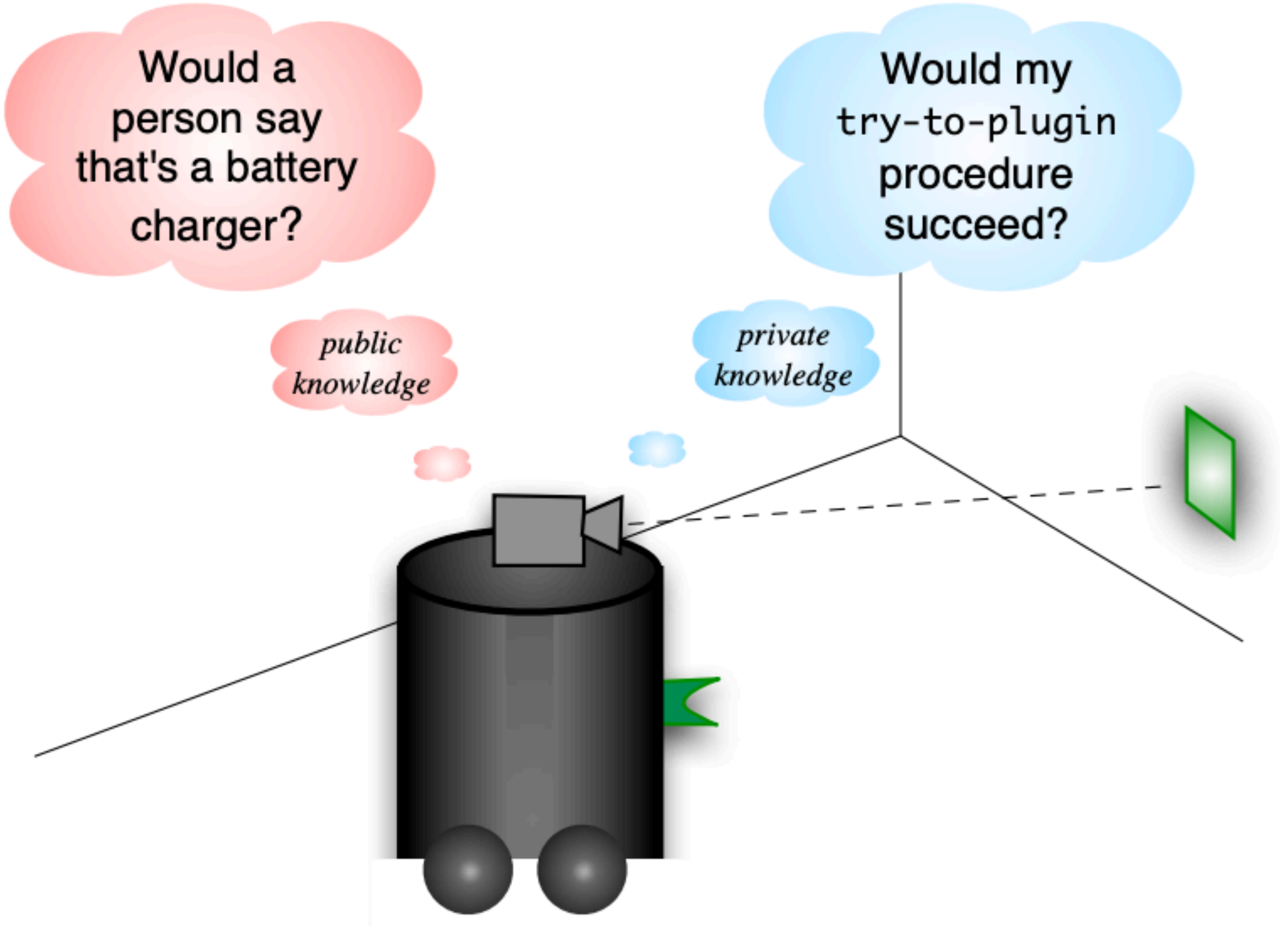
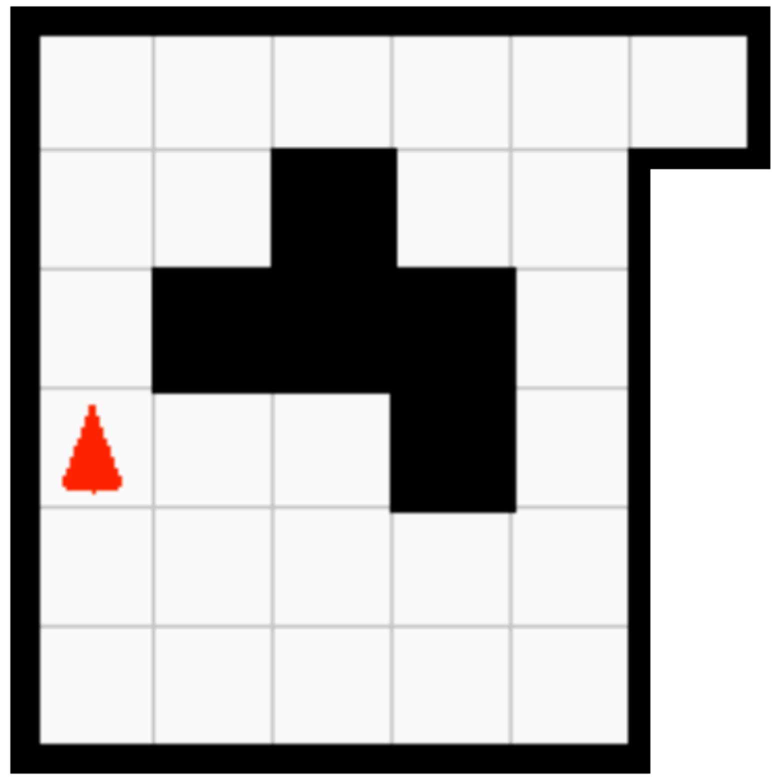
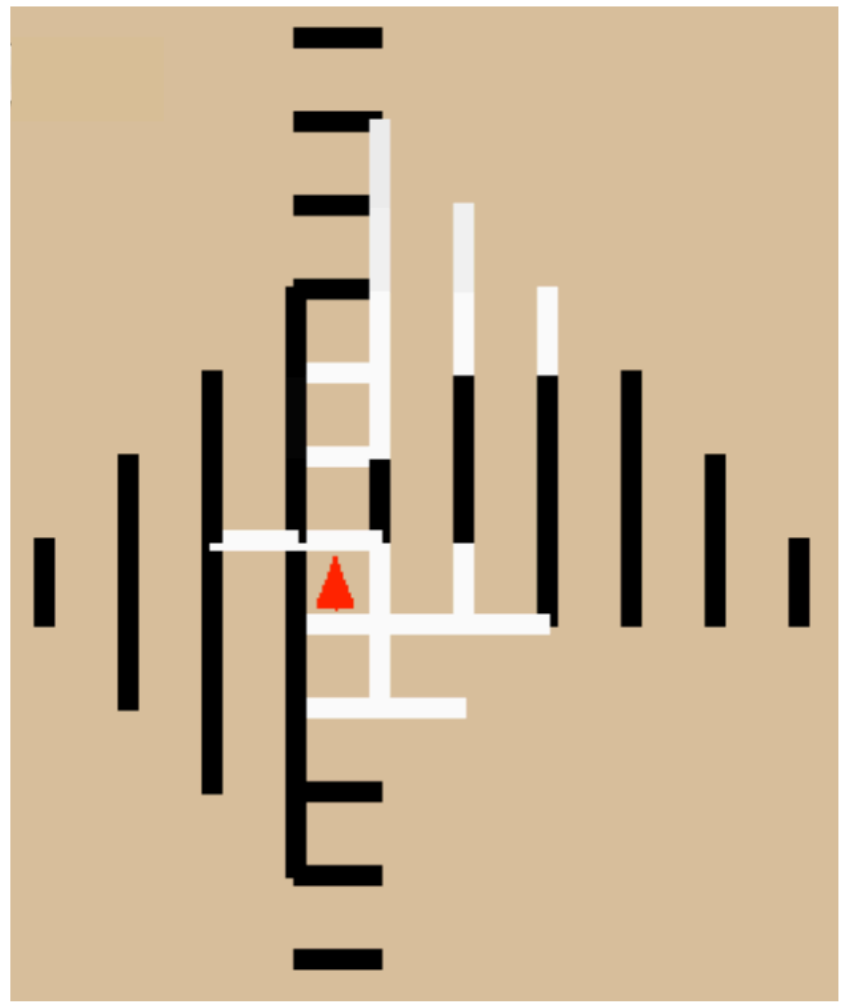
Start recording …

