Carnegie Mellon University
Language Technologies Institute

Intro to Reinforcement Learning Part I

11-777 Multimodal Machine Learning Fall 2021

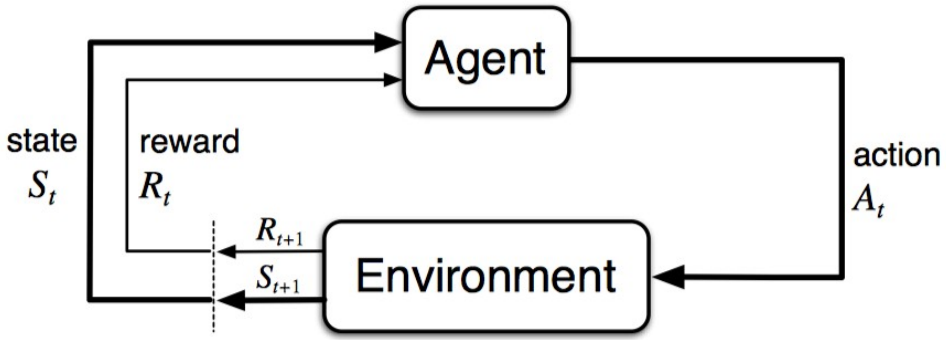
Amir Zadeh
Slides from Paul Liang

Used Materials

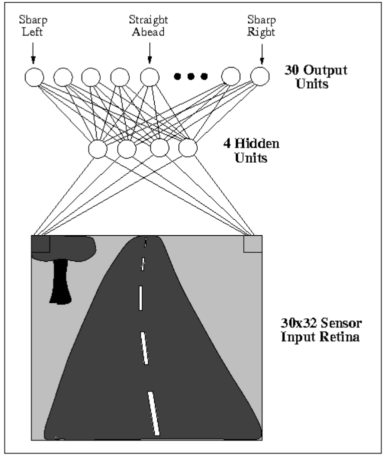
Acknowledgement: Some of the material and slides for this lecture were borrowed from the Deep RL Bootcamp at UC Berkeley organized by Pieter Abbeel, Yan Duan, Xi Chen, and Andrej Karpathy, as well as Katerina Fragkiadaki and Ruslan Salakhutdinov's 10-703 course at CMU, who in turn borrowed much from Rich Sutton's class and David Silver's class on Reinforcement Learning.

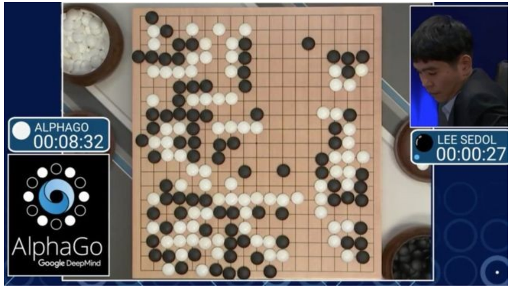
Contents

- Introduction to RL
- Markov Decision Processes (MDPs)
- Solving known MDPs using value and policy iteration
- Solving unknown MDPs using function approximation and Q-learning



Reinforcement Learning



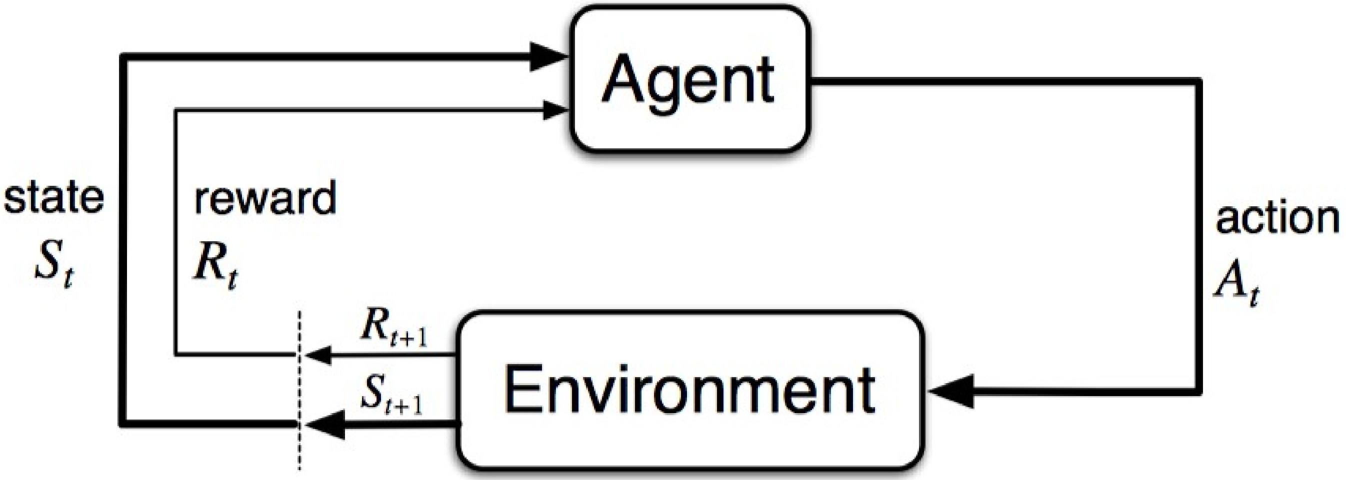


ALVINN, 1989

AlphaGo, 2016

DQN, 2015

Reinforcement Learning



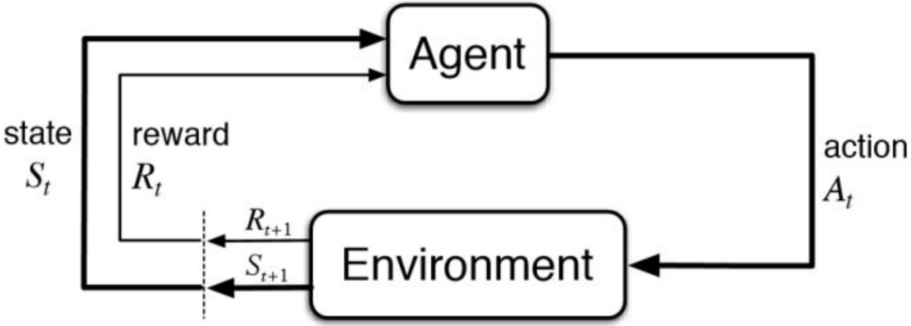
Trajectory

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots$$

Markov Decision Process (MDPs)

An MDP is defined by:

- Set of states S
- Set of actions A
- Transition function $P(s' \mid s, a)$
- Reward function $R(s, a, s')$
- Start state s_0
- Discount factor γ
- Horizon H



Trajectory

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots$$

Markov assumption + Fully observable

A state should summarize all past information and have the **Markov property.**

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

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

We should be able to throw away the history once state is known

- If some information is only partially observable: Partially Observable MDP (POMDP)

Return

We aim to maximize *total discounted reward*:

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Discount factor

γ close to 0 leads to "myopic" evaluation

γ close to 1 leads to "far-sighted" evaluation

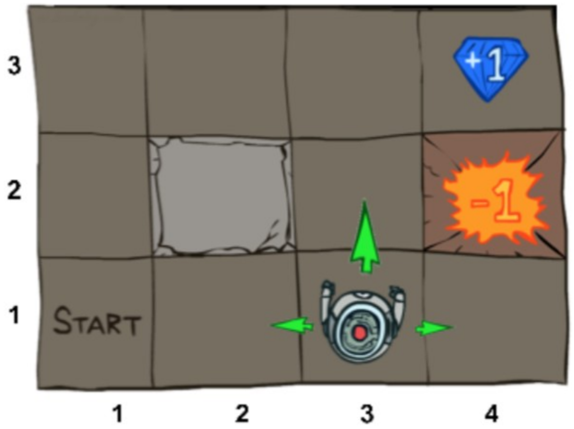
Policy

Definition: A policy is a distribution over actions given states

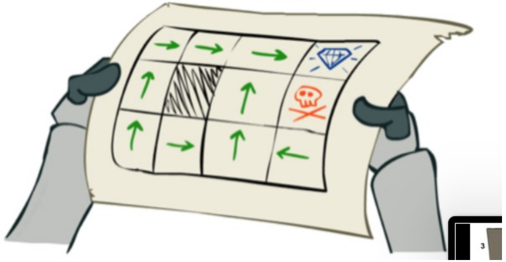
$$\pi(a \mid s) = \mathbf{Pr}(A_t = a \mid S_t = s), \forall t$$

- A policy fully defines the behavior of an agent
- The policy is stationary (time-independent)
- During learning, the agent changes its policy as a result of experience

Special case: deterministic policies



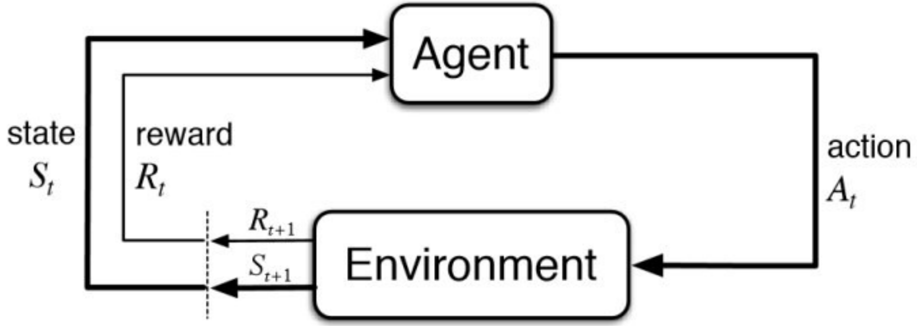
π:



Learn the optimal policy to maximize return

An MDP is defined by:

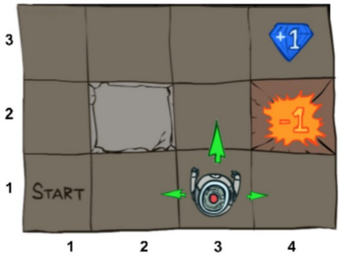
- Set of states S
- Set of actions A
- Transition function $P(s' \mid s, a)$
- Reward function $R(s, a, s')$
- Start state s_0
- Discount factor γ
- Horizon H



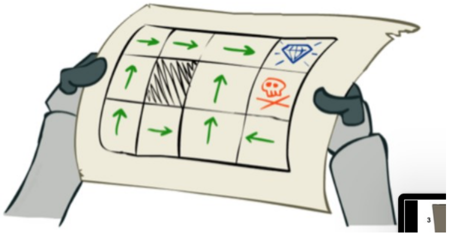
Return:

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Goal: $$\arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R_t | \pi\right]$$



π:



Reinforcement Learning vs Supervised Learning

Reinforcement Learning

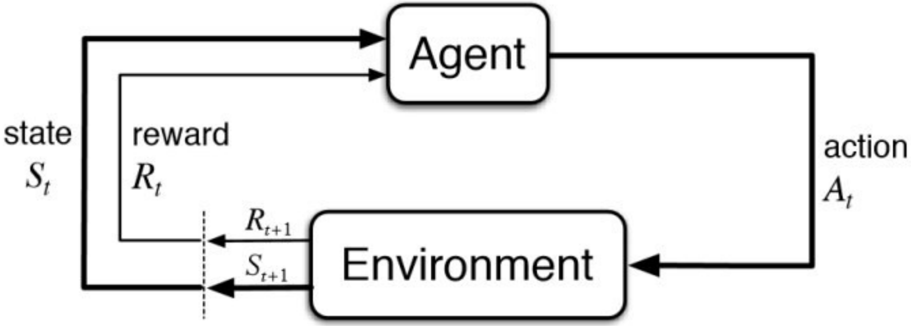
- Sequential decision making
- Maximize cumulative reward
- Sparse rewards
- Environment maybe unknown

Supervised Learning

- One-step decision making
- Maximize immediate reward
- Dense supervision
- Environment always known

Intersection between RL and supervised learning

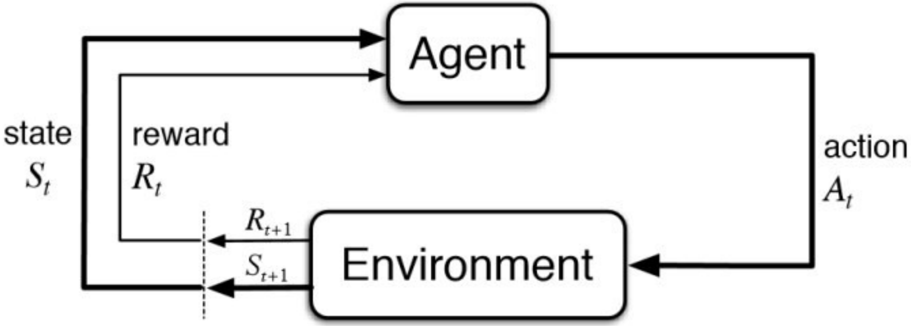
Imitation learning!



Learn the optimal policy to maximize return

An MDP is defined by:

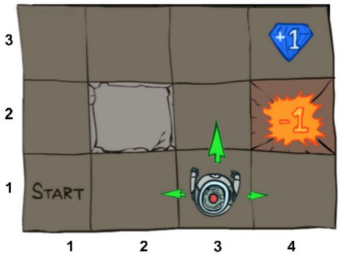
- Set of states S
- Set of actions A
- Transition function $P(s' \mid s, a)$
- Reward function $R(s, a, s')$
- Start state s_0
- Discount factor γ
- Horizon H



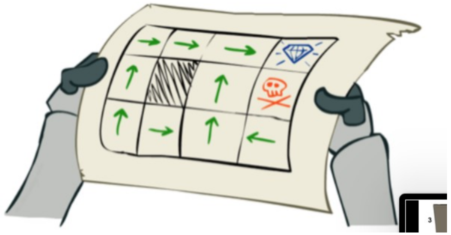
Return:

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Goal: $$\arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R_t | \pi\right]$$



π:



State and action value functions

- Definition**:** the **state-value function** $V^{\pi}(s)$ of an MDP is the expected return starting from state s, and following policy

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[G_t | S_t = s\right]$$

Captures long term reward

- Definition: the **action-value function** $Q^{\pi}(s,a)$ is the expected return starting from state s, taking action a, and then following policy

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[G_t | S_t = s, A_t = a\right]$$

Captures long term reward

Relationships between state and action values

State value functions

Action value functions

$$V^\pi(s) = \sum_a \pi(a|s) Q^\pi(s,a)$$

$$V^*(s) = \max_\pi V^\pi(s)$$

$$\begin{tikzcd}
\boxed{V^\pi(s)} \arrow[d] & \boxed{Q^\pi(s,a)} \arrow[l] \arrow[d] \\
\boxed{V^*(s)} & \boxed{Q^*(s,a)} \arrow[l]
\end{tikzcd}$$

$$Q^*(s,a) = \max_\pi Q^\pi(s,a)$$

$$V^*(s) = \max_a Q^*(s,a)$$

Obtaining the optimal policy

Optimal policy can be found by maximizing over Q*(s,a)

$$\pi^*(a|s) = \begin{cases} 1, & \text{if } a = \arg\max_a \ Q^*(s,a) \\ 0, & \text{else} \end{cases}$$

Obtaining the optimal policy

Optimal policy can be found by maximizing over Q*(s,a)

$$\pi^*(a|s) = \begin{cases} 1, & \text{if } a = \arg\max_a \ Q^*(s,a) \\ 0, & \text{else} \end{cases}$$

Optimal policy can also be found by maximizing over V*(s') with **one-step look ahead**

$$\pi^*(a|s) = \begin{cases} 1, & \text{if } a = \arg\max_a \mathbb{E}_{s'} \left[r(s,a,s') + \gamma V^*(s') \right] \\ 0, & \text{else} \end{cases}$$

$$\pi^*(a|s) = \begin{cases} 1, & \text{if } a = \arg\max_a \left[\sum_{s'} p(s'|s,a)(r(s,a,s') + \gamma V^*(s')) \right] \\ 0, & \text{else} \end{cases}$$

Policy Iteration

1. Policy evaluation

Iterate until convergence:

$$V^{\pi}_{[k+1]}(s) = \sum_{a} \pi_{[k]}(a|s) \sum_{s'} p(s'|s,a) \left[r(s,a,s') + \gamma V^{\pi}_{[k]}(s') \right]$$

2. Policy Improvement

Find the best action according to one-step look ahead

$$\pi_{[k+1]}(a|s) = \arg\max_{a} \sum_{s'} p(s'|s,a) \left[r(s,a,s') + \gamma V^{\pi}_{[k]}(s') \right]$$

Value Iteration

Algorithm:

Start with $V_0^*(s) = 0$ for all s.

For k = 1, ... , H:

For all states s in S:

$$V_k^*(s) \leftarrow \max_a \sum_{s'} P(s'|s,a) \left(R(s,a,s') + \gamma V_{k-1}^*(s') \right)$$

Value Iteration

Algorithm:

Start with $V_0^*(s) = 0$ for all s.

For k = 1, … , H:

For all states s in S:

$$V_k^*(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)\left(R(s,a,s') + \gamma V_{k-1}^*(s')\right)$$

$$\pi_k^*(s) \leftarrow \arg\max_a \sum_{s'} P(s'|s,a)\left(R(s,a,s') + \gamma V_{k-1}^*(s')\right)$$

Find the best action according to one-step look ahead

This is called a value update or Bellman update/back-up

Slides from Fragkiadaki

Value Iteration

Repeat until policy converges. Guaranteed to converge to optimal policy.

Algorithm:

Start with $V_0^*(s) = 0$ for all s.

