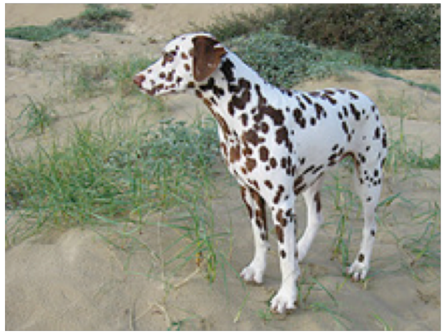
Embodied cognition

Recognition today

- Large dataset of isolated, labeled images
- Where do the images come from?
- What do we do with the labels?

{( , 1), ( , 1), ( , 0) , ... }

Embodied agents

- Agents that perceive and act in the world
- Input images come from the physical world
- “Recognition” is used to act

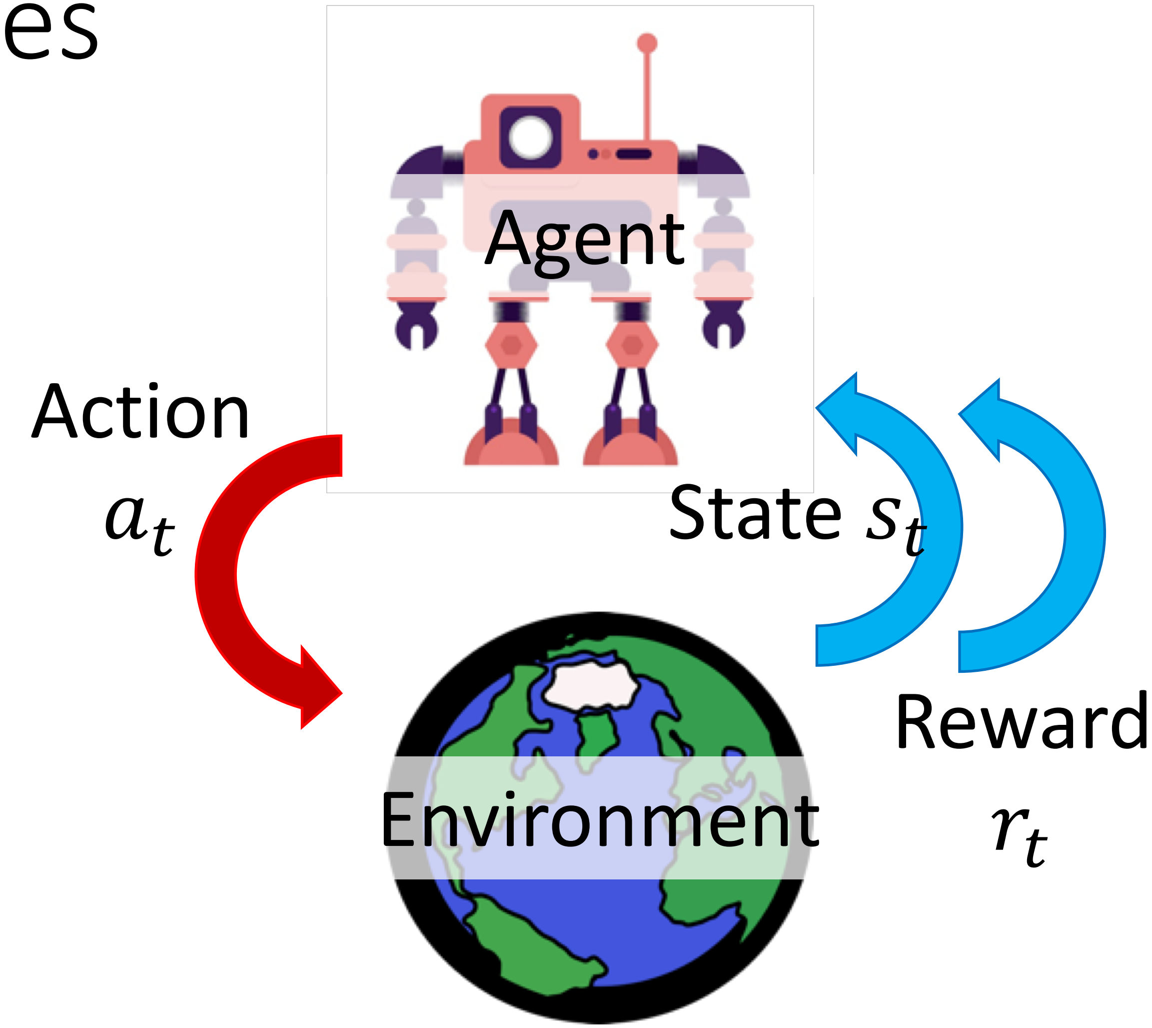
Embodied cognition

- Perception now depends on action: state of agent or state of the world
  - No more collections of isolated images
- Feedback to agent comes through consequences of actions
  - No well-defined labels