How should we represent the agent's knowledge of the world?


Would a person say that's a battery charger?
public knowledge
Would my try-to-plugin procedure succeed?
private knowledge

Everything the agent knows should be a statement about the data-stream





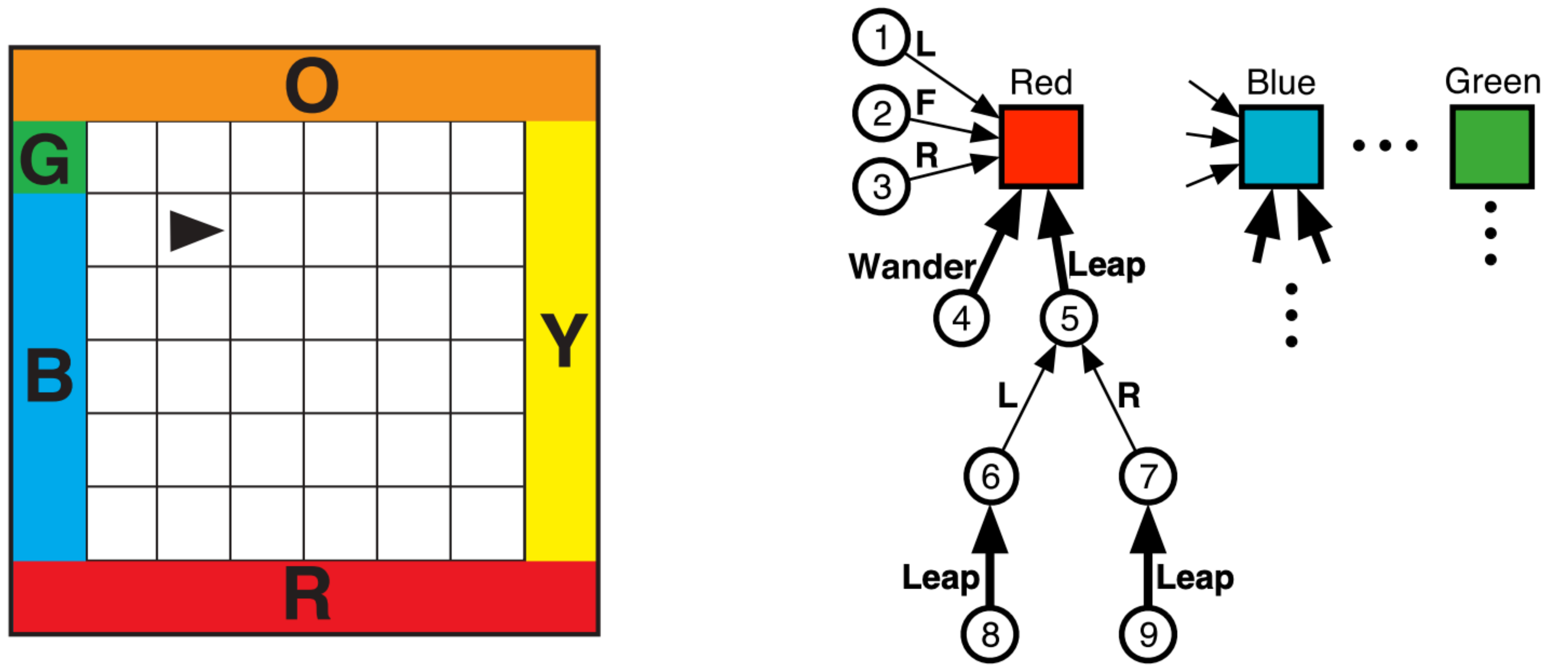
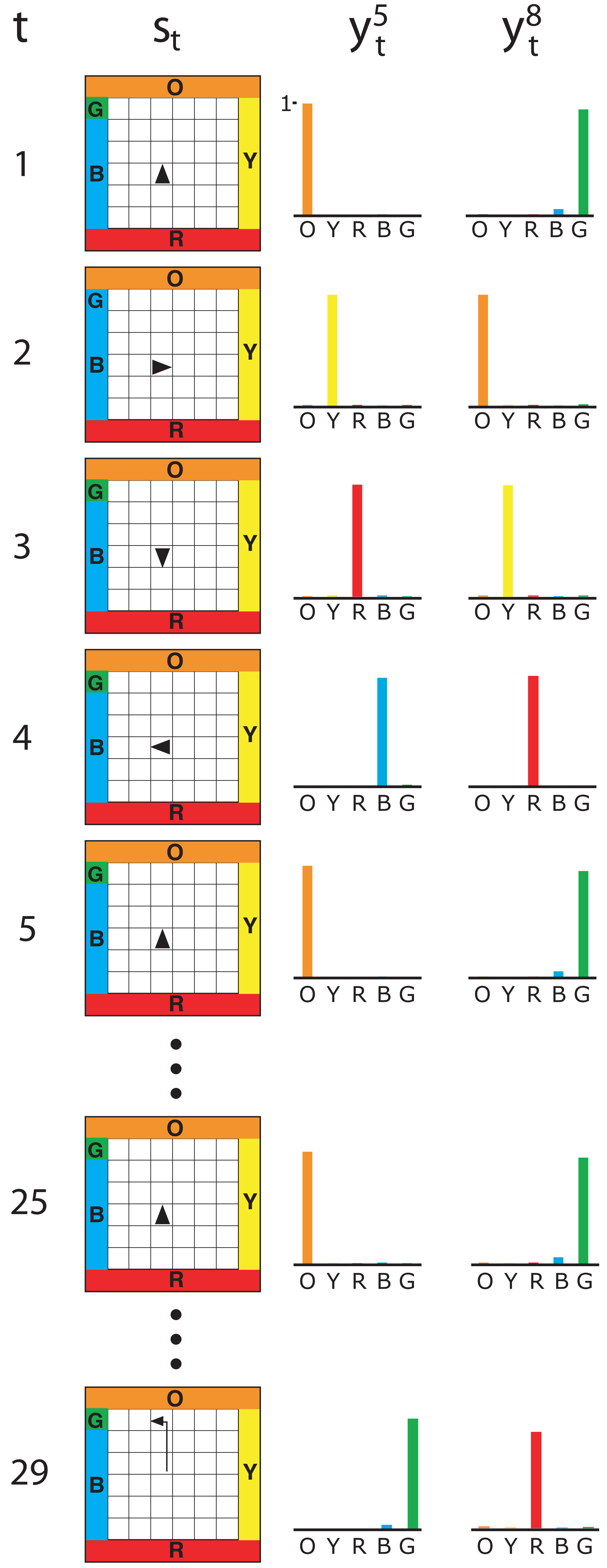