For k = 1, ... , H:

For all states s in S:

$$V_k^*(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)\left(R(s,a,s') + \gamma V_{k-1}^*(s')\right)$$

$$\pi_k^*(s) \leftarrow \arg\max_a \sum_{s'} P(s'|s,a)\left(R(s,a,s') + \gamma V_{k-1}^*(s')\right)$$

Find the best action according to one-step look ahead

This is called a value update or Bellman update/back-up

Slides from Fragkiadaki

Q-Value Iteration

$Q^*(s, a)$ = expected utility starting in s, taking action a, and (thereafter) acting optimally

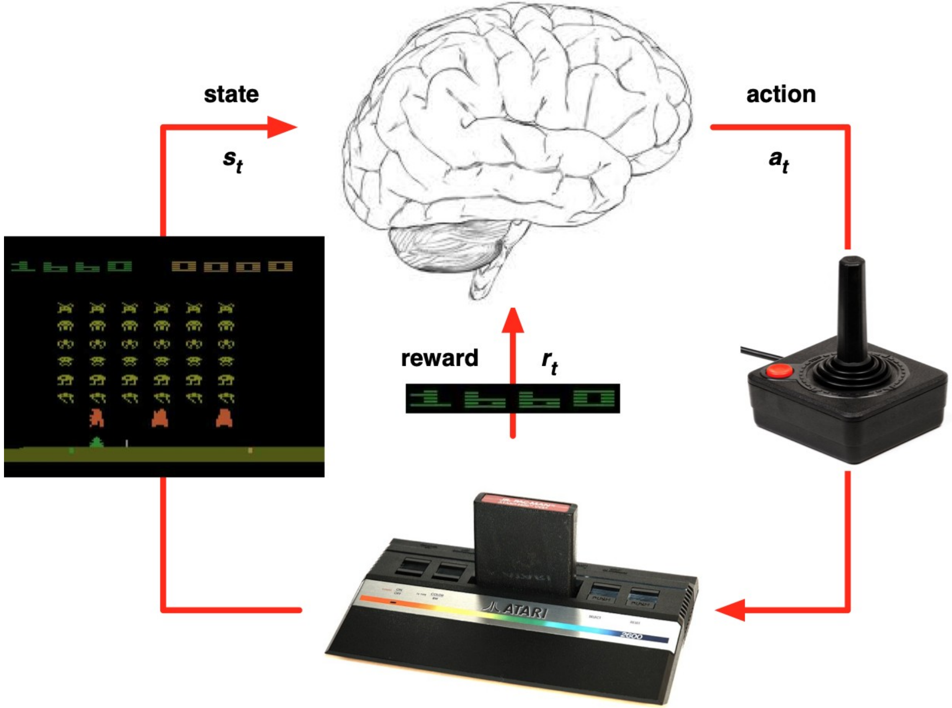
Bellman Equation:

$$Q^*(s,a) = \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q^*(s',a'))$$

Q-Value Iteration:

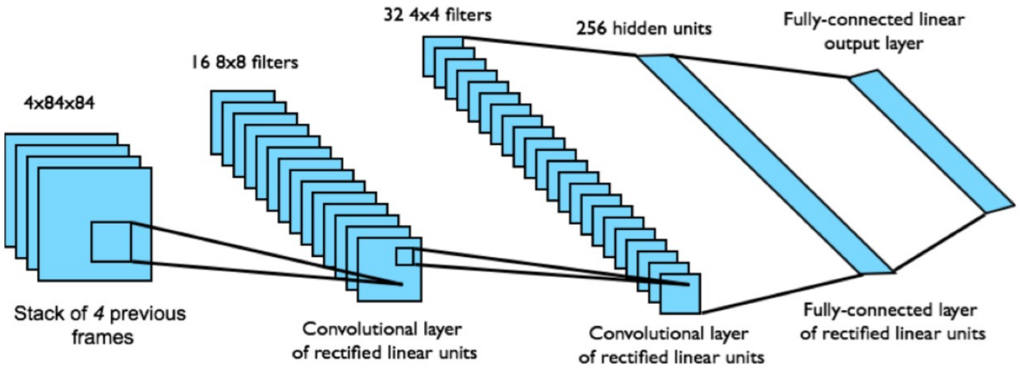
$$Q^*_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q^*_k(s',a'))$$

Deep Q-learning for Atari



Deep Q-learning for Atari

- End-to-end learning of values Q(s,a) from pixels s
- Input state s is stack of raw pixels from last 4 frames
- Output is Q(s,a) for 18 joystick/button positions
- Reward is change in score for that step

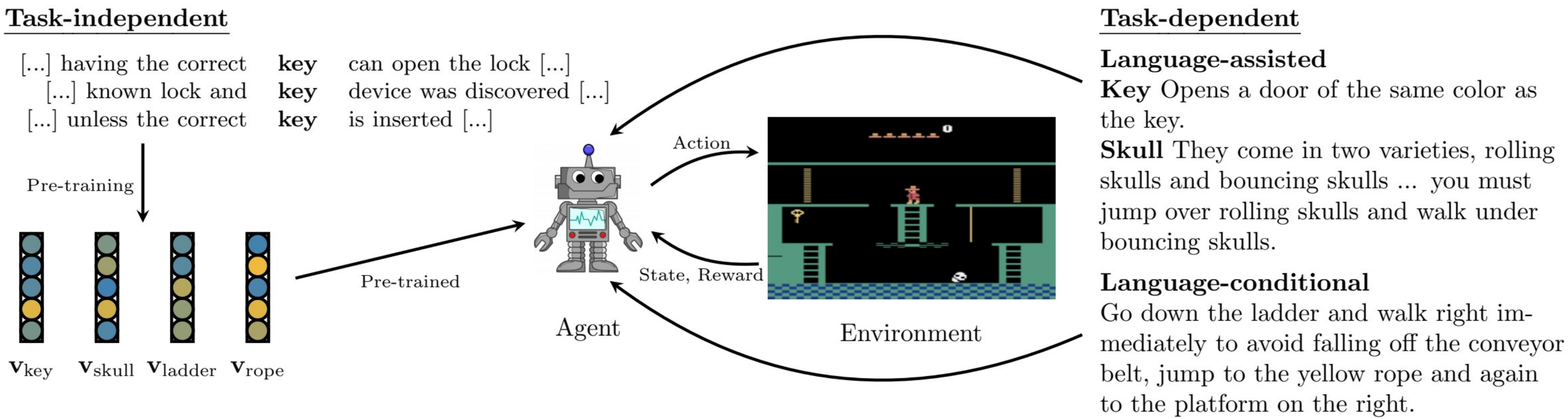


- Network architecture and hyperparameters fixed across all games

Slides from Fragkiadaki

Mnih et.al., Nature, 2014

RL and Language



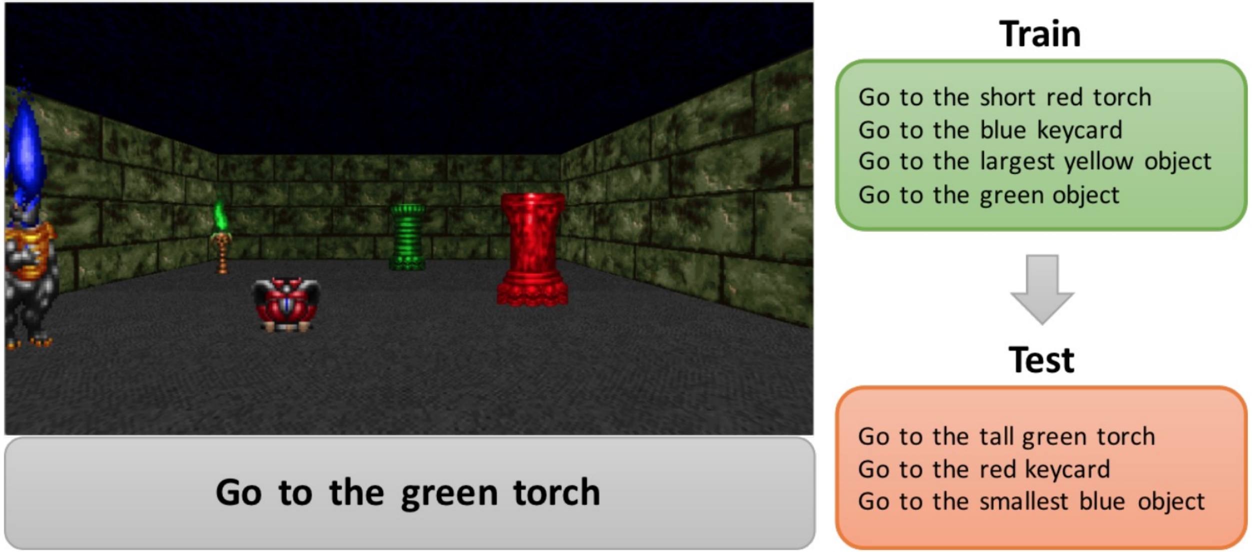
Luketina et. al., IJCAI 2019

Language-conditional RL

- Instruction following
- Rewards from instructions
- Language in the observation and action space

Language-conditional RL: Instruction following

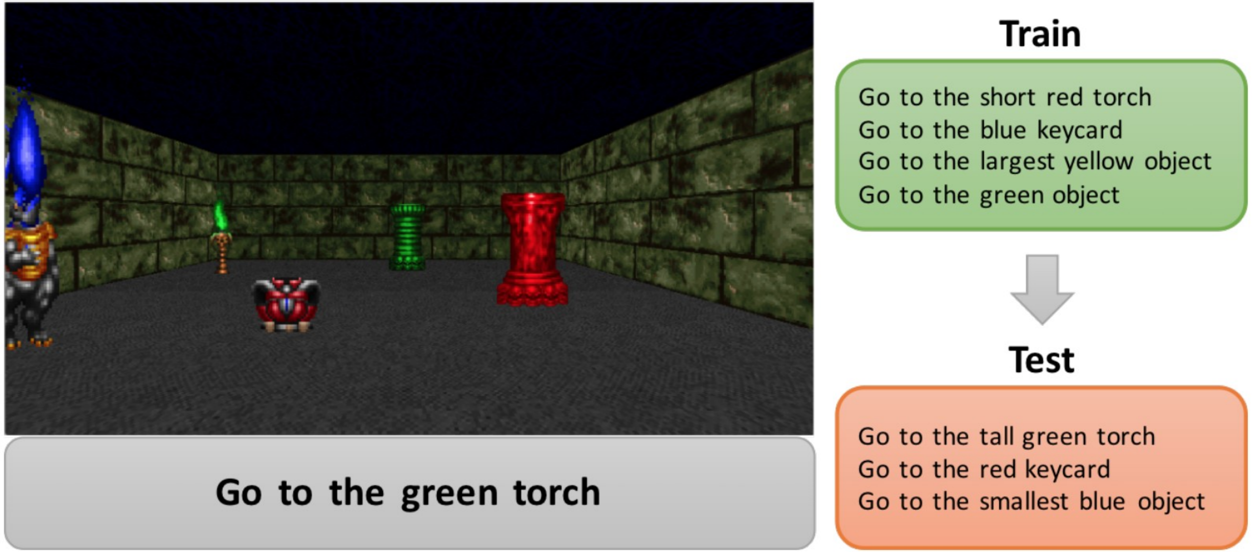
- Navigation via instruction following



Chaplot et. al., AAAI 2018
Misra et. al., EMNLP

Language-conditional RL: Instruction following

- Navigation via instruction following



Fusion
Alignment
Ground language
Recognize objects
Navigate to objects
Generalize to unseen objects

