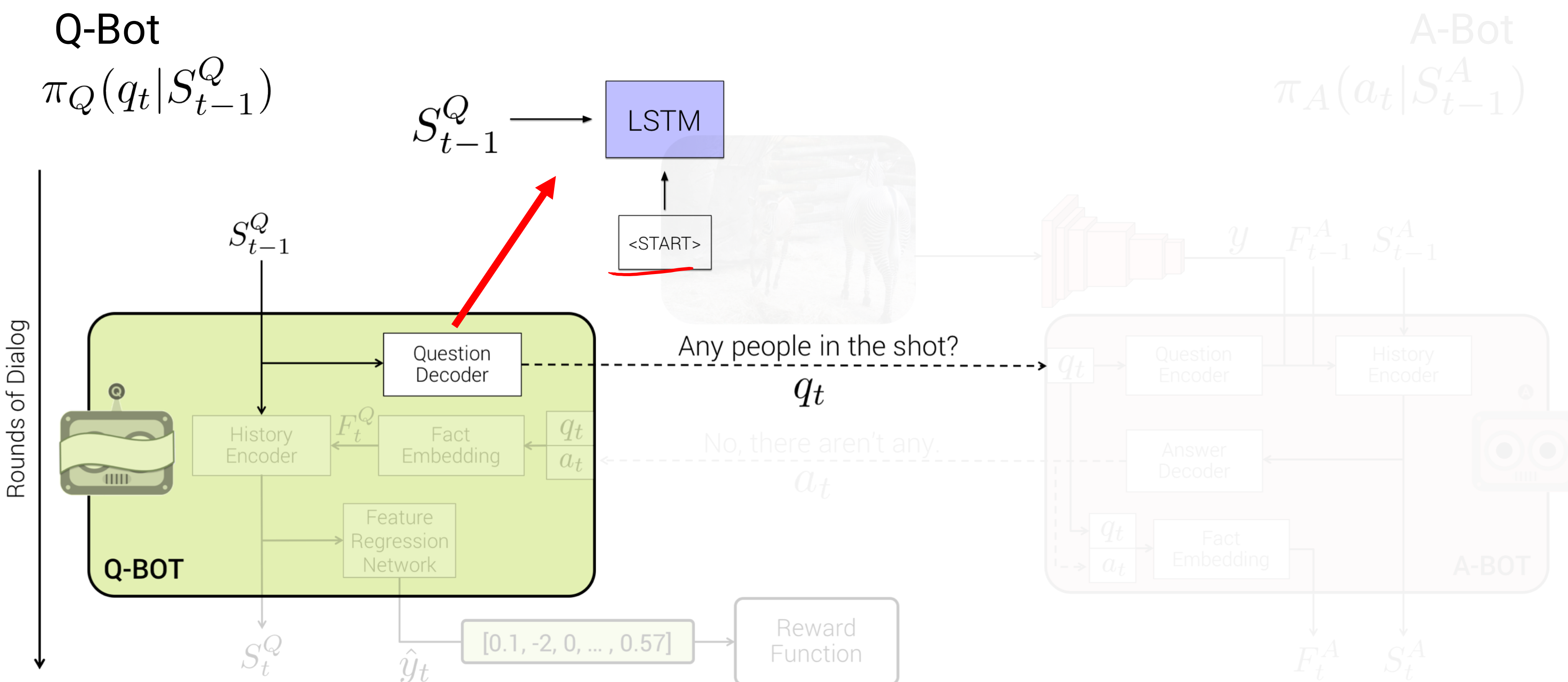
Policy Networks

Q-Bot
$\pi_Q(q_t|S^Q_{t-1})$

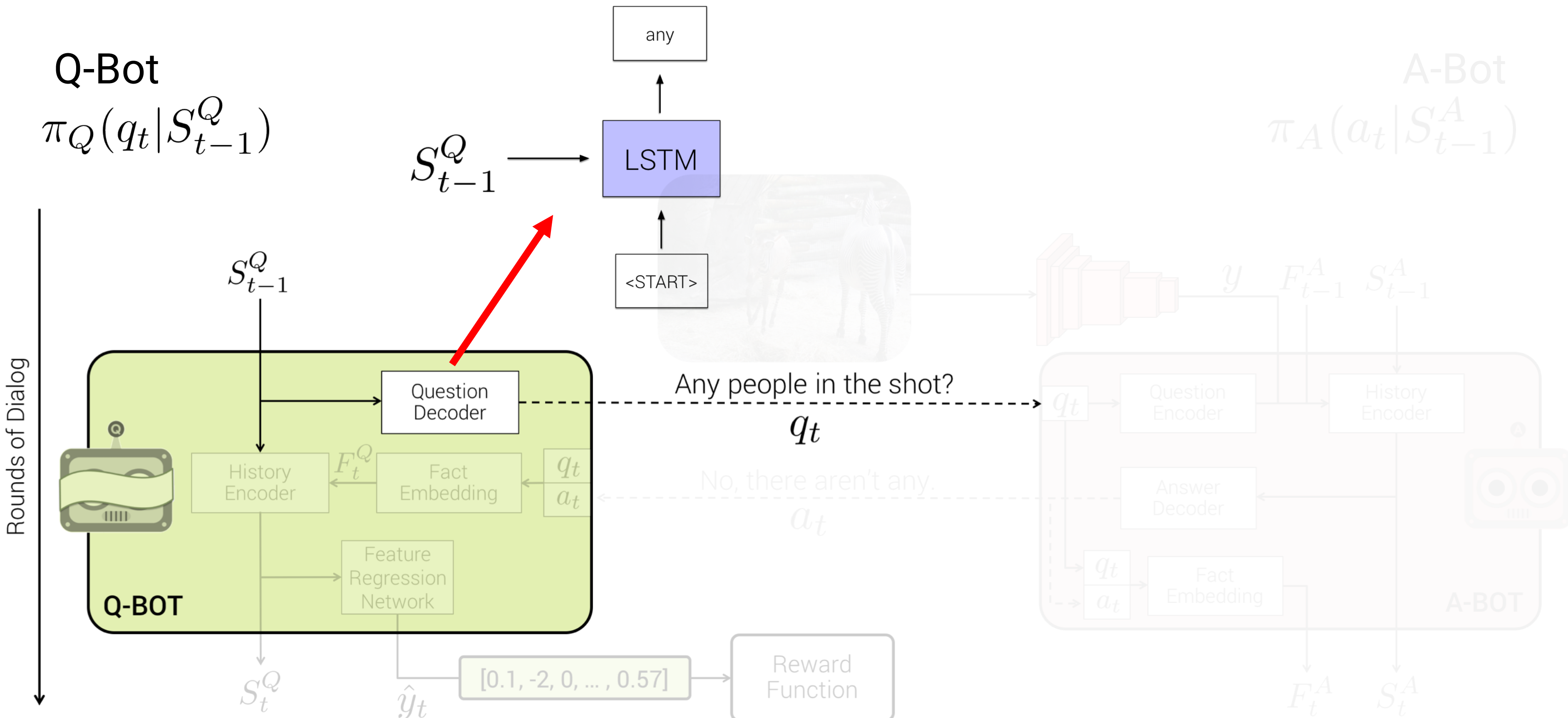
A-Bot
$\pi_A(a_t|S^A_{t-1})$



Policy Networks



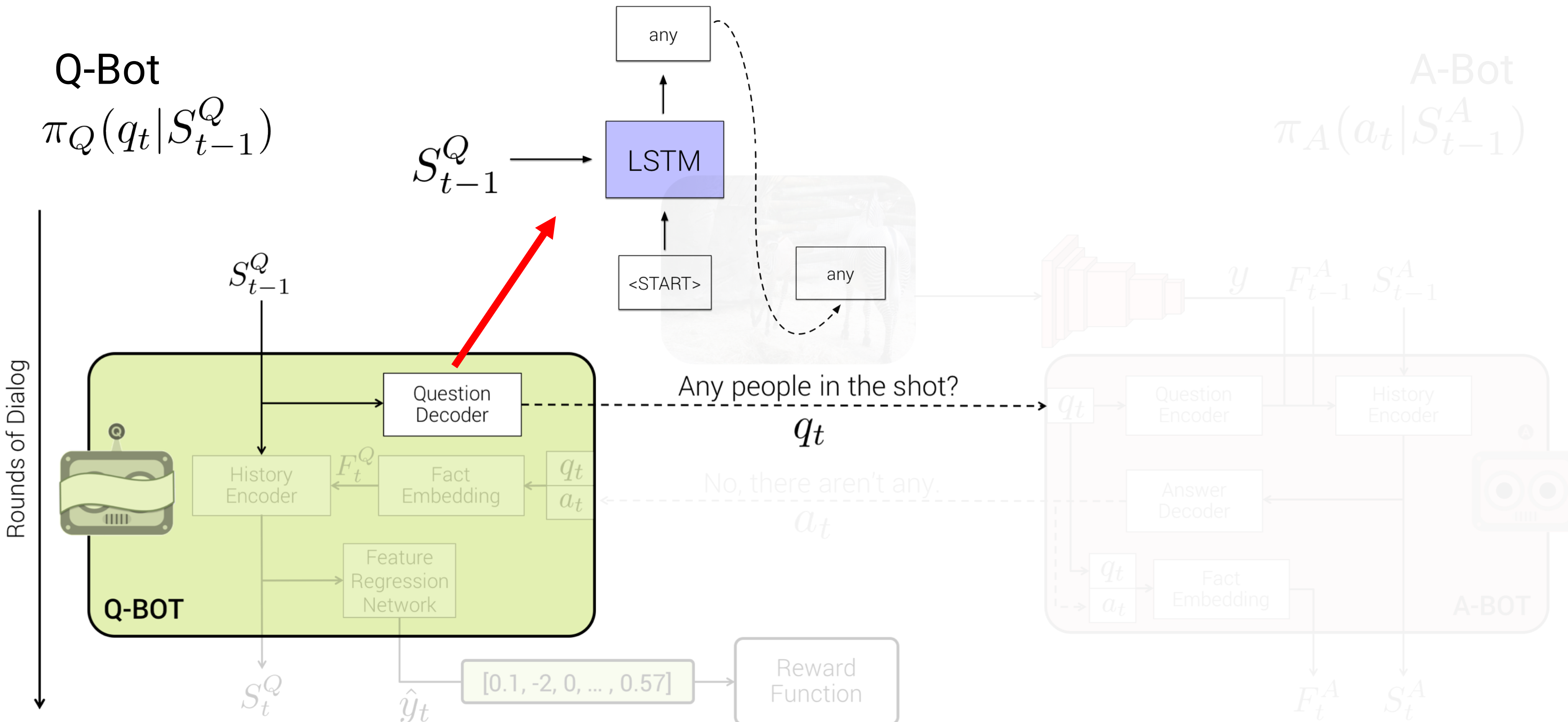
Policy Networks



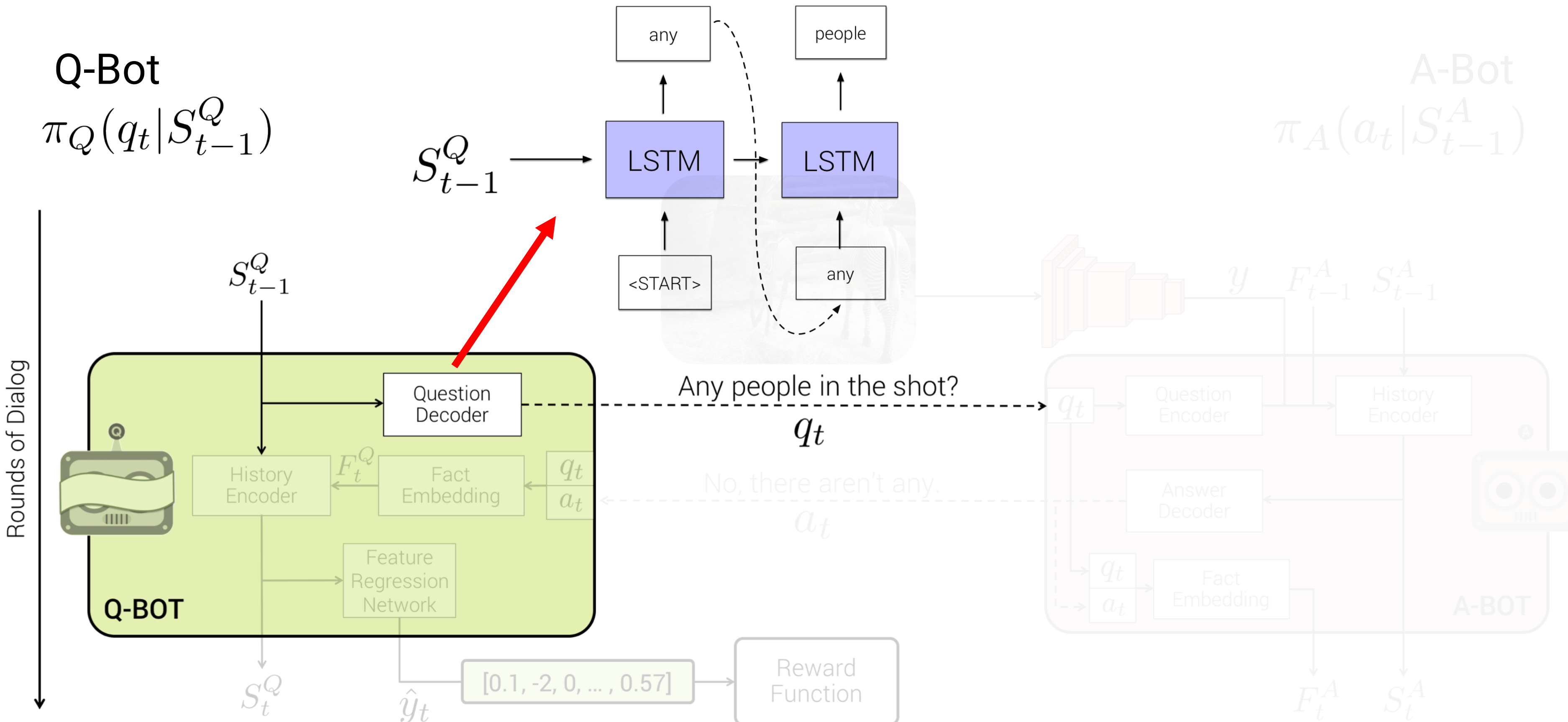
Policy Networks

Q-Bot

$\pi_Q(q_t | S^Q_{t-1})$

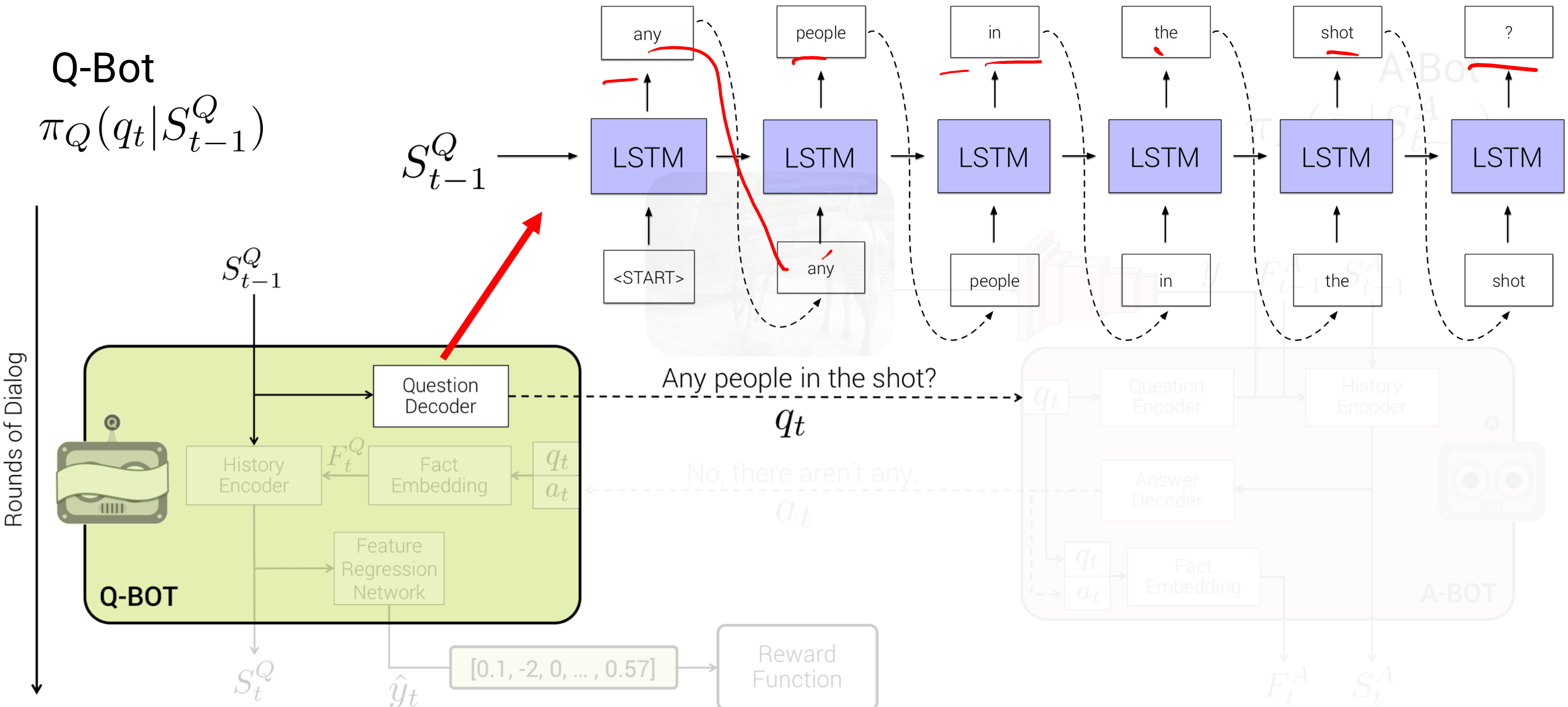


Policy Networks



Policy Networks

Q-Bot
$\pi_Q(q_t | S^Q_{t-1})$



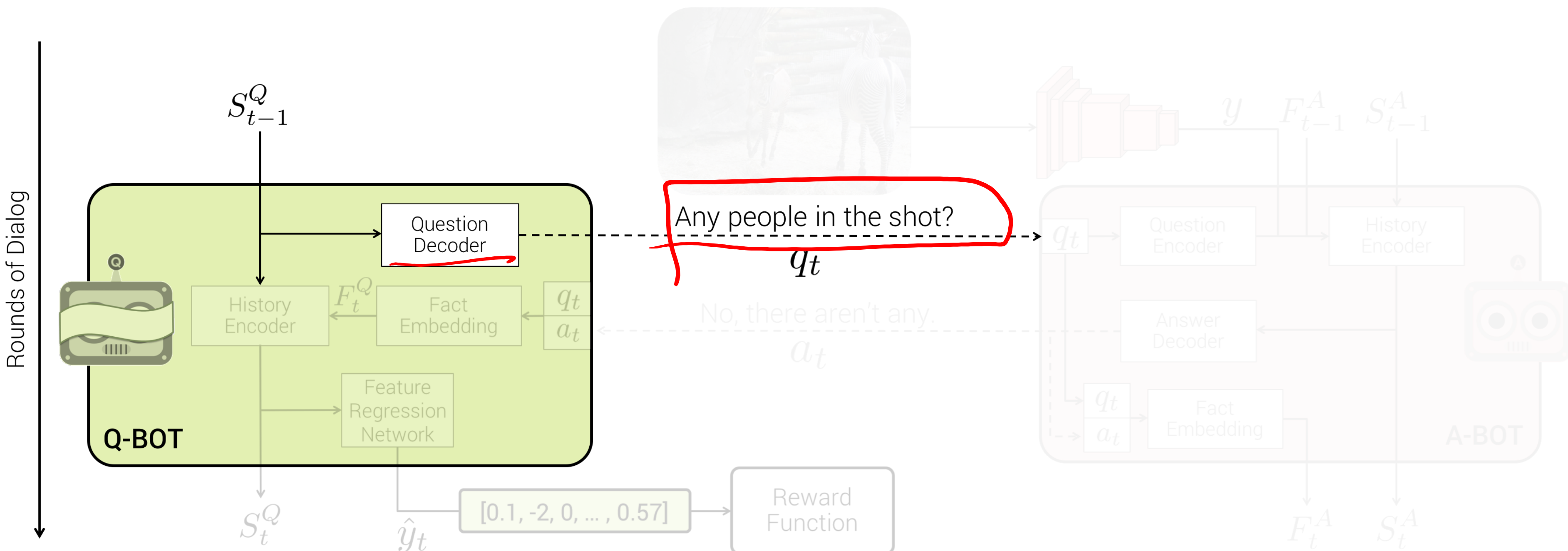
Policy Networks

Q-Bot