Figure 2: The compass world (left) and a portion of the



- Y5 is leap
- Y8 is leap, L, leap

How much larger would the TD network have to be to handle a 100x100 gridworld? The answer is *none at all*. The same question network applies to any size problem. If the layout of the colored walls remain the same, then even the answer network transfers across worlds of widely varying sizes. We have used the same TD network to make all long-term predictions correctly on a 100x100 version of this problem.

Admin

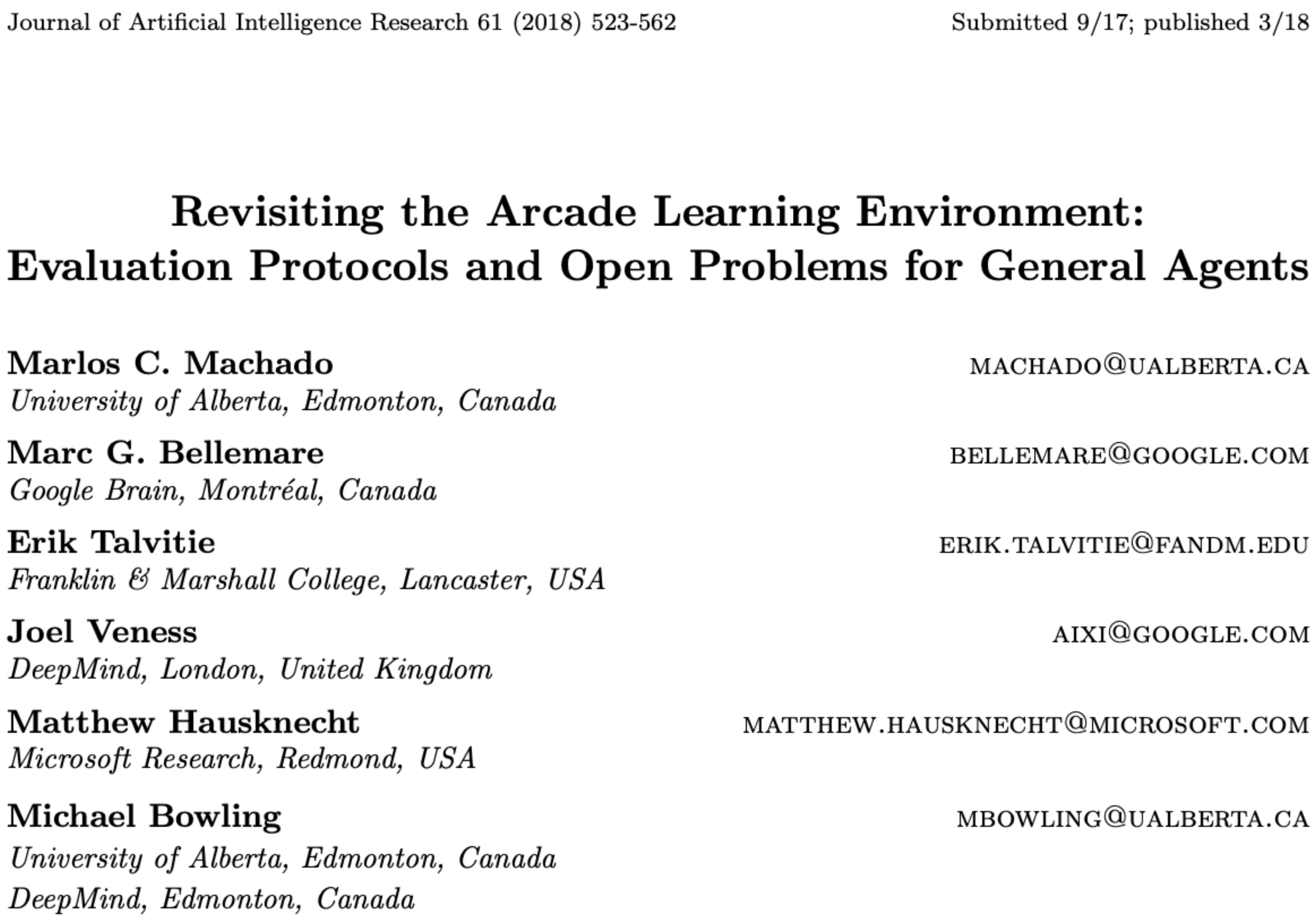
- No office hours this week
- Project team list sheet:
    - https://docs.google.com/spreadsheets/d/1f-QybvJk5V5dilsHOL9f6eNx5fAR0gFEuUekHS94MhE
- Session moderators for today: **Johnstonbaugh, Kerrick & Burega, Bradley Thomas**
    - https://docs.google.com/spreadsheets/d/1dbmlvduupZUCDjxU4HW2_350OVrVG-g1FoEAG-uWhMk

Let’s start discussing making things better

The Good, ~~the bad, and the ugly~~

Two papers about methodology and better RL experiments

- A recent Deep RL paper
- An older (pre deep learning hype) paper
- Both from serious algorithm development experts who also happen to be very serious empiricists
- One focused on the Arcade Learning Environment (ALE) or Atari
- The other worried about overfitting to class RL control tasks
- Both ultimately focused on general purpose agents that can solve many tasks!



