



Dopamine

Flexible Deep Reinforcement Learning Research

Pablo Samuel Castro

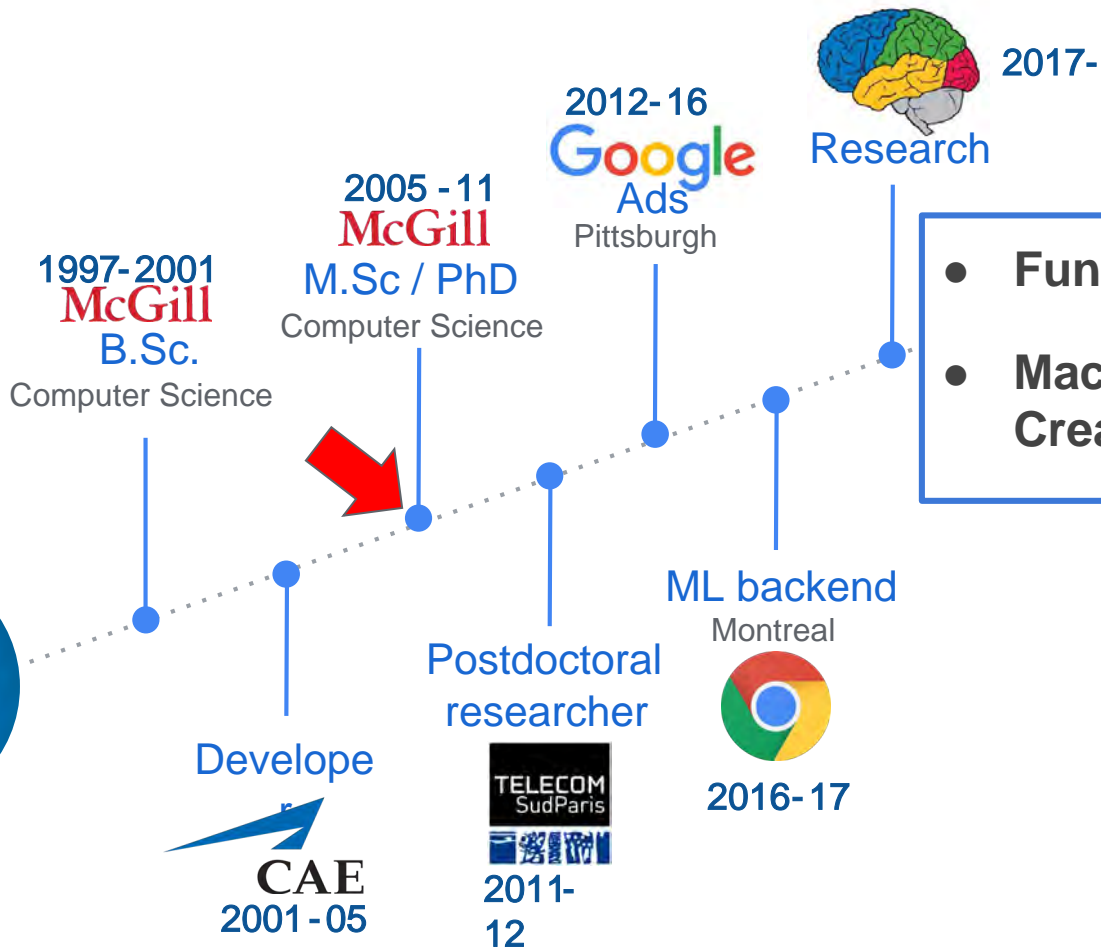
@pcastr 

señor swesearcher

Google Research, Brain Team



Pablo
Google MON



- **Fundamental RL**
- **Machine Learning + Creativity**

Machine Learning + Creativity

Combining Learned Lyrical Structures and Vocabulary for Improved Lyric Generation

Pablo Samuel Castro
Google Brain
psc@google.com

Maria Attarian
Google
jmattarian@google.com

Sparkle and Shine 2.0



Performing Structured Improvisations with Pre-existing Generative Musical Models

Pablo Samuel Castro
Google Brain
psc@google.com



Understanding distributional RL

Distributional reinforcement learning with linear function approximation



CLOUD AI INNOVATION SECURITY MORE ▾ NEWSLETTERS ALL WRITERS

MUST READ: [Microsoft publishes SECCON framework for securing Windows 10](#)

A

Google explores AI's mysterious polytope

1g

Researchers at Google Brain and DeepMind go in quest of better "representations" of the world by AI, through exploration of the polytope, a Euclidean geometric form that represents the possible solutions to a game of strategy.



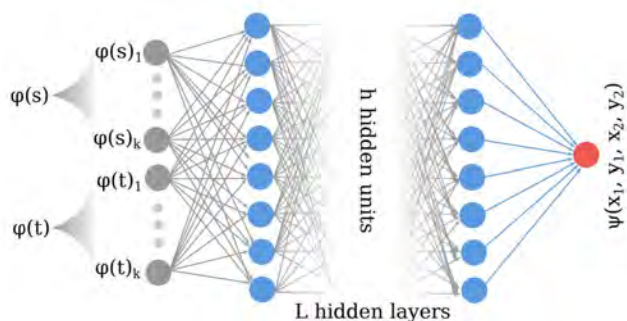
By [Tiernan Ray](#) | February 6, 2019 -- 18:46 GMT (10:46 PST) | Topic: [Artificial Intelligence](#)

A Geometric Perspective on Optimal Representations for Reinforcement Learning

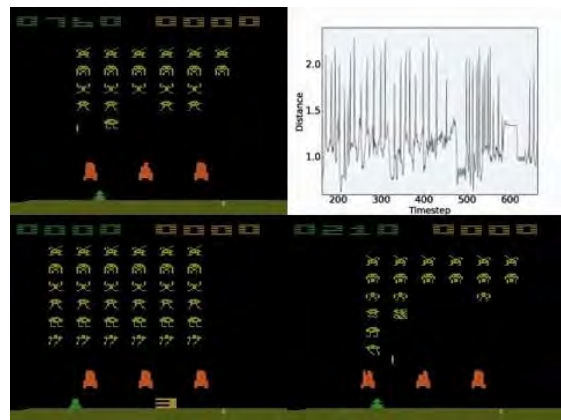
Marc G. Bellemare¹ Will Dabney² Robert Dadashi¹ Adrien Ali Taïga^{1,3} Pablo Samuel Castro¹
Nicolas Le Roux¹ Dale Schuurmans^{1,4} Tor Lattimore² Clare Lyle⁵

Bisimulation metrics for MDPs

Scalable methods for computing state similarity in deterministic Markov Decision Processes



Pablo Samuel Castro
Google Brain
psc@google.com



$$\mathcal{F}(d)(s, t) = \max_{a \in \mathcal{A}} (|\mathcal{R}(s, a) - \mathcal{R}(t, a)| + \gamma \mathcal{W}(d)(\mathcal{P}(s, a), \mathcal{P}(t, a))) \longrightarrow \max \left[|\mathcal{R}(s, a) - \mathcal{R}(t, a)| + \gamma \psi_{\theta_i^-}([\phi(\mathcal{N}(s, a)), \phi(\mathcal{N}(t, a))]), \psi_{\theta_i^-}([\phi(s), \phi(t)]) \right]$$

Reinforcement Learning

Decision making under uncertainty

Typically formulated as a **Markov Decision Process (MDP)** :