- Three problems:
  - Learning to act to achieve goals, with perception in the loop
  - Self-supervised learning: learning perception from action
  - Active perception: learning to act for perception

Learning to act to achieve goals

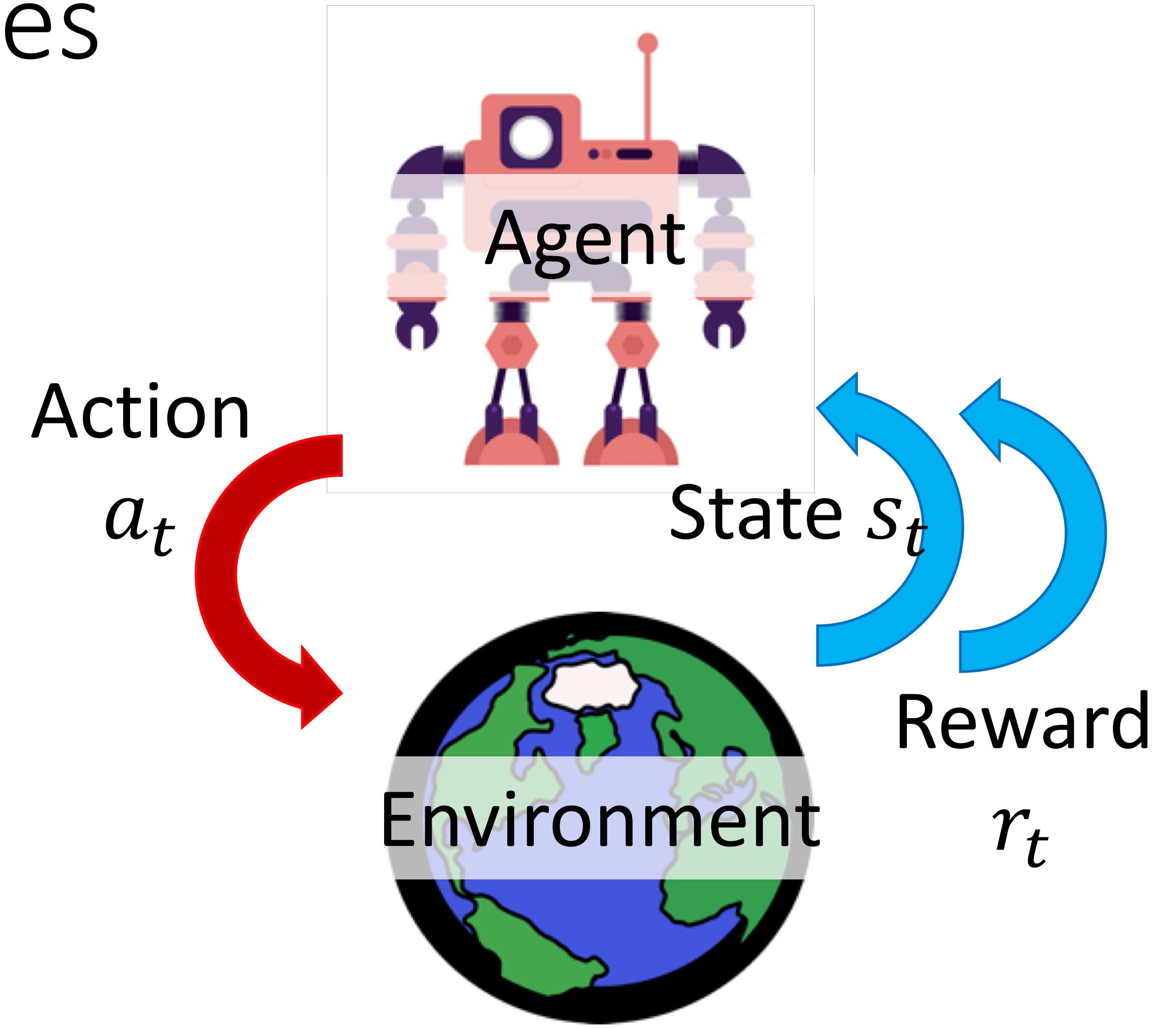
Markov decision processes

- Agent acting in an environment
- Environment has **states** that are known to agent
- Agent takes **actions**
- Action + random stuff leads to a **state transition**
- Environment gives agent **reward**
- Agent's goal: maximize **return** = total reward over time
- Agent's **policy:** mapping from states to actions



