Google AI

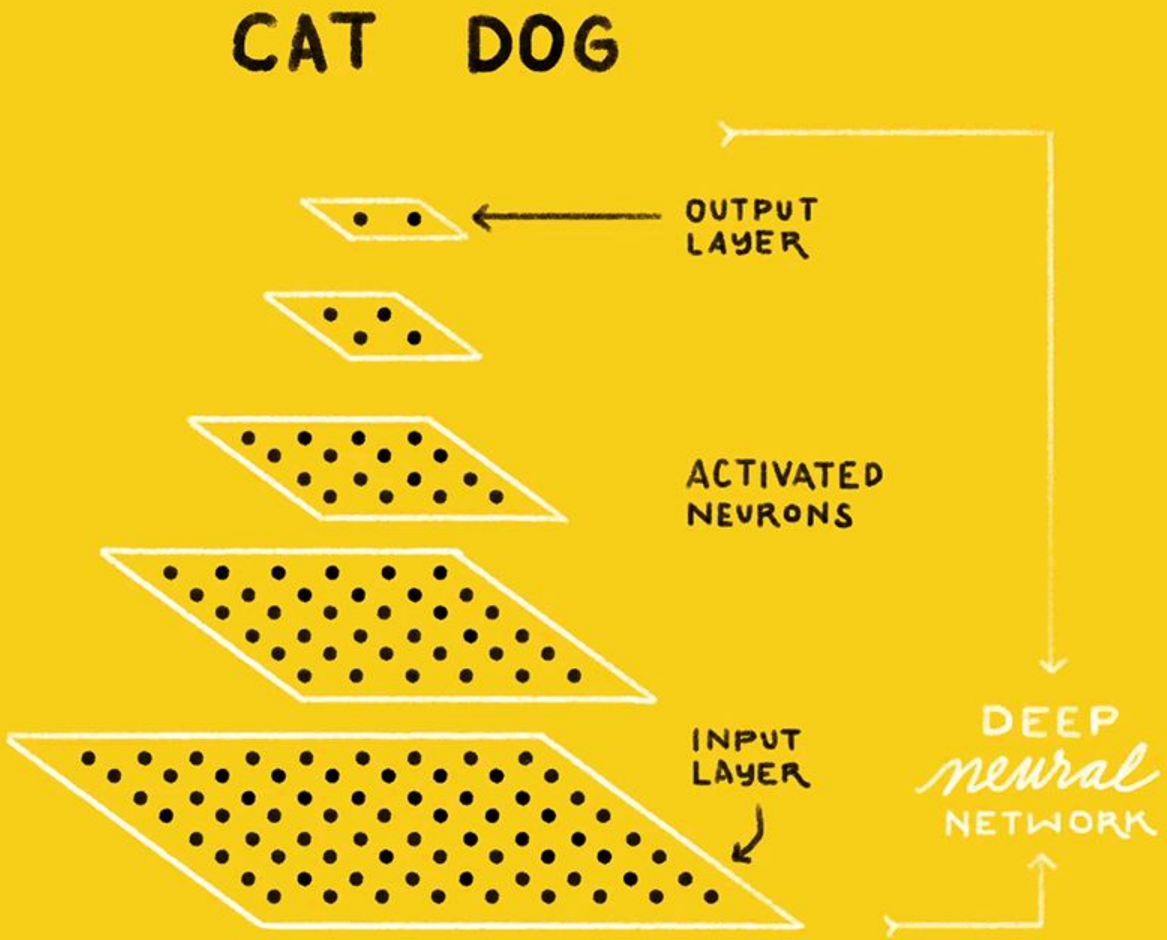
Reinforcement Learning

PhD. Sergio Guadarrama
Google Research

@sguada

IS THIS A

CAT or DOG?



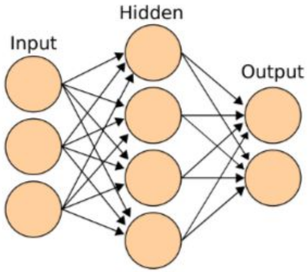
Google AI

Programming is not enough

Programming

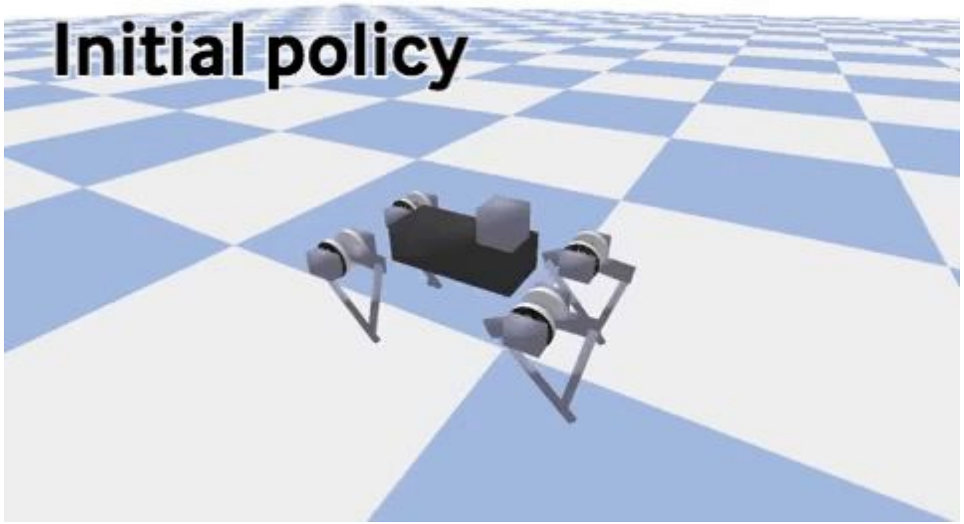
+ Automates execution
- Needs full specification of state and control flow

Supervised Learning



+ Learns representation and maps to a decision
- Needs expert label for every decision

Learning to Walk



Google AI

Not all who wander are lost...