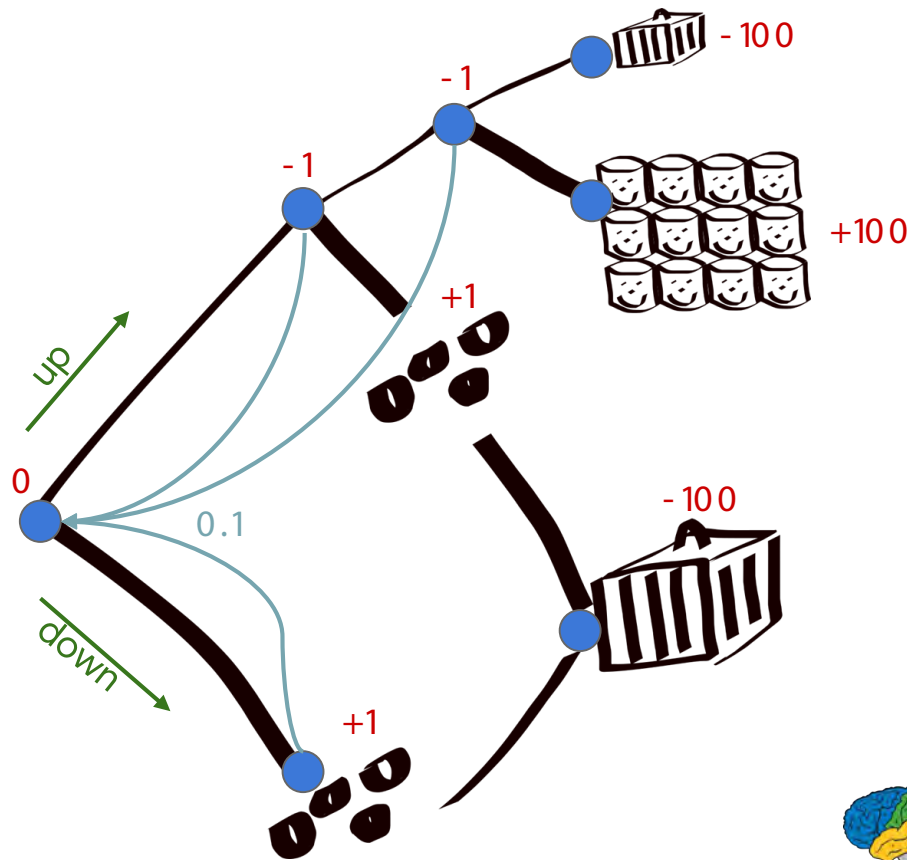
States

Actions

Rewards

Transition dynamics

Discount factor γ



$$V^*(s) = \max_a [R(s, a) + \gamma \sum_{s'} P(s'|s, a) V^*(s')]$$



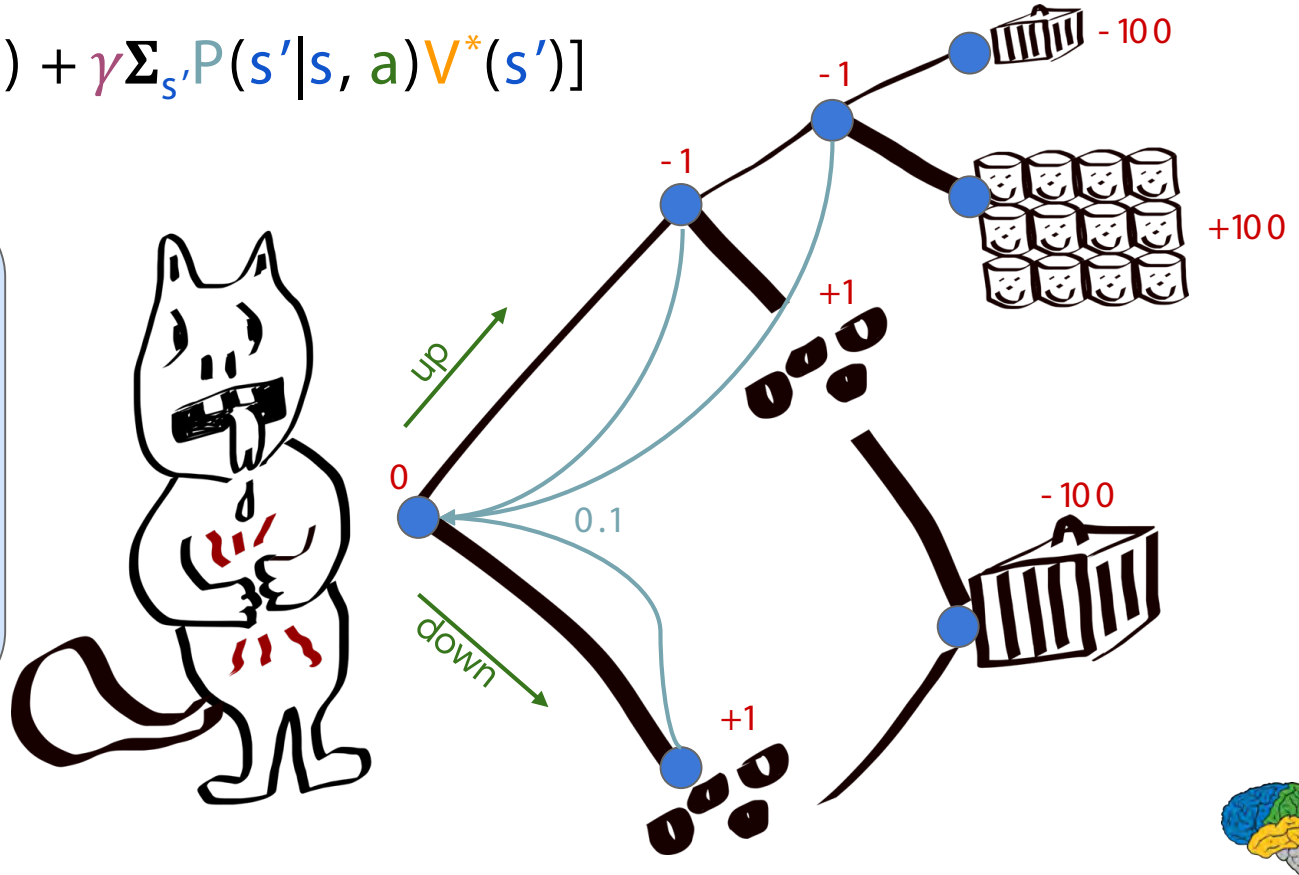
Value Iteration

$$V^*(s) = \max_a [R(s, a) + \gamma \sum_{s'} P(s'|s, a) V^*(s')]$$

Richard E. Bellman

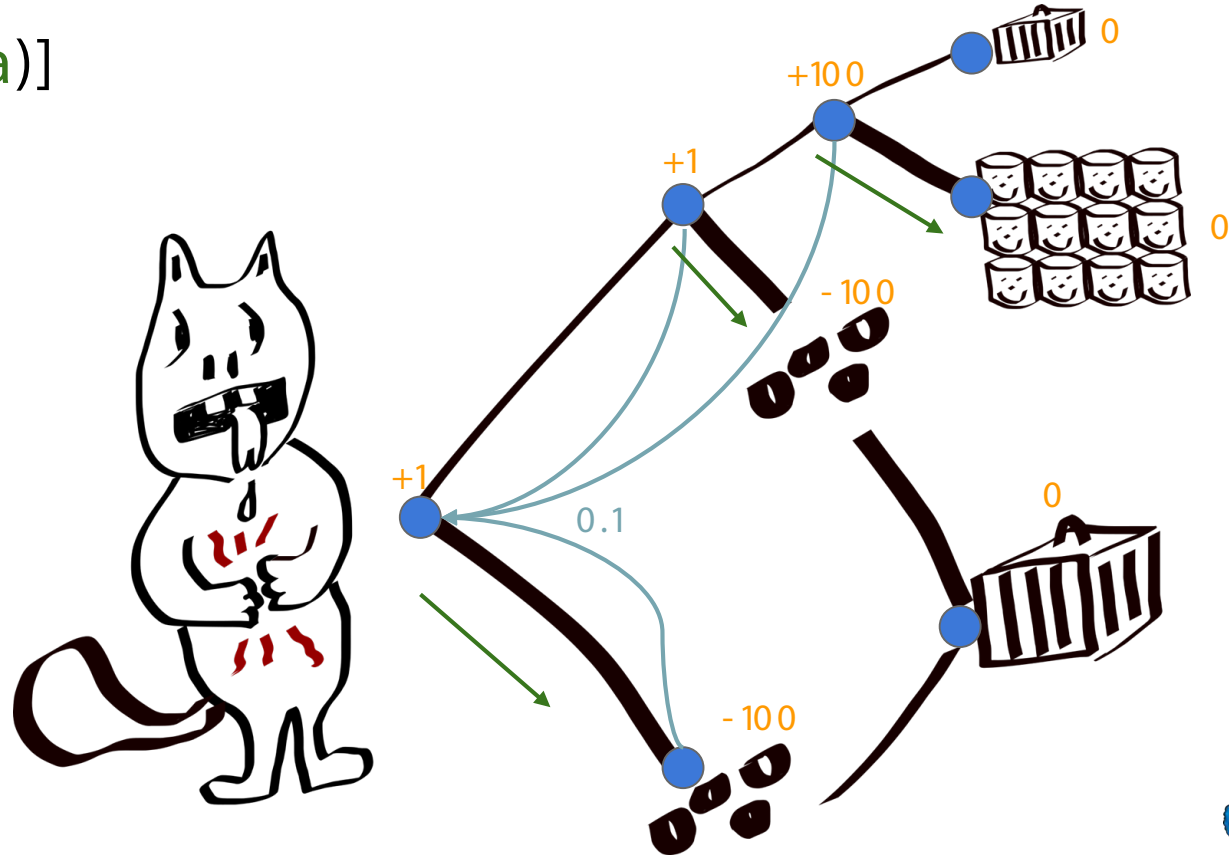


Dynamic programming!
1957



Value Iteration

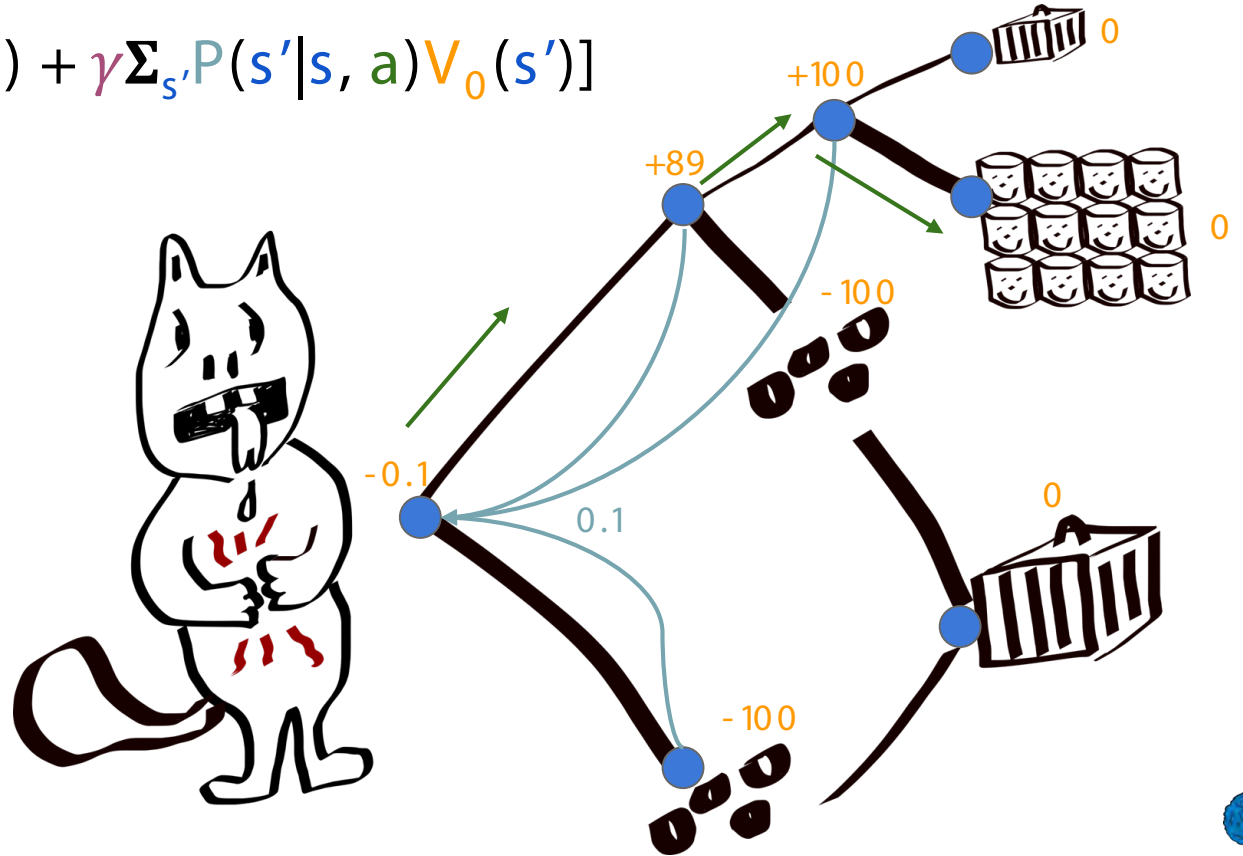
$$V_0(s) = \max_a [R(s, a)]$$



Value Iteration

$$V_1(s) = \max_a [R(s, a) + \gamma \sum_{s'} P(s'|s, a) V_0(s')]$$

$$\gamma = 0.9$$

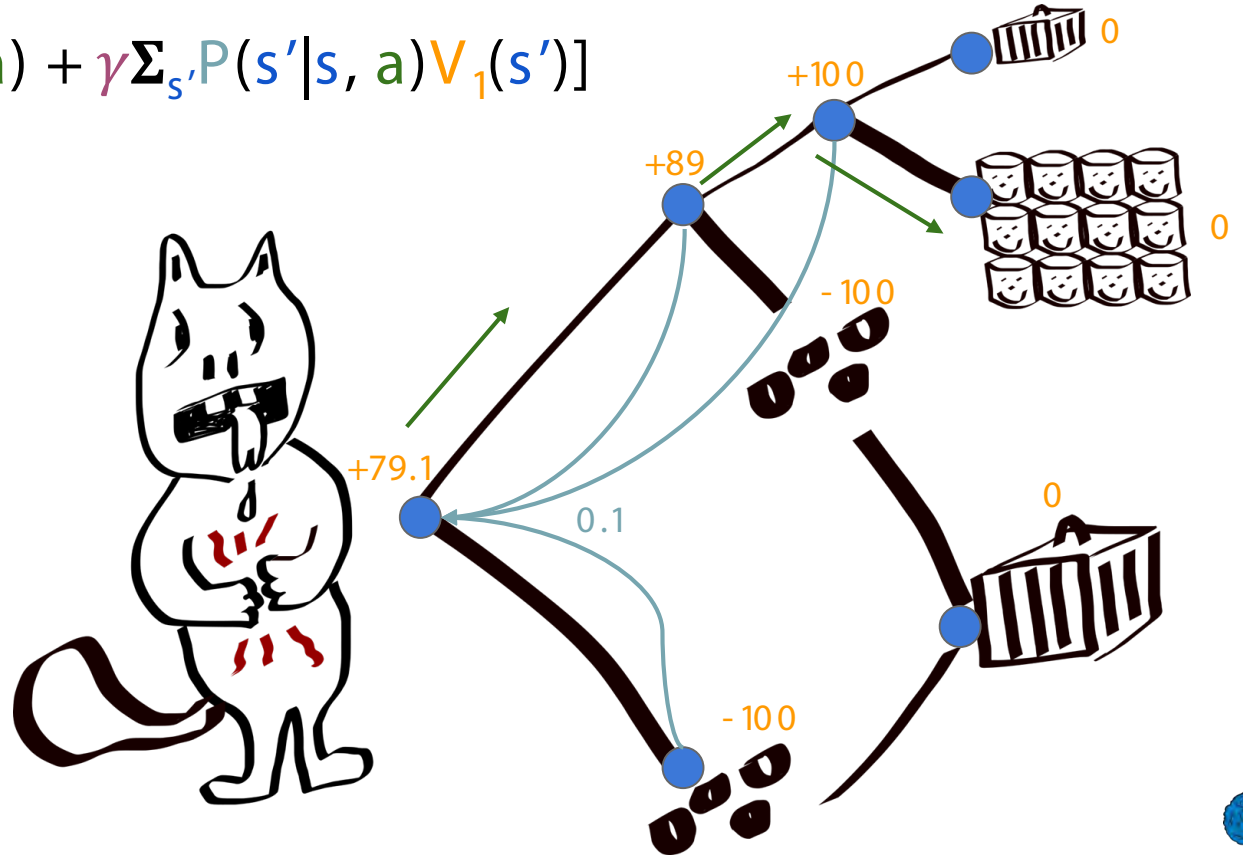