Markov decision processes

- Set of states S
- Set of actions A
- Transition function
  $T(s, a, s') = P(s_{t+1} = s' | s_t = s, a_t = a)$
- Reward function
  $$R(s) = \mathbf{E}(r_t | s_t = s)$$
- Return (with discount)
  $$\sum_t \gamma^t r_t \,, \qquad \gamma \in [0,1)$$
- Must learn **policy:** $\pi: S \times A \to [0,1]$



Why is this hard?

- Stochasticity
  - T and R are non-deterministic
- Unknown dynamics
  - T and R are unknown: don’t know which action will lead to which state
- Partial feedback
  - Only get to see consequences of action taken
- Long term consequences
  - Actions can influence rewards several time steps later

Example: self-driving cars

- Stochasticity
  - Pressing gas pedal does not always cause acceleration
  - Weather
- Unknown dynamics
  - Do not know where roads lead
  - Do not know what other cars might do
- Long term consequences
  - E.g., may only realize later that took a wrong turn earlier

Model-based control

- MDPs:
  - States S, actions A, transitions $T(s, a, s')$, reward function $R(s)$
- What is known, what is unknown?
- Case 1: S, A, T, R known, only policy unknown
  - T, R: *model dynamics*
  - Example: self-driving in an empty city with a map
- Problem becomes one of *planning* or *optimal control*

Model-based control

- If states are discrete
  - Algorithms from AI : *search*
- If states are continuous
  - Algorithms from control theory : *optimization*
- Why is this hard?
  - Large space of states, most not good
  - Stochasticity, e.g., wheels might slip
- What to do when we don’t know model dynamics?
  - Learn dynamics! (*System identification* in control theory)
  - Reinforcement learning

Markov decision processes

- Value function $V^{\pi}(s)$:
  - What return will I get if I start from state s and follow policy π thereafter?
  - $V^{*}(s) = \max_{\pi} V^{\pi}(s) = V^{\pi^{*}}(s)$ where $\pi^{*}$is the optimal policy
- Action-value function $Q^{\pi}(s, a)$:
  - What return will I get if I start from state s, take action a and follow policy π thereafter
  - $Q^{*}(s, a) = \max_{\pi} Q^{\pi}(s, a) = Q^{\pi^{*}}(s, a)$

Bellman equations

- $Q^{\pi}(s,a) = R(s\ ) + \gamma \sum_{s'} T(s,a,s')\ \sum_{a'} \pi(s',a') Q^{\pi}\ (s',a')$
- $V^{\pi}(s) = R(s) + \gamma \sum_{a} \pi(s,a)\ \sum_{s'} T(s,a,s') V^{\pi}(s')$
- $Q^{*}(s,a) = R(s) + \gamma \sum_{s'} T(s,a,s') \max_{a'} Q^{*}(s',a')$
- $V^{*}(s) = R(s) + \gamma \max_{a} \sum_{s'} T(s,a,s') V^{*}(s')$

Basic RL algorithms I: Policy iteration

- Start with a random policy
- Repeat:
    1. Policy evaluation: Evaluate Q^{π}
    2. Policy improvement: update policy
        - $\pi(s,a) = \mathbb{I}(a = \arg\max_a Q^{\pi}(s,a))$
        - But deterministic policies may not explore all states
        - ϵ-greedy: with small probability choose random action instead

Policy evaluation: (Q-) value iteration

- If we know T and R, policy evaluation is "easy"
- Convert bellman equation into recurrence
- $Q^{\pi}(s, a) = R(s\,) + \gamma \sum_{s'} T(s, a, s')\ \sum_{a'} \pi(s', a') Q^{\pi}\ (s', a')$
- $Q_t^{\pi}(s, a) \leftarrow R(s\,) + \gamma \sum_{s'} T(s, a, s')\ \sum_{a'} \pi(s', a') Q_{t-1}^{\pi}\ (s', a')$
- Can be done "in the head"