Journal of Artificial Intelligence Research 61 (2018) 523-562 Submitted 9/17; published 3/18

Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents

Marlos C. Machado MACHADO@UALBERTA.CA
University of Alberta, Edmonton, Canada

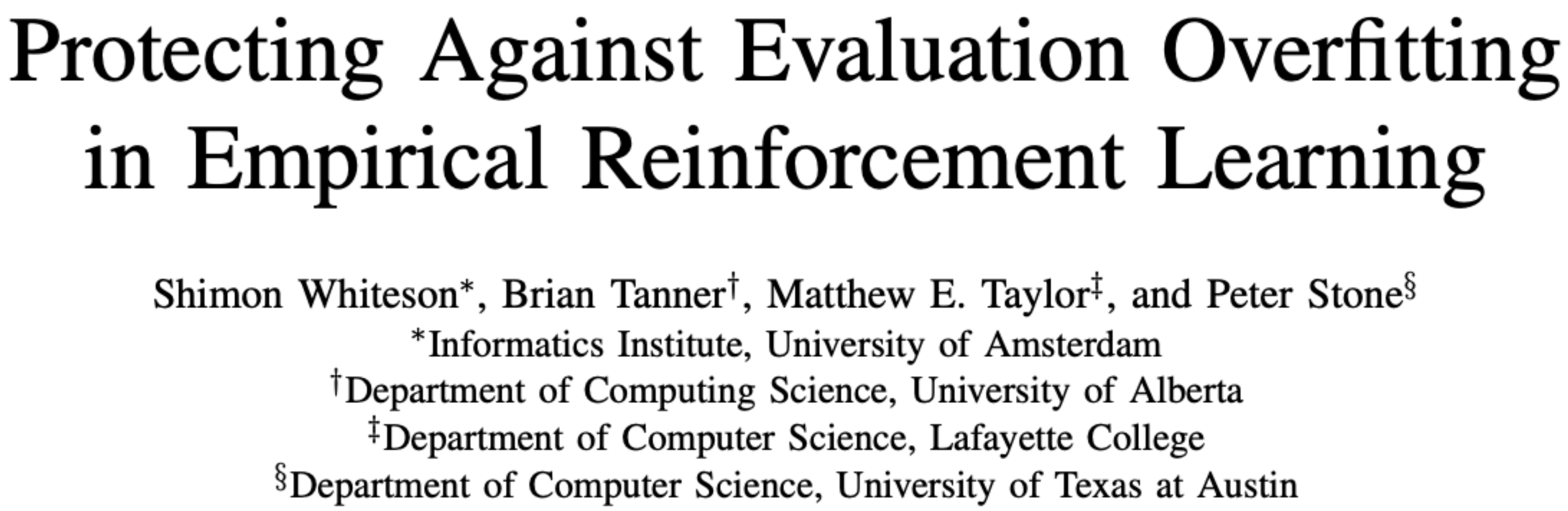
Marc G. Bellemare BELLEMARE@GOOGLE.COM
Google Brain, Montréal, Canada

Erik Talvitie ERIK.TALVITIE@FANDM.EDU
Franklin & Marshall College, Lancaster, USA

Joel Veness AIXI@GOOGLE.COM
DeepMind, London, United Kingdom

Matthew Hausknecht MATTHEW.HAUSKNECHT@MICROSOFT.COM
Microsoft Research, Redmond, USA

Michael Bowling MBOWLING@UALBERTA.CA
University of Alberta, Edmonton, Canada
DeepMind, Edmonton, Canada



Protecting Against Evaluation Overfitting in Empirical Reinforcement Learning

Shimon Whiteson*, Brian Tanner[†], Matthew E. Taylor[‡], and Peter Stone[§]
*Informatics Institute, University of Amsterdam
[†]Department of Computing Science, University of Alberta
[‡]Department of Computer Science, Lafayette College
[§]Department of Computer Science, University of Texas at Austin

305 citations between them

Quick history lesson on ALE

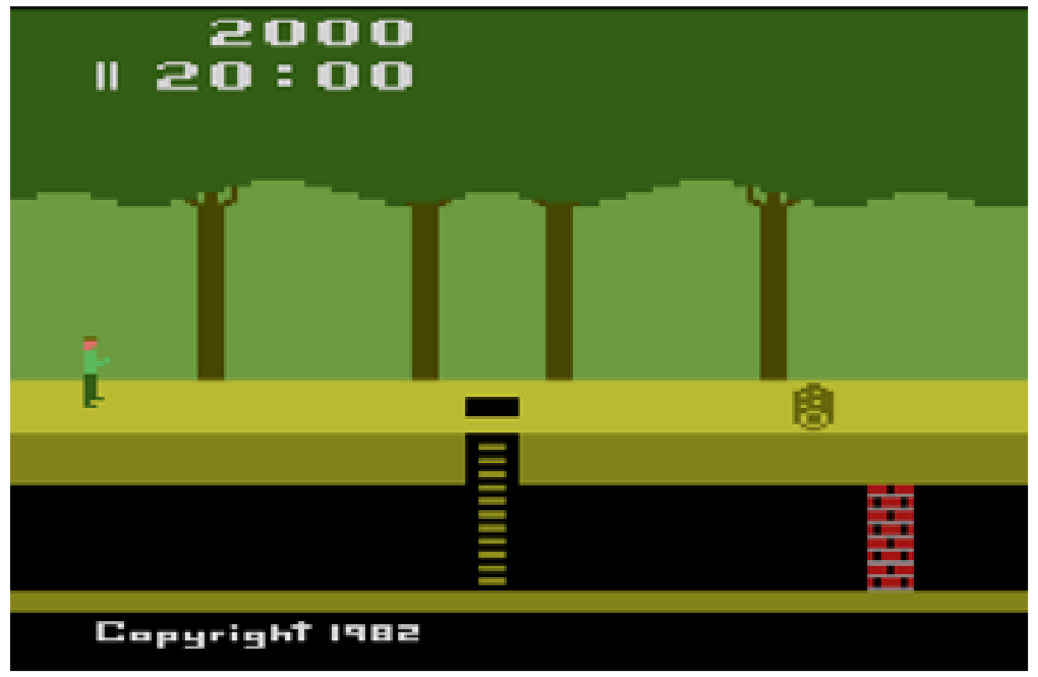
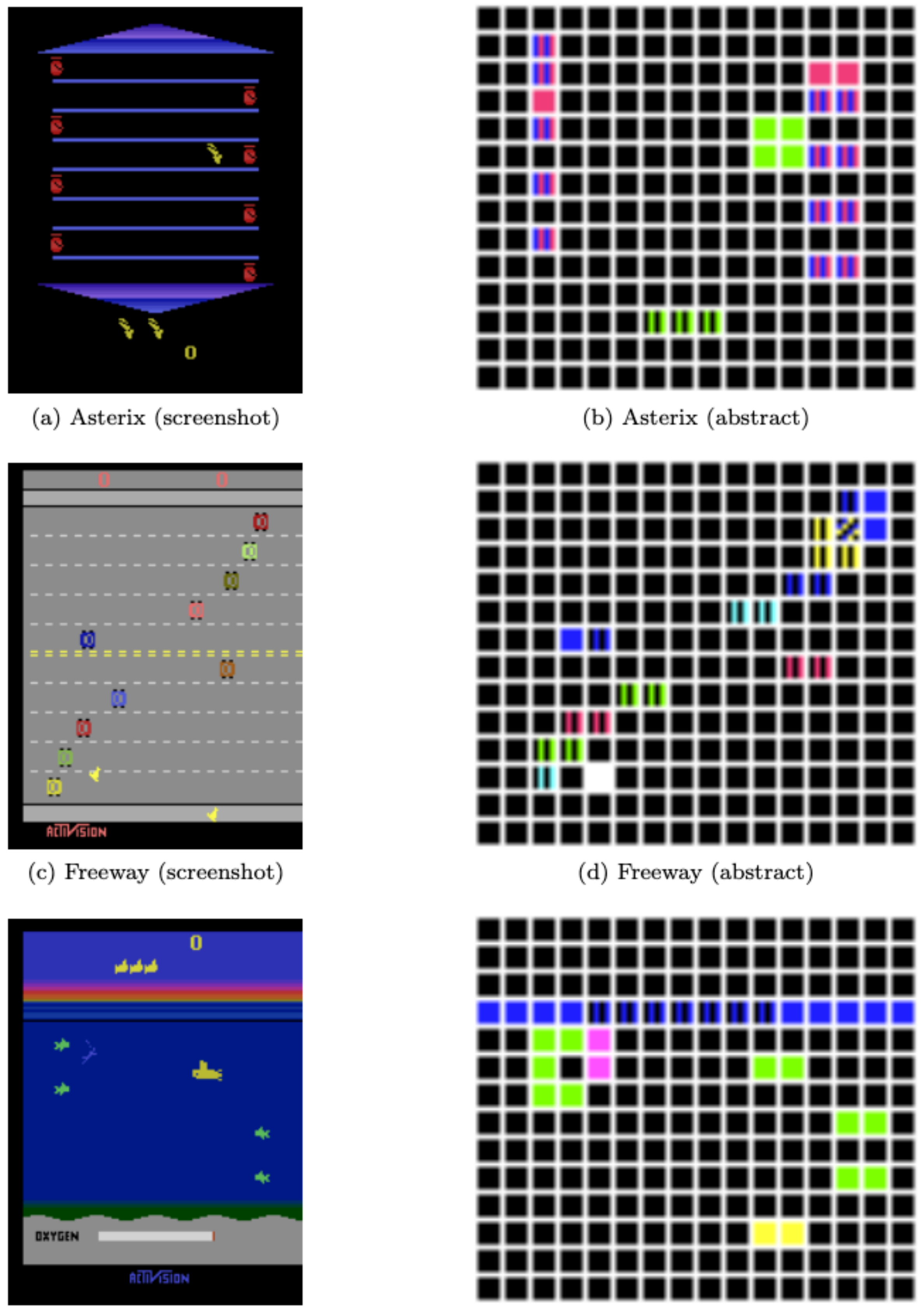