Chaplot et. al., AAAI 2018
Misra et. al., EMNLP

Applications: Hard attention

Hard attention 'gates' (0/1) rather than soft attention (softmax between 0-1)
- Can be seen as discrete layers in between differentiable neural net layers

Sentiment analysis, emotion recognition

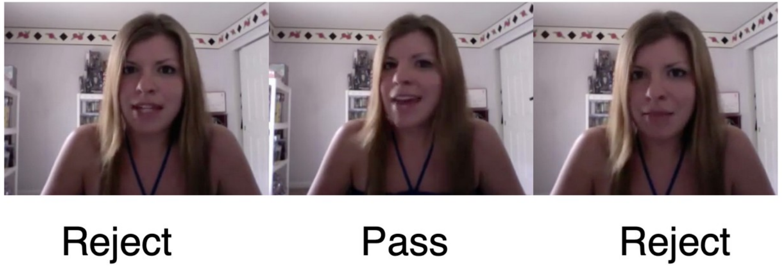
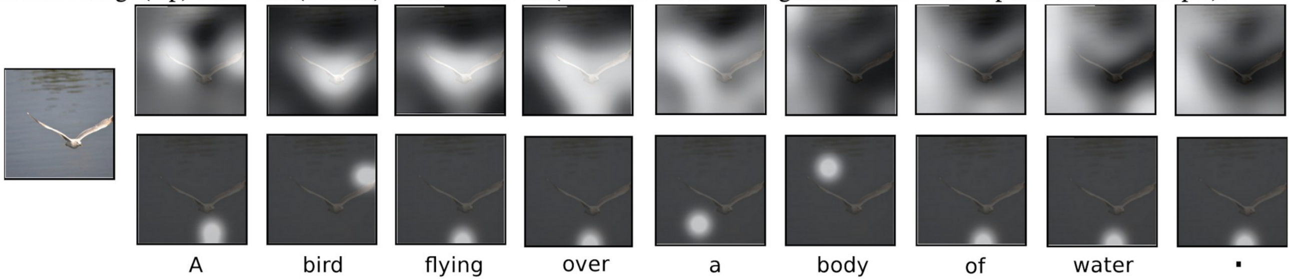


Figure 3. Visualization of the attention for each generated word. The rough visualizations obtained by upsampling the attention weights and smoothing. (top)"soft" and (bottom) "hard" attention (note that both models generated the same captions in this example).

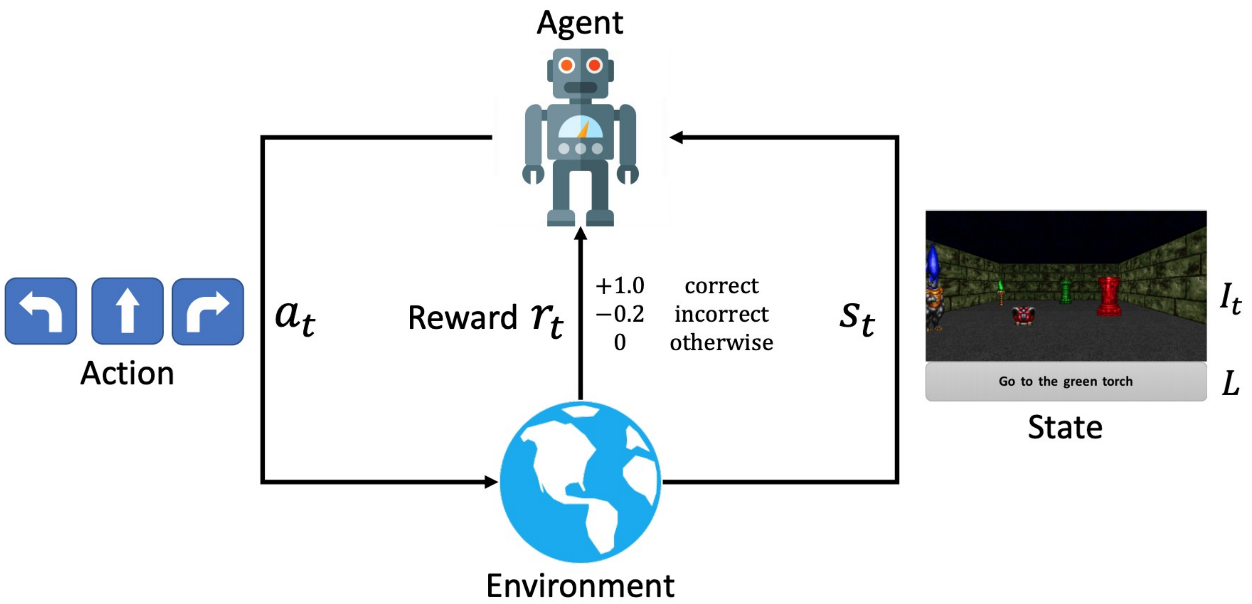
Image captioning



[Xu et. al., ICML 2015]
[Chen et al., ICMI 2017]