Policy evaluation: sarsa

- If we don't know T and R, have to deal with samples and try out in the world
- Try current policy to get sequence $(\ldots, s_t, a_t, r_t, s_{t+1}, a_{t+1}, \ldots)$
- Convert Bellman equation into an update
- Bellman equation:
  $Q^\pi(s,a) = R(s\,) + \gamma \mathbb{E}_{s' \sim T(s,a,\cdot)}\, \mathbb{E}_{a' \sim \pi(s',\cdot)} Q^\pi(s',a')$
- Sample update:
  $$Q_t^\pi(s_t, a_t) \leftarrow (1-\alpha) Q_{t-1}^\pi(s_t, a_t) + \alpha(r_t + \gamma Q_{t-1}^\pi(s_{t+1}, a_{t+1}))$$

Basic RL algorithms II: Q-learning

- Policy iteration can be slow
- Can update policy without waiting for full evaluation
- Alternative: learn Q* directly
- Act using some random policy, but use observations to find Q*

Q-learning

- $Q^*(s,a) = R(s) + \gamma \sum_{s'} T(s,a,s') \max_{a'} Q^*(s',a')$
- Every iteration, agent is in state s, takes action a, receives reward r and reaches state s'
- $Q^t(s,a) \leftarrow (1-\alpha)Q^{t-1}(s,a) + \ \alpha(r + \gamma \max_{a'} Q^{t-1}(s',a'))$
- This converges to Q*
- How do we get the policy from Q*?
- $\pi^* \ (s) = \max_{a} Q^*(s,a)$

Basic RL algorithms III: Policy gradient

- How about directly optimizing the policy instead of going through Q?
- Pipeline:
  - Use π_θ to choose action
  - Get reward
  - Take step along gradient to produce better policy
- Problem: choosing action is a non-differentiable function of the policy
- How do we get the gradient?