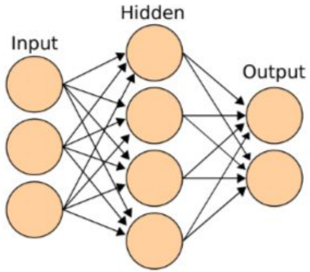
Supervised Learning

Reinforcement Learning

Supervised Learning is not enough

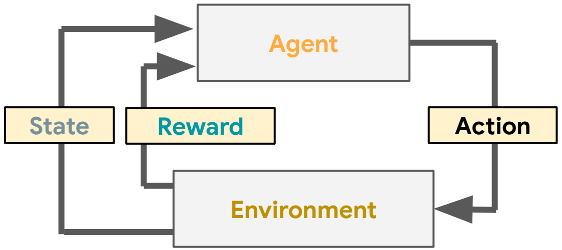
Google AI

Supervised Learning



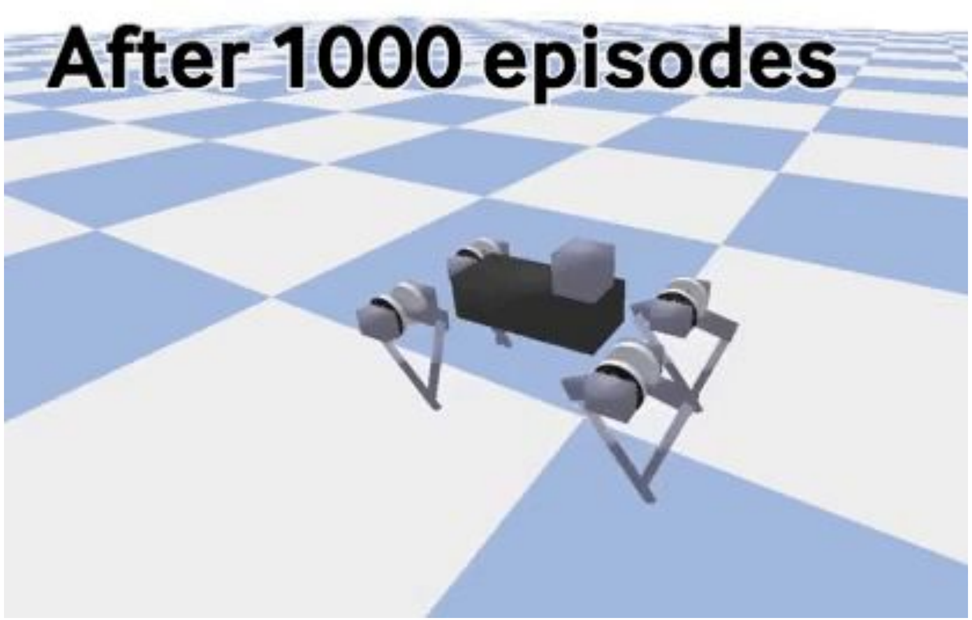
- Needs expert label for every decision

Reinforcement Learning



+ Agent learns from trial and error

Learning to Walk



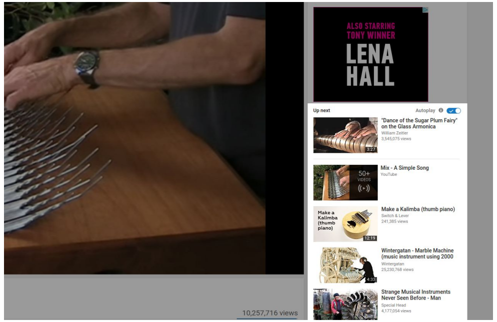
Learning to walk

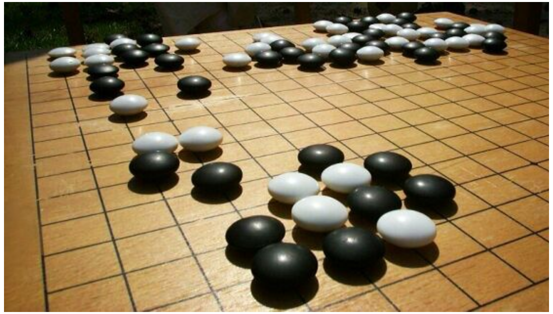
Learning to Walk

Learning to walk

Google AI

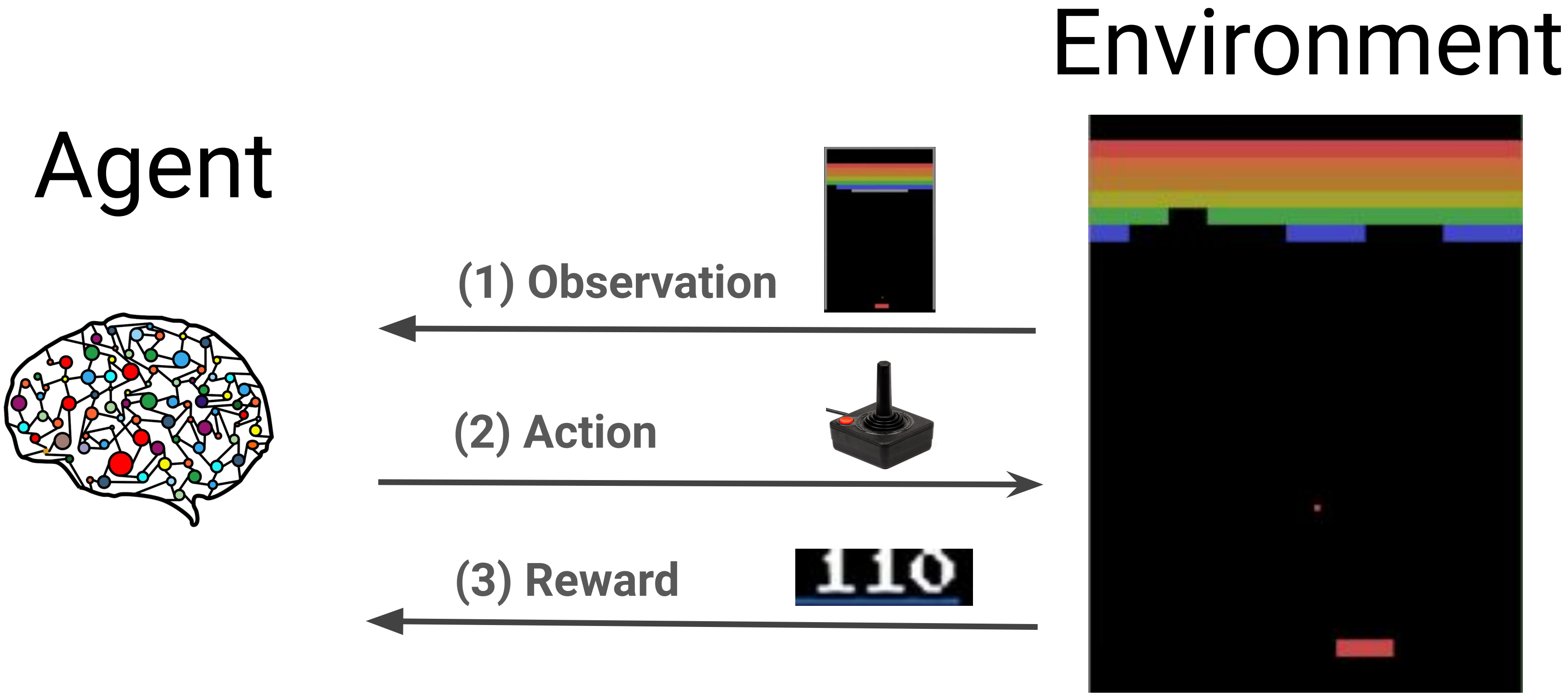
Reinforcement Learning applications



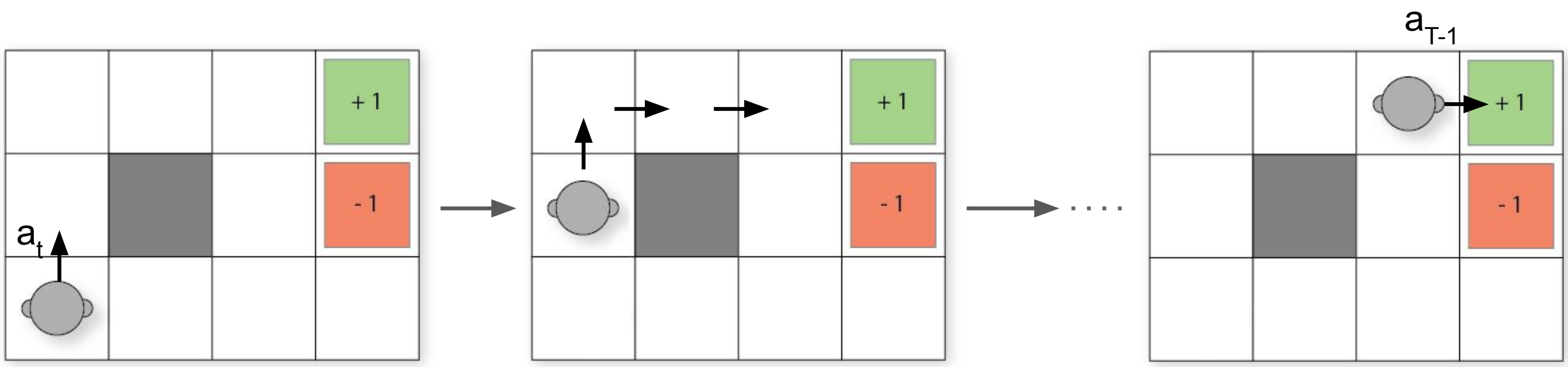


How does RL work?