Value Iteration

$$V_2(s) = \max_a [R(s, a) + \gamma \sum_{s'} P(s'|s, a) V_1(s')]$$

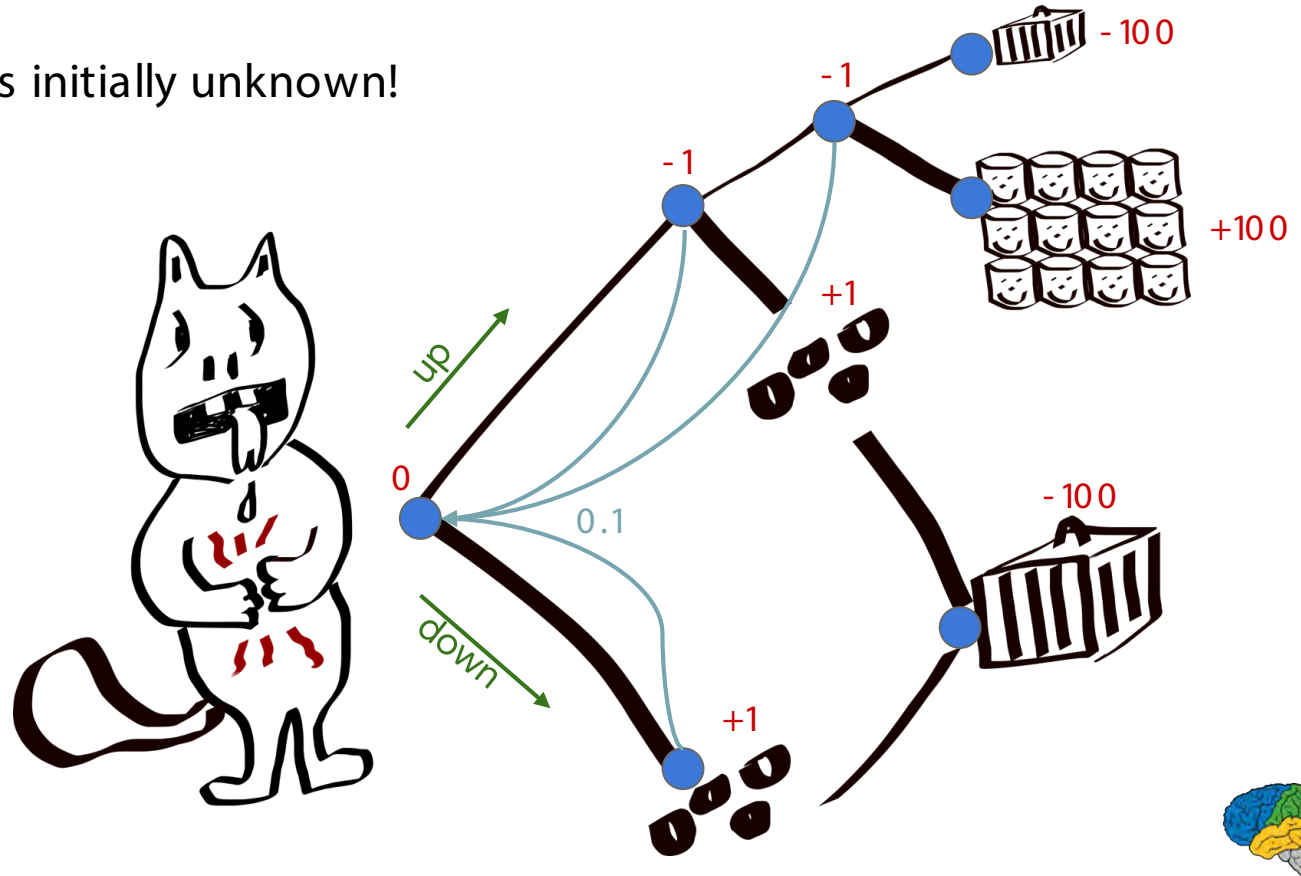
$$\gamma = 0.9$$

$$V^* = V_2$$



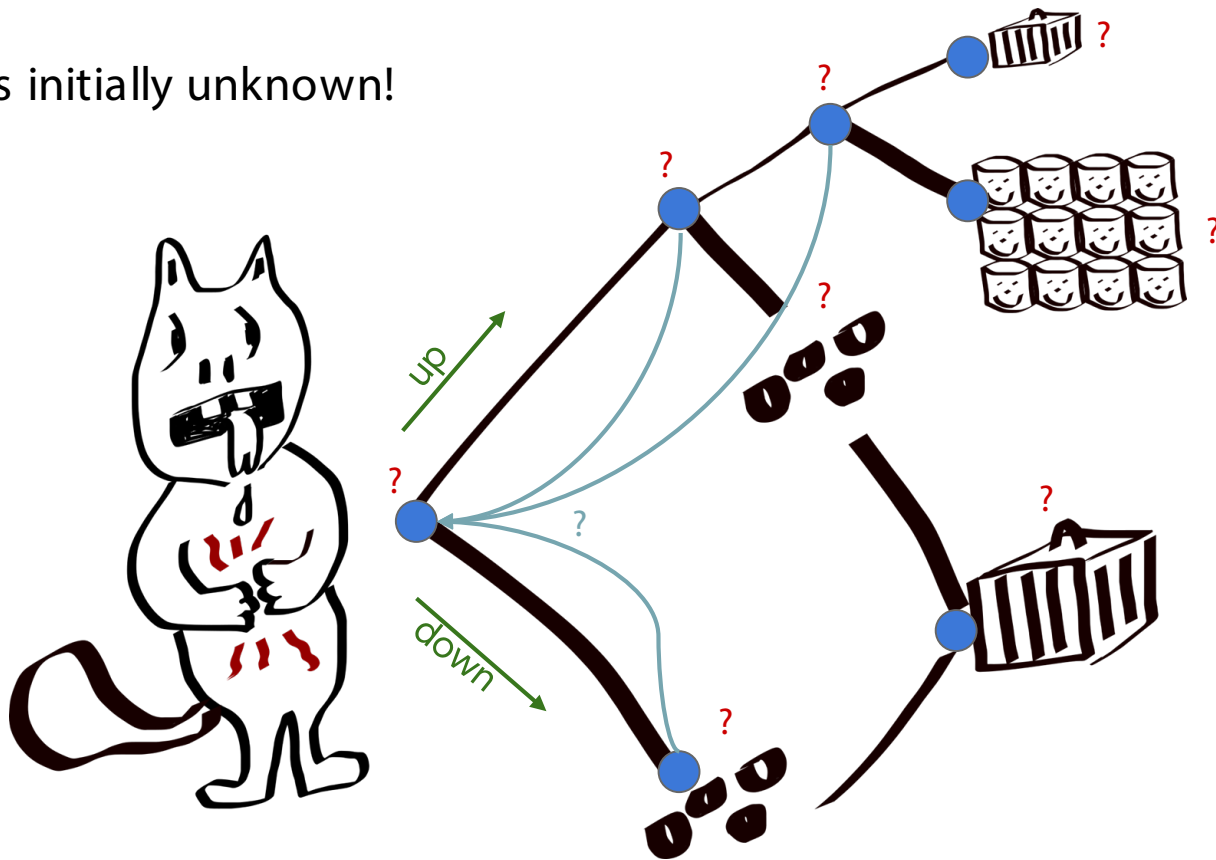
Reinforcement Learning

Rewards and dynamics initially unknown!



Reinforcement Learning

Rewards and dynamics initially unknown!



Reinforcement Learning

Rewards and dynamics initially unknown!

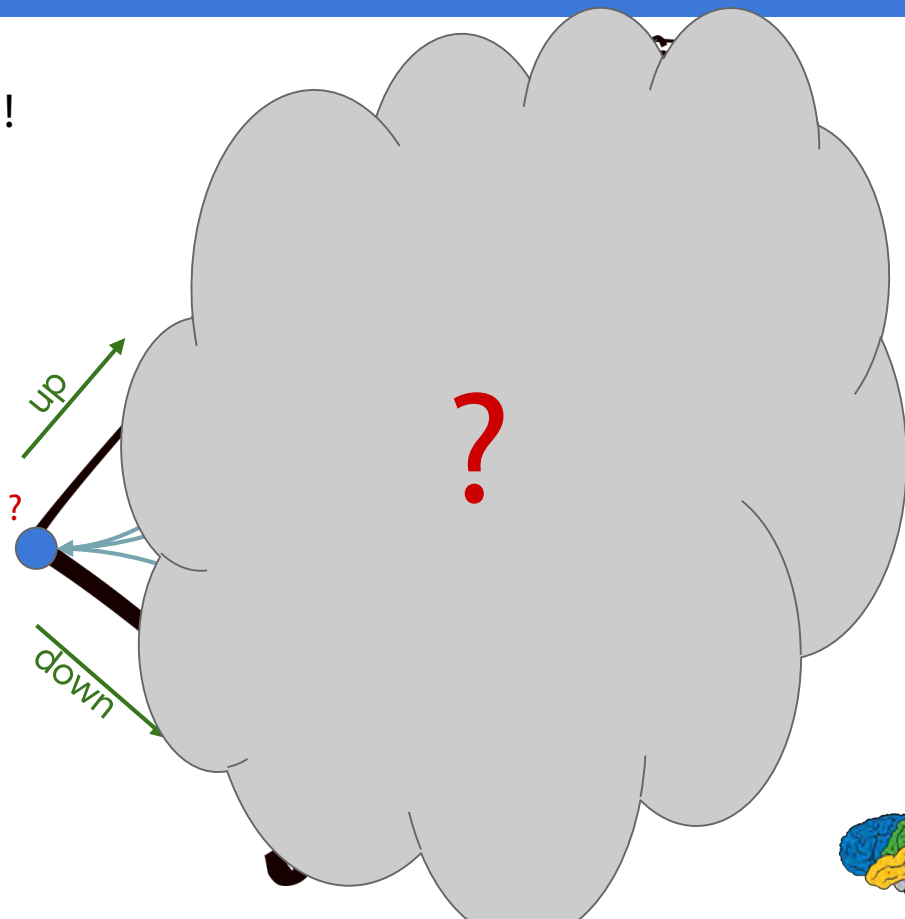
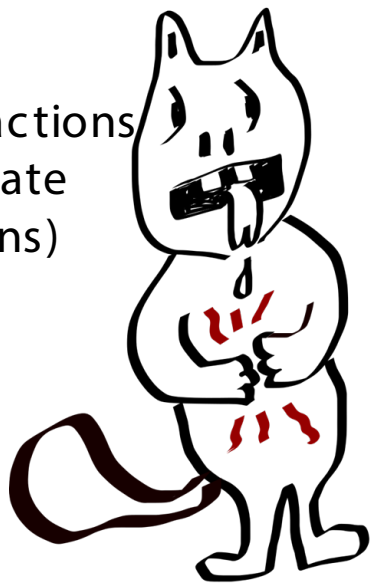
A few ways to handle this.

Maintain and update a

- Model estimate

- Policy for choosing actions

- Value function estimate
(use to choose actions)