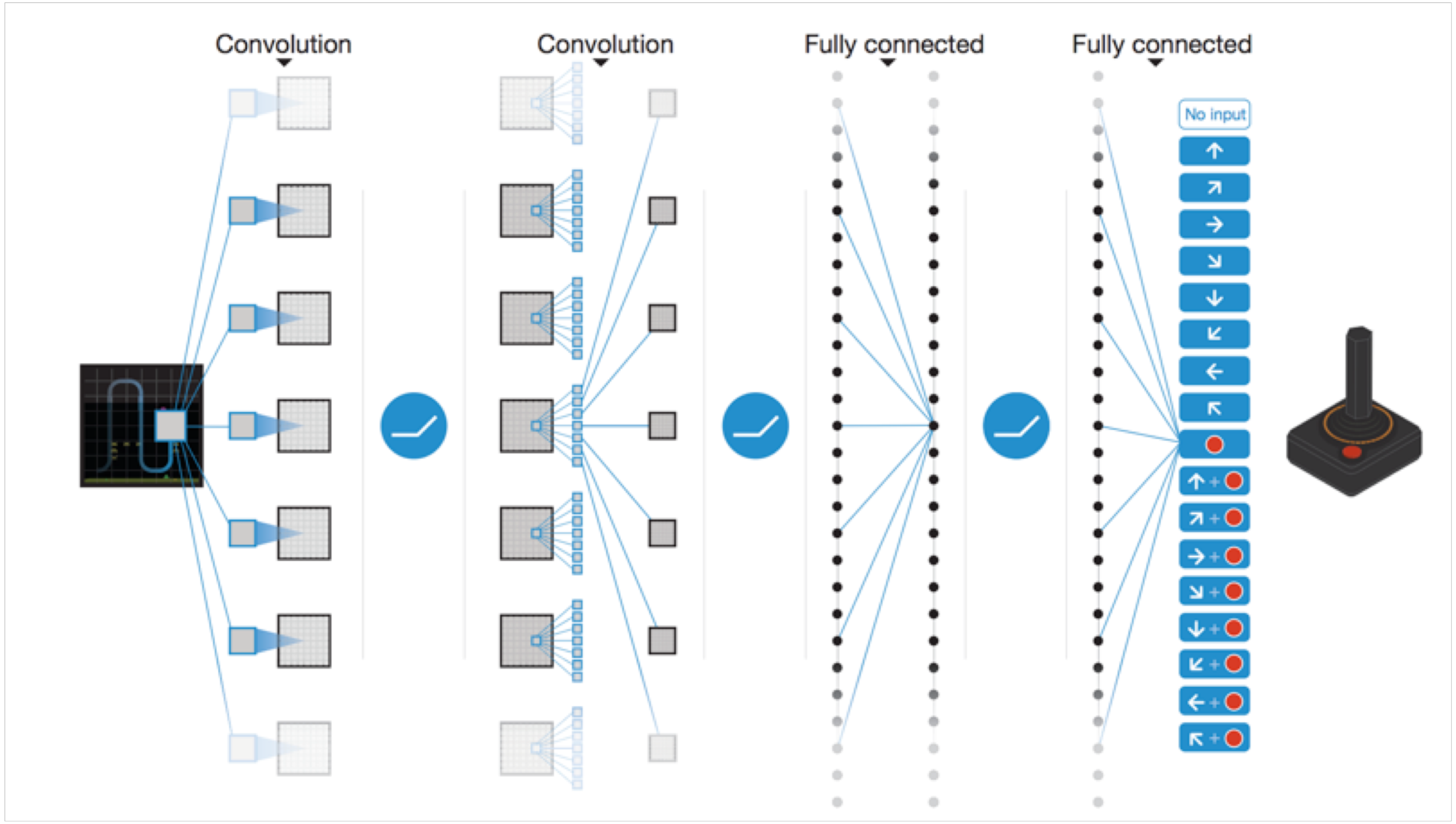
REINFORCE

- Suppose the agent takes a sequence τ of actions a_t and goes through states s_t under policy π
- Probability of sequence = $\pi_\theta(\tau) = \prod_t \pi_\theta(s_t, a_t)$
- Total return = $r(\tau) = \sum_t \gamma^t r_t$
- $J(\theta) = E_\pi(r) = \sum_\tau r(\tau)\pi_\theta(\tau)$
- A single run gives a sample τ and an estimate of $J(\theta)$
- $\nabla_\theta J(\theta) = \sum_\tau r(\tau)\ \nabla_\theta \pi_\theta(\tau)$
- How do we compute $\nabla_\theta J(\theta)$ with samples?

REINFORCE

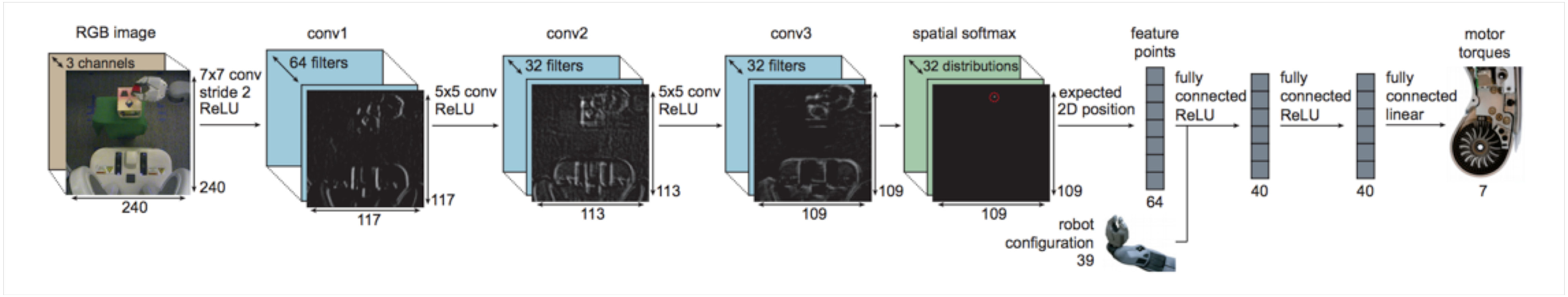
- $\nabla_\theta J(\theta) = \sum_\tau r(\tau)\ \nabla_\theta \pi_\theta(\tau)$
- Identity: $\nabla_\theta\ \pi_\theta(\tau)\ = \pi_\theta(\tau)\frac{\nabla_\theta\ \pi_\theta(\tau)}{\pi_\theta(\tau)} = \pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau)$
- Thus $\nabla_\theta\ J(\theta) = \sum_\tau r(\tau)\, \pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) = \mathrm{E}_\pi[r(\tau)\nabla_\theta \log \pi_\theta(\tau)]$
- Thus to get gradient, simply compute $r(\tau)\nabla_\theta \log \pi_\theta(\tau)$ for every run and average

Learning to play Atari games



Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." Nature518.7540 (2015): 529-533.