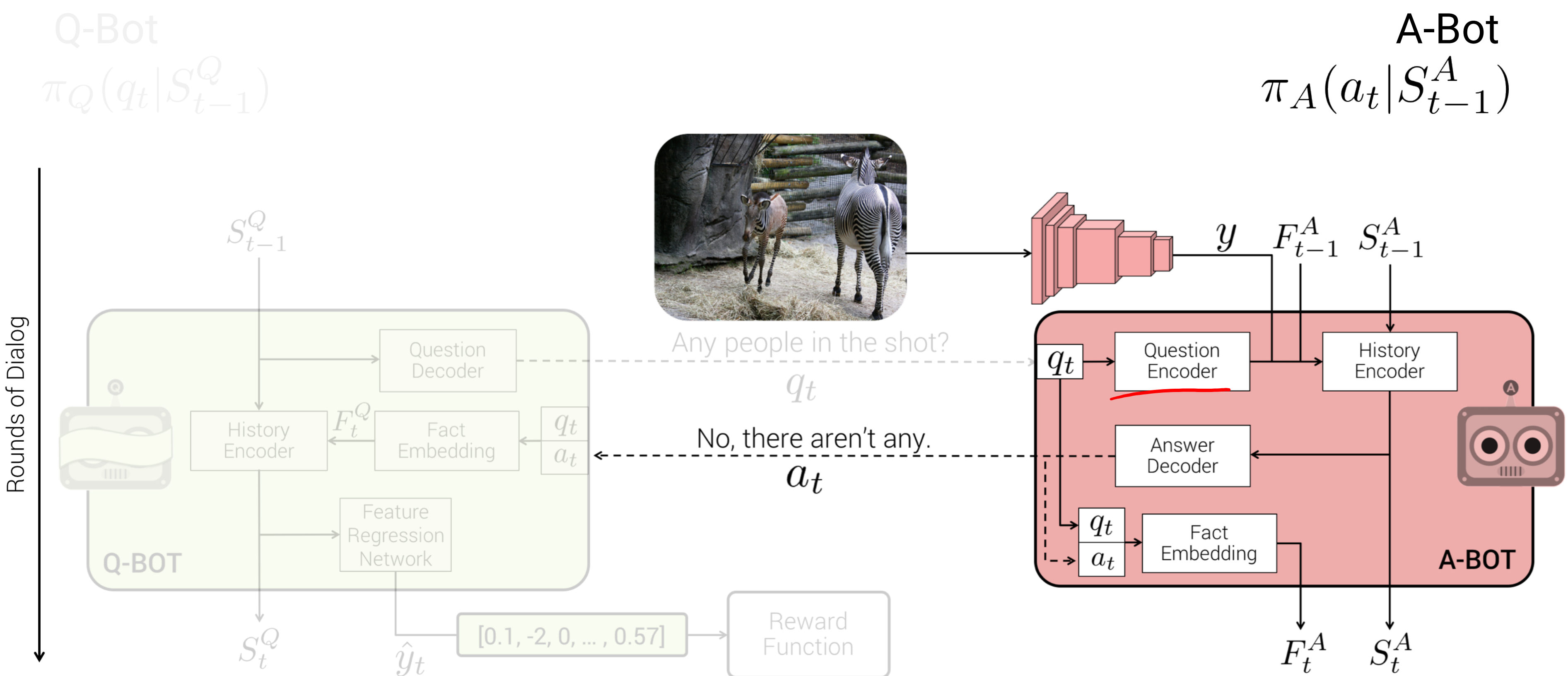
$\pi_Q(q_t | S^Q_{t-1})$

A-Bot

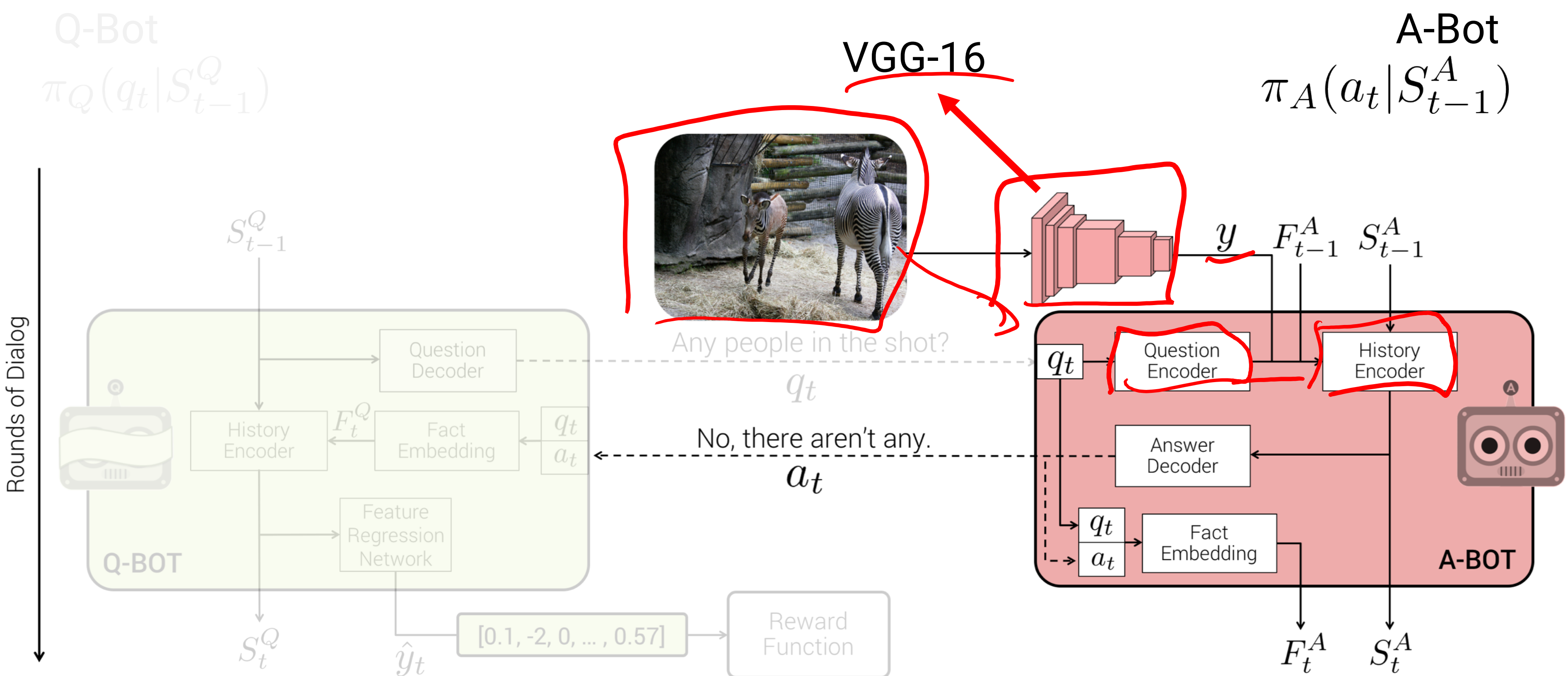
$\pi_A(a_t | S^A_{t-1})$



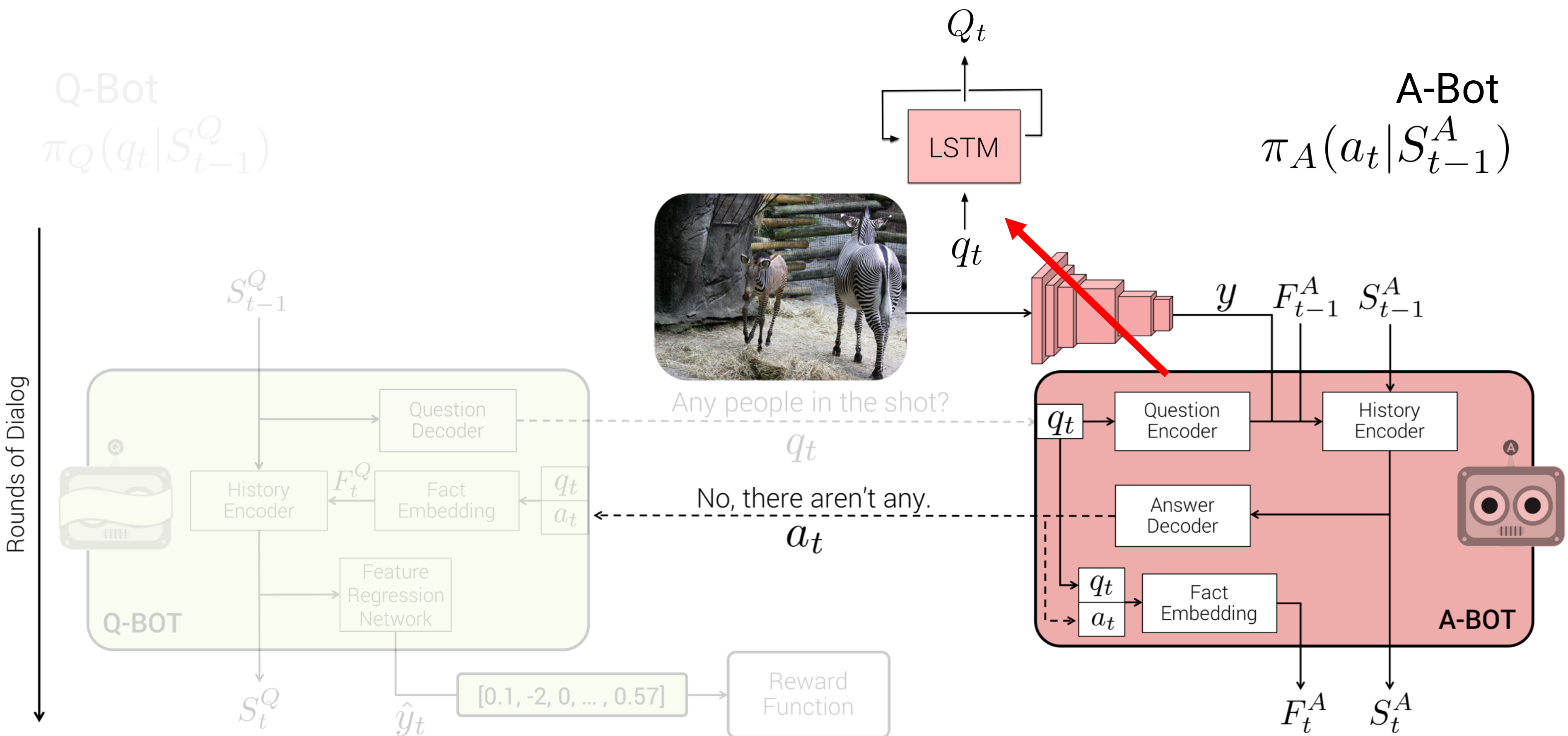
Policy Networks



Policy Networks



Policy Networks



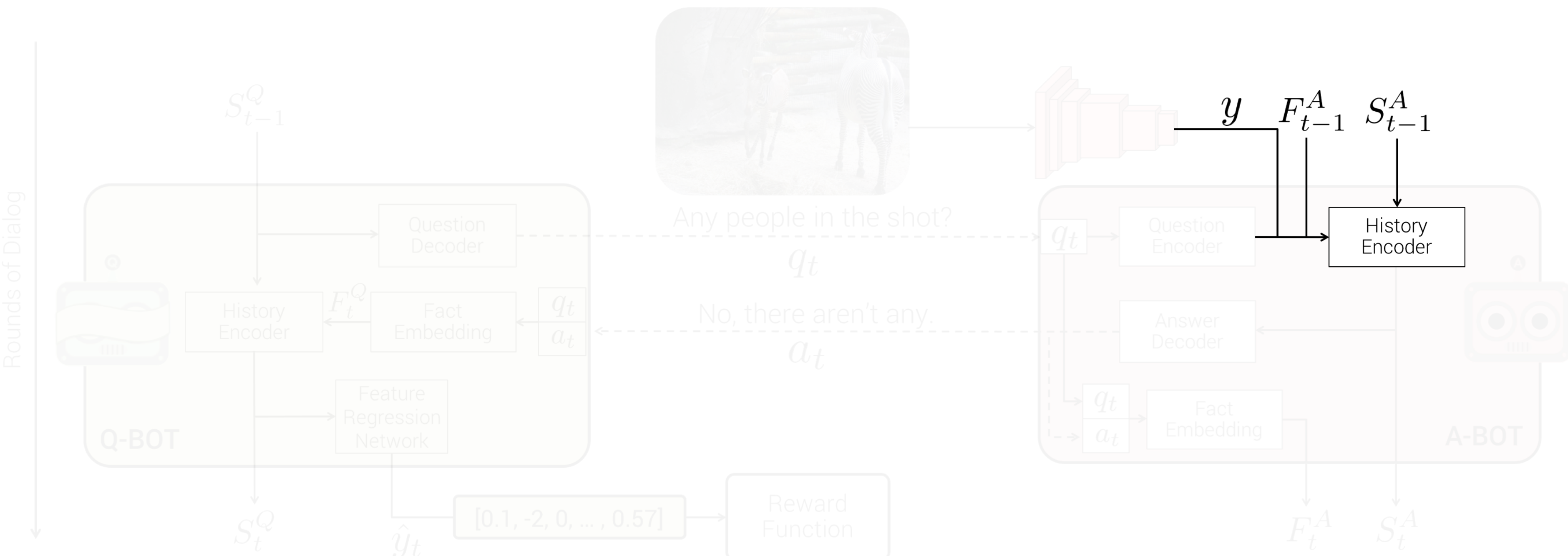
Policy Networks

Q-Bot

$\pi_Q(q_t | S_{t-1}^Q)$

A-Bot

$\pi_A(a_t | S_{t-1}^A)$

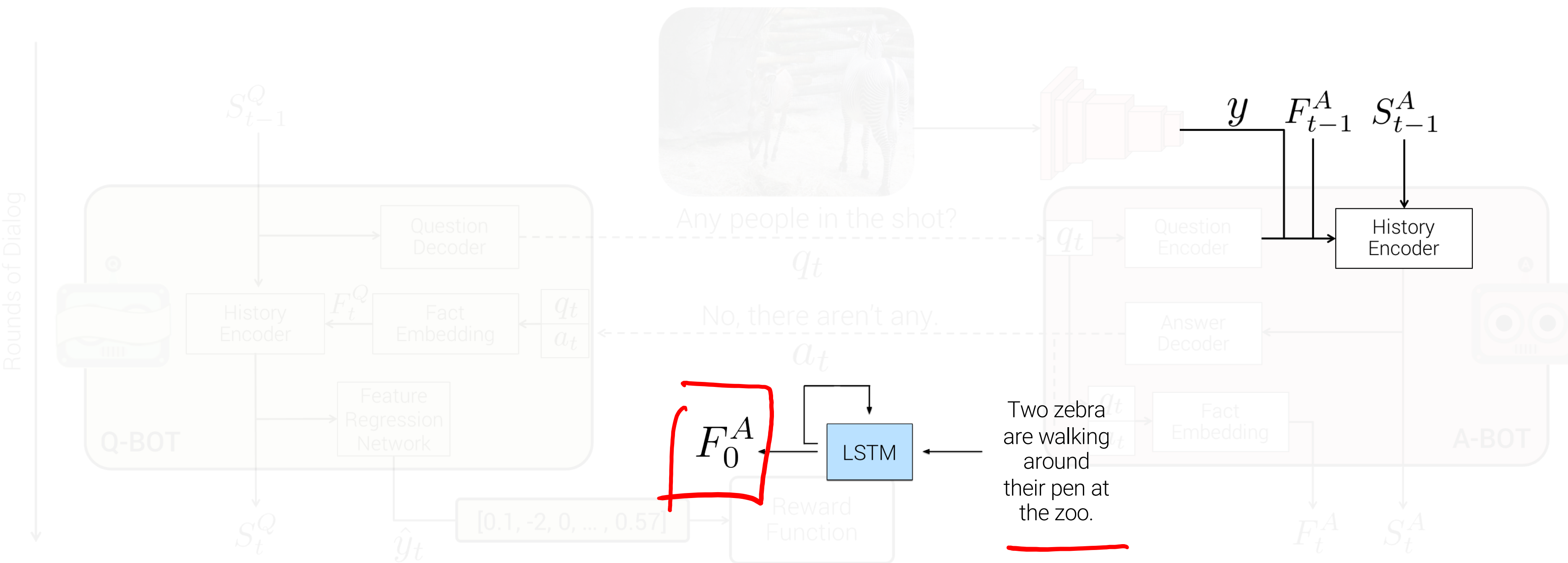


Policy Networks

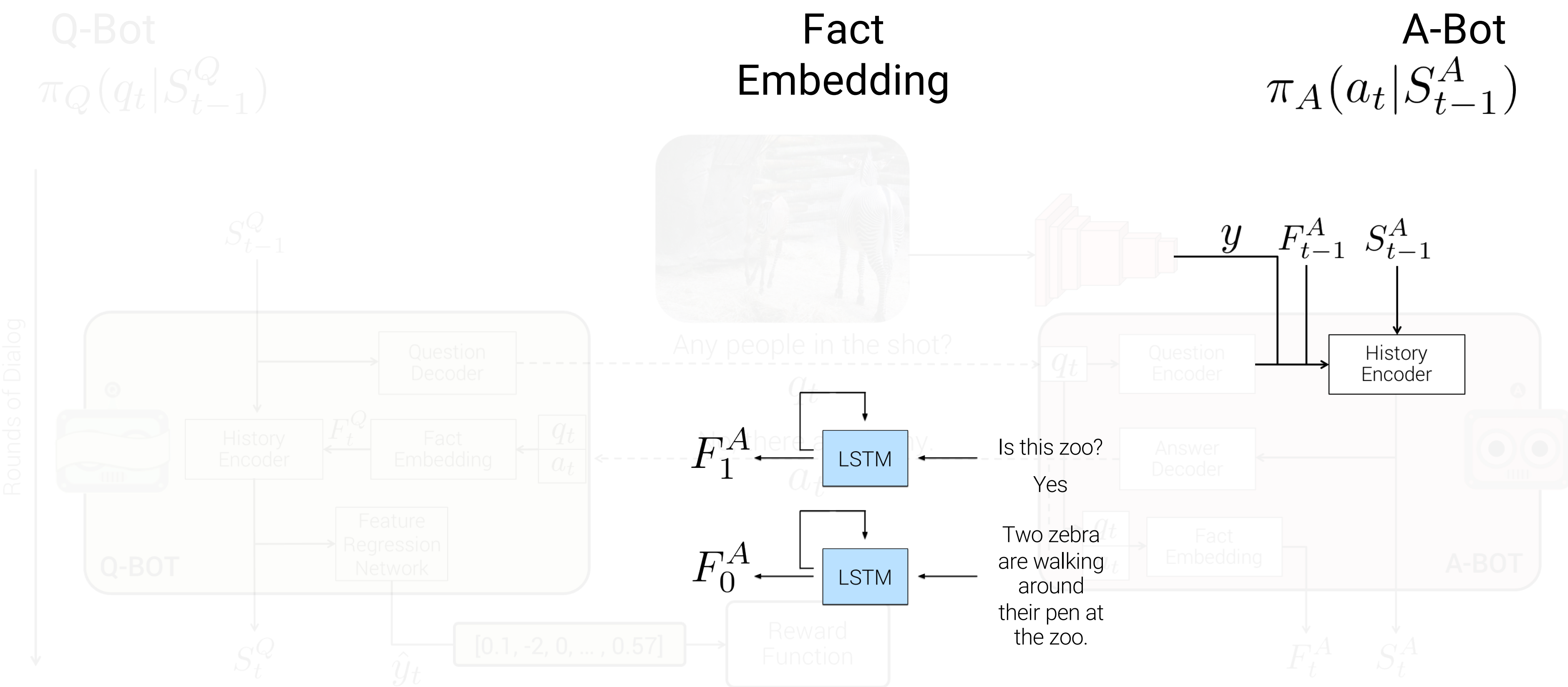
Fact Embedding

A-Bot

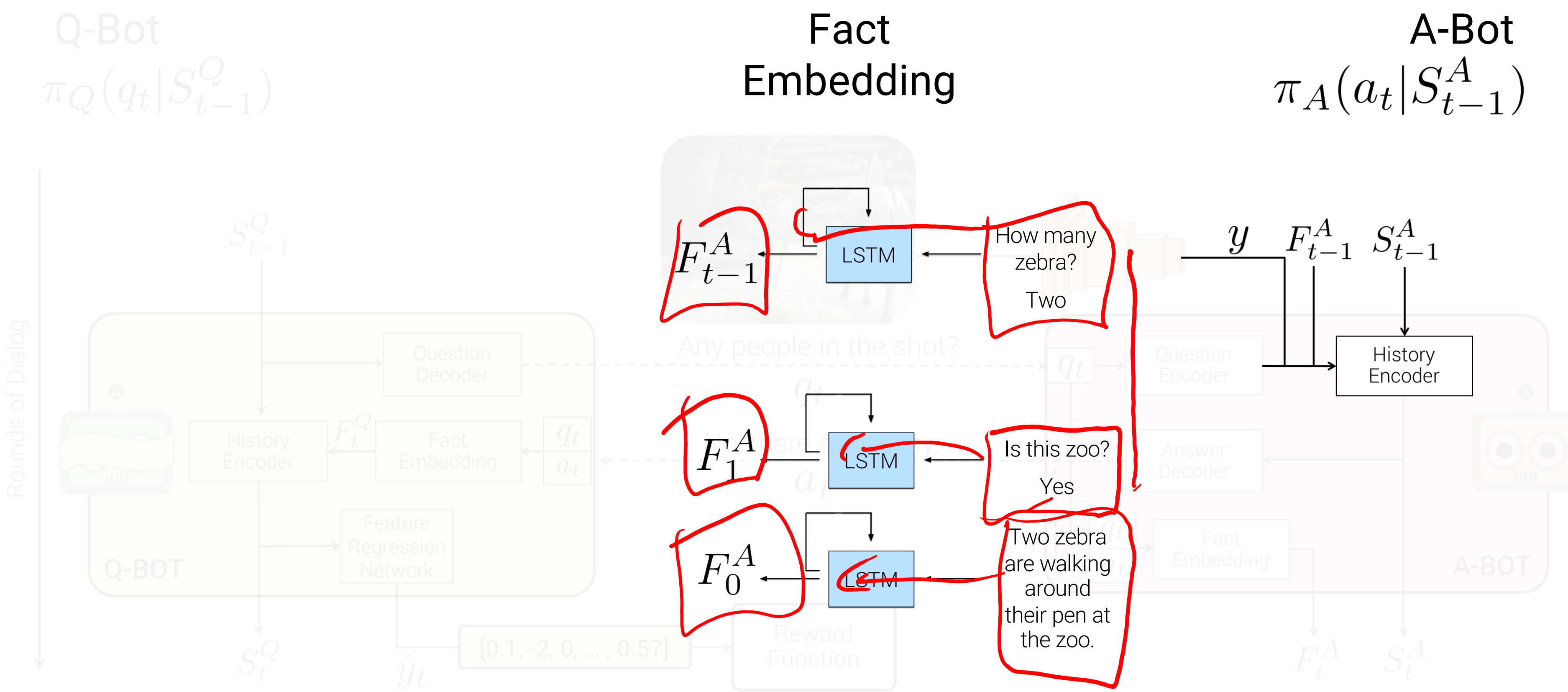