Language-conditional RL: Instruction following

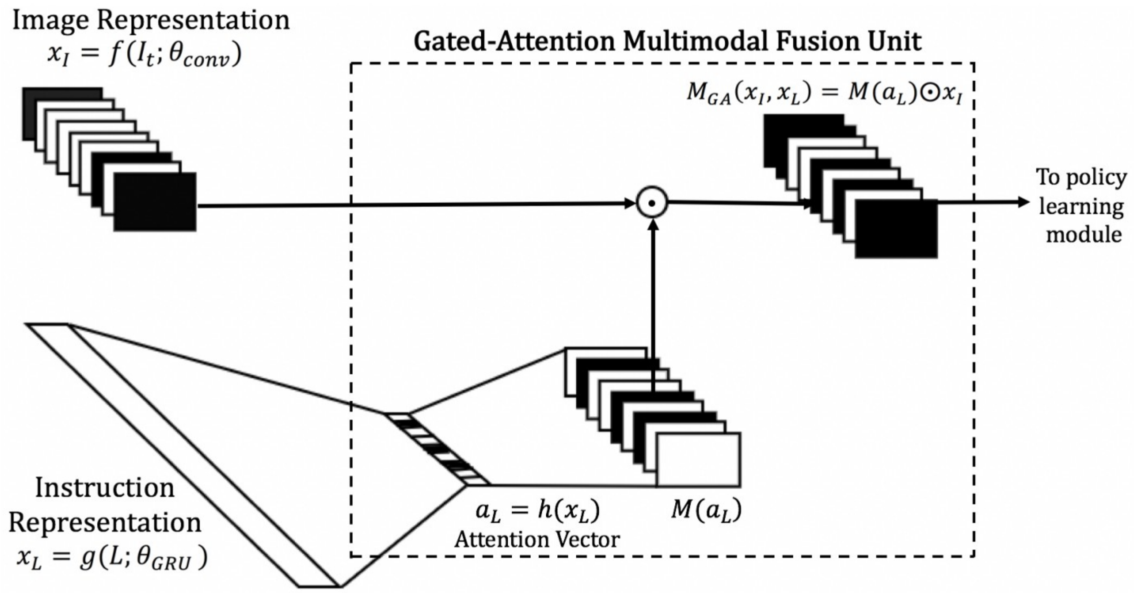
- Interaction with the environment



Chaplot et. al., AAAI 2018

Language-conditional RL: Instruction following

- Gated attention via element-wise product

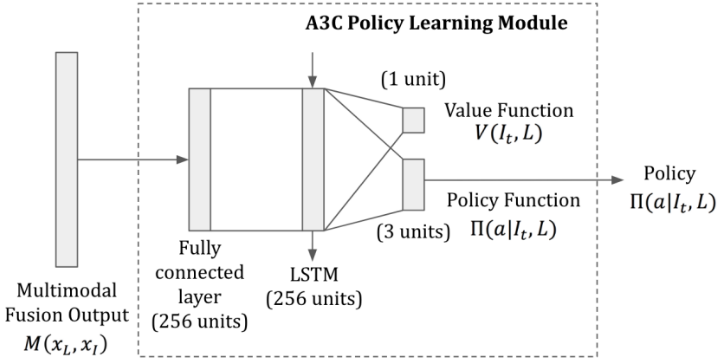


Fusion
Alignment
Ground language
Recognize objects

Chaplot et. al., AAAI 2018

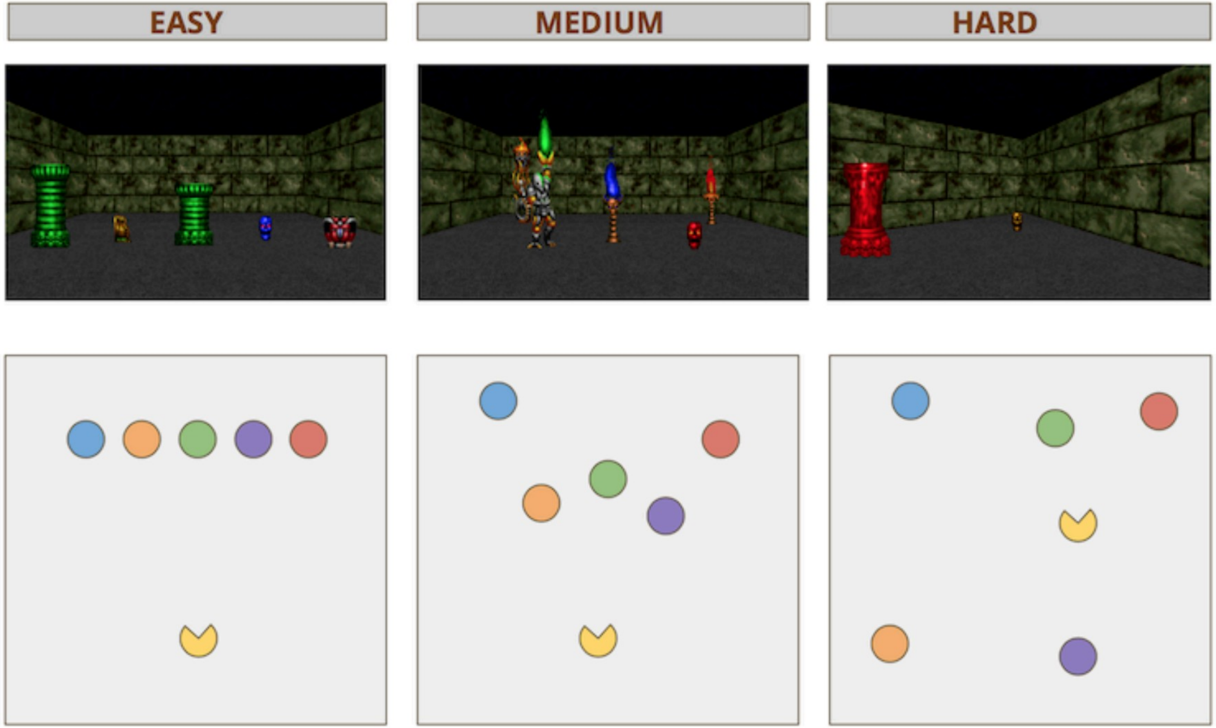
Language-conditional RL: Instruction following

- Policy learning
  - Asynchronous Advantage Actor-Critic (A3C) (Mnih et al.)
    - uses a deep neural network to parametrize the policy and value functions and runs multiple parallel threads to update the network parameters.
    - use **entropy regularization** for improved exploration
    - use **Generalized Advantage Estimator** to reduce the variance of the policy gradient updates (Schulman et al.)



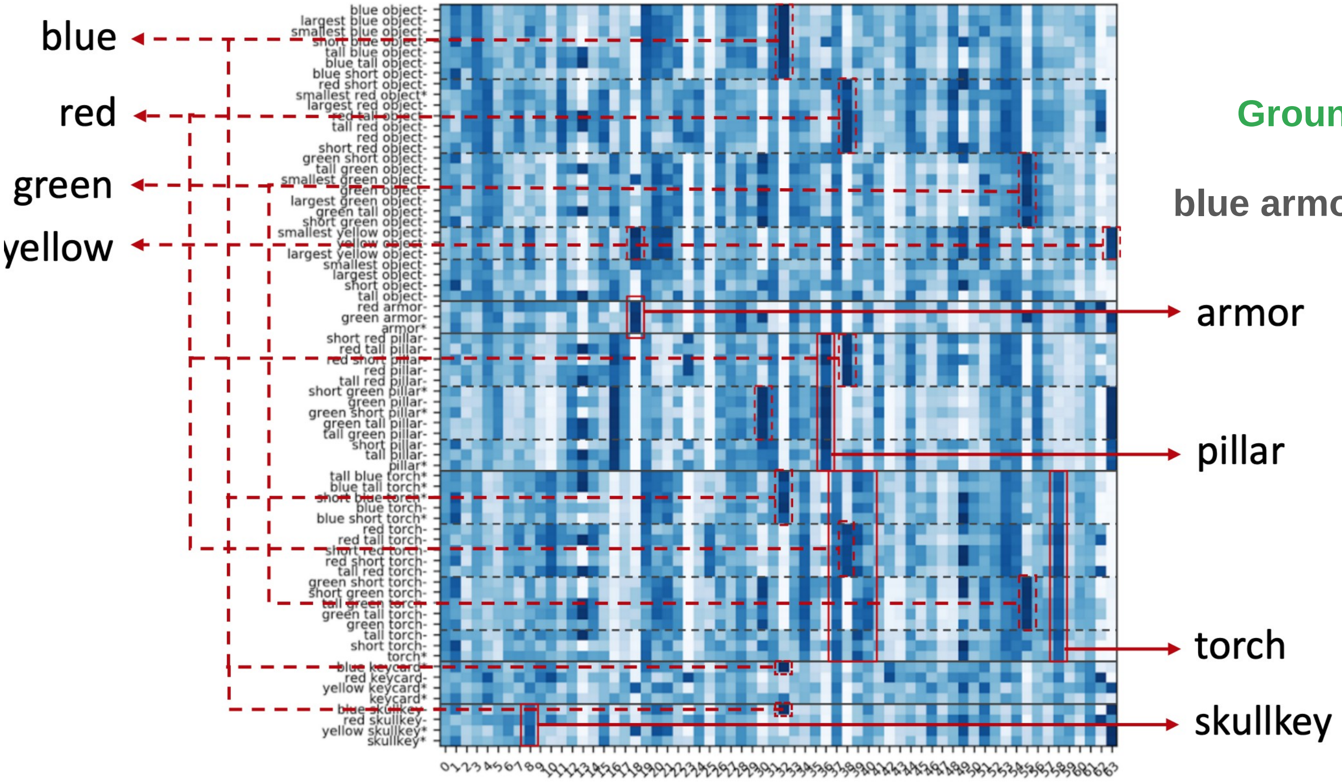
Chaplot et. al., AAAI 2018

Language-conditional RL: Instruction following



Chaplot et. al., AAAI 2018

Language-conditional RL: Instruction following

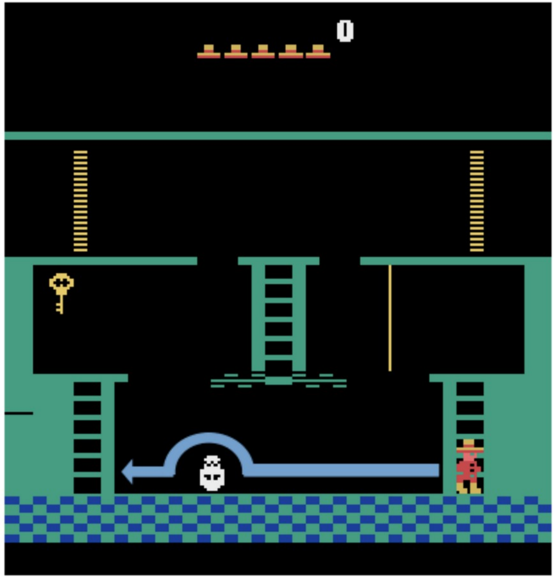


Grounding is important for generalization

blue armor, red pillar -> blue pillar

Chaplot et. al., AAAI 2018

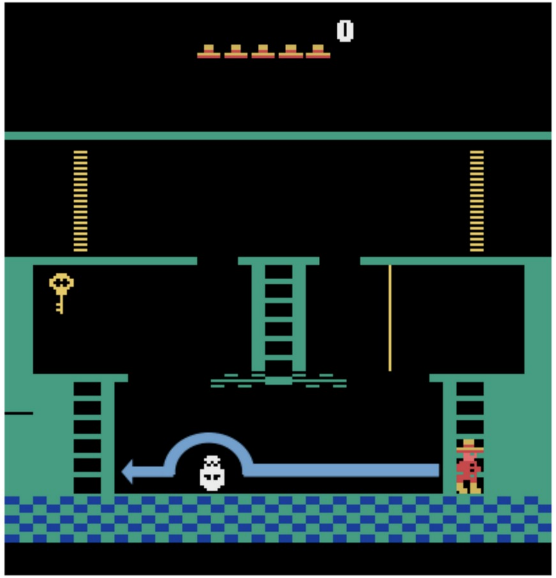
Language-conditional RL: Rewards from instructions



Montezuma's revenge

Sparse, long-term reward problem
General solution: reward shaping via auxiliary rewards

Language-conditional RL: Rewards from instructions



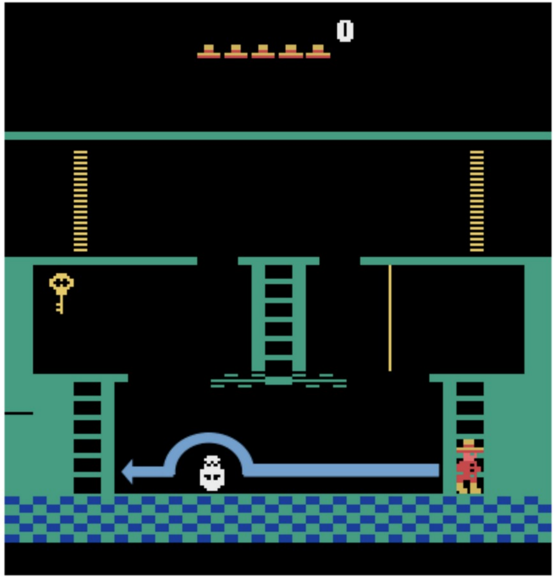
Montezuma's revenge

Sparse, long-term reward problem
General solution: reward shaping via auxiliary rewards

Encourages agent to explore its environment by maximizing **curiosity.**
How well can I **predict** my environment?
1. Less training data
2. Stochastic
3. Unknown dynamics
So I should **explore more**.