Learning to do robotic tasks

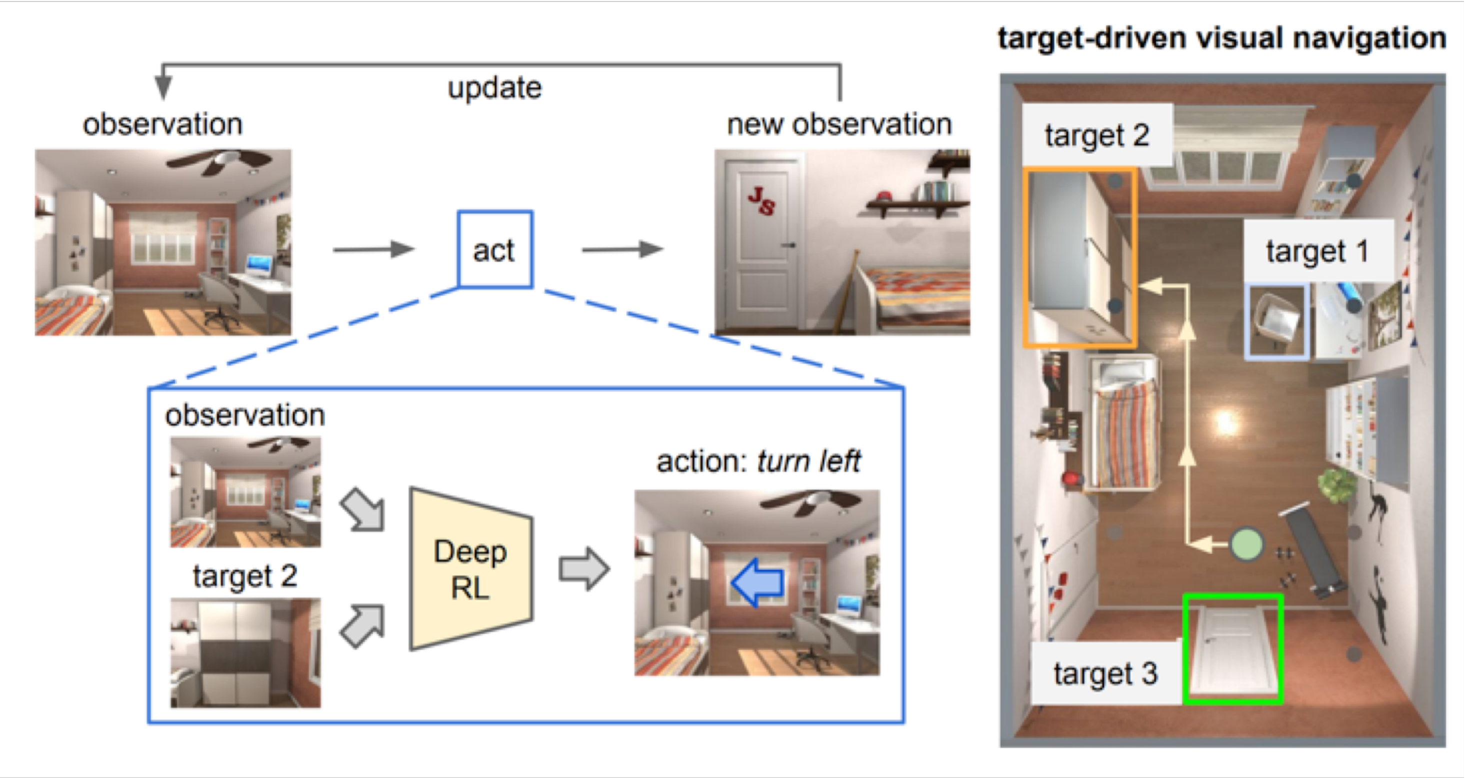


End-to-End Training of Deep Visuomotor Policies. S. Levine, C. Finn, T. Darrell, P. Abbeel. In *JMLR,* 2016

Reinforcement learning and generalization

- RL learns a *policy*
- Policies are specific to *goals*
- Reinforcement learning = learning to play a particular game
- Separate model for each game
    - “Close this bottle”
    - “Peel this banana”
- General model?