$\pi_A(a_t | S^A_{t-1})$



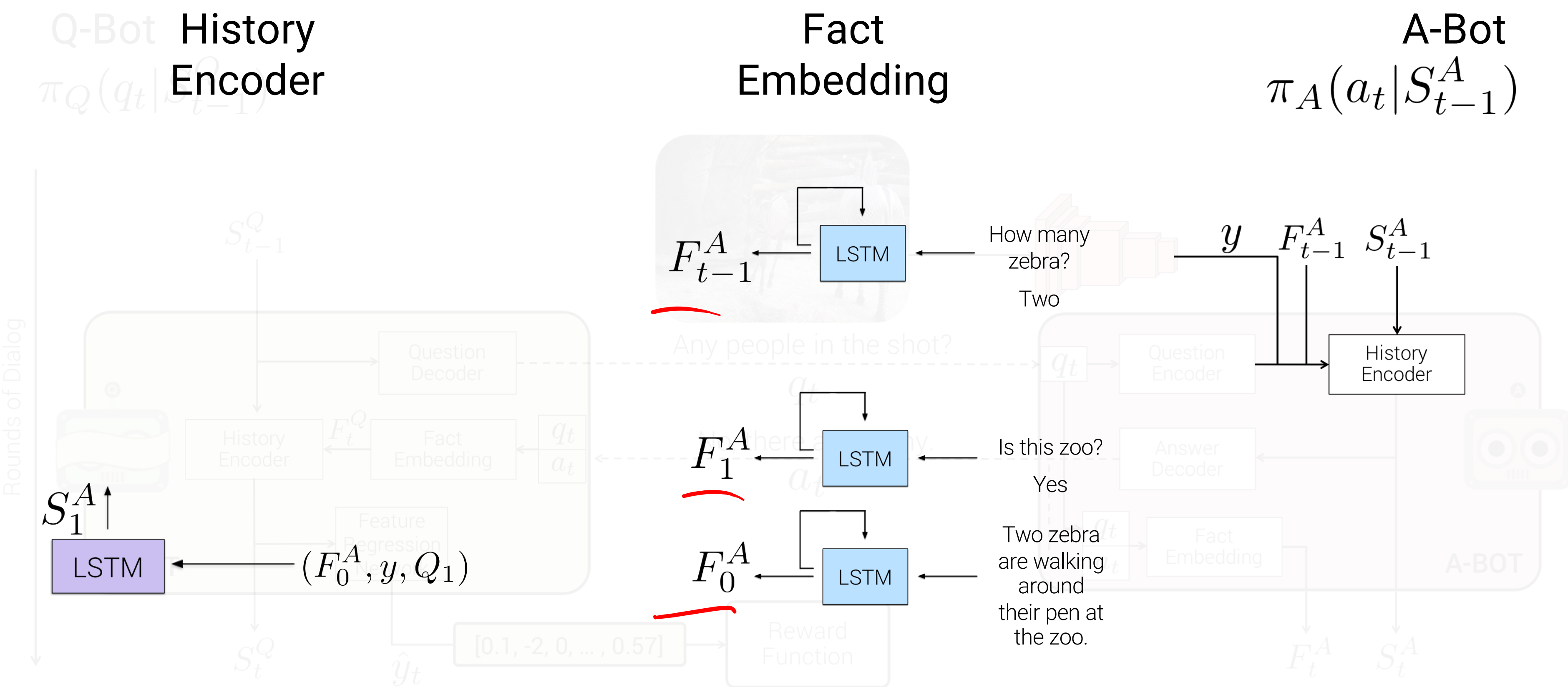
Policy Networks



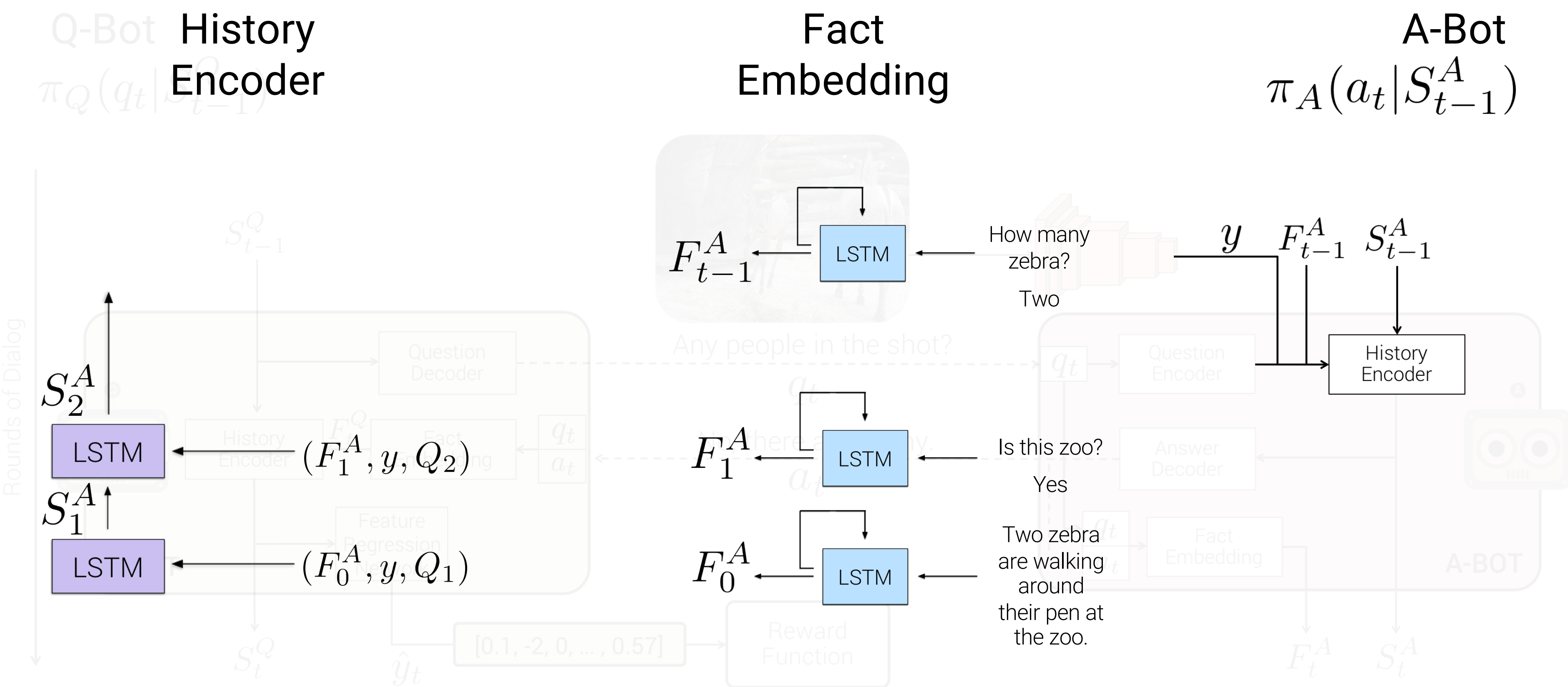
Policy Networks



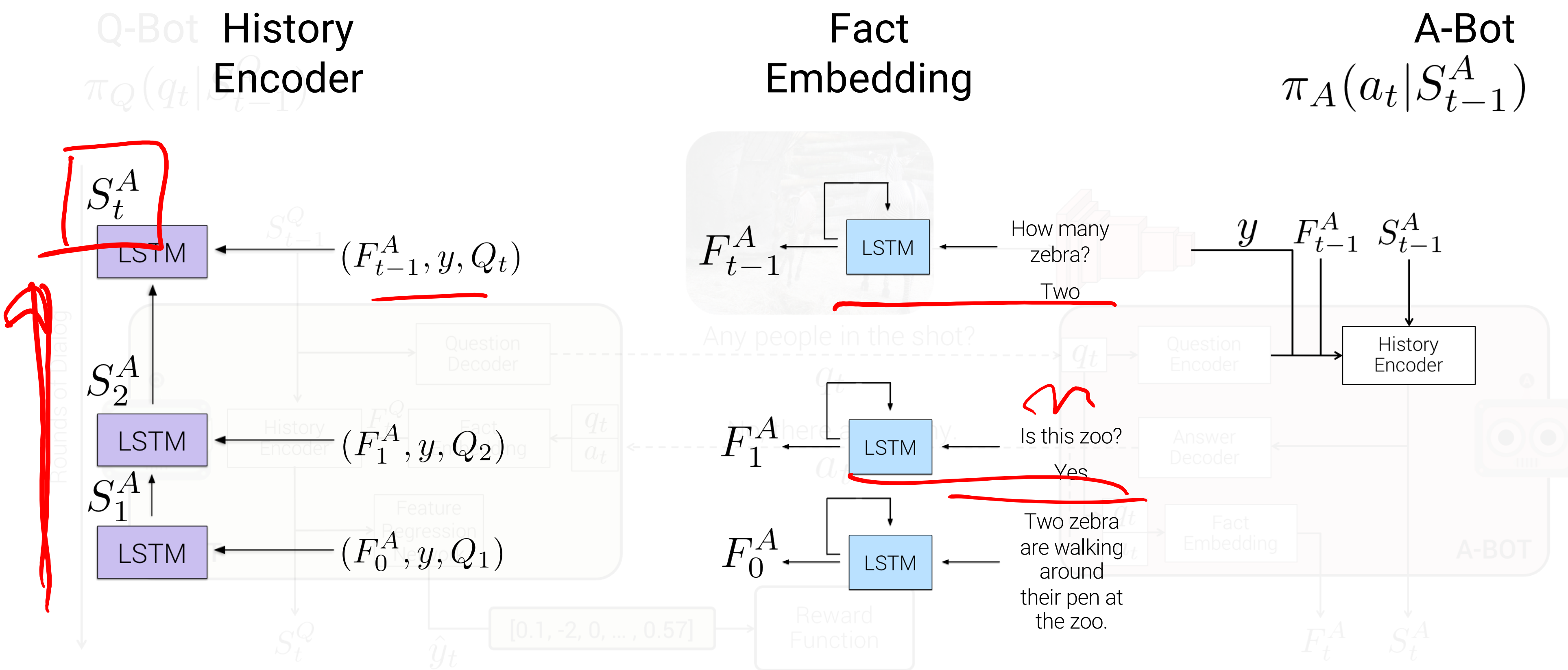
Policy Networks



Policy Networks



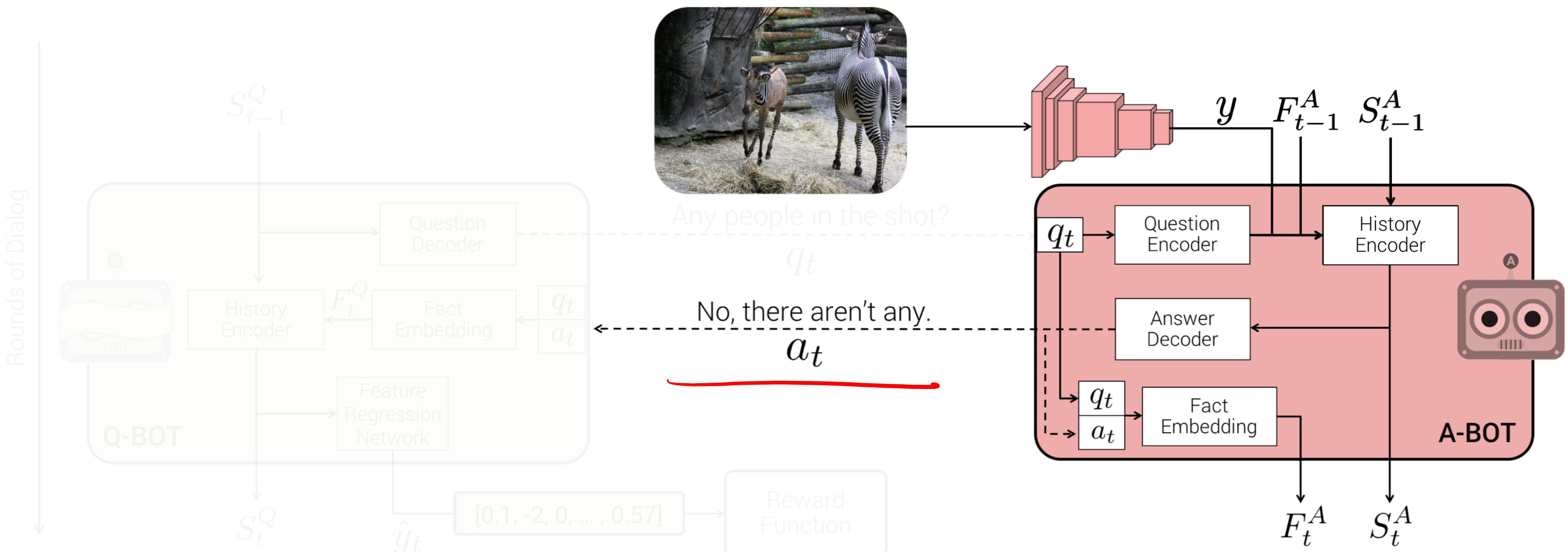
Policy Networks



Policy Networks

Q-Bot
$\pi_Q(q_t|S^Q_{t-1})$

A-Bot
$\pi_A(a_t|S^A_{t-1})$



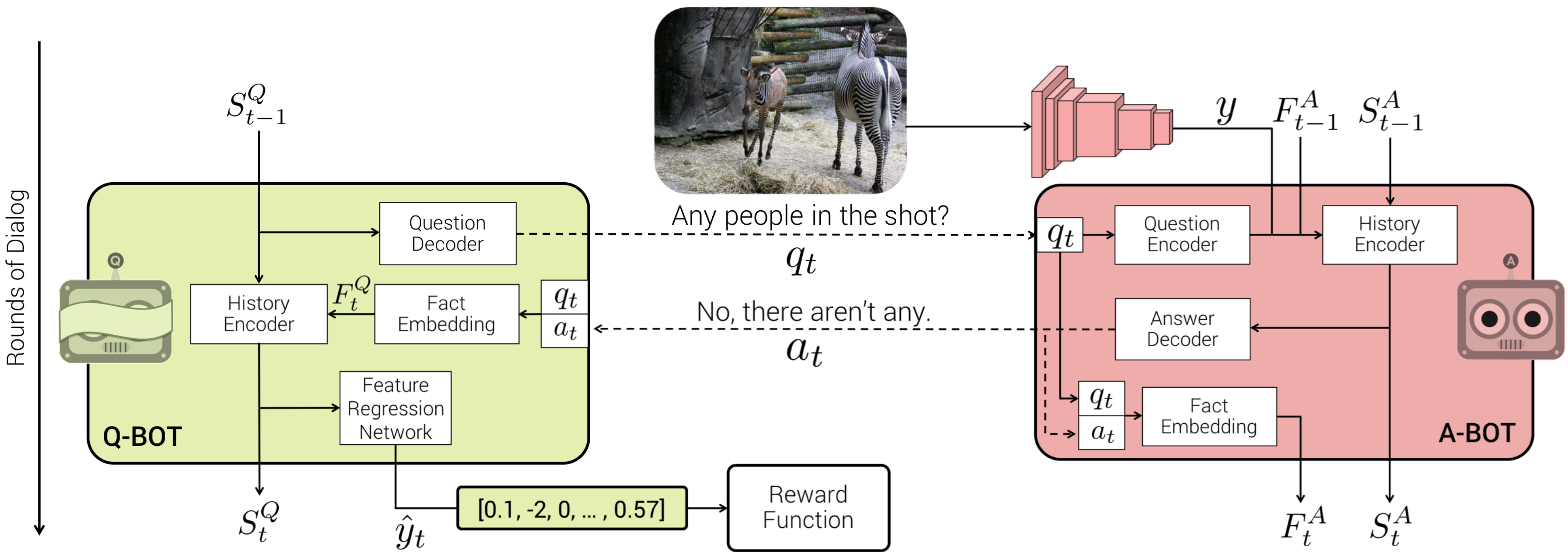
Policy Networks

Q-Bot

$\pi_Q(q_t | S^Q_{t-1})$

A-Bot

$\pi_A(a_t | S^A_{t-1})$



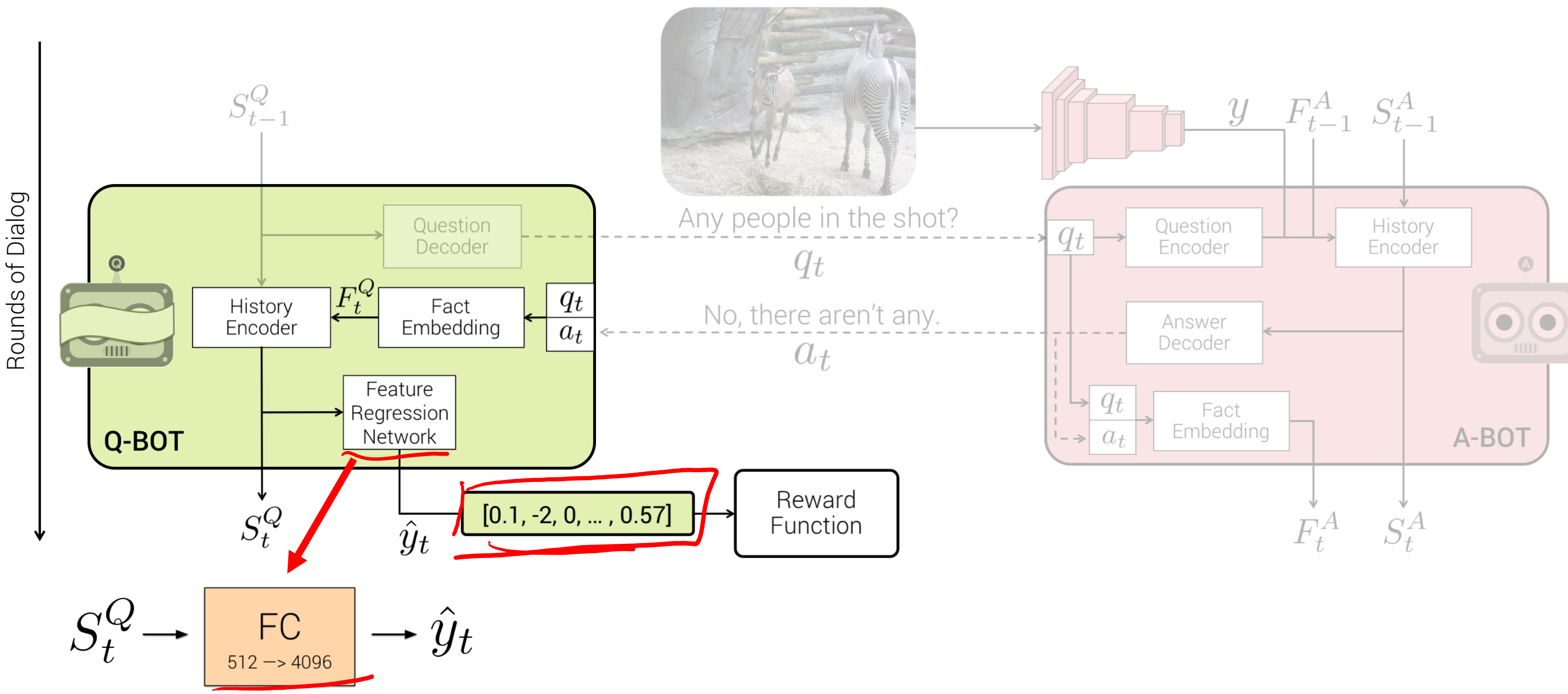