Reinforcement Learning

Common difficulties

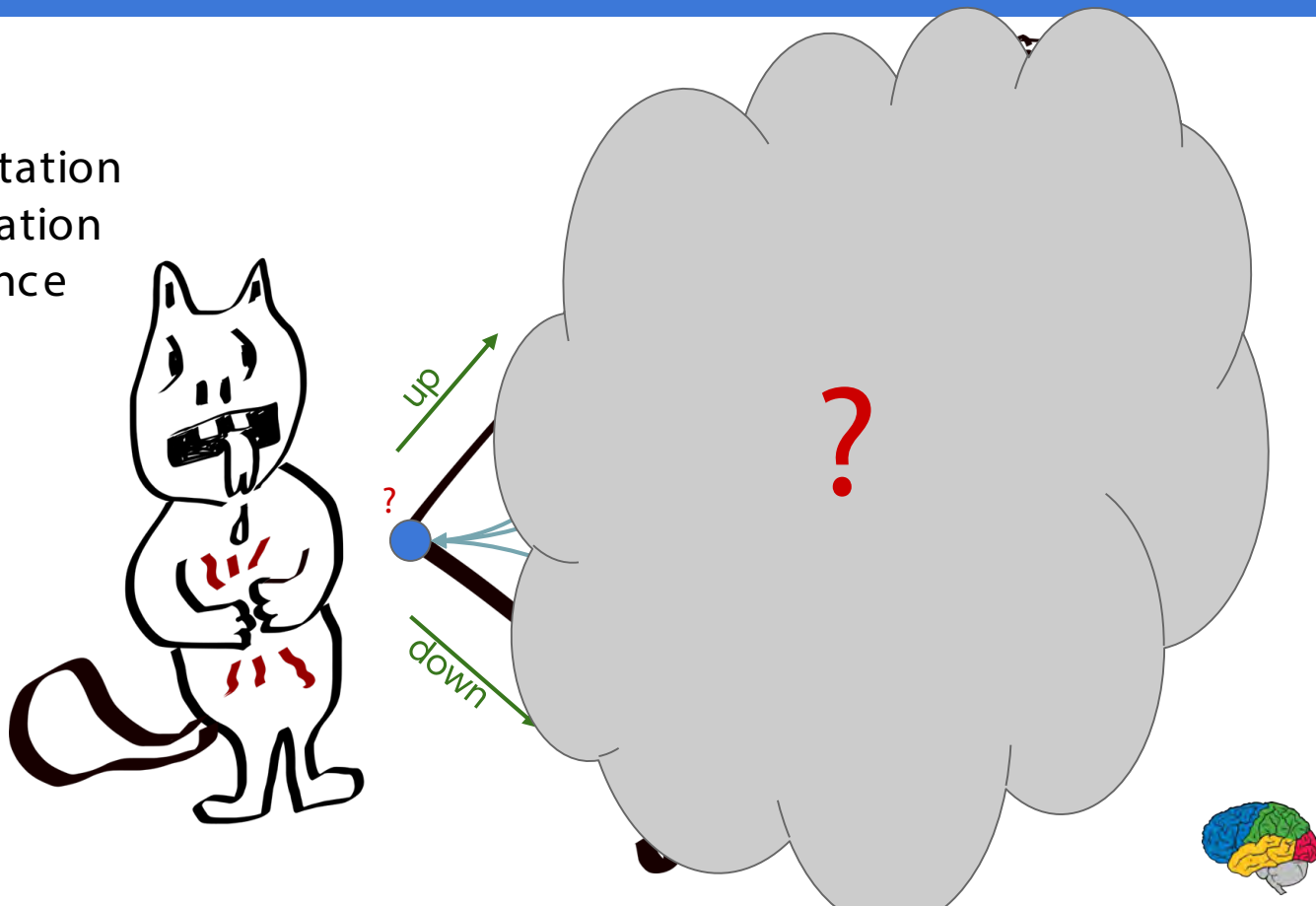
Exploration / exploitation

Function approximation

Off-policy divergence

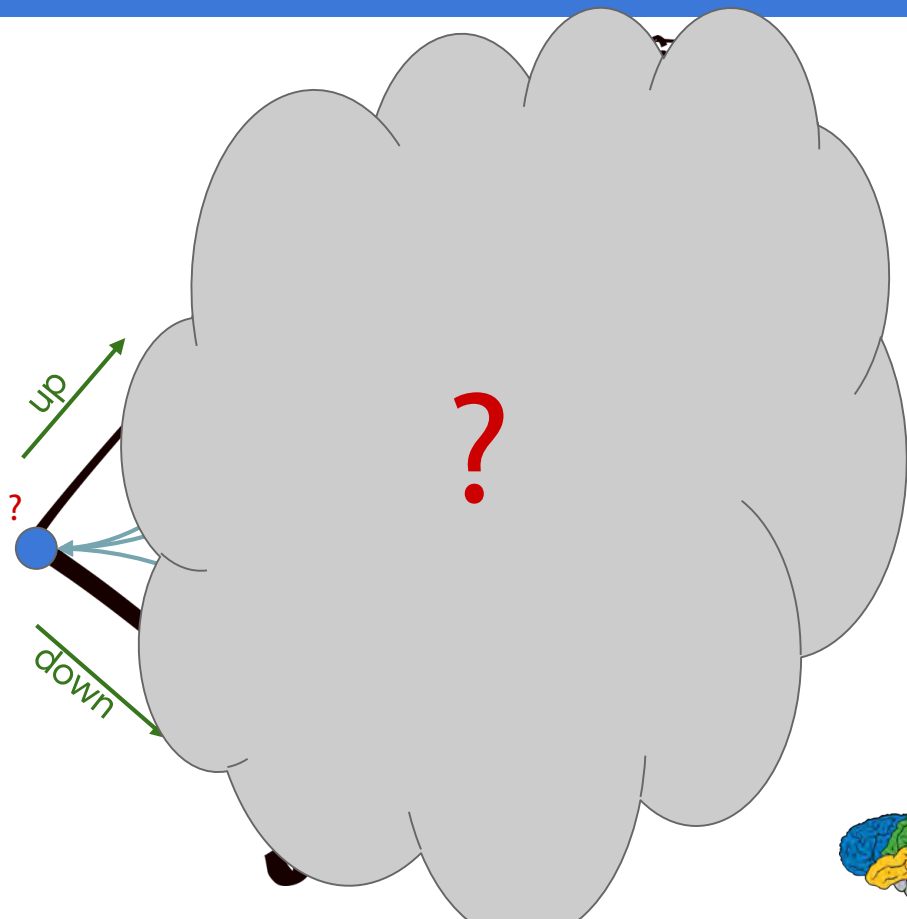
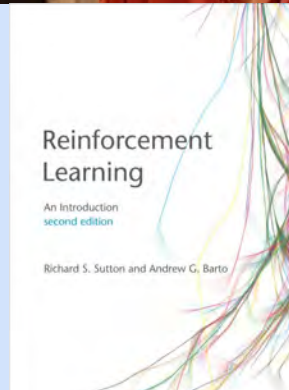
Sample efficiency

...

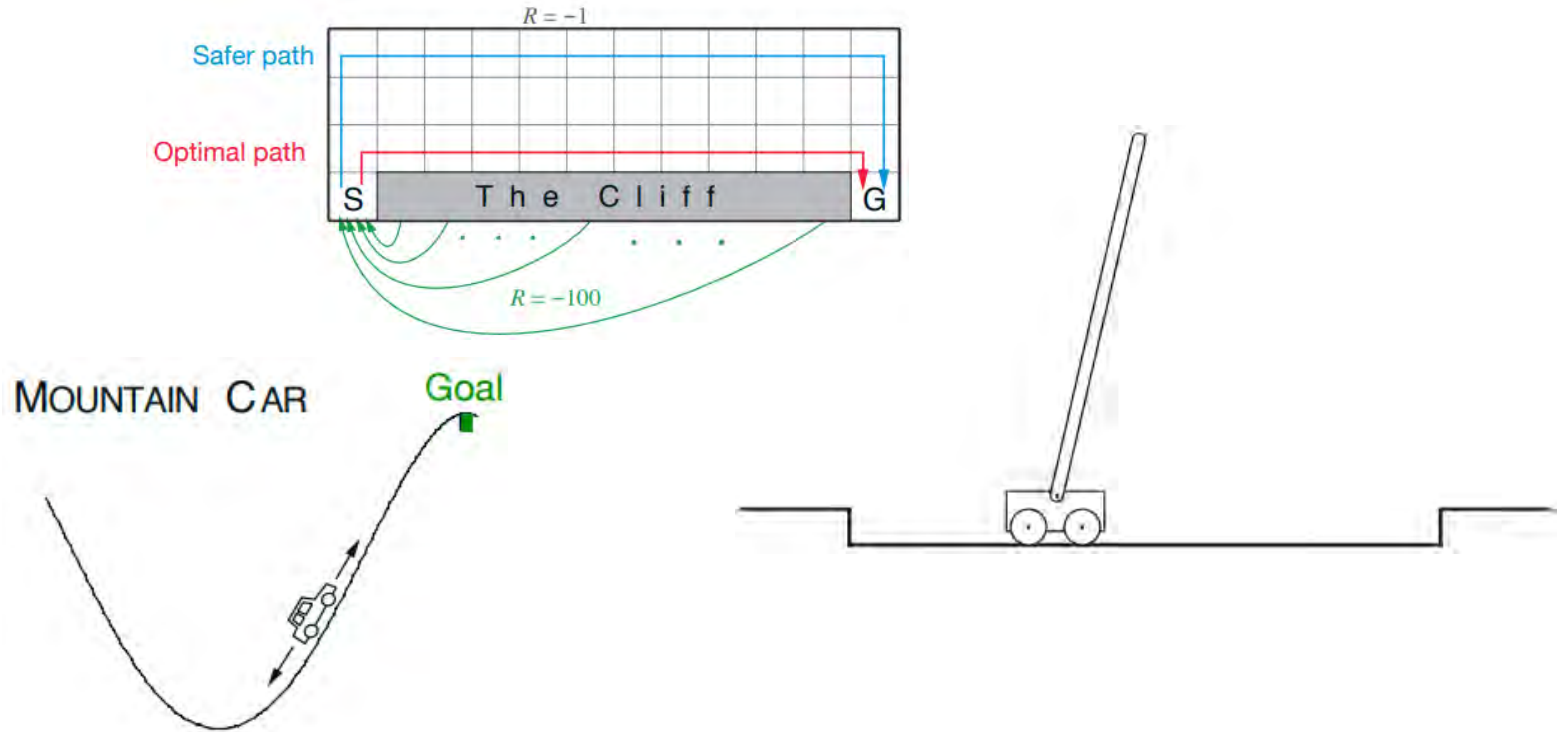


Reinforcement Learning

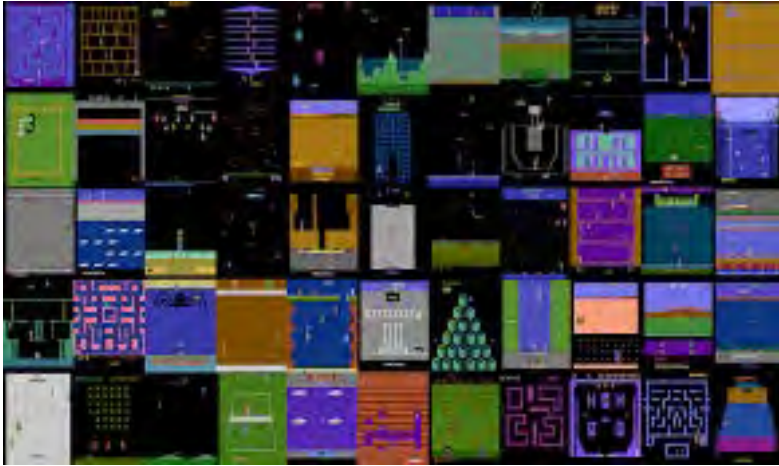
Reinforcement Learning
Sutton & Barto
1998, 2018



Environments



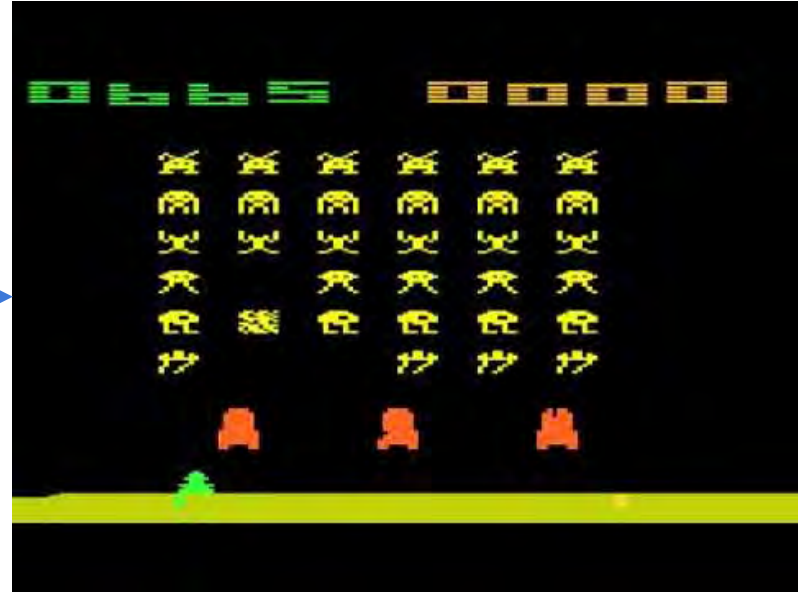
Arcade Learning Environment



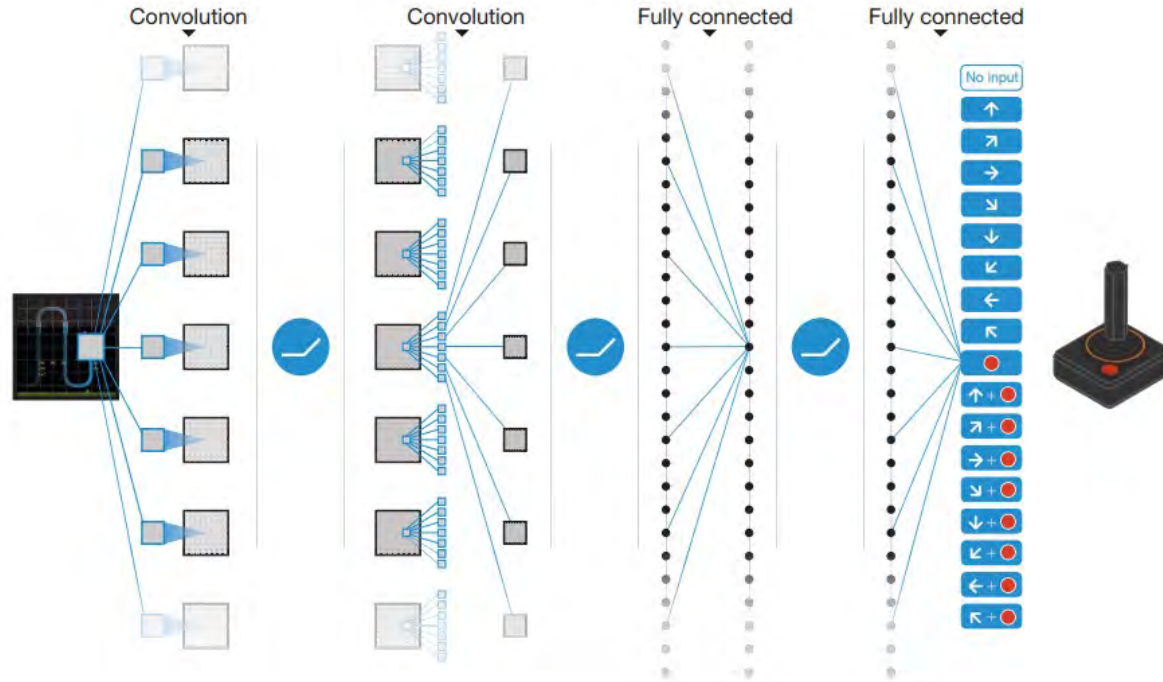
Bellemare et al., (2012) -
<https://arxiv.org/abs/1207.4708>



Arcade Learning Environment



DQN



Mnih et al. (2015, Nature) - Human-level control through deep reinforcement learning



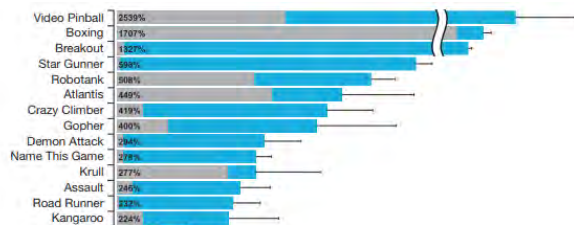
DQN



Mnih et al. (2015, Nature) - Human-level control through deep reinforcement learning



DQN

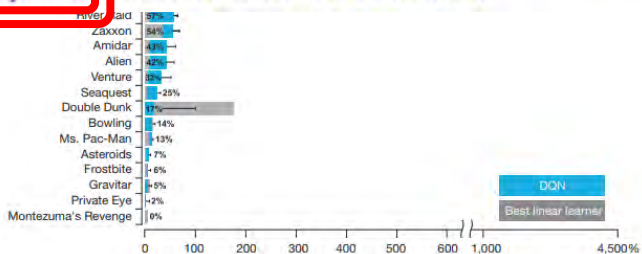


Human-level control through deep reinforcement learning

[V Mnih](#), [K Kavukcuoglu](#), [D Silver](#), [AA Rusu](#), [J Veness](#)... - Nature, 2015 - nature.com

The theory of reinforcement learning provides a normative account 1, deeply rooted in psychological 2 and neuroscientific 3 perspectives on animal behaviour, of how agents may optimize their control of an environment. To use reinforcement learning successfully in situations approaching real-world complexity, however, agents are confronted with a difficult task: they must derive efficient representations of the environment from high-dimensional sensory input, and use these to generalize past experience to new situations. Remarkably ...