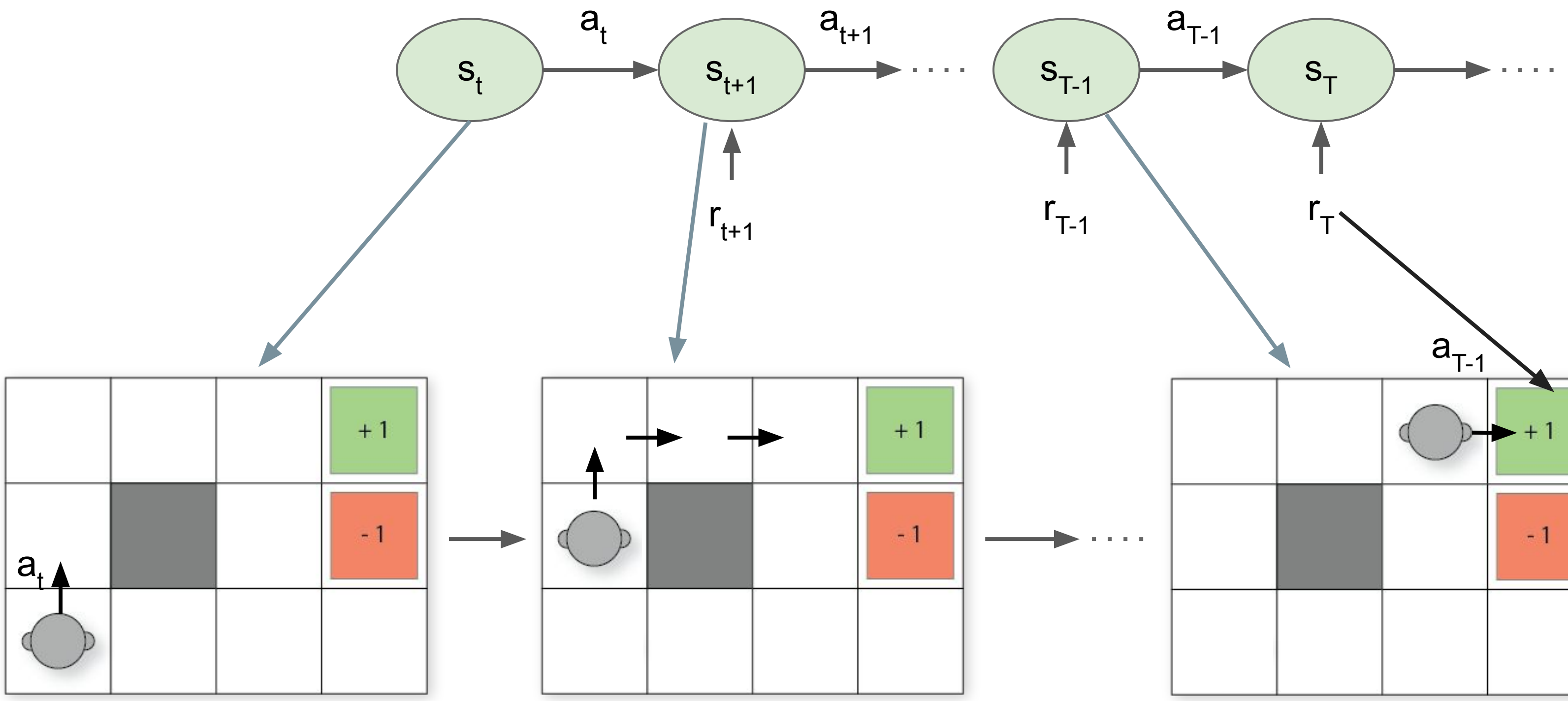
Agent vs Environment



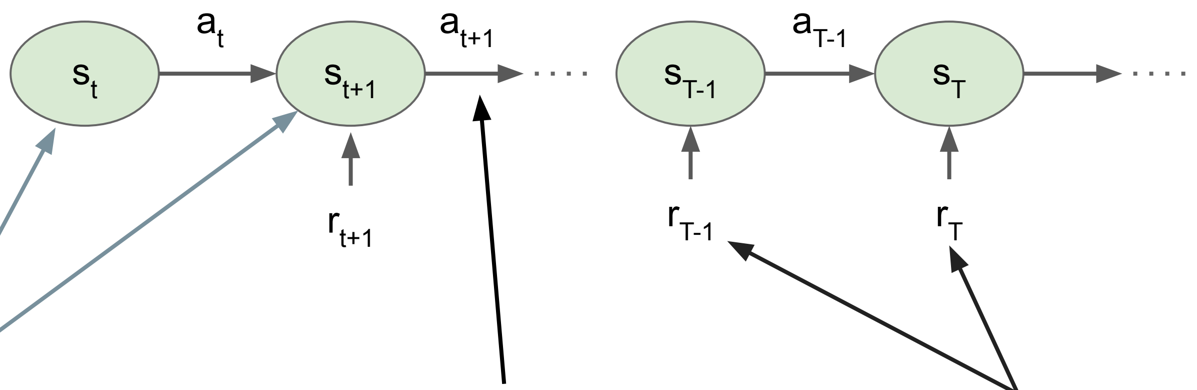
Foundation: Sequence of decisions



Foundation: Sequence of decisions



Foundation: Markov Decision Processes



$s_t, s_{t+1} \in S$

Environment states

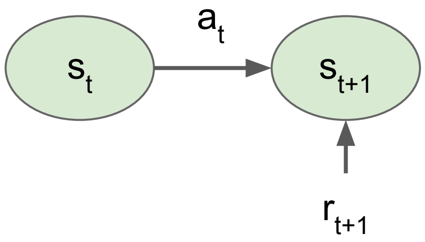
$a_t \in A$

Agent actions

$\mathcal{R}_{s_t}^{a_t}$

Rewards

Transition and Reward functions



$$\mathcal{P}^{a}_{s_t, s_{t+1}} = P(s_{t+1} | s_t, a_t)$$

Current state, Action → Next state

$$\mathcal{R}^{a_t}_{s_t, s_{t+1}} = E(r_t | s_t, s_{t+1}, a_t)$$

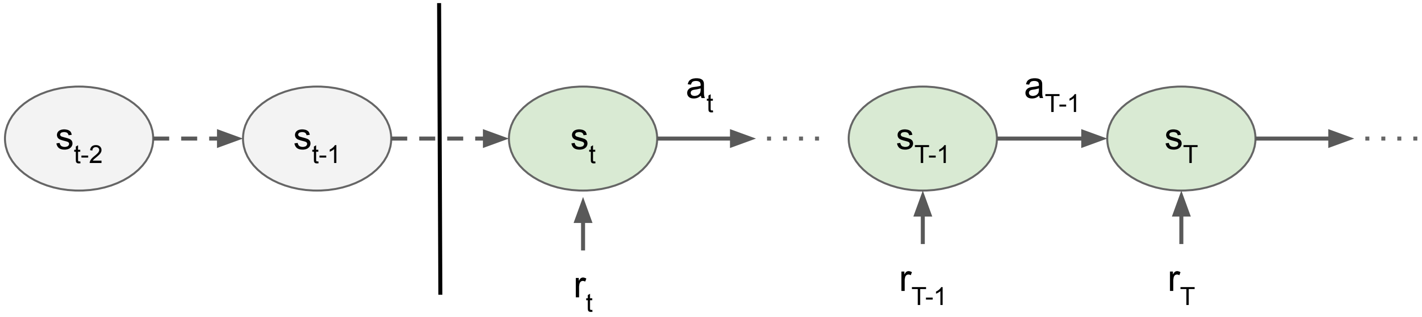
Current state, Action, Next state → Reward

The Markov Property

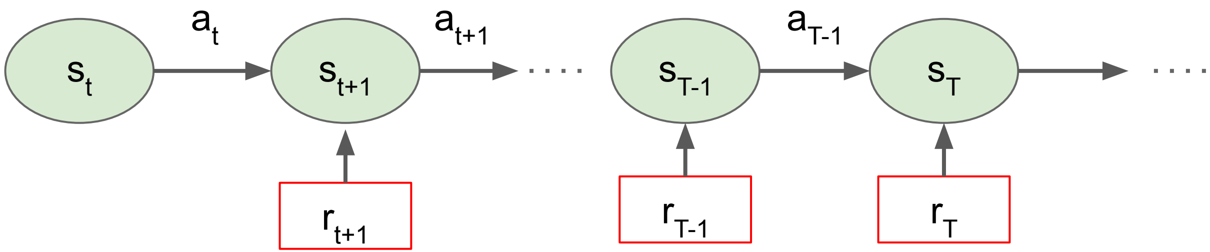
Assumption:

Given the **present**, the **future** is independent of the **past**

i.e., no matter how you **reached** a state,