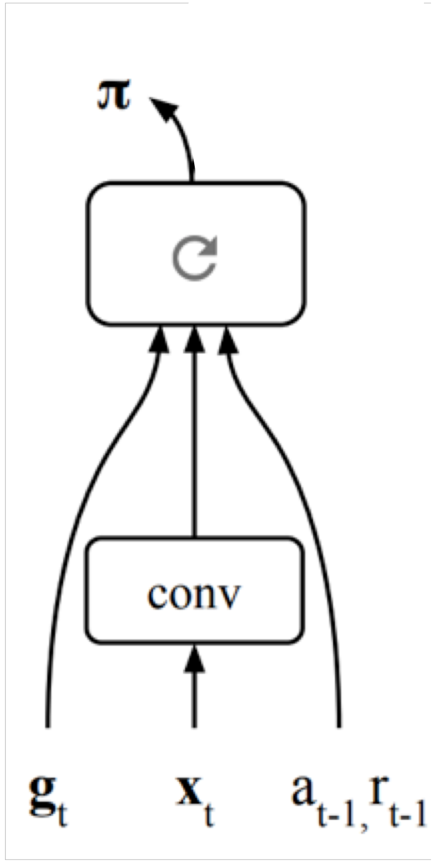
Conditioning on input and target



Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning. Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-fei, A. Farhadi. In *ICRA,* 2017

Partial observations

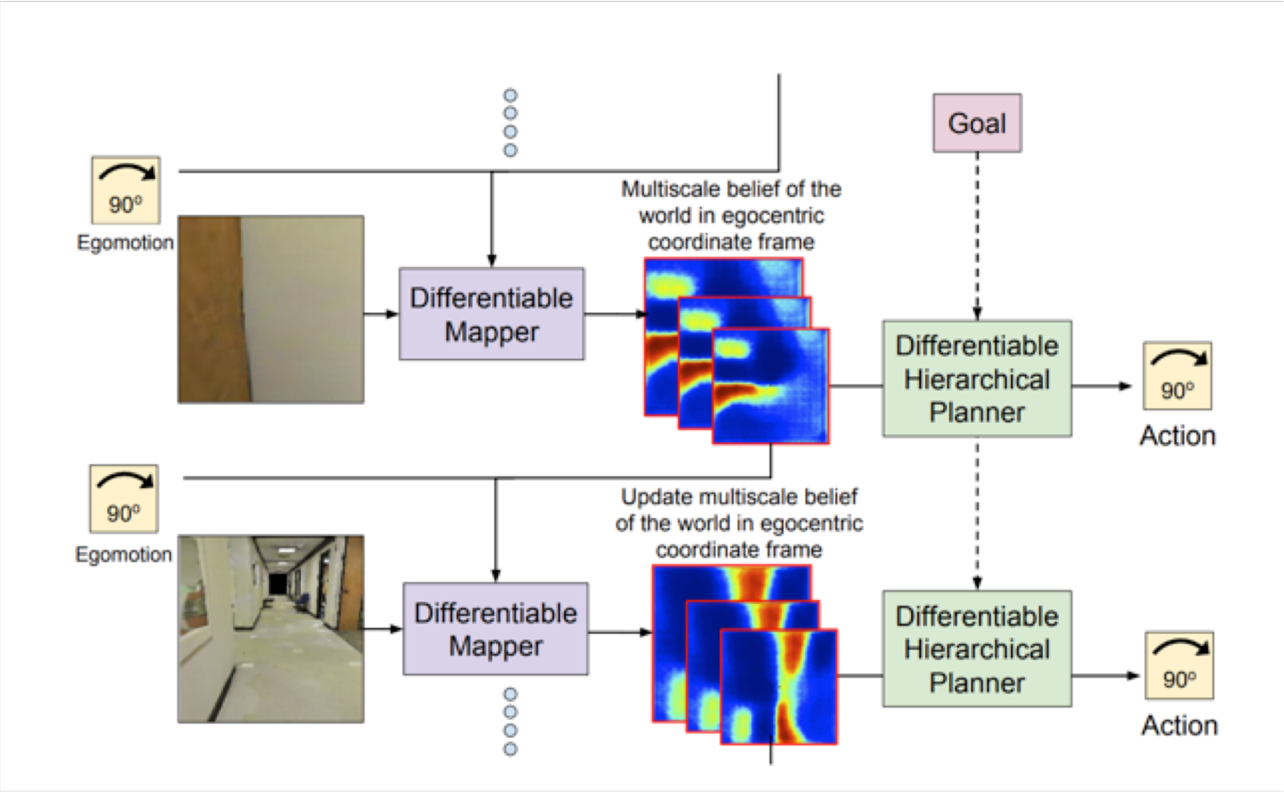
- Assumption: image conveys the entire state
- True for Atari, not true for real worls
- Simple idea: let neural network maintain state internally