☆ 59 **Cited by 5215** Related articles All 41 versions

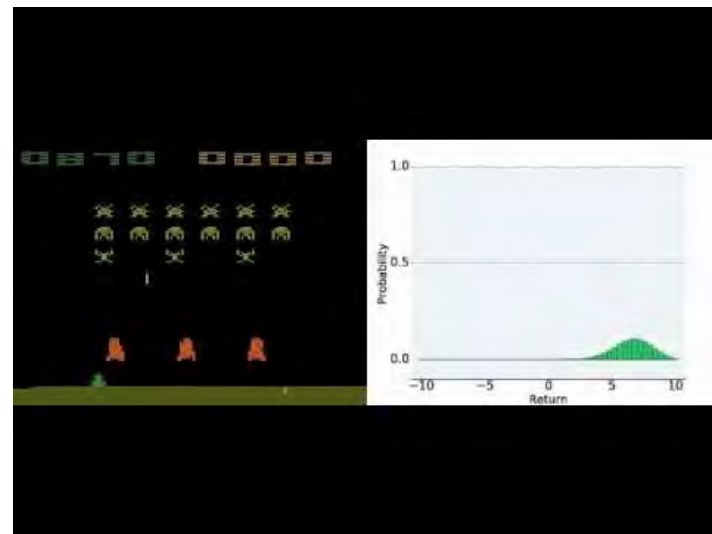
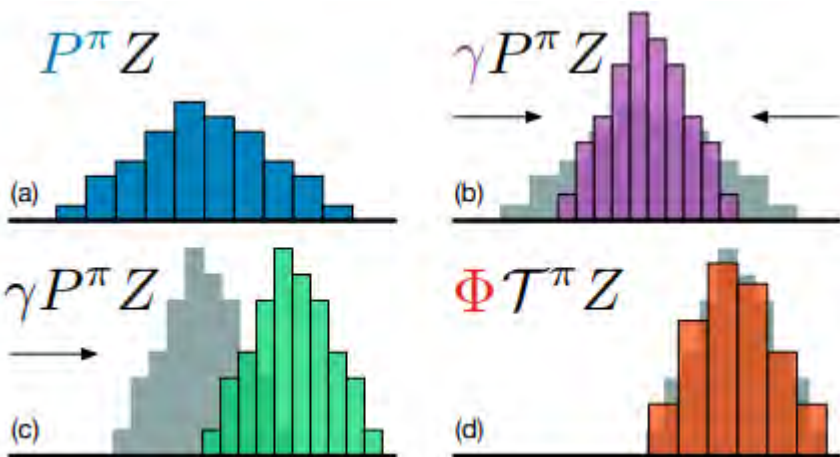


Mnih et al. (2015, Nature) - Human-level control through deep reinforcement learning



Distributional RL

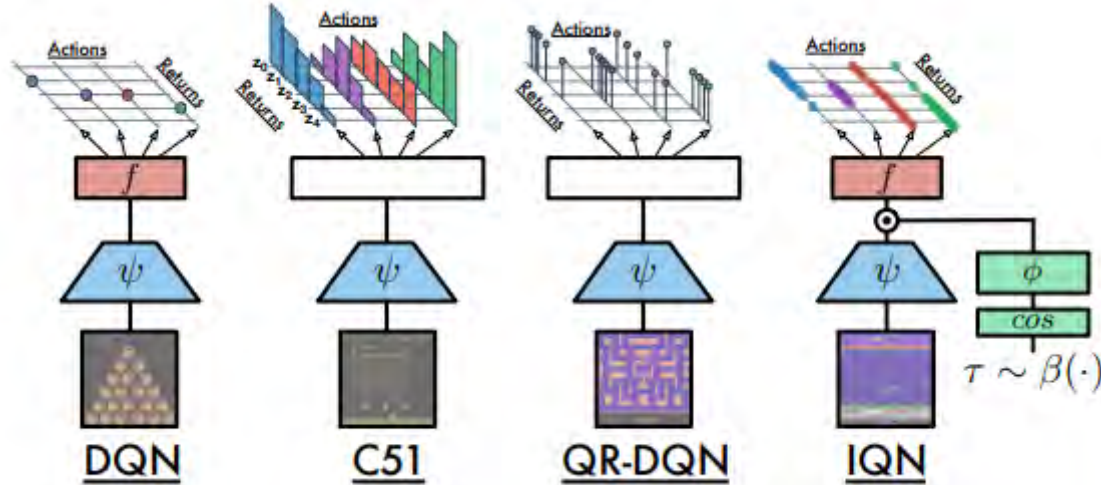
$$Q(x, a) = \mathbb{E} R(x, a) + \gamma \mathbb{E} Q(X', A') \longrightarrow Z(x, a) \stackrel{D}{=} R(x, a) + \gamma Z(X', A')$$



C51



Distributional RL



Rainbow: Hessel et al. (2018) - <https://arxiv.org/abs/1710.02298>

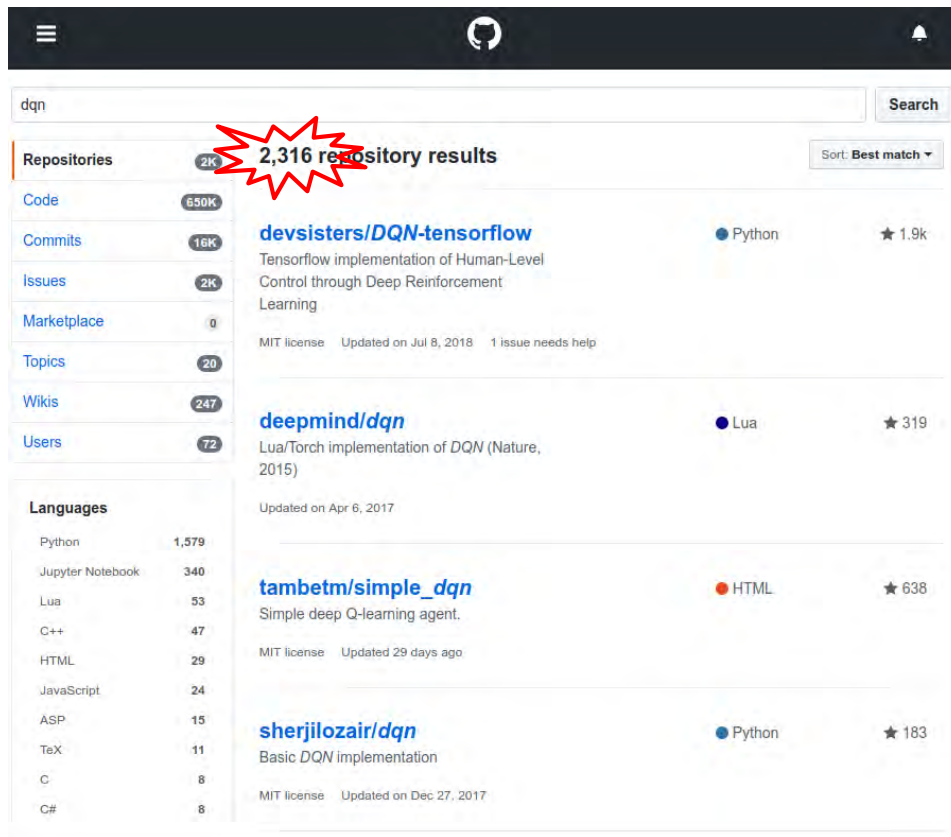
IQN: Dabney et al. (2018) - <https://arxiv.org/abs/1806.06923>



Let's do some research!
What's a good DQN implementation to start from?



DQN implementations



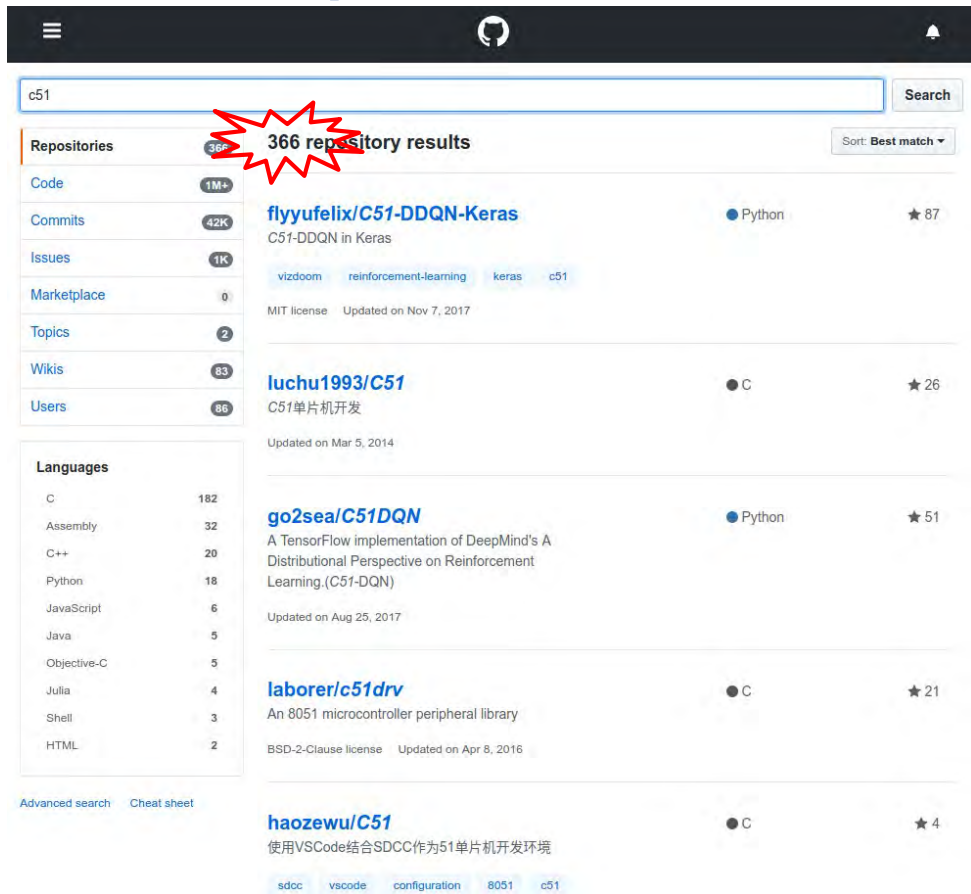
The screenshot shows the GitHub search interface for the query 'dqn'. A red starburst graphic highlights the text '2,316 repository results' in the top left of the results area. The interface includes a search bar at the top, a sidebar with filters for Repositories, Code, Commits, Issues, Marketplace, Topics, Wikis, and Users, and a list of repository results. The results list shows the top three repositories: 'devsisters/DQN-tensorflow', 'deepmind/dqn', and 'tambetm/simple_dqn'. Each result includes the repository name, a brief description, the programming language, and the number of stars.

Repository	Description	Language	Stars
devsisters/DQN-tensorflow	Tensorflow implementation of Human-Level Control through Deep Reinforcement Learning	Python	1.9k
deepmind/dqn	Lua/Torch implementation of DQN (Nature, 2015)	Lua	319
tambetm/simple_dqn	Simple deep Q-learning agent.	HTML	638

Languages

Language	Count
Python	1,579
Jupyter Notebook	340
Lua	53
C++	47
HTML	29
JavaScript	24
ASP	15
TeX	11
C	8
C#	8

C51 implementations



The screenshot shows the GitHub search interface for the query 'c51'. The search bar at the top contains 'c51' and a 'Search' button. Below the search bar, a red starburst graphic highlights the text '366 repository results'. The left sidebar shows filters for Repositories (366), Code (1M+), Commits (42K), Issues (1K), Marketplace (0), Topics (2), Wikis (83), and Users (86). The main content area displays a list of repositories. The first repository is 'flyyufelix/C51-DDQN-Keras' by flyyufelix, a Python implementation of C51-DDQN in Keras, with 87 stars. The second is 'luchu1993/C51' by luchu1993, a C implementation for C51 microcontroller development, with 26 stars. The third is 'go2sea/C51DQN' by go2sea, a TensorFlow implementation of DeepMind's A Distributional Perspective on Reinforcement Learning (C51-DQN) in Python, with 51 stars. The fourth is 'laborer/c51drv' by laborer, a C implementation of an 8051 microcontroller peripheral library, with 21 stars. The fifth is 'haozewu/C51' by haozewu, a C implementation using VSCode and SDCC for 51 microcontroller development, with 4 stars. A 'Languages' sidebar on the left lists various programming languages and their counts: C (182), Assembly (32), C++ (20), Python (18), JavaScript (6), Java (5), Objective-C (5), Julia (4), Shell (3), and HTML (2). At the bottom, there are links for 'Advanced search' and 'Cheat sheet'.