Figure 2. Initial screen of Pitfall.

- Based on video game console from the 80's
- First used by Diol et al (2008) to investigate object oriented representations for RL in Pitfall!
  - From each frame a list of objects was generated (e.g., Man, Hole, Ladder, Log, Wall and Tree)
  - With attributes: width and height, and direction for the man
  - Seven actions: WalkRight, **WalkLeft**, JumpLeft, JumpRight, Up, Down and JumpUp
    - **WalkLeft** requires four frames: one to tell Pitfall to move the Man to the left, and three frames where no action is taken to allow for the animation of the Man to complete
- Raw inputs are huge, but object representations reduces it to 6^(640x420)
- Agent learned the optimal policy from only 494 actions, or 4810 game frames
  - SOTA in pitfall was zero until 2020: specialized distributed architectures trained for 5 billion frames or 950 days per game

History of ALE

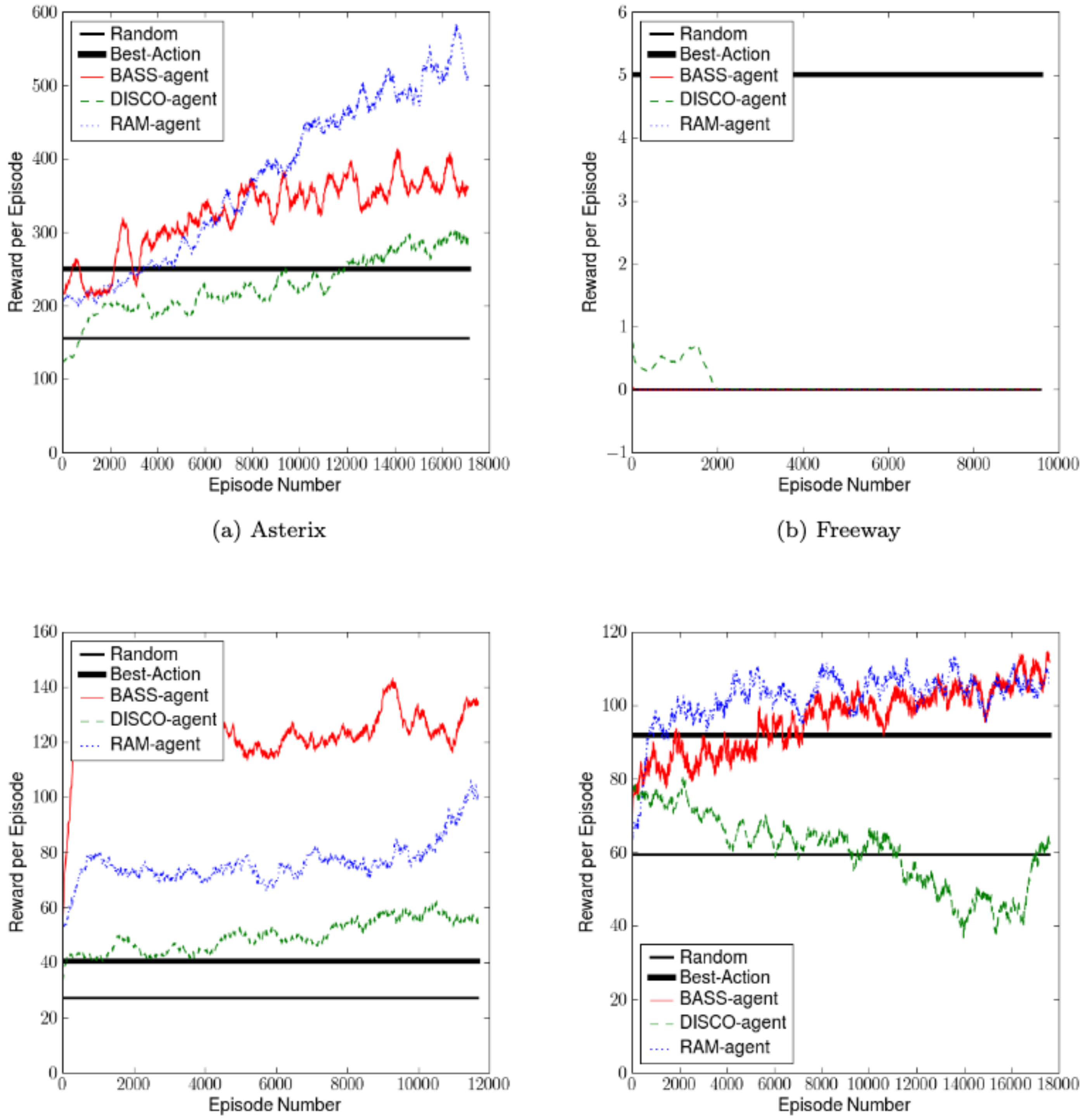
- Next Naddaf undertook connecting the Atari 2600 emulator to RL agents as an MSc project with Bowling at UofA