all **future** states can be predicted from your **current** state and **future** actions



Objective: Learn Policy that Maximize “Return”



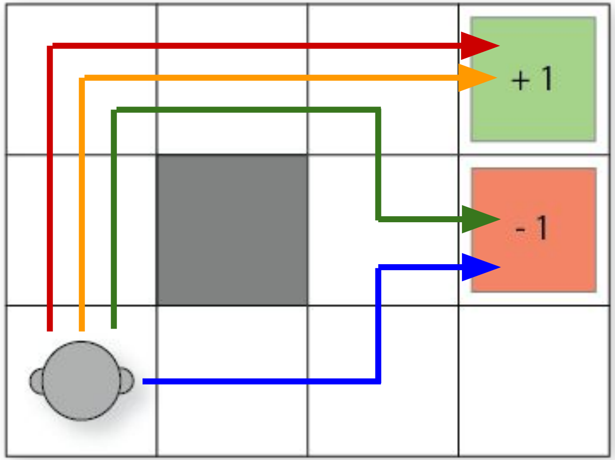
Return = Expected sum of discounted future rewards

$$E[R_t] = E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots]$$

Policy

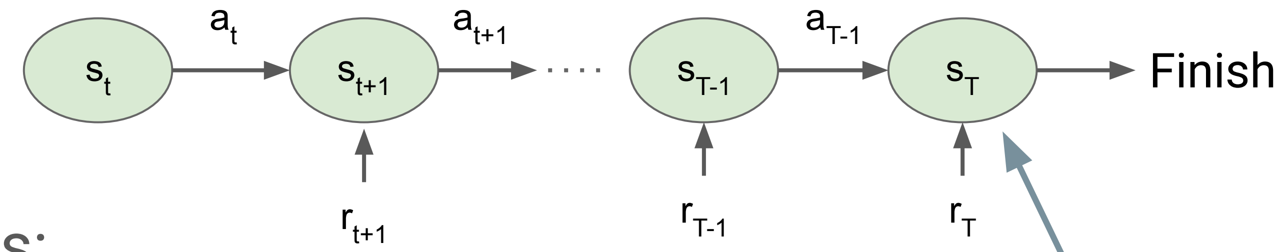
A distribution π over actions given a state.

$$a_t \sim \pi(a|s_t)$$



Google AI

Episodes



Termination reasons:

- Goal acquired
- Fatal mistake
- Step limit reached

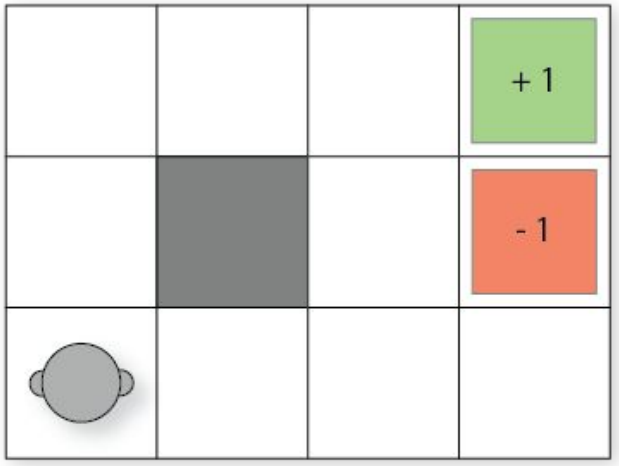
Terminal State

$$E[R_t] = E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{T-t} r_T]$$

Reasoning under uncertainty

Unknown Environment model	Multi-armed Bandits	**Reinforcement Learning**
Known Environment model	Decision Theory	Planning with MDP
	Actions do not change world state	Actions change world state