Policy Networks

Q-Bot

$\pi_Q(q_t|S^Q_{t-1})$

A-Bot

$\pi_A(a_t|S^A_{t-1})$



Slide Credit: Abhishek Das

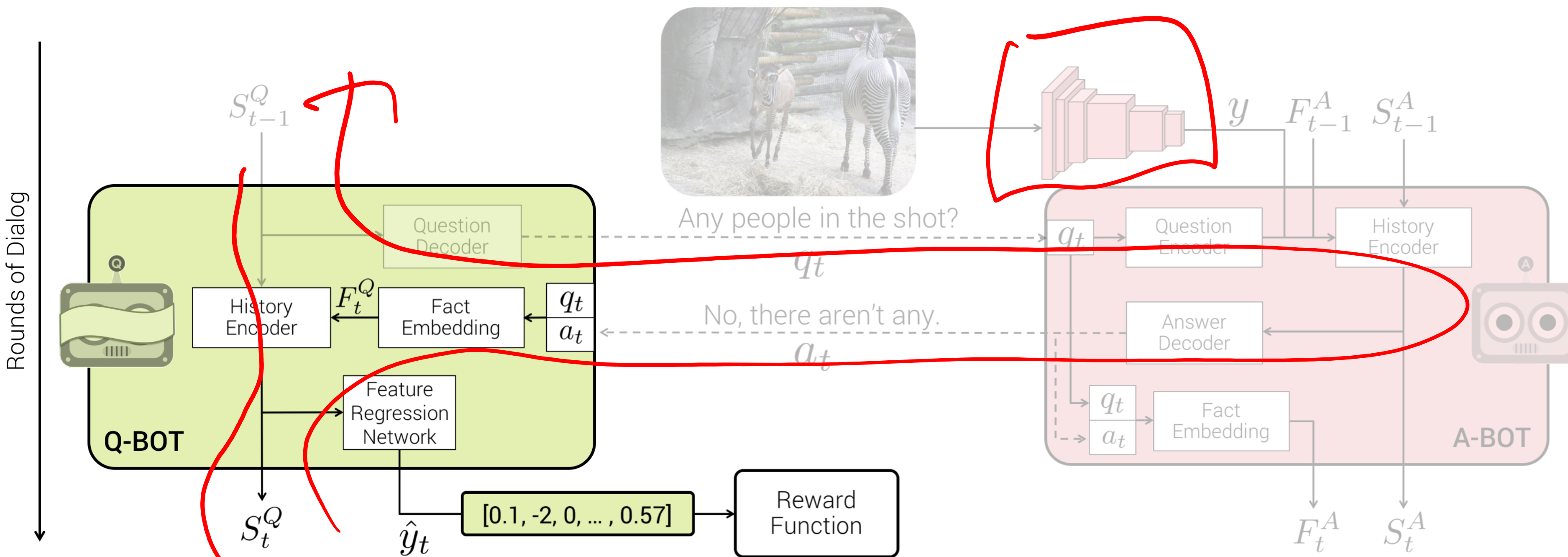
Policy Networks

Q-Bot

$\pi_Q(q_t | S_{t-1}^Q)$

A-Bot

$\pi_A(a_t | S_{t-1}^A)$



$$r_t\Big(\underbrace{s_t^Q}_{\text{state}}, \underbrace{(q_t, a_t, y_t)}_{\text{action}}\Big) = \underbrace{\ell\left(\hat{y}_{t-1}, y^{gt}\right)}_{\text{distance at } t\text{-}1} - \underbrace{\ell\left(\hat{y}_t, y^{gt}\right)}_{\text{distance at } t}$$

Policy Gradients