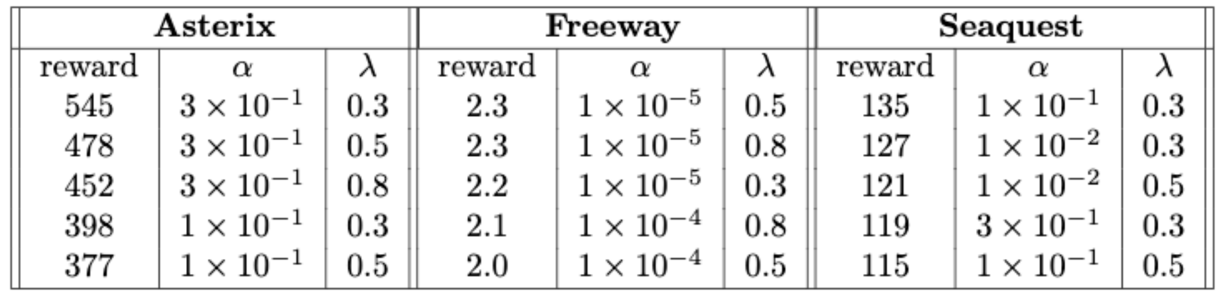
(a) Asterix (screenshot) (b) Asterix (abstract)

(c) Freeway (screenshot) (d) Freeway (abstract)

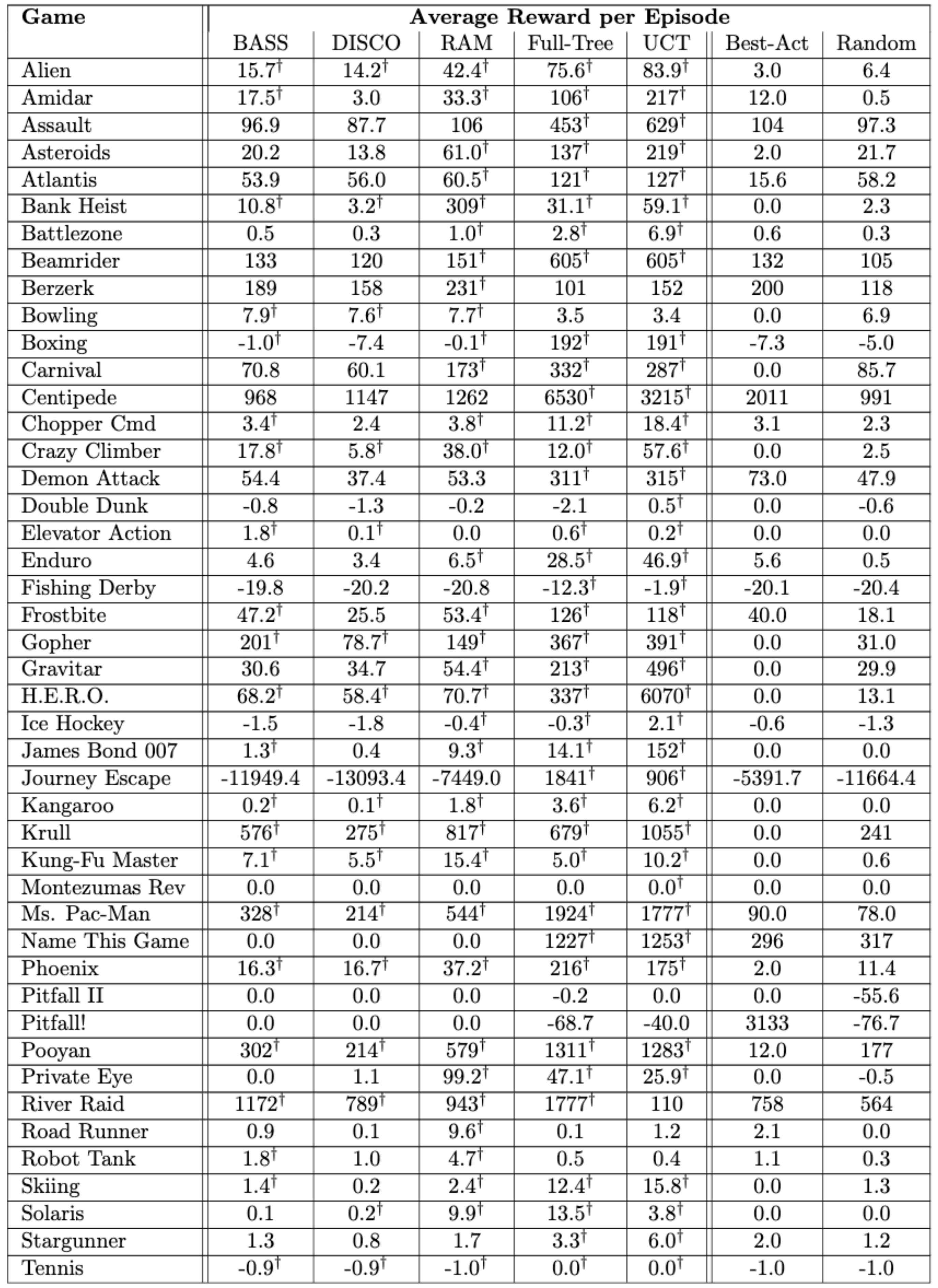
(e) Seaquest (screenshot) (f) Seaquest (abstract)



(a) Asterix (b) Freeway



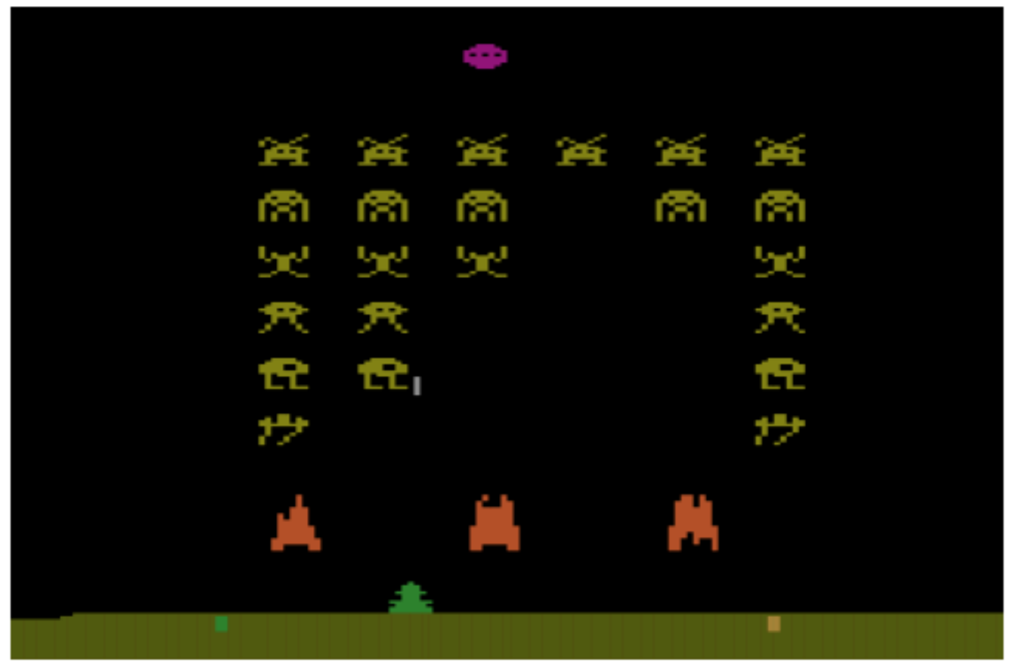
Asterix			Freeway			Seaquest		
reward	α	λ	reward	α	λ	reward	α	λ
545	3×10^{-1}	0.3	2.3	1×10^{-5}	0.5	135	1×10^{-1}	0.3
478	3×10^{-1}	0.5	2.3	1×10^{-5}	0.8	127	1×10^{-2}	0.3
452	3×10^{-1}	0.8	2.2	1×10^{-5}	0.3	121	1×10^{-2}	0.5
398	1×10^{-1}	0.3	2.1	1×10^{-4}	0.8	119	3×10^{-1}	0.3
377	1×10^{-1}	0.5	2.0	1×10^{-4}	0.5	115	1×10^{-1}	0.5