Search results for **c51** (366 repository results)

Sort: Best match

Repository	Language	Stars
flyyufelix/C51-DDQN-Keras C51-DDQN in Keras vizdoom reinforcement-learning keras c51 MIT license Updated on Nov 7, 2017	Python	87
luchu1993/C51 C51单片机开发 Updated on Mar 5, 2014	C	26
go2sea/C51DQN A TensorFlow implementation of DeepMind's A Distributional Perspective on Reinforcement Learning.(C51-DQN) Updated on Aug 25, 2017	Python	51
laborer/c51drv An 8051 microcontroller peripheral library BSD-2-Clause license Updated on Apr 8, 2016	C	21
haozewu/C51 使用VSCode结合SDCC作为51单片机开发环境 sdoc vscode configuration 8051 c51	C	4

Advanced search Cheat sheet

Let's do some research!
What's a good DQN implementation to start from?

...



Let's do some research!
What's a good DQN implementation to start from?

...

Anything internal and reliable?



Let's do some research!
What's a good DQN implementation to start from?

...

Anything internal and reliable?

There's a team building a bunch of RL algorithms, I
could add it there.



Let's do some research!
What's a good DQN implementation to start from?

...

Anything internal and reliable?

There's a team building a bunch of RL algorithms, I
could add it there.

I'd like to add a counter for the number of backups
performed and print it to terminal occasionally.
Can we do that?





Let's do some research!
What's a good DQN implementation to start from?

...

Anything internal and reliable?

There's a team building a bunch of RL algorithms, I
could add it there.

I'd like to add a counter for the number of backups
performed and print it to terminal occasionally.
Can we do that?

Sure, just create this **tf.placeholder** , add these
chained **tf.control_dependencies** and
tf.variable_scope (make sure they're in the right
order!), then pass it through a **tf.print** on an op that
you know will be executed, and you're good to go!



We need our own thing.
Let's build it!

Dopamine



github.com/google/dopamine

Core team



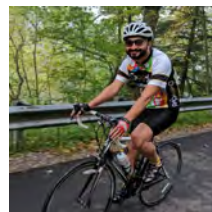
Pablo
Samuel
Castro



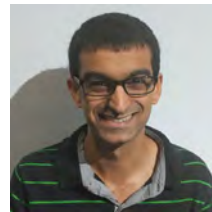
Marc G.
Bellemare



Carles
Gelada



Subhodeep
Moitra

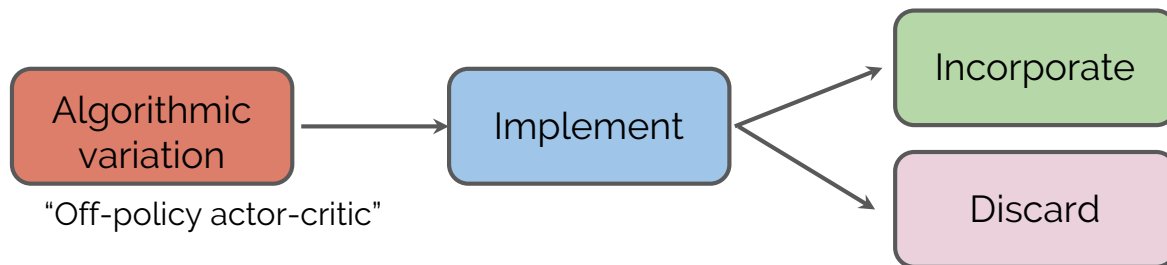


Saurabh Kumar

And many other contributors.

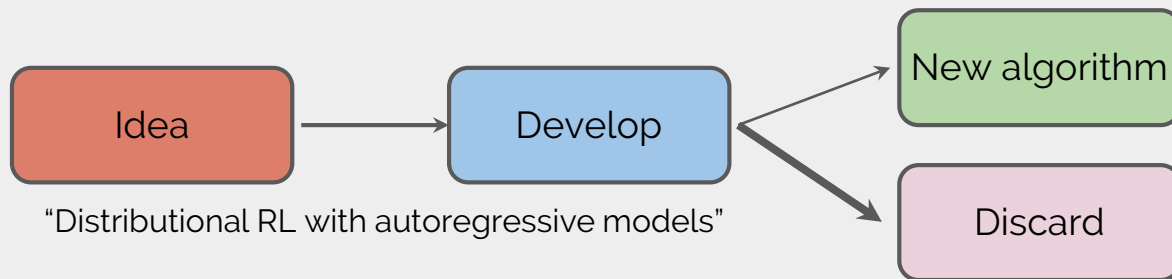


"Double DQN leads to more conservative policies in fast-paced Atari games"



"Off-policy actor-critic"

Dopamine



"Distributional RL with autoregressive models"

Desiderata

Built for researchers

Fast prototyping

Easily experiment with wild ideas

Simple is beautiful

Our approach: Focus

Small (but well-tested) codebase

Single environment (ALE)

Value-based agents

Code Specifics

14 python files (excluding tests)

Around 2000 lines in total

98% code coverage

4 agents:

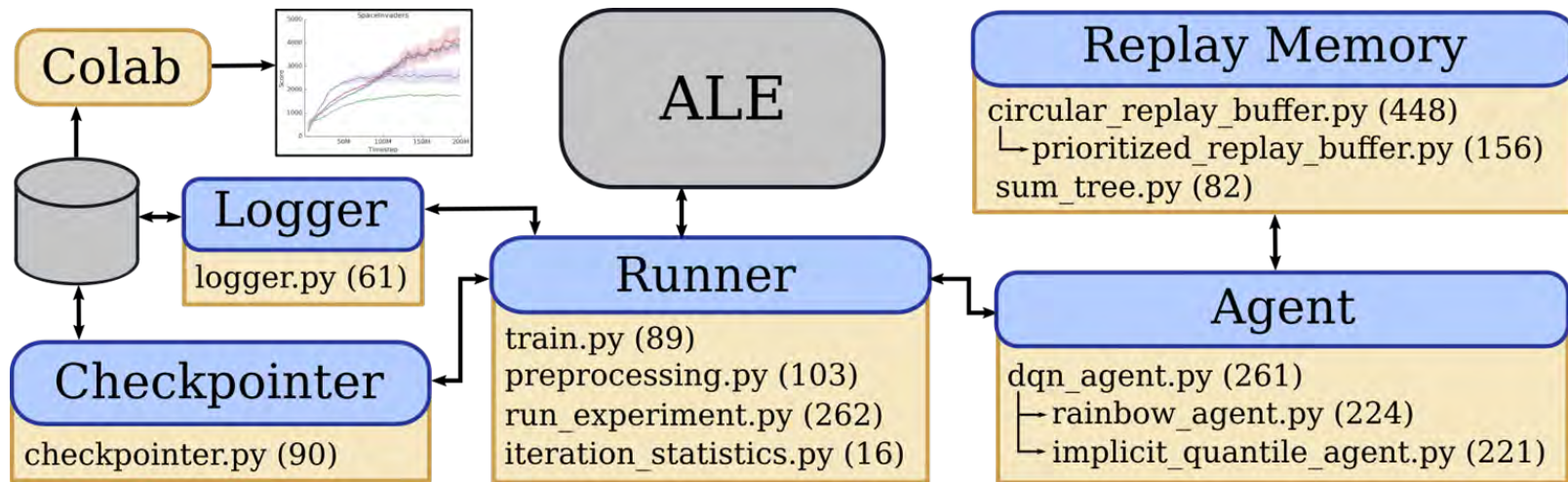
DQN

C51

Rainbow

IQN

Code Design



New agent based on DQN

[illegible]

New agent from scratch

```
class StickyAgent(object):
    """This agent randomly selects an action and sticks to it. It will change
    actions with probability switch_prob."""
    def __init__(self, sess, num_actions, switch_prob=0.1):
        self.sess = sess
        self.num_actions = num_actions
        self.switch_prob = switch_prob
        self.last_action = np.random.randint(num_actions)
        self.eval_mode = False

    def _choose_action(self):
        if np.random.random() <= self.switch_prob:
            self.last_action = np.random.randint(self.num_actions)
        return self.last_action

    def bundle_and_checkpoint(self, unused_checkpoint_dir, unused_iteration):
        pass

    def unbundle(self, unused_checkpoint_dir, unused_checkpoint_version,
                 unused_data):
        pass

    def begin_episode(self, unused_observation):
        return self._choose_action()

    def end_episode(self, unused_reward):
        pass

    def step(self, reward, observation):
        return self._choose_action()

def create_sticky_agent(sess, environment, summary_writer=None):
    """The Runner class will expect a function of this type to create an agent."""
    return StickyAgent(sess, num_actions=environment.action_space.n,
                       switch_prob=0.2)

sticky_runner = run_experiment.Runner(LOG_PATH,
```

But what about
reproducibility?

Deep Reinforcement Learning that Matters

**Peter Henderson^{1*}, Riashat Islam^{1,2*}, Philip Bachman²
Joelle Pineau¹, Doina Precup¹, David Meger¹**

¹ McGill University, Montreal, Canada

² Microsoft Maluuba, Montreal, Canada

{peter.henderson, riashat.islam}@mail.mcgill.ca, phbachma@microsoft.com
{jpineau, dprecup}@cs.mcgill.ca, dmeger@cim.mcgill.ca

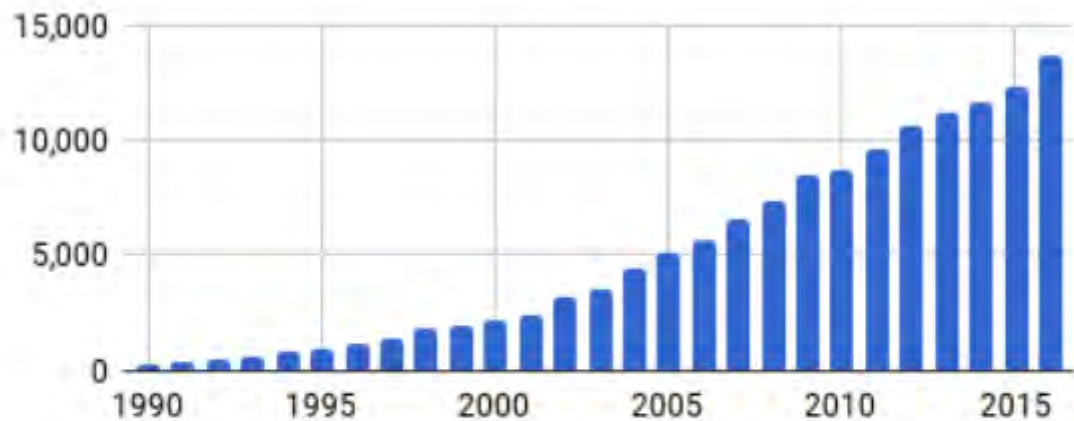


Figure 1: Growth of published reinforcement learning papers. Shown are the number of RL-related publications (y-axis) per year (x-axis) scraped from Google Scholar searches.

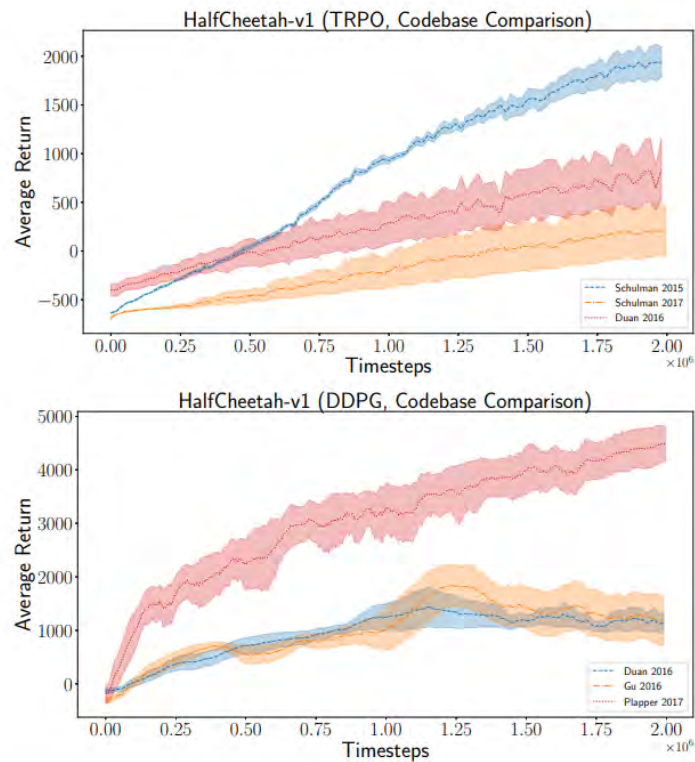


Figure 6: TRPO codebase comparison using our default set of hyperparameters (as used in other experiments).