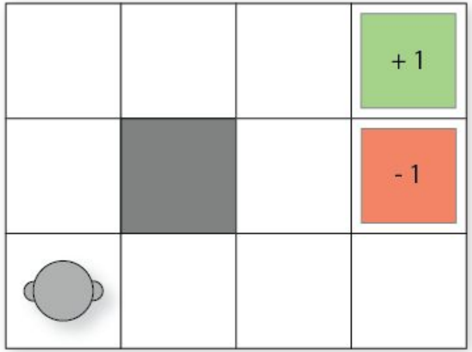
Value Functions & Q-Learning



Value Function

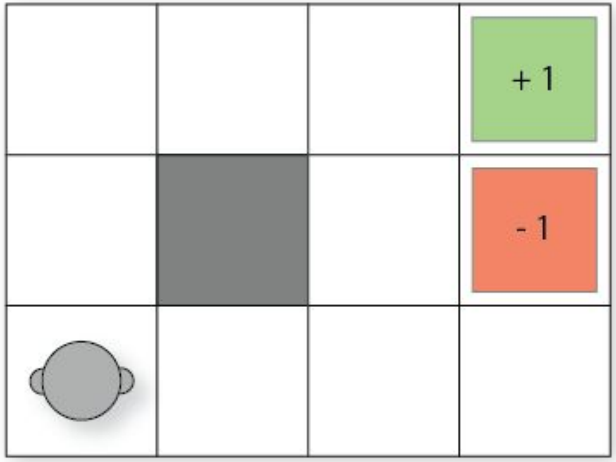
Goal: Choose **actions** that maximize **return**

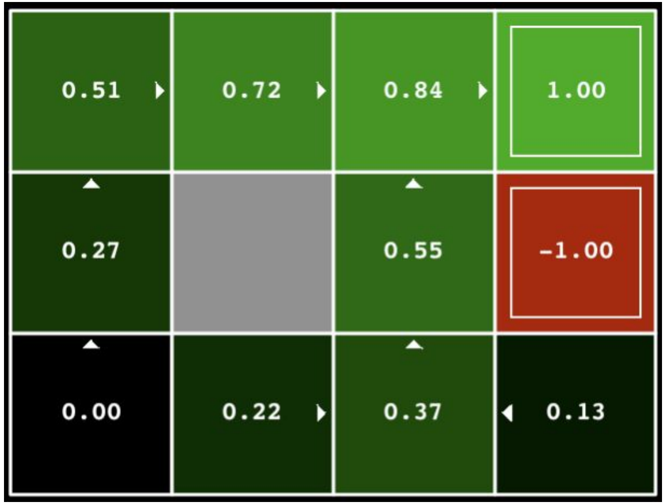
OR: Visit **states** that maximize **return**



$$V^{\pi}(s) = E_{\pi}[R_t | s_t = s]$$

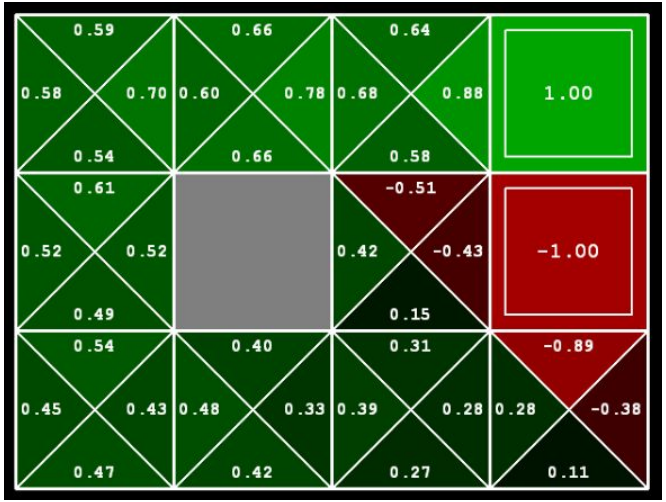
Value Function





Action-Value Function (Q-function)

What is the expected return of taking an **action *a*** in a **state *s*** and then following a **policy** π?

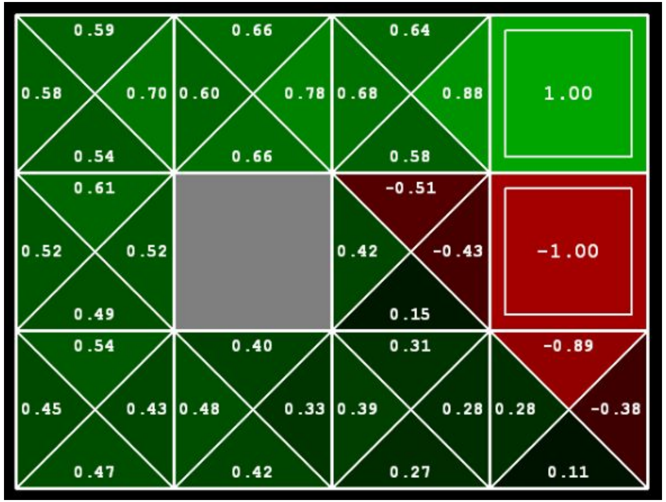


$$Q^{\pi}(s,a) = E_{\pi}[R_t | s_t = s, a_t = a]$$

Q-function: Concept

$$Q^{\pi}(s,a) = E_{\pi}[R_t | s_t = s, a_t = a]$$

Knowing reward requires knowing environment, so *Q*-function depends on agent's knowledge of the environment



Greedy Policy with Value-based RL

From the current state, choose action that has the highest estimated Q-value:

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

Bellman Equation: Concept

- Q improves with experience
- When is Q "good enough"?
- Realize Q is recursive, meaning

$$Q(s,a) = R(s,a) + \gamma * \max Q(s',a')$$

Bellman Equation: Concept

$Q(s,a)$ after taking action "a" in state "s"

$$= \quad R(s,a) + \gamma * \max Q(s')$$

$$= \quad \sum_{s'} [R(s,a,s') + \gamma * \max Q(s')]$$

$$= \quad \sum (P(s'|s,a) * [R(s,a,s') + \gamma * \max Q(s')])$$

Bellman Equation: Math

Define *Q** as the optimal *Q*-function which satisfies:

$$Q^*(s,a) = \sum_{s'} P(s' \mid s,a)\left[R(s,a,s') + \gamma \max_{a'} Q^*(s',a')\right]$$