Game	Average Reward per Episode						
	BASS	DISCO	RAM	Full-Tree	UCT	Best-Act	Random
Alien	15.7†	14.2†	42.4†	75.6†	83.9†	3.0	6.4
Amidar	17.5†	3.0	33.3†	106†	217†	12.0	0.5
Assault	96.9	87.7	106	453†	629†	104	97.3
Asteroids	20.2	13.8	61.0†	137†	219†	2.0	21.7
Atlantis	53.9	56.0	60.5†	121†	127†	15.6	58.2
Bank Heist	10.8†	3.2†	309†	31.1†	59.1†	0.0	2.3
Battlezone	0.5	0.3	1.0†	2.8†	6.9†	0.6	0.3
Beamrider	133	120	151†	605†	605†	132	105
Berzerk	189	158	231†	101	152	200	118
Bowling	7.9†	7.6†	7.7†	3.5	3.4	0.0	6.9
Boxing	-1.0†	-7.4	-0.1†	192†	191†	-7.3	-5.0
Carnival	70.8	60.1	173†	332†	287†	0.0	85.7
Centipede	968	1147	1262	6530†	3215†	2011	991
Chopper Cmd	3.4†	2.4	3.8†	11.2†	18.4†	3.1	2.3
Crazy Climber	17.8†	5.8†	38.0†	12.0†	57.6†	0.0	2.5
Demon Attack	54.4	37.4	53.3	311†	315†	73.0	47.9
Double Dunk	-0.8	-1.3	-0.2	-2.1	0.5†	0.0	-0.6
Elevator Action	1.8†	0.1†	0.0	0.6†	0.2†	0.0	0.0
Enduro	4.6	3.4	6.5†	28.5†	46.9†	5.6	0.5
Fishing Derby	-19.8	-20.2	-20.8	-12.3†	-1.9†	-20.1	-20.4
Frostbite	47.2†	25.5	53.4†	126†	118†	40.0	18.1
Gopher	201†	78.7†	149†	367†	391†	0.0	31.0
Gravitar	30.6	34.7	54.4†	213†	496†	0.0	29.9
H.E.R.O.	68.2†	58.4†	70.7†	337†	6070†	0.0	13.1
Ice Hockey	-1.5	-1.8	-0.4†	-0.3†	2.1†	-0.6	-1.3
James Bond 007	1.3†	0.4	9.3†	14.1†	152†	0.0	0.0
Journey Escape	-11949.4	-13093.4	-7449.0	1841†	906†	-5391.7	-11664.4
Kangaroo	0.2†	0.1†	1.8†	3.6†	6.2†	0.0	0.0
Krull	576†	275†	817†	679†	1055†	0.0	241
Kung-Fu Master	7.1†	5.5†	15.4†	5.0†	10.2†	0.0	0.6
Montezumas Rev	0.0	0.0	0.0	0.0	0.0†	0.0	0.0
Ms. Pac-Man	328†	214†	544†	1924†	1777†	90.0	78.0
Name This Game	0.0	0.0	0.0	1227†	1253†	296	317
Phoenix	16.3†	16.7†	37.2†	216†	175†	2.0	11.4
Pitfall II	0.0	0.0	0.0	-0.2	0.0	0.0	-55.6
Pitfall!	0.0	0.0	0.0	-68.7	-40.0	3133	-76.7
Pooyan	302†	214†	579†	1311†	1283†	12.0	177
Private Eye	0.0	1.1	99.2†	47.1†	25.9†	0.0	-0.5
River Raid	1172†	789†	943†	1777†	110	758	564
Road Runner	0.9	0.1	9.6†	0.1	1.2	2.1	0.0
Robot Tank	1.8†	1.0	4.7†	0.5	0.4	1.1	0.3
Skiing	1.4†	0.2	2.4†	12.4†	15.8†	0.0	1.3
Solaris	0.1	0.2†	9.9†	13.5†	3.8†	0.0	0.0
Stargunner	1.3	0.8	1.7	3.3†	6.0†	2.0	1.2
Tennis	-0.9†	-0.9†	-1.0†	0.0†	0.0†	-1.0	-1.0

Why ALE?