$$J(\theta_A, \theta_Q) = \mathop{\mathbb{E}}_{\pi_Q, \pi_A} \left[r_t \left(s_t^Q, (q_t, a_t, y_t) \right) \right]$$

REINFORCE Gradients

$$\nabla_{\theta_Q} J = \nabla_{\theta_Q} \left[\mathop{\mathbb{E}}_{\pi_Q, \pi_A} [r_t(\cdot)] \right]$$

$$= \sum_{q_t, a_t} \pi_Q \left(q_t | s_{t-1}^Q \right) \nabla_{\theta_Q} \log \pi_Q \left(q_t | s_{t-1}^Q \right) \pi_A \left(a_t | s_t^A \right) r_t(\cdot)$$

$$= \mathop{\mathbb{E}}_{\pi_Q, \pi_A} \left[r_t(\cdot) \nabla_{\theta_Q} \log \pi_Q \left(q_t | s_{t-1}^Q \right) \right]$$

Turing Test

How old is this man?	He looks like late teens
Does he have short hair or long hair?	Medium
What color is his hair?	Brown
Is he wearing glasses?	No
Does he have facial hair?	No
How is he dressed?	He's white dress shirt with red tie ... school uniform
Is he larger man or is he skinny?	He's skinny
Is this picture taken outside?	Inside
Can you tell if it's daytime?	I'm not sure
Can you tell what room he is standing in?	He's in hallway

A man making the live long and prosper sign from star trek.