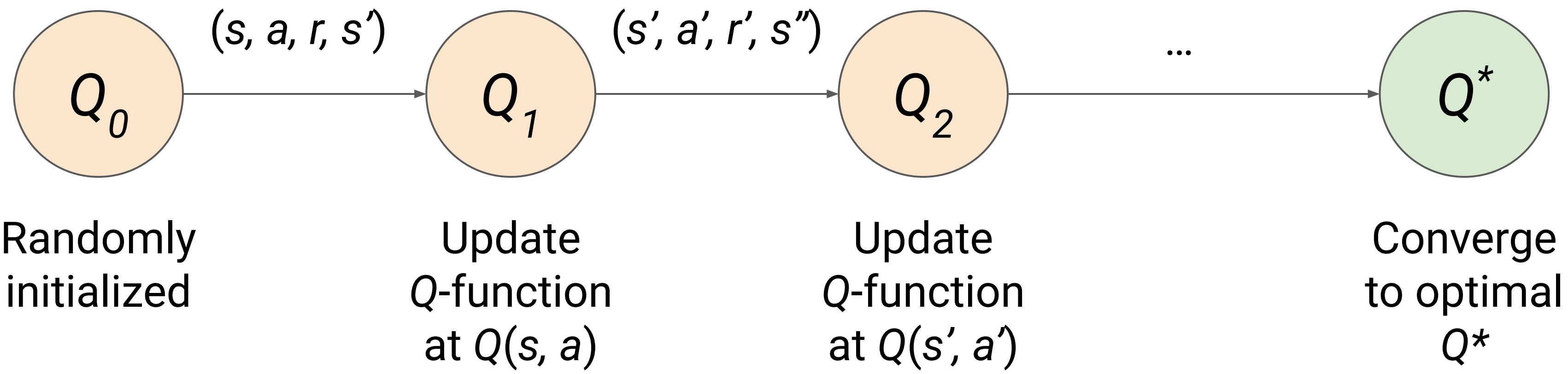
$$= \mathbb{E}_{s'}\left[R(s,a,s') + \gamma \max_{a'} Q^*(s',a') \mid s,a\right]$$

Q-Learning Algorithm

Act in the world and collect an experience sample:

state, action, reward, next state (s, a, r, s')

Update Q function based on the new sample



Bellman Error Gradient

Goal: Minimize Bellman Error

Discount

Old estimate

$$\underbrace{r + \gamma \max_{a'} Q(s', a')}_{\text{New estimate}} - \overbrace{Q(s, a)}^{\text{Old estimate}}$$

Actual Reward

Q-Learning update

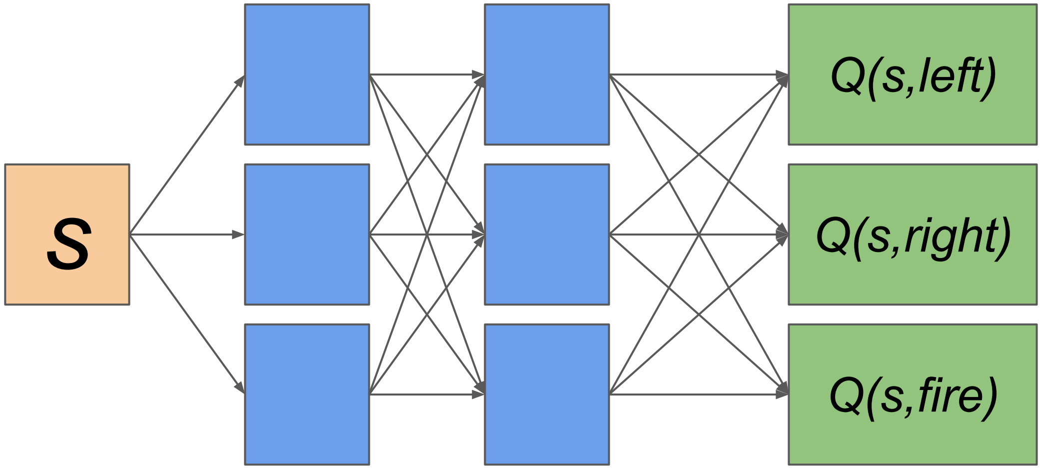
Goal: Minimize Bellman Error

$$Q(s,a) \leftarrow Q(s,a) + \alpha \overbrace{\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)}^{\text{Bellman Error Gradient}}$$

Learning Rate

Discount

Deep Q-Learning with DQNs



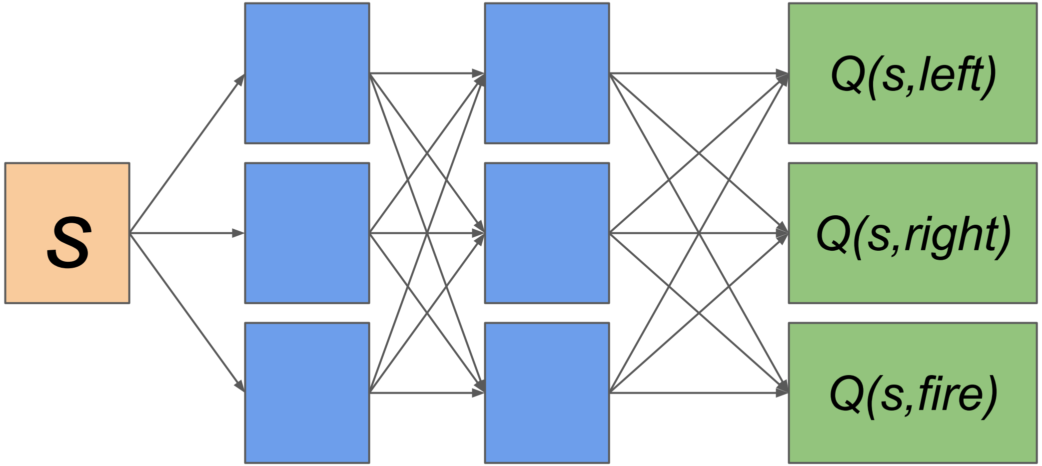
Let's add Neural Networks to Q-Learning!

Why?

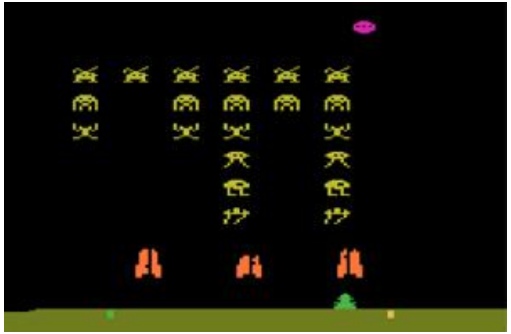
- Neural nets are universal function approximators
- They can approximate $Q(s,a)$ with a deep neural network
- They can scale to large state spaces, e.g., image pixels

DQN

Neural net that finds Q-values for every possible action



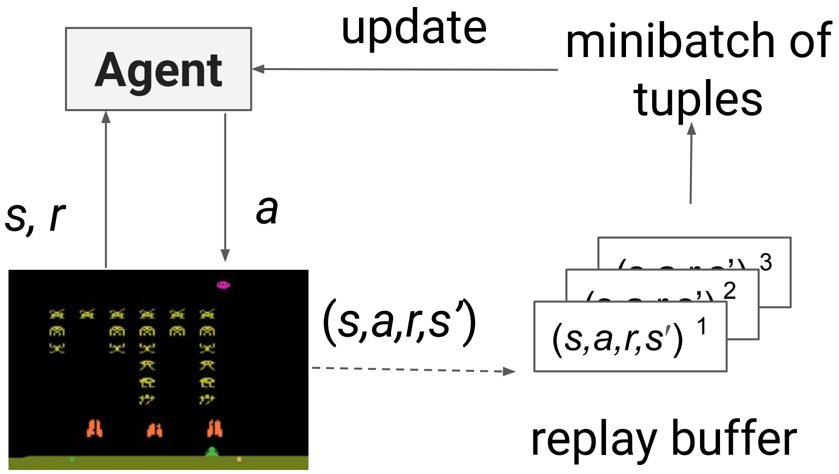
Introduced by DeepMind to play Atari games



DQN Training: Naive approach

- Use DQN Network to decide action
- Record experience and update Q
- Train DQN on new Q
- Problems:
  - DQN is unstable
  - Experience is not i.i.d.