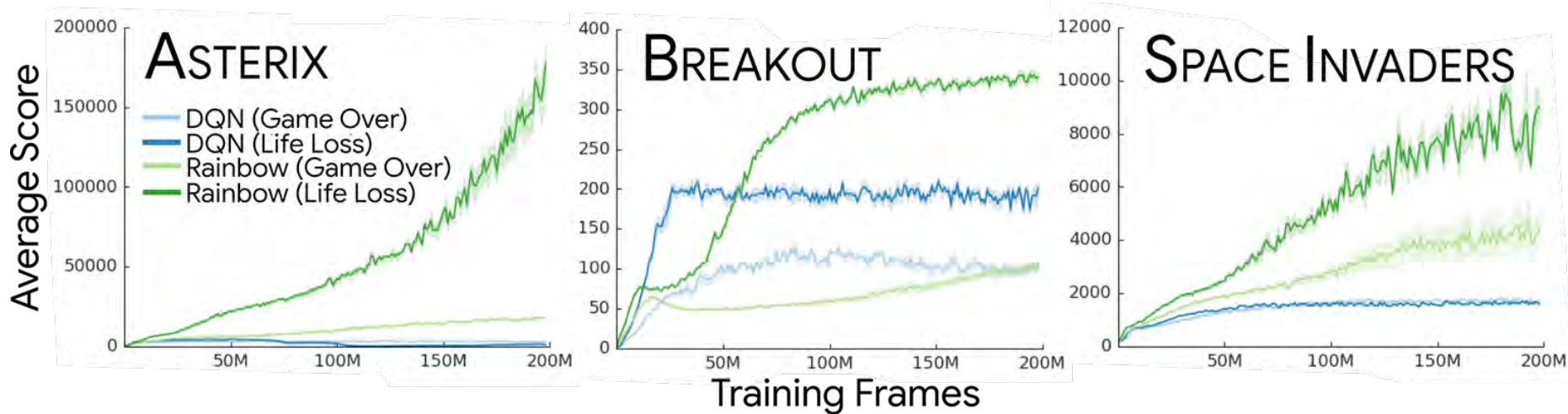
How can Dopamine help
with reproducibility?

Hyperparameter sanity: gin-config

```
1 DQNAgent.gamma = 0.99
2 DQNAgent.update_horizon = 1
3 DQNAgent.min_replay_history = 20000 # agent steps
4 DQNAgent.update_period = 4
5 DQNAgent.target_update_period = 8000 # agent steps
6 DQNAgent.epsilon_train = 0.01
7 DQNAgent.epsilon_eval = 0.001
8 DQNAgent.epsilon_decay_period = 250000 # agent steps
9 DQNAgent.tf_device = '/gpu:0' # use '/cpu:0' for non-GPU version
10 DQNAgent.optimizer = @tf.train.RMSPropOptimizer()
11
12 tf.train.RMSPropOptimizer.learning_rate = 0.00025
13 tf.train.RMSPropOptimizer.decay = 0.95
14 tf.train.RMSPropOptimizer.momentum = 0.0
15 tf.train.RMSPropOptimizer.epsilon = 0.00001
16 tf.train.RMSPropOptimizer.centered = True
17
18 Runner.game_name = 'Pong'
19 Runner.sticky_actions = True
20 Runner.num_iterations = 200
21 Runner.training_steps = 250000 # agent steps
22 Runner.evaluation_steps = 125000 # agent steps
23 Runner.max_steps_per_episode = 27000 # agent steps
24
25 WrappedReplayBuffer.replay_capacity = 1000000
26 WrappedReplayBuffer.batch_size = 32
```

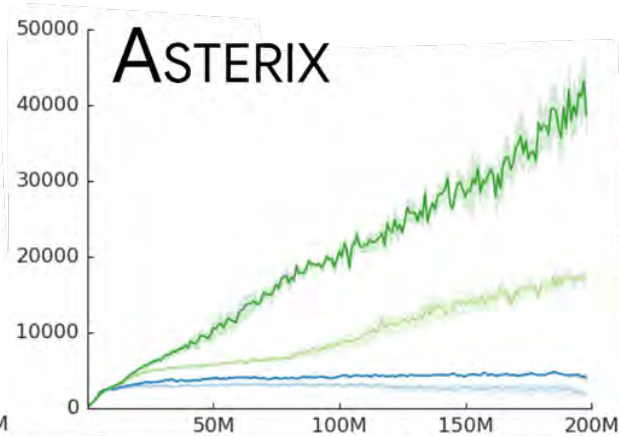
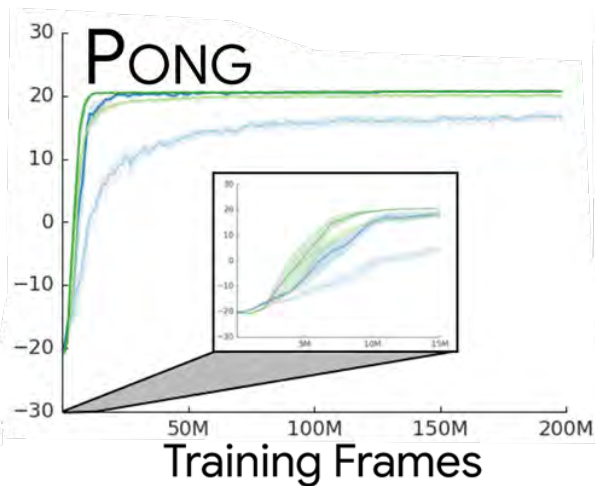
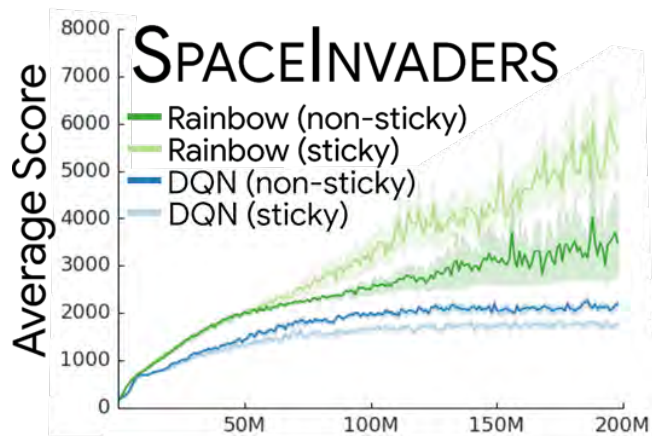

Episode ends: LifeLoss vs GameOver

```
AtariPreprocessing.terminal_on_life_loss = True
```



Sticky vs non-sticky actions

$$A_t = \begin{cases} a, & \text{with prob. } 1 - \zeta, \\ a_{t-1}, & \text{with prob. } \zeta. \end{cases}$$



```
Runner.sticky_actions = False
```

Stable baselines

We ran 5 independent runs for all 4 agents on all 60 Atari games

We provide:

- TensorFlow checkpoints for each of these runs

- pickle files to easily visualize in colab (or anywhere)

- Tensorboard event files

- JSON files with data for plotting

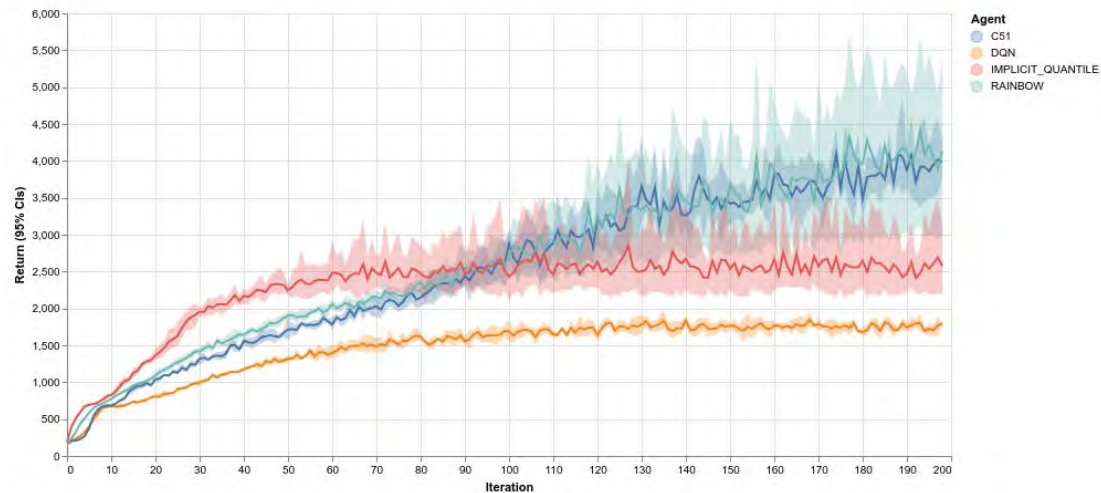
Colabs for extra documentation and instruction

Baselines plots



spaceinvaders ▼

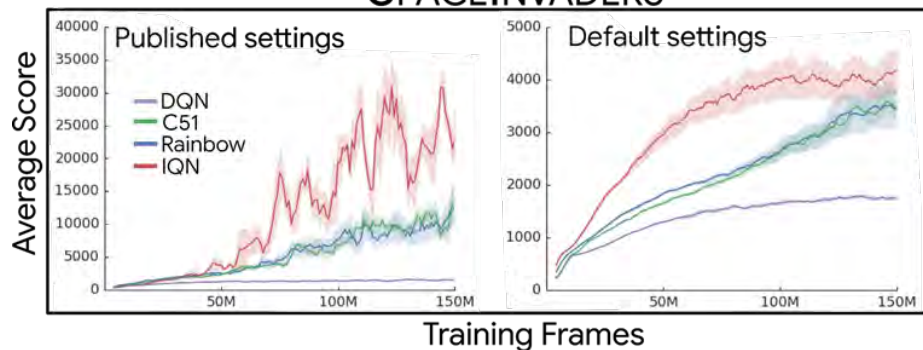
spaceinvaders



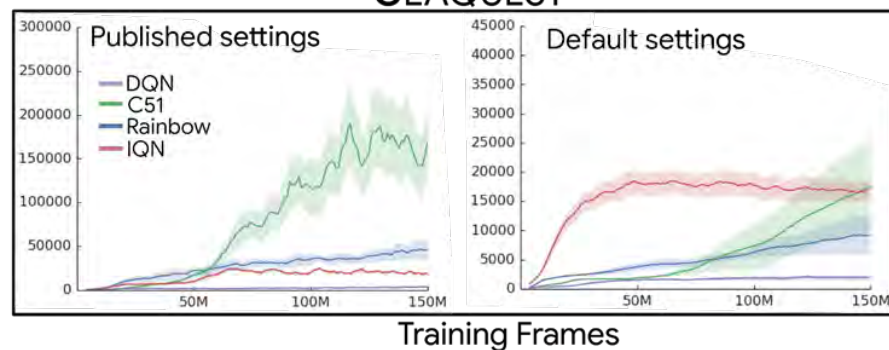
[Save as SVG](#) [Save as PNG](#) [View Source](#) [Open in Vega Editor](#)

Comparison with published settings

SPACE INVADERS



SEAQUEST



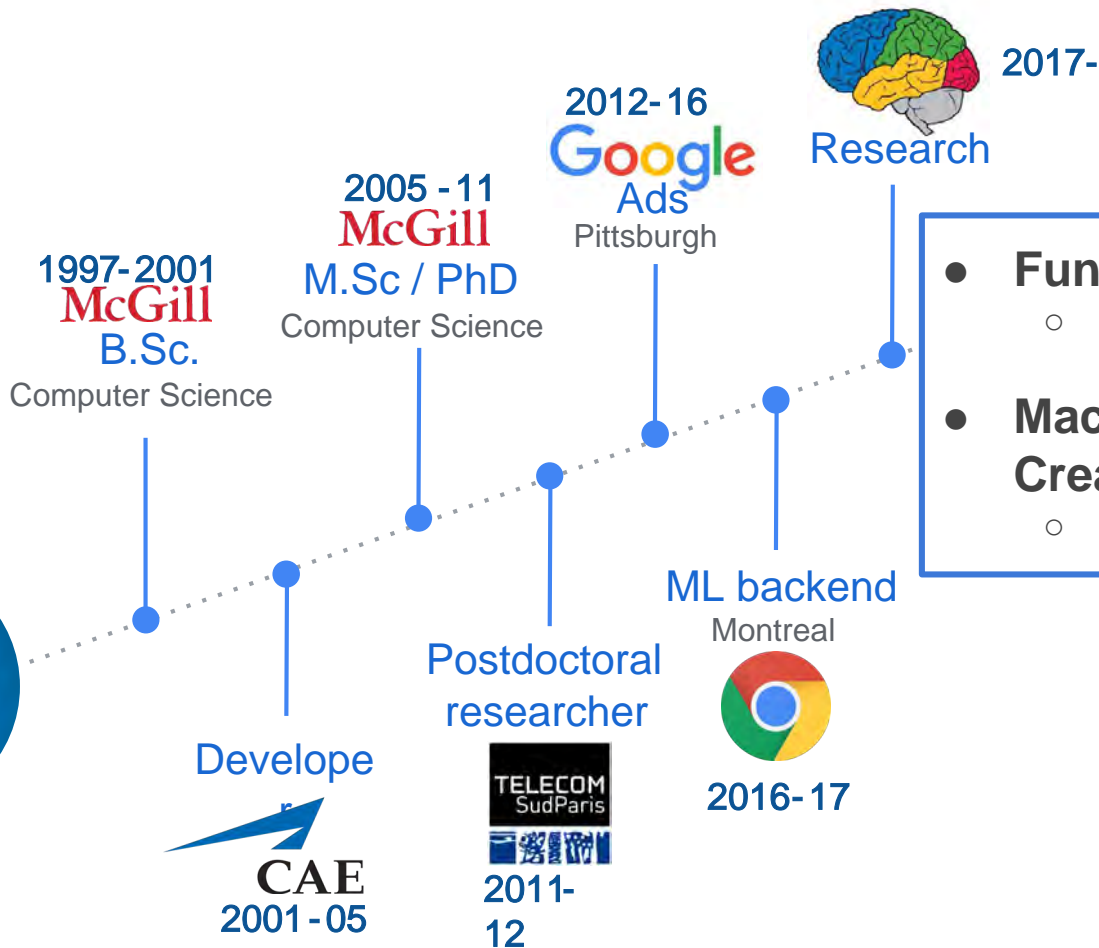
DOPAMINE: A RESEARCH FRAMEWORK FOR DEEP REINFORCEMENT LEARNING

Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar & Marc G. Bellemare
Google Brain
{psc, smoitra, cgel, kumasaurabh, bellemare}@google.com

<https://arxiv.org/abs/1812.06110>



Pablo
Google MON



- **Fundamental RL**
 - May 7th, 9-9:45 R01
- **Machine Learning + Creativity**
 - May 7th, noon, R09



2005 - 11
McGill
M.Sc / PhD
Computer Science

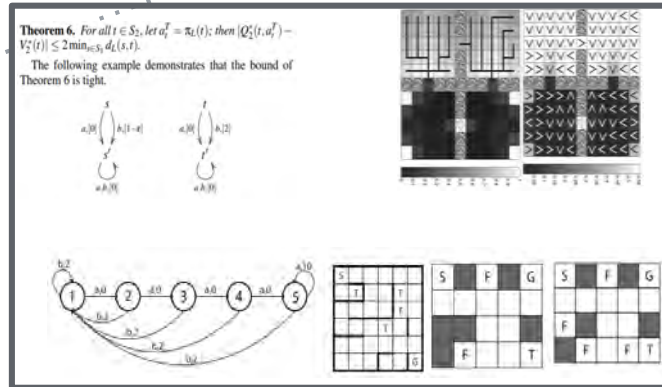
 2017-
Research



2017-

Research

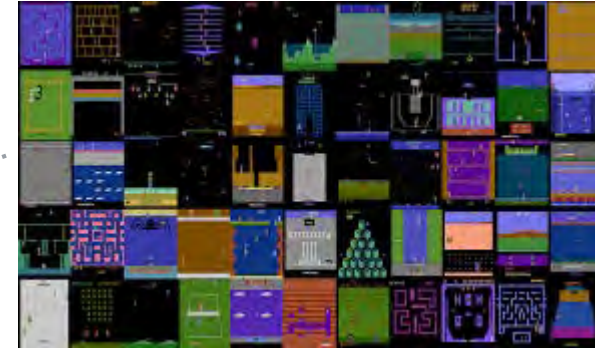
2005 - 11
McGill
M.Sc / PhD
Computer Science





2017-

Research



2005 - 11
McGill
M.Sc / PhD
Computer Science



Theorem 6. For all $t \in S_2$, let $a_t^x = \pi_t(t)$; then $|Q_t^x(t, a_t^x) - V_t^x(t)| \leq 2 \min_{c \in S_1} d_t(s, t)$.

The following example demonstrates that the bound of Theorem 6 is tight.

S	T	T

S	F	G

S	F	G

There is now a stronger
emphasis on empirical
validation.

Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning

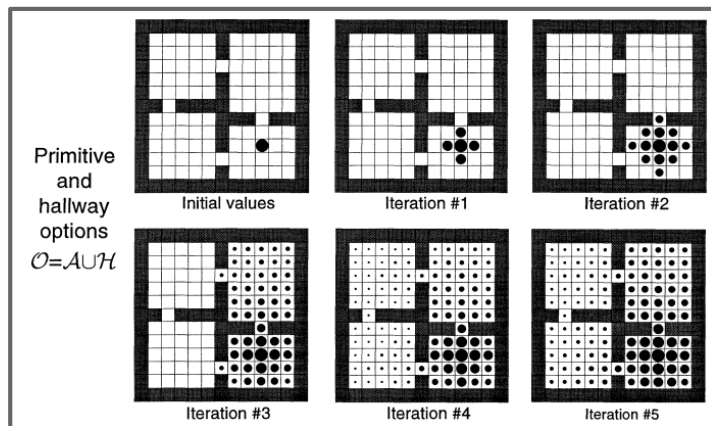
Richard S. Sutton^{a,*}, Doina Precup^b, Satinder Singh^a

^a AT&T Labs.-Research, 180 Park Avenue, Florham Park, NJ 07932, USA

^b Computer Science Department, University of Massachusetts, Amherst, MA 01003, USA

Received 1 December 1998

Theorem 3 (Convergence of intra-option Q-learning). *For any set of Markov options, \mathcal{O} , with deterministic policies, one-step intra-option Q-learning converges with probability 1 to the optimal Q-values, Q^* , for every option regardless of what options are executed during learning, provided that every action gets executed in every state infinitely often.*



The Option-Critic Architecture

Pierre-Luc Bacon and **Jean Harb** and **Doina Precup**

Reasoning and Learning Lab, School of Computer Science

McGill University

{pbacon, jharb, dprecup}@cs.mcgill.ca

Theorem 1 (Intra-Option Policy Gradient Theorem). *Given a set of Markov options with stochastic intra-option policies differentiable in their parameters θ , the gradient of the expected discounted return with respect to θ and initial condition (s_0, ω_0) is:*

$$\sum_{s, \omega} \mu_{\Omega}(s, \omega | s_0, \omega_0) \sum_a \frac{\partial \pi_{\omega, \theta}(a | s)}{\partial \theta} Q_U(s, \omega, a),$$

where $\mu_{\Omega}(s, \omega | s_0, \omega_0)$ is the probability of observing state s and option ω along a trajectory starting from (s_0, ω_0) .

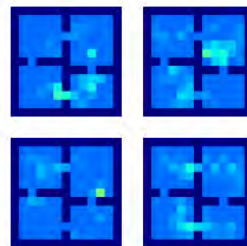


Figure 2: Termination probabilities for the option-critic. The darkest color represents the highest probability, and lighter colors encode lower probabilities.

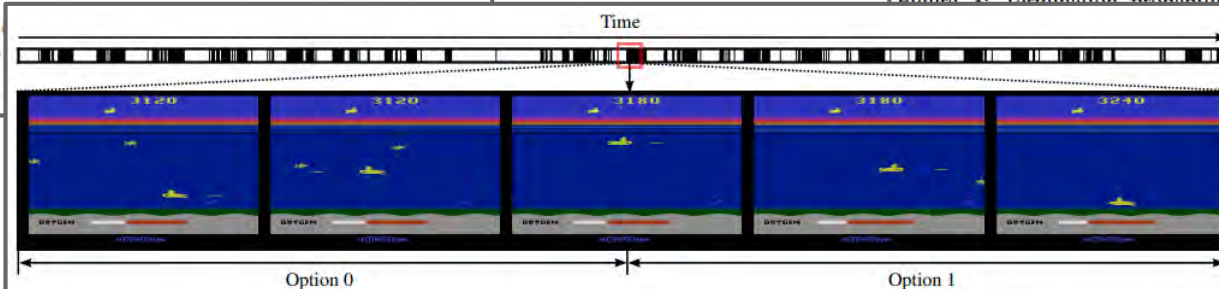


Figure 9: Up/down specialization in the solution found by option-critic when learning with 2 options in Seaquest. The top bar shows a trajectory in the game, with “white” representing a segment during which option 1 was active and “black” for option 2.

Both types of papers are
trying to get the same
thing out of their readers...

Trust

Reproducibility checklist

For any **theoretical claim**, check if you include:

- ☐ A statement of the result.
- ☐ A clear explanation of any assumptions.
- ☐ A complete proof of the claim.

For all **figures** and **tables** that present empirical results, check if you include:

- ☐ A complete description of the data collection process, including sample size.
- ☐ A link to a downloadable version of the dataset or simulation environment.
- ☐ An explanation of any data that were excluded, description of any pre-processing step.
- ☐ An explanation of how samples were allocated for training / validation / testing.
- ☐ The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results.
- ☐ The exact number of evaluation runs.
- ☐ A description of how experiments were run.
- ☐ A clear definition of the specific measure or statistics used to report results.
- ☐ Clearly defined error bars.
- ☐ A description of results with central tendency (e.g. mean) & variation (e.g. stddev).
- ☐ A description of the computing infrastructure used.

Readers trust Theorems by looking at the
proof.

Proofs are fully contained in the paper!



By <http://www-groups.dcs.st-and.ac.uk/~history/PictDisplay/Fermat.html>, Public Domain, <https://commons.wikimedia.org/w/index.php?curid=36804>

Trust on empirical results in RL?

Running on well-understood environments (ALE, etc.)

Reporting well-understood metrics

Reporting hyperparameters chosen

Multiple runs, confidence intervals

Hyperparameter optimization for baselines?

Providing source code

...



Theorem 1. *The set \mathbb{Q} of rational numbers is countable.*

Theorem 1. The set \mathbb{Q} of rational numbers is countable.

Proof 3

For each $n \in \mathbb{N}$, define S_n to be the set:

$$S_n := \left\{ \frac{m}{n} : m \in \mathbb{Z} \right\}$$

By **Integers are Countably Infinite**, each S_n is countably infinite.

Because each rational number can be written down with a positive denominator, it follows that:

$$\forall q \in \mathbb{Q} : \exists n \in \mathbb{N} : q \in S_n$$

which is to say:

$$\bigcup_{n \in \mathbb{N}} S_n = \mathbb{Q}$$

By **Countable Union of Countable Sets is Countable**, it follows that \mathbb{Q} is countable.

Since \mathbb{Q} is manifestly infinite, it is countably infinite.



Proof 4

Let $Q_{\pm} = \{q \in \mathbb{Q} : \pm q > 0\}$.

For every $q \in Q_{+}$, there exists at least one pair $(m, n) \in \mathbb{N} \times \mathbb{N}$ such that $q = \frac{m}{n}$.

Therefore, we can find an injection $i : Q_{+} \rightarrow \mathbb{N} \times \mathbb{N}$.

By **Cartesian Product of Natural Numbers with Itself is Countable**, $\mathbb{N} \times \mathbb{N}$ is countable.

Hence Q_{+} is countable, by **Domain of Injection to Countable Set is Countable**.

The map $- : q \mapsto -q$ provides a bijection from Q_{-} to Q_{+} , hence Q_{-} is also countable.

Hence \mathbb{Q} is countable.

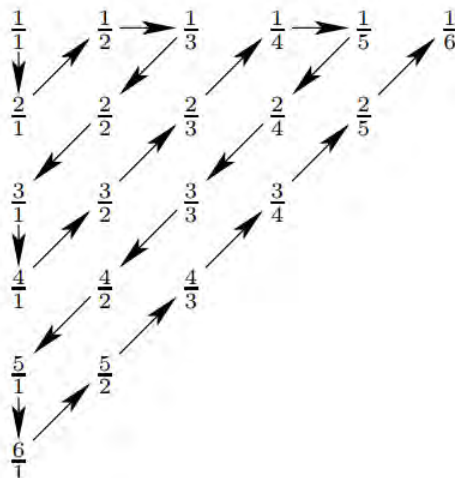


Redirection!

Theorem 1. *The set \mathbb{Q} of rational numbers is countable.*

■ **Proof.** By listing the set \mathbb{Q}^+ of positive rationals as suggested in the figure in the margin, but leaving out numbers already encountered, we see that \mathbb{Q}^+ is countable, and hence so is \mathbb{Q} by listing 0 at the beginning and $-\frac{p}{q}$ right after $\frac{p}{q}$. With this listing

$$\mathbb{Q} = \{0, 1, -1, 2, -2, \frac{1}{2}, -\frac{1}{2}, \frac{1}{3}, -\frac{1}{3}, 3, -3, 4, -4, \frac{3}{2}, -\frac{3}{2}, \dots\}. \quad \square$$



We should write code
worthy of being included
in The Book!

¡Gracias!

github.com/google/dopamine

Pablo Samuel Castro

@pcastr 

psc@google.com