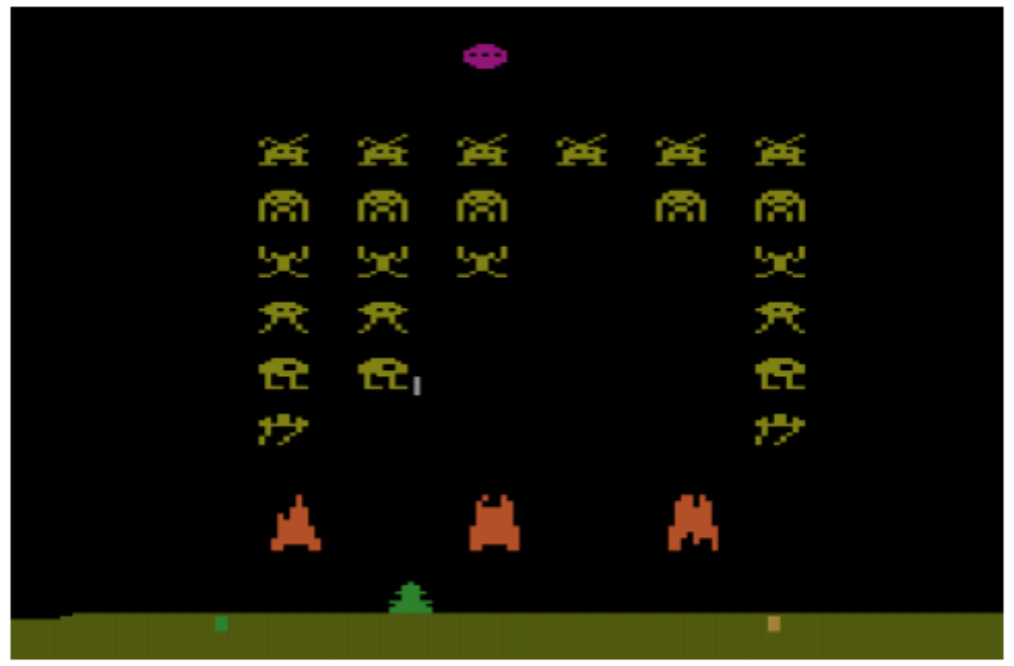
Great platform for research



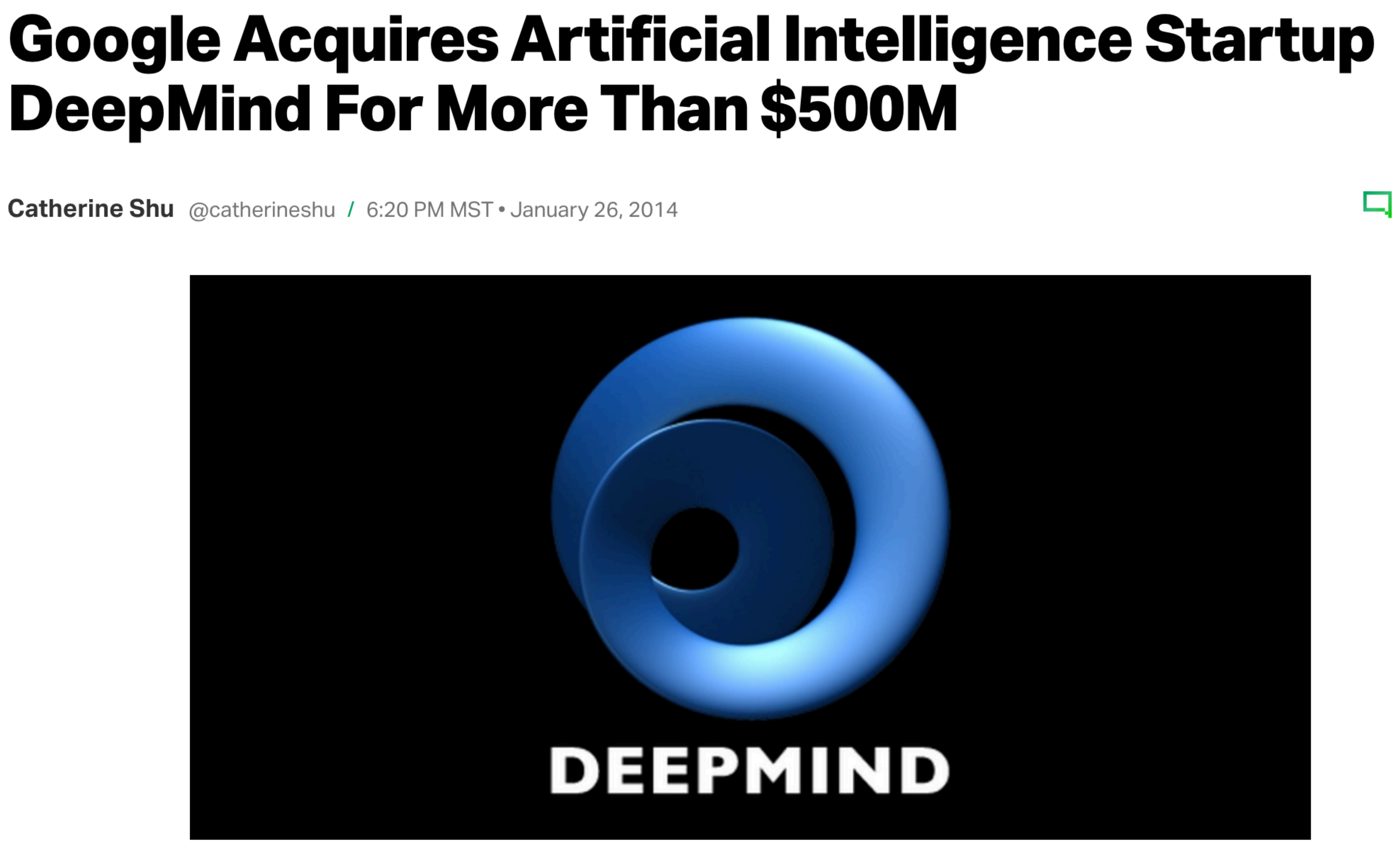
- The inputs are massive; state-construction is needed
- All games have the same interface, but very different dynamics
  - 7 bit x 160x210, 18 discrete actions
- Clear reward signal: score (huge variety)
- Naturally episodic: when the game ends
- And its a simulator so saving and storing the agent (weights), resets, teleports all possible—hopefully mostly for debugging
- 55 different games >> total number of classic benchmarks in RL

Why ALE?

It’s all about general purpose agents



- Fascinating for representation learning: using vision or RAM
- Fascinating potential for generalization and transfer: 5 paddle-based games!
- Fascinating for intrinsic motivation: humans don’t seem to look at the score
- True challenge for exploration-exploitation: agent can die but must progress; some games are particularly hard for exploration
- One agent (function approximation, learning algorithm, set of hyper-parameters) for 55 environments
- The games were not made by RL researchers; designed to be hard and interesting!
- Initial results showed it was very very hard



Google Acquires Artificial Intelligence Startup DeepMind For More Than $500M

Catherine Shu @catherineshu / 6:20 PM MST • January 26, 2014

ALE & DQN was a breakthrough

INNOVATIONS IN
The microbiome
nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
LEARNING CURVE
Self-taught AI software attains human-level performance in video games
PAGES 486 & 529
EPIDEMIOLOGY
SHARE DATA IN OUTBREAKS
Forge open access to sequences and more
PAGE 477
COSMOLOGY
A GIANT IN THE EARLY UNIVERSE
A supermassive black hole at a redshift of 6.3
PAGES 490 & 512
QUANTUM PHYSICS
TELEPORTATION FOR TWO
Transferring two properties of a single photon
PAGES 491 & 516
NATURE.COM/NATURE
26 February 2015 £10
Vol. 518, No. 7540

Revisiting ALE (2018)

A check-in and a refocus

- Discuss the usage and impact of ALE
- Highlight some problems
- Provide direction on how best to use ALE to drive research in RL

Troubling trends in ALE

A roadmap

- Divergent Evaluation Methodologies
- Performance measures
- Determinism and stochasticity
- A new gold-standard set of benchmark results for ALE
- Algorithmic areas for future ALE research

Divergent Evaluation Methodologies

How we use ALE

- Definition of episodes and termination
    - game over (some games have multiple lives—hard to learn about)
    - Nature DQN: at death (but not during evaluation)
- Setting hyperparameters
    - split games into train and test set (tune on training set)
    - Train on all the games
    - Tune hypers per game!!

Divergent Evaluation Methodologies

How we use ALE

- Measuring training data
    - Fixed number of episodes (usual challenges associated with)
    - Total number of frames
    - Do we skip frames?
- Summarizing performance; everyone does it differently
    - How do we compare fairly? Re-run every time?!?
    - Reporting performance during learning is ideal
- ALE is totally deterministic!

Summarizing performance