Trick 1 of 2: Replay Buffers



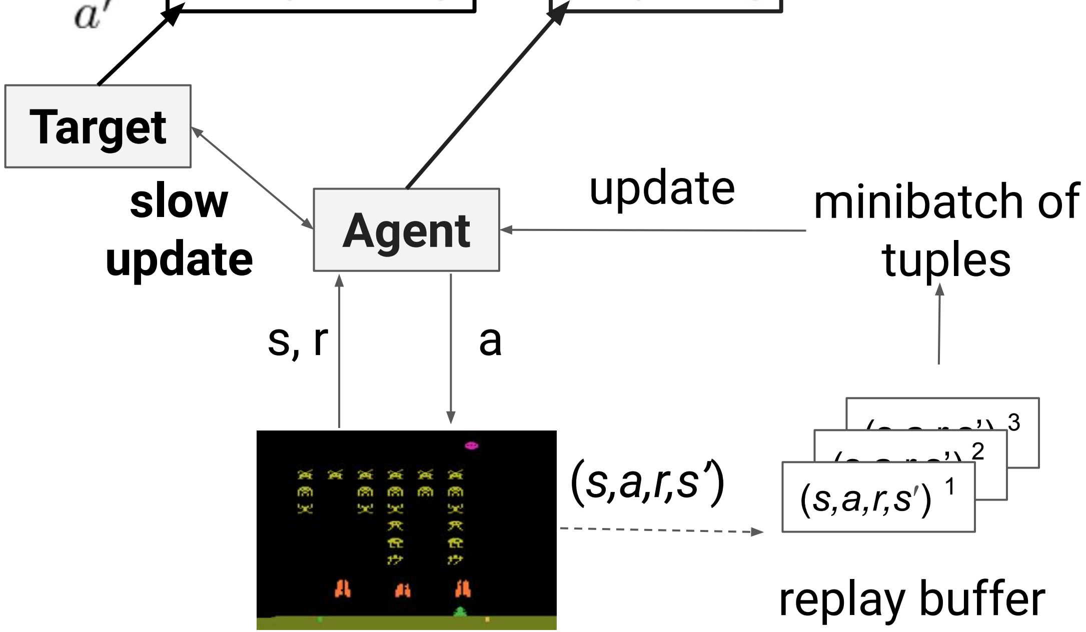
Idea:

Collect experience,
then randomly sample from it,
to avoid temporal correlation in updates

Trick 2 of 2: Target Networks

Bellman Error Gradient:

$$r + \gamma \max_{a'} \boxed{Q^T(s', a')} - \boxed{Q(s, a)}$$

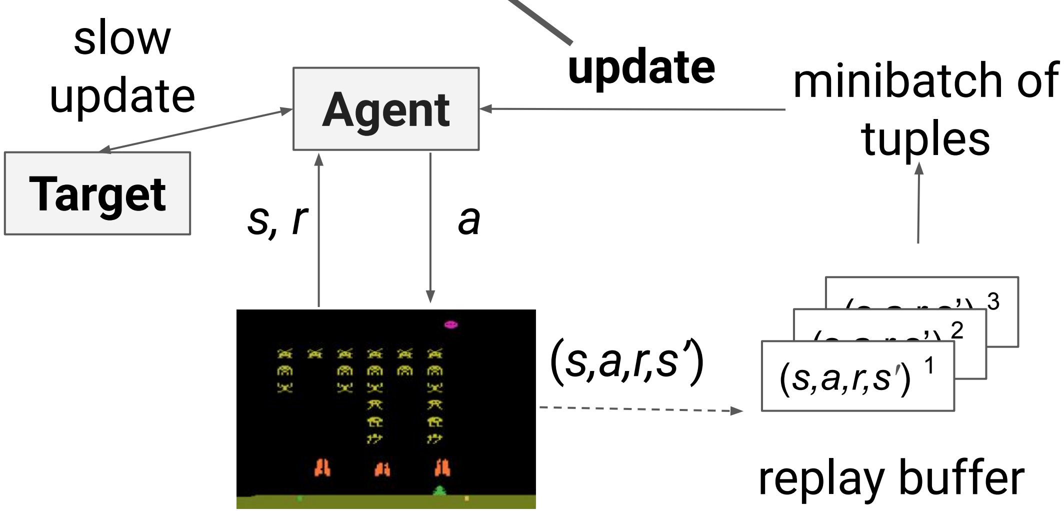


Idea:
Use separate net to compute error.
Update it slowly for more stability

Putting it together...

Minibatch SGD with experience tuples, minimizes Bellman error

Deep Q-Learning
==
Supervised Learning of Q-Network !!!



Comparison

Q-Learning:

Minimize discrepancy in Q-values of states

Terminate on reaching a "fixed-point" of Bellman equation

Supervised Learning:

Minimize deviation from training labels

Terminate on reaching a minimum of loss function

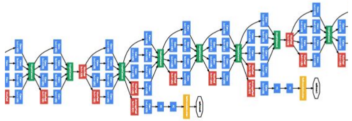
What do you need to do RL?

- Algorithm: DQN, PPO, SAC, ...
- Policy, Networks
- Environments
- Replay Buffers
- Training
- Metrics
- Bellman updates
- ...

[Python]
Environment/Task

tf.data.Dataset

Collect Policy



Replay
Buffer

Agent

Google AI

TF-Agents available in GitHub

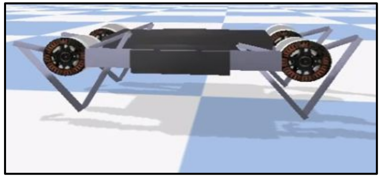
https://github.com/tensorflow/agents

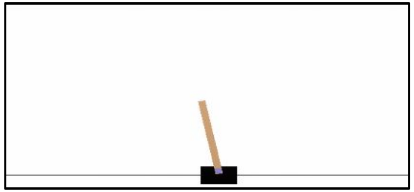
pip install tf-agents

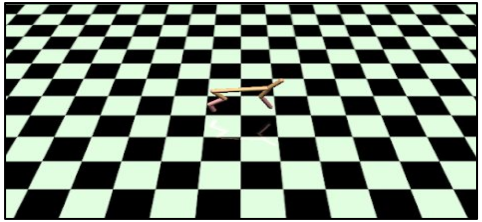
- Learn using Colabs DQN-Cartpole or SAC-Minitaur.
- Ready to solve important problems
- Contributions and PRs are welcome:
  - Environments, Algorithms, ...

Available Environment Suites

- **Gym**
- **Atari**
- Mujoco
- PyBullet
- DM-Control
- Yours?







Available Agents

- **DQN**, DDQN, DQN-RNN, **C51**
- **DDPG, TD3**
- **PPO**, PPO-RNN
- REINFORCE
- **SAC**
- Behavioral Cloning
- Contextual Bandits
- More coming soon! Yours?