Where is man located?	Looks like classroom of some sort
What is man wearing?	Black t shirt and jeans
Are there any other people in picture?	Yes, quite few people in background
What color is table?	Picture is black and white, but it's wood table
Is there anything else on table?	Not that I can see
What are other people doing?	They are standing in front of him, but I don't ...
Are there any windows?	Not that I can see
What color are walls?	I can't see walls
What time of day do you think it is?	I can't tell
What is man wearing?	Black t shirt and jeans

How old is man?	I can't see his face, I don't know
What is he wearing?	shorts and t shirt
What color is his shirt?	white
what color are his pants?	black
what color is his shirt?	blue
What color are his shoes?	Black
What color are his shoes?	Black
What color are his shoes?	Black
What color are his shoes?	Black
What color are his shoes?	Black

What color is skateboard?	It is hard to tell but I think it's black
Is he wearing helmet?	Yes
Is he wearing knee pads?	No, no knee pads
Is it day or night?	It is daytime
Is it sunny?	It is bright out, but I see no shadows, so I can not
Can you see any other people?	There are 2 other people in background
Is it day or night?	It is daytime
Is it sunny?	Yes, it is sunny
What color is skateboard?	It is hard to tell but I think it's black
Is he wearing helmet?	Yes, he is wearing helmet

SL Agents

RL Agents

Plan for Today

- (Deep) Reinforcement Learning
  - Policy gradients

- Closing the loop

So what *is* Deep (Machine) Learning?

- A few different ideas:

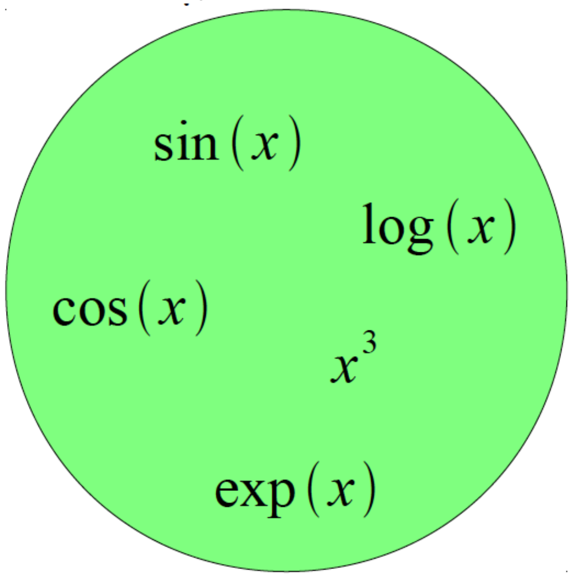
- (Hierarchical) Compositionality
  - Cascade of non-linear transformations
  - Multiple layers of representations

- End-to-End Learning
  - Learning (goal-driven) representations
  - Learning to feature extraction

- Distributed Representations
  - No single neuron "encodes" everything
  - Groups of neurons work together

Building A Complicated Function

Given a library of simple functions

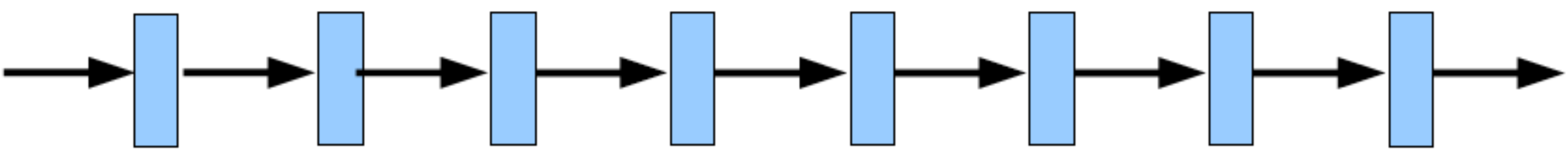


Compose into a complicate function

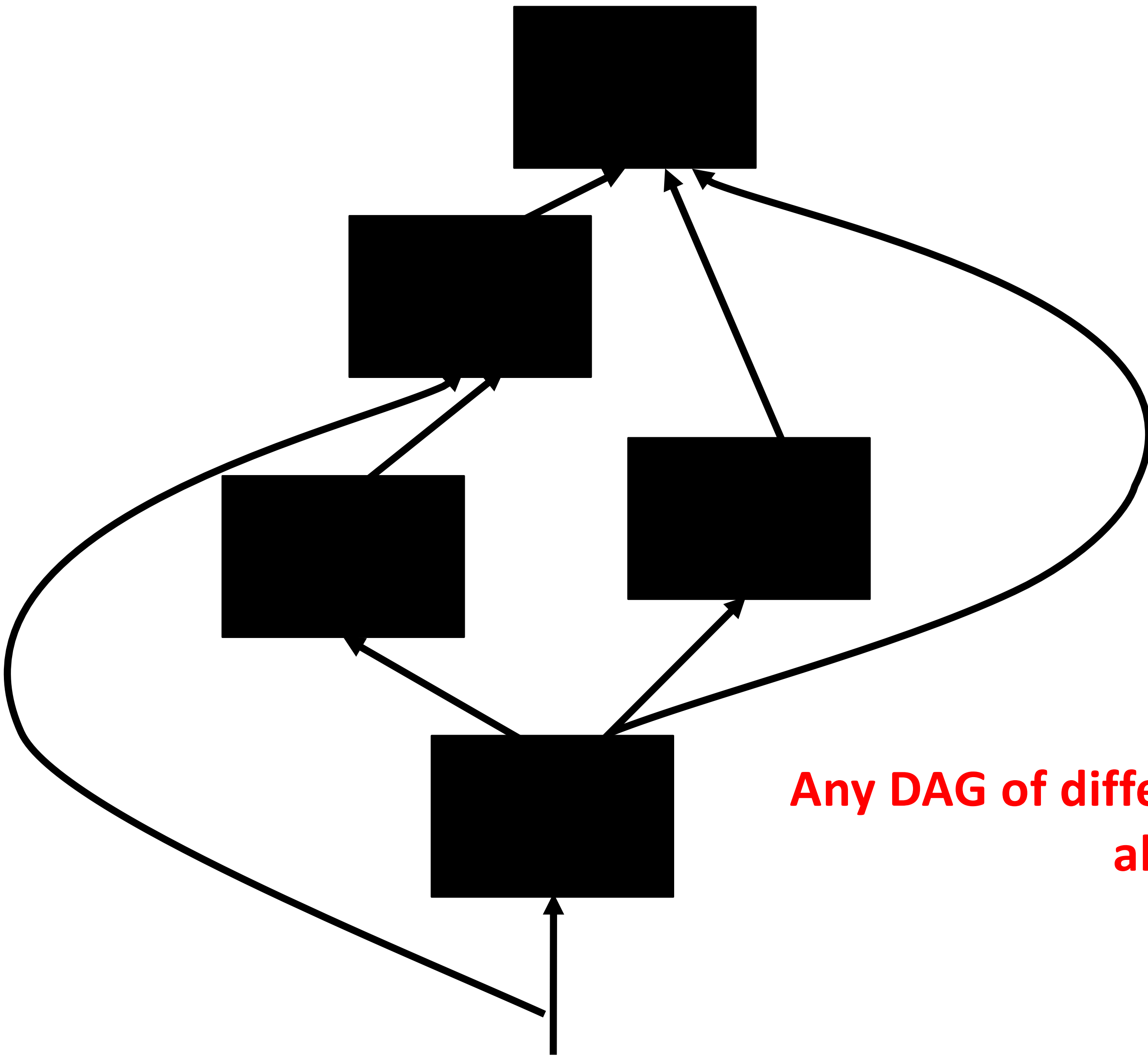
Idea 2: Compositions

- Deep Learning
- Grammar models
- Scattering transforms…

$$f(x) = g_1(g_2(\ldots(g_n(x)\ldots))$$



Differentiable Computation Graph



Any DAG of differentialble modules is allowed!

So what *is* Deep (Machine) Learning?

- A few different ideas:

- (Hierarchical) Compositionality
  - Cascade of non-linear transformations
  - Multiple layers of representations

- End-to-End Learning
  - Learning (goal-driven) representations
  - Learning to feature extraction

- Distributed Representations
  - No single neuron "encodes" everything
  - Groups of neurons work together

"Shallow" vs Deep Learning

- "Shallow" models

hand-crafted Feature Extractor → "Simple" Trainable Classifier

fixed — learned

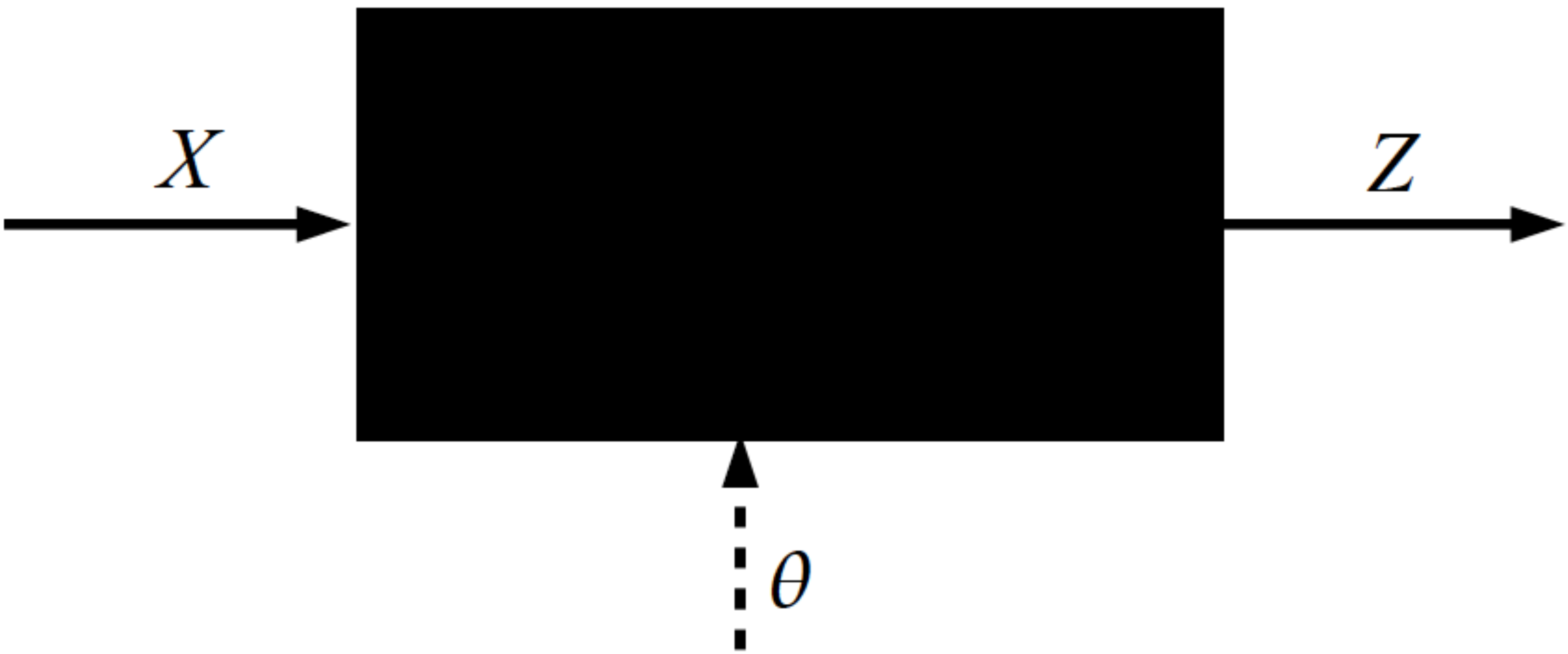
- Deep models

Trainable Feature-Transform / Classifier → ... → Trainable Feature-Transform / Classifier → Trainable Feature-Transform / Classifier →