Mirowski, Piotr, et al. "Learning to Navigate in Cities Without a Map." *arXiv preprint arXiv:1804.00168* (2018).

Incorporating domain knowledge

- Should we rely on learning entirely?
- E.g. for navigation, maintain a map of the environment and of agent's state within it
- Classical solution: SLAM (Simultaneous localization and mapping)



Gupta, Saurabh, et al. "Cognitive mapping and planning for visual navigation." *arXiv preprint arXiv:1702.03920* 3 (2017).

Reflex agents

- Reflex agents
  - Map states to actions
  - Are feedforward
  - Cannot explore / back-track *unless state records history*
  - Have to be trained on each environment

Planning

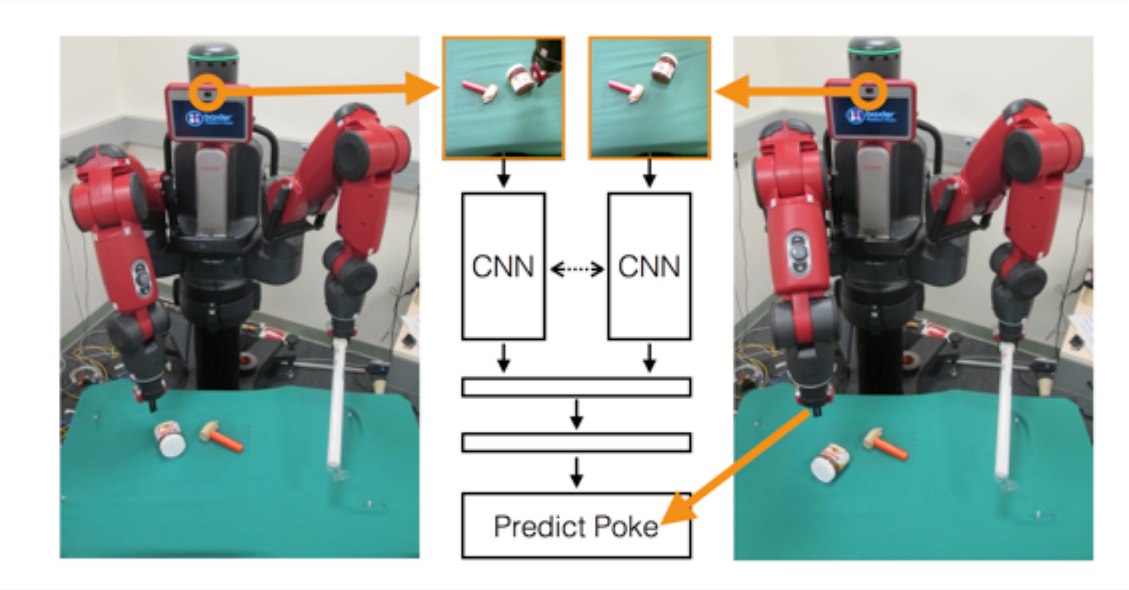
- If we can predict consequences of actions, we can plan

Model-based RL

- *Learn* a ”forward model” of how states evolve
  - E.g., $T(s, a, s') = P(s_{t+1} = s' | s_t = s, a_t = a)$
- Then we can optimize π offline for reaching the goal
- Open-loop planning:
  - Take the first action, see where we land
  - Re-optimize

Inverse model

- Predicting image pixels is hard
- Can also have inverse model: given s and s’, what action takes me from s to s’?
- Use inverse model to find an initial action, perform action, then re-evaluate.



Agrawal, Pulkit, et al. "Learning to poke by poking: Experiential learning of intuitive physics." *Advances in Neural Information Processing Systems*. 2016.