Fully tested

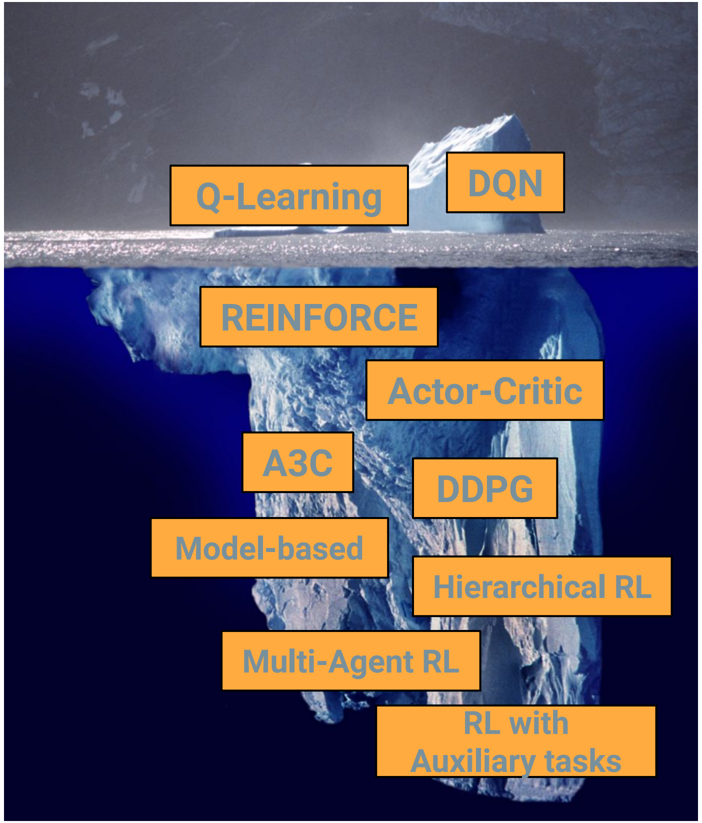
Model quality regression tests

Speed regression tests

Applications,

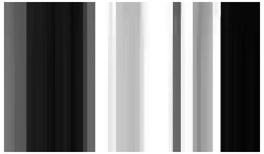
Challenges,

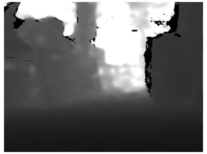
Next Steps...

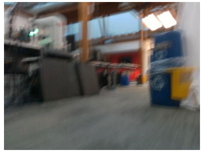


Google AI

Variety of skills on a variety of robots







MuJoCo Fetch Freight Mintaur Car Racer

Nonlinear dynamics.
High degree of freedom.
No sensors.

Differential drive.
Noisy sensors.
Inertia.

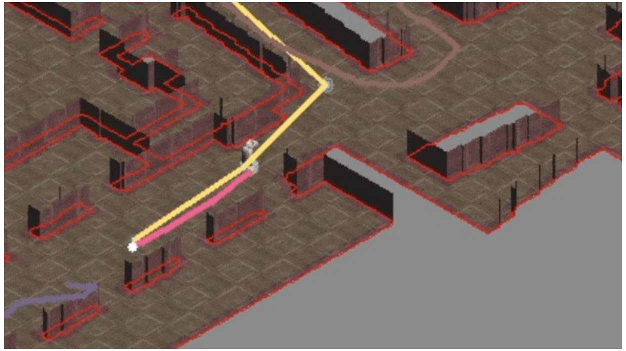
Kinodynamic constraints.
Noisy sensors.

Safe and feasible motion in dynamic world at scale.

[Lyapunov-based Safe Policy Optimization for Continuous Control,"
Chow, Nachum, Faust, Ghavamzadeh, Duenez-Guzman, under submission] [pdf, Video]
[Long-Range Indoor Navigation with PRM-RL, Francis, Faust, Chiang, Hsu, Kew, Fiser, Lee @ under submission] [pdf, Video,]
[Learning Navigation Behaviors End to End with AutoRL, Chiang*, Faust,* Fiser, Francis, RA-L/ICRA 2019] [pdf, Video]
[PRM-RL: Long-range Robotic Navigation Tasks by Combining Reinforcement Learning and Sampling-based Planning, Faust, Ramirez, Fiser, Oslund, Francis, Davidson, Tapia @ ICRA 2018)] [pdf, Video]

Google AI

Learn end-to-end motion of complex robots dynamic world



RL for safe obstacle avoidance in simple environments

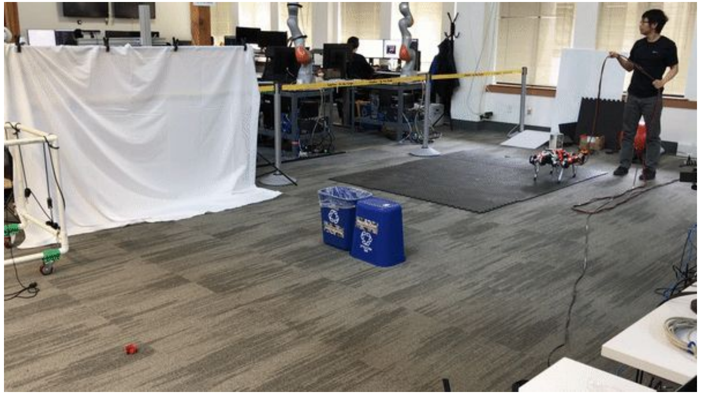
Direct transfer

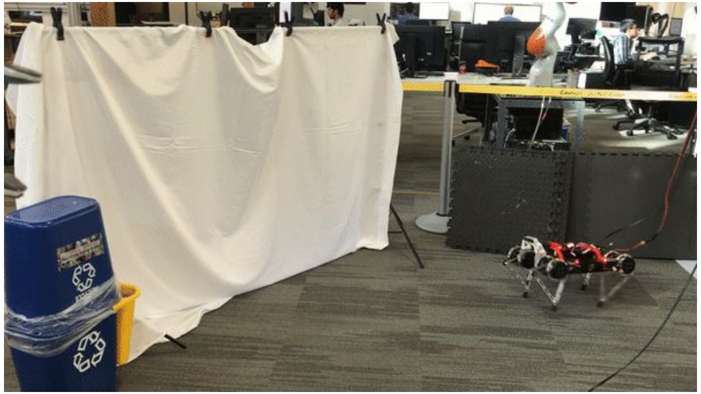
Run in real world, unseen environments
Adapt to changes on the go

Areas of interest:

- Articulated robots
- Nonlinear dynamics
- Multi-agent systems
- Task distribution
- Adaptation
- Safe RL

Learn end-to-end navigation and locomotion





- Locomotion coupled with navigation
- Navigation from depth maps
  - (and soon RGB)
- Sim2real and online training
- Navigation without localization

[On-going research, Ichter, Tan, Zhang, Faust]

Research Interns

Google Research is looking for 2021 Summer Research Interns.

https://cutt.ly/2gmkcUP

Google AI Residency Program

The Google AI Residency Program is a 12-18 months research training role designed to jumpstart or advance your career in machine learning research.

g.co/airesidency

Thanks