Pathak et. al., ICML 2017
Burda et. al., ICLR 2019

Language-conditional RL: Rewards from instructions



Montezuma's revenge

Sparse, long-term reward problem
General solution: reward shaping via auxiliary rewards

Natural language for reward shaping

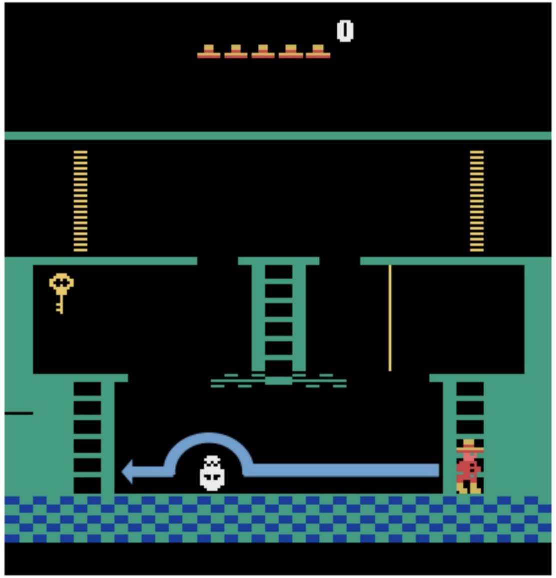
"Jump over the skull while going to the left"

from Amazon Mturk :-(
asked annotators to play the
game and describe entities

Intermediate rewards to speed up learning

Goyal et. al., IJCAI 2019

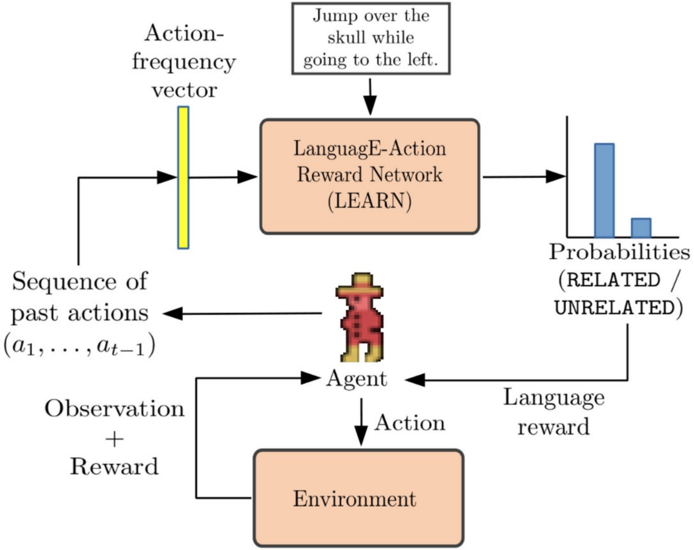
Language-conditional RL: Rewards from instructions



Montezuma's revenge

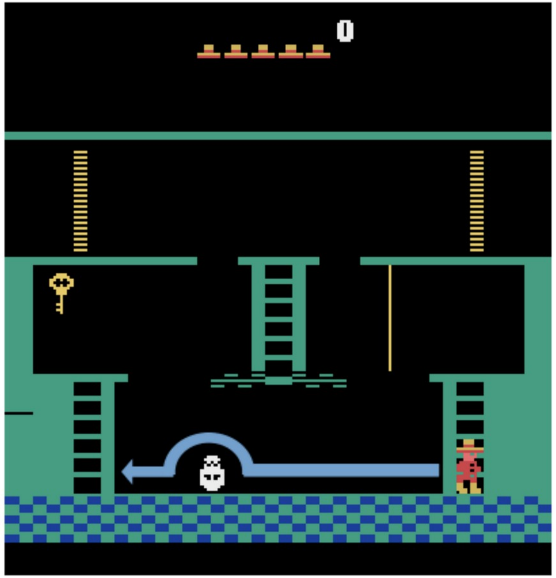
Natural language for reward shaping

Encourages agent to take actions related to the instructions



Goyal et. al., IJCAI 2019

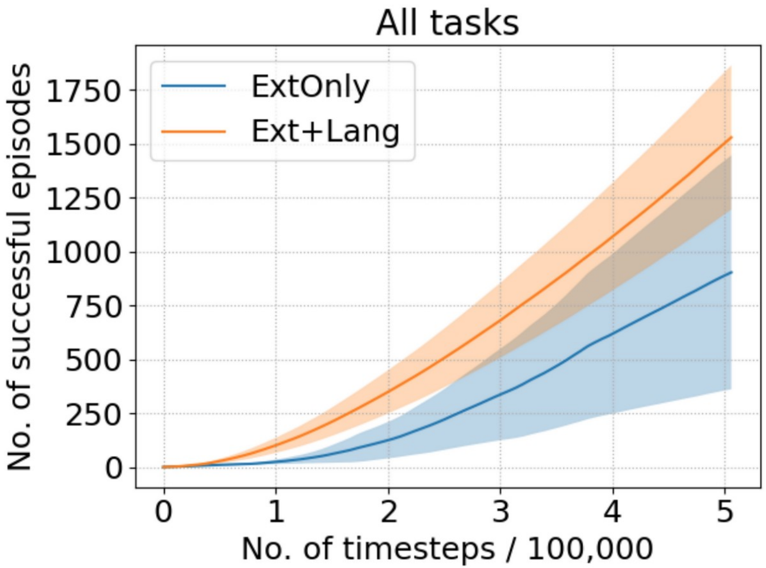
Language-conditional RL: Rewards from instructions



Montezuma's revenge

Natural language for reward shaping

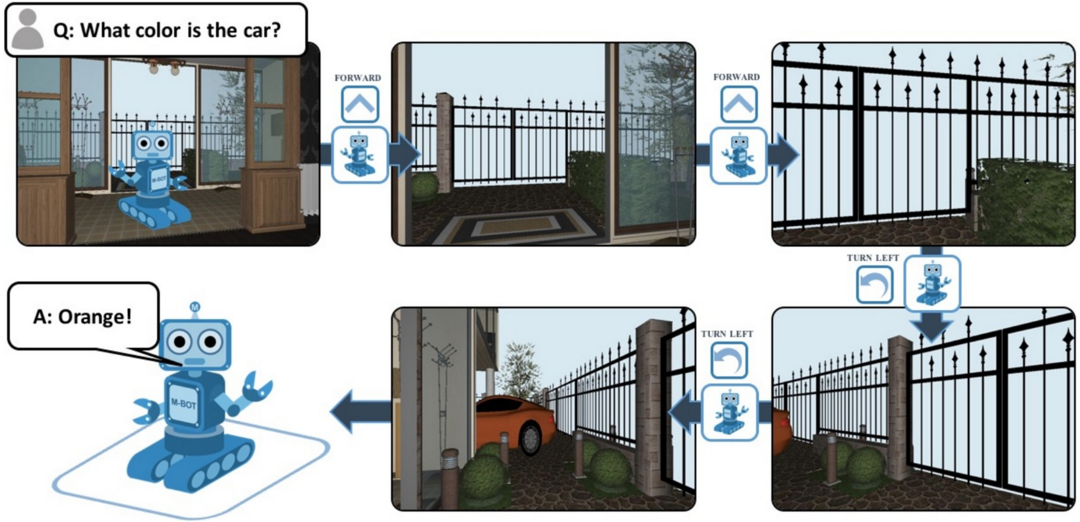
Encourages agent to take actions related to the instructions



Goyal et. al., IJCAI 2019

Language-conditional RL: Language in S and A

- Embodied QA: Navigation + QA



Most methods similar to instruction following

Das et. al., CVPR 2018

Language-assisted RL

- Language for communicating domain knowledge
- Language for structuring policies

Language-assisted RL: Domain knowledge

- Properties of entities in the environment are annotated by language

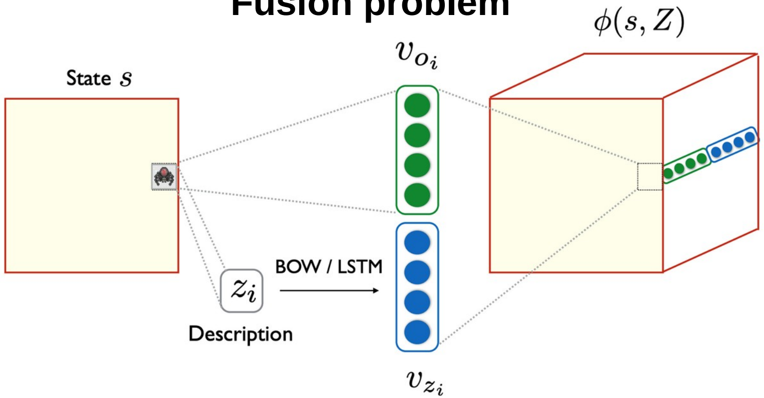
from Amazon Mturk :-(
asked annotators to play the game and describe entities