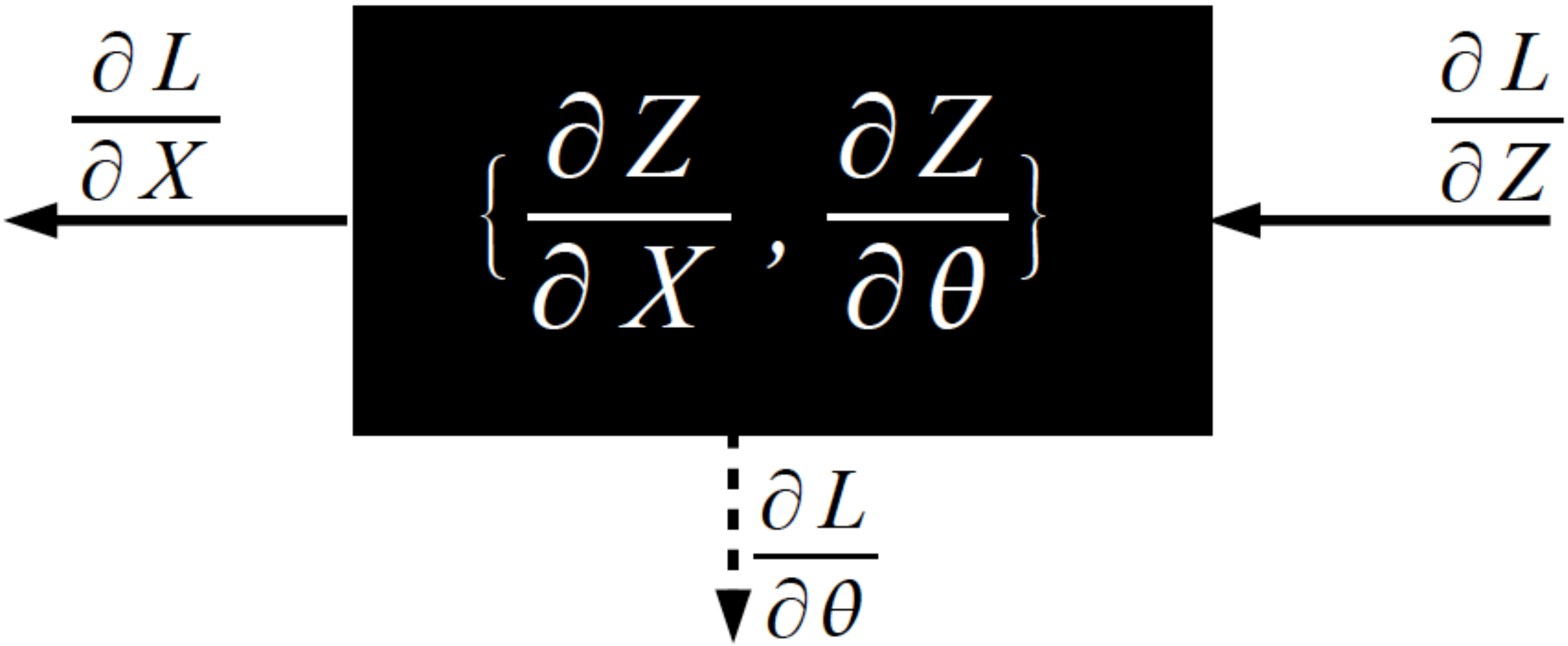
Learned Internal Representations

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Key Computation: Forward-Prop



Key Computation: Back-Prop



So what *is* Deep (Machine) Learning?

- A few different ideas:

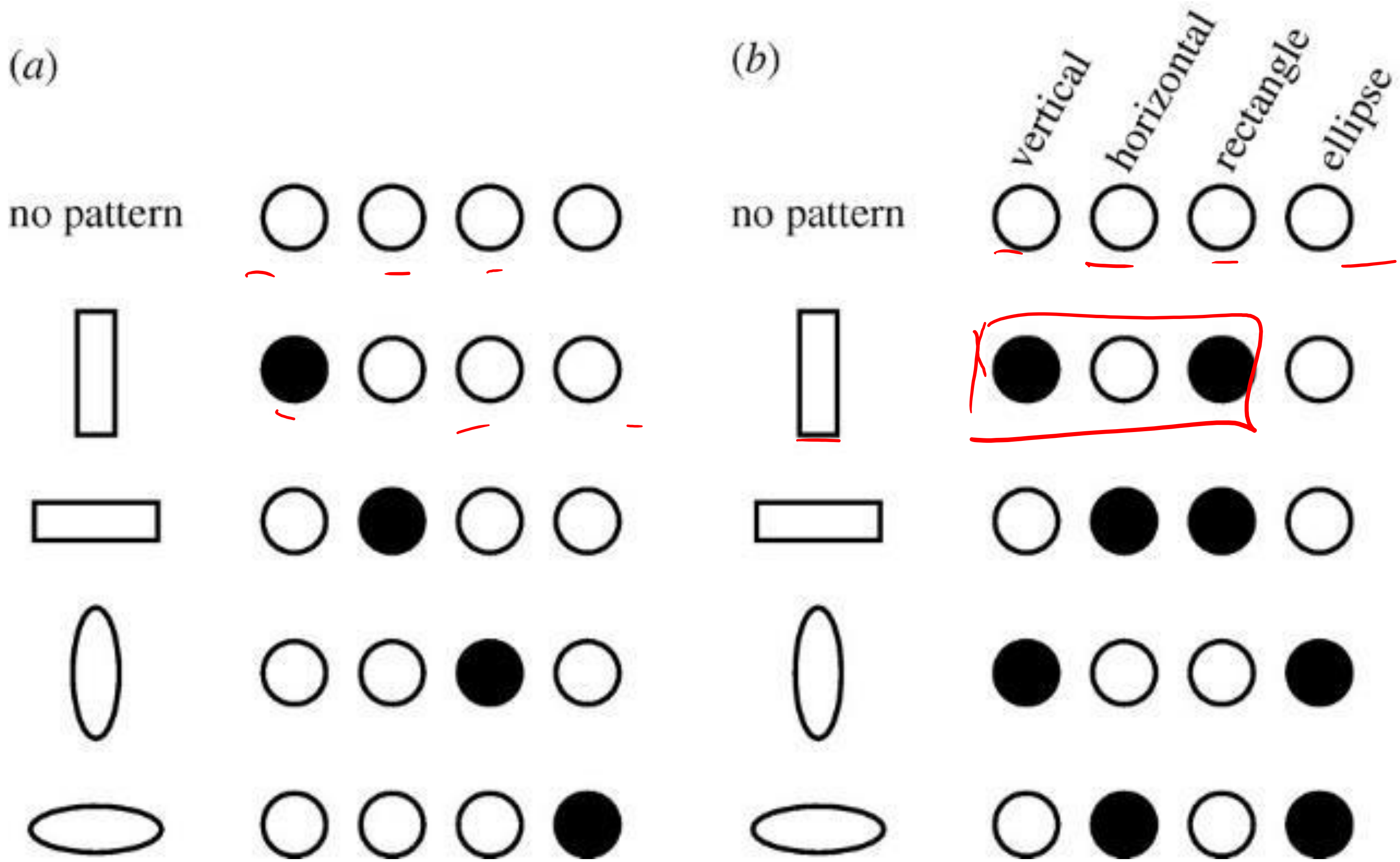
- (Hierarchical) Compositionality
  - Cascade of non-linear transformations
  - Multiple layers of representations

- End-to-End Learning
  - Learning (goal-driven) representations
  - Learning to feature extraction

- Distributed Representations
  - No single neuron “encodes” everything
  - Groups of neurons work together

Distributed Representations Toy Example

- Can we interpret each dimension?



Power of distributed representations!

Local ● ● ○ ● = VR + HR + HE = ?

Distributed ● ● ○ ● = V + H + E $\approx$ ○

What is this class about?

What is this class about?

- Introduction to Deep Learning

- Goal:
  - After finishing this class, you should be ready to get started on your first DL research project.
    - CNNs
    - RNNs
    - Deep Reinforcement Learning
    - Generative Models (VAEs, GANs)

- Target Audience:
  - Senior undergrads, MS-ML, and new PhD students

What did we learn?

- Background & Basics
  - Neural Networks, Backprop, Optimization (SGD)
- Module 1: Convolutional Neural Networks (CNNs)
  - Architectures, Pre-training, Fine-tuning
  - Visualizations, Fooling CNSS, Adversarial examples
  - Different tasks: detection CNNs, segmentation CNNs
- Module 2: Recurrent Neural Networks (RNNs)
  - Difficulty of learning; “Vanilla” RNNs, LSTMs, GRU
  - RNNs for Sequence-to-Sequence (machine translation & image captioning, VQA, Visual Dialog)
- Module 3: Deep Reinforcement Learning
  - Overview, policy gradients
  - Optimizing Neural Sequence Models for goal-driven rewards
- Module 4: Deep Structured Prediction
  - Crash course on Bayes Nets, Variational Inference
  - Variational Auto Encoders (VAEs)
- Module 5: Advanced Topics
  - GANs, Adversarial Learning

Arxiv Fire Hose

PhD Student

Deep Learning papers

Cornell University Library
arXiv.org

Feedback

CIOS Help

Inbox - GT November 26, 2018 at 1:37 AM

CH

GT Course Instructor Opinion Survey (CIOS) now open

To: Batra, Dhruv,

Reply-To: cioshelp@gatech.edu

Siri found new contact info in this email: Help Cios evaluations@smartevals.com add to Contacts...

Dear Dhruv,

Good morning. The Course/Instructor Opinion Survey (CIOS) is now available for the following courses. Your courses, their survey start and end dates, and your current response rate are shown in the table below.

Eval	Course Prefix	Course Number	Sec	Type	Name	Begin	End	Not Resp.	Resp.	Tot.
Preview	CS	4803	DL	A	Special Topics	11-26	12-16	20	0	20
Preview	CS	7643	A	A	Deep Learning	11-26	12-16	84	0	84

Students have received an announcement indicating that surveys have begun, and they will continue to receive periodic reminder emails with all of the necessary information to complete the survey. However, you can ALSO set up additional reminders within the system that would come from you. Simply login at the link below, click the "Not Set" button near the left of the table, and follow the directions to set up auto-email reminders for your all of your courses.

Reports with your results will be available 5 days after full semester grades are due and you will receive an email with report access information at that time.information at that time.
If you would like to view your response rates at any time, you can log in with your GT account here: http://b.gatech.edu/cios

If you have any problems with the survey system, please email cioshelp@gatech.edu.

Thanks!