Narasimhan et. al., JAIR 2018

Language-assisted RL: Domain knowledge

- Properties of entities in the environment are annotated by language

Fusion problem

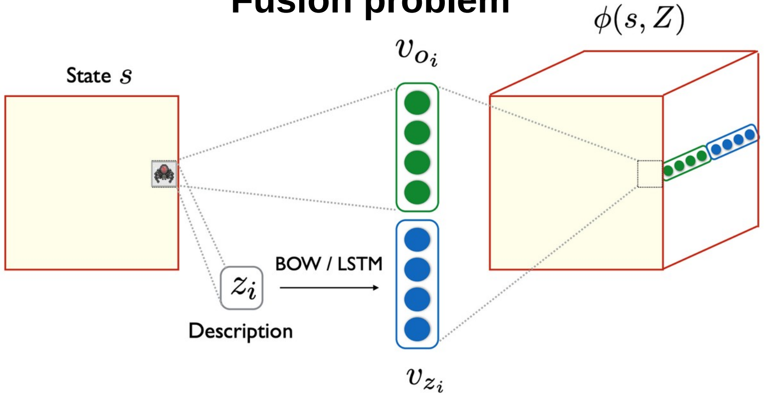


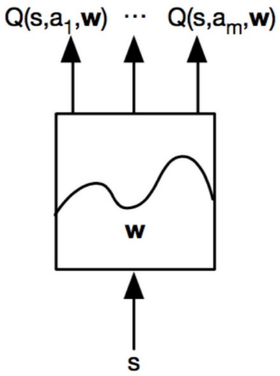
Narasimhan et. al., JAIR 2018

Language-assisted RL: Domain knowledge

- Properties of entities in the environment are annotated by language

Fusion problem



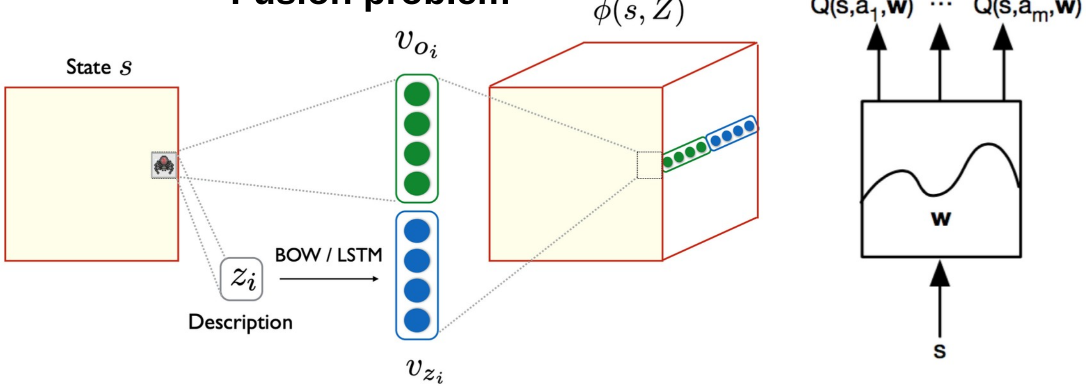


Narasimhan et. al., JAIR 2018

Language-assisted RL: Domain knowledge

- Properties of entities in the environment are annotated by language

Fusion problem



Grounded language learning
Helps to ground the meaning of text to the dynamics, transitions, and rewards
Language helps in multi-task learning and transfer learning

Narasimhan et. al., JAIR 2018

Language-assisted RL: Domain knowledge

- Learning to read instruction manuals

The natural resources available where a population settles affects its ability to produce food and goods. Build your city on a plains or grassland square with a river running through it if possible.

Figure 1: An excerpt from the user manual of the game Civilization II.

Branavan et. al., JAIR 2012

Language-assisted RL: Domain knowledge

- Learning to read instruction manuals

The natural resources available where a population settles affects its ability to produce food and goods. Build your city on a plains or grassland square with a river running through it if possible.

1. Choose **relevant** sentences
2. Label words into **action-description**, **state-description**, or **background**

Branavan et. al., JAIR 2012

Language-assisted RL: Domain knowledge

- Learning to read instruction manuals

The natural resources available where a population settles affects its ability to produce food and goods. Build your city on a plains or grassland square with a river running through it if possible.

Map tile attributes:
- Terrain type (e.g. grassland, mountain, etc)
- Tile resources (e.g. wheat, coal, wildlife, etc)

City attributes:
- City population
- Amount of food produced

Unit attributes:
- Unit type (e.g., worker, explorer, archer, etc)
- Is unit in a city ?

1. Choose **relevant** sentences
2. Label words into **action-description**, **state-description**, or **background**

Branavan et. al., JAIR 2012

Language-assisted RL: Domain knowledge

- Learning to read instruction manuals

The natural resources available where a population settles affects its ability to produce food and goods. Build your city on a plains or grassland square with a river running through it if possible.

Map tile attributes:
- Terrain type (e.g. grassland, mountain, etc)
- Tile resources (e.g. wheat, coal, wildlife, etc)

City attributes:
- City population
- Amount of food produced

Unit attributes:
- Unit type (e.g., worker, explorer, archer, etc)
- Is unit in a city ?

1. Choose **relevant** sentences
2. Label words into **action-description**, **state-description**, or **background**

Branavan et. al., JAIR 2012

Language-assisted RL: Domain knowledge

- Learning to read instruction manuals

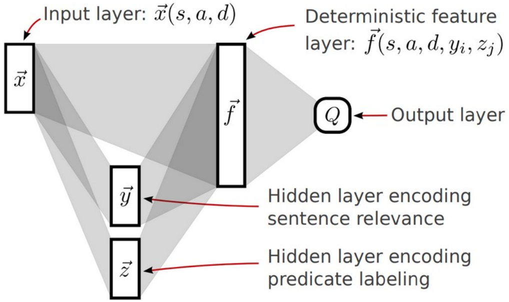
The natural resources available where a population settles affects its ability to produce food and goods. Build your city on a plains or grassland square with a river running through it if possible.

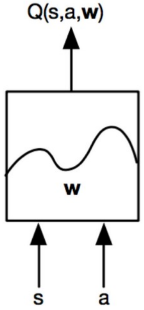
Map tile attributes:
- Terrain type (e.g. grassland, mountain, etc)
- Tile resources (e.g. wheat, coal, wildlife, etc)

City attributes:
- City population
- Amount of food produced

Unit attributes:
- Unit type (e.g., worker, explorer, archer, etc)
- Is unit in a city ?

1. Choose **relevant** sentences
2. Label words into **action-description**, **state-description**, or **background**





Branavan et. al., JAIR 2012

Language-assisted RL: Domain knowledge

- Learning to read instruction manuals

- Phalanxes are twice as effective at defending cities as warriors.
- Build the city on plains or grassland with a river running through it.
- You can rename the city if you like, but we'll refer to it as washington.
- There are many different strategies dictating the order in which advances are researched

Relevant sentences

- After the road is built, use the settlers to start improving the terrain.
  S S S A A A A A
- When the settlers becomes active, chose build road.
  S S S A A A
- Use settlers or engineers to improve a terrain square within the city radius
  A S A A S A S S S S

A: action-description
S: state-description

Branavan et. al., JAIR 2012

Language-assisted RL: Domain knowledge

- Learning to read instruction manuals

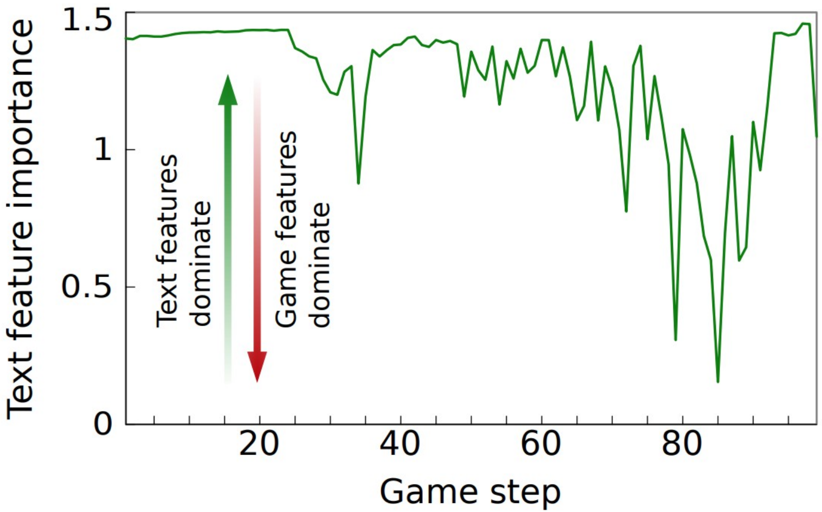
Method	% Win	% Loss	Std. Err.
Random	0	100	—
Built-in AI	0	0	—
Game only	17.3	5.3	± 2.7
Sentence relevance	46.7	2.8	± 3.5
Full model	**53.7**	5.9	± 3.5
Random text	40.3	4.3	± 3.4
Latent variable	26.1	3.7	± 3.1

Grounded language learning
Ground the meaning of text to the dynamics, transitions, and rewards
Language helps in learning

Branavan et. al., JAIR 2012

Language-assisted RL: Domain knowledge

- Learning to read instruction manuals

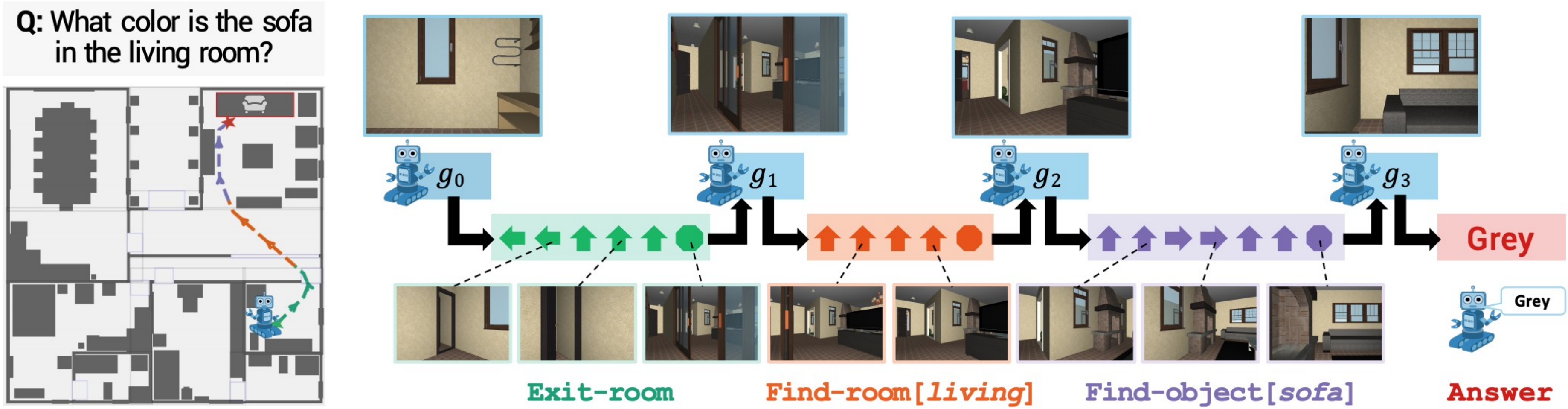


Language is most important at the start when you don't have a good policy
Afterwards, the model relies on game features

Branavan et. al., JAIR 2012

Language for structuring policies

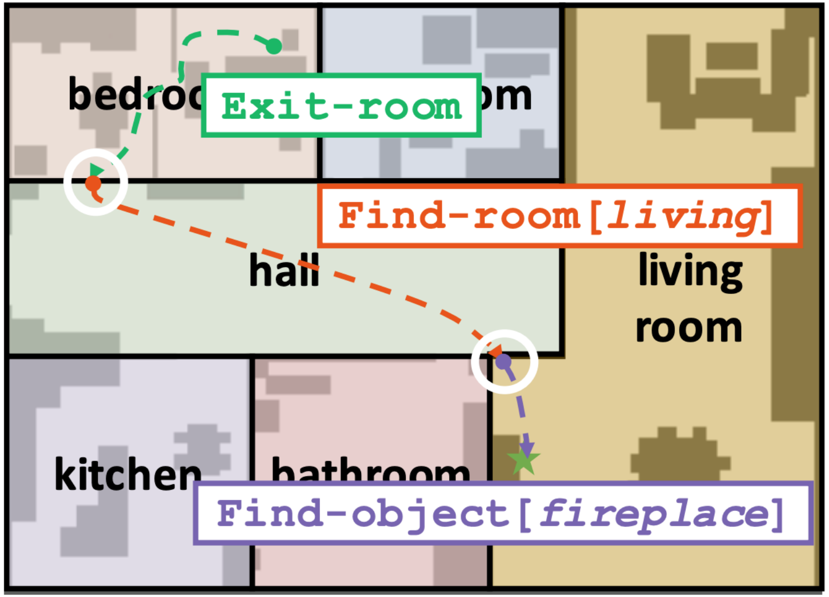
- Composing modules for Embodied QA

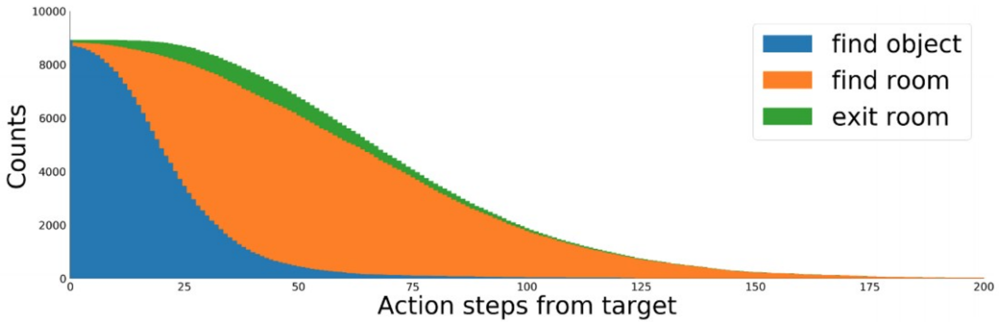


Das et. al., CoRL 2018

Language for structuring policies

- Composing modules for Embodied QA





Das et. al., CoRL 2018

Summary of applications

Stochastic optimization

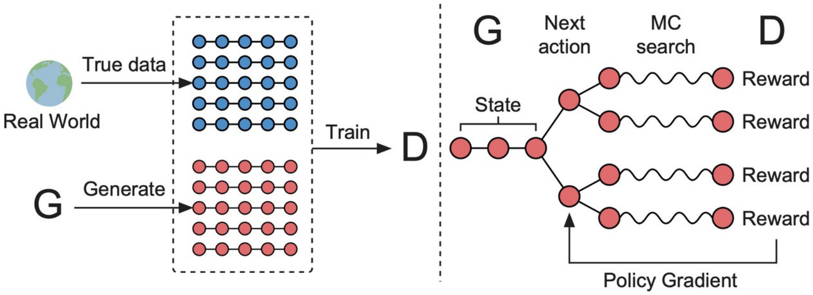
sentences LM

Sample $\boxed{\mathbf{z}^1, \cdots, \mathbf{z}^K}$ from $\boxed{q_\phi(\mathbf{z})}$ and estimate

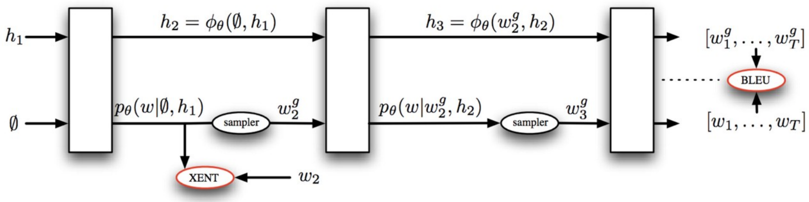
$$\nabla_\phi E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] \approx \frac{1}{K} \sum_k \boxed{f(\mathbf{z}^k)} \nabla_\phi \log q_\phi(\mathbf{z}^k)$$

disc reward

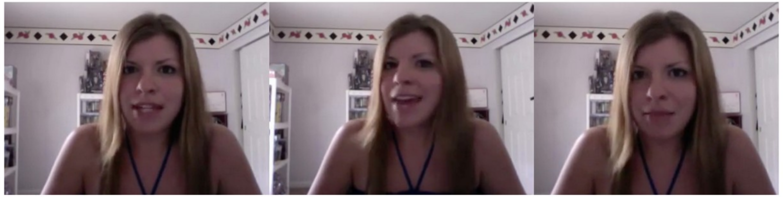
Text generation



General reward functions



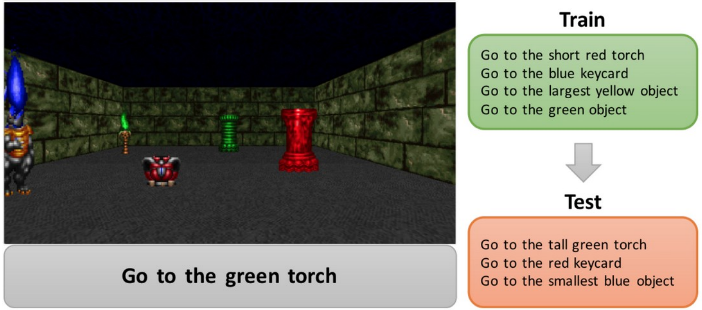
Discrete layers



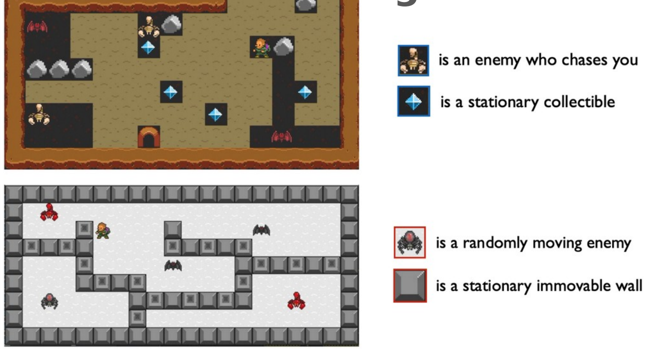
Reject Pass Reject

Summary of applications

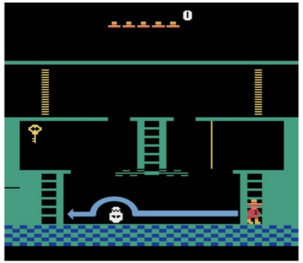
Instruction following



Language as domain knowledge



Language for rewards



"Jump over the skull while going to the left"

Language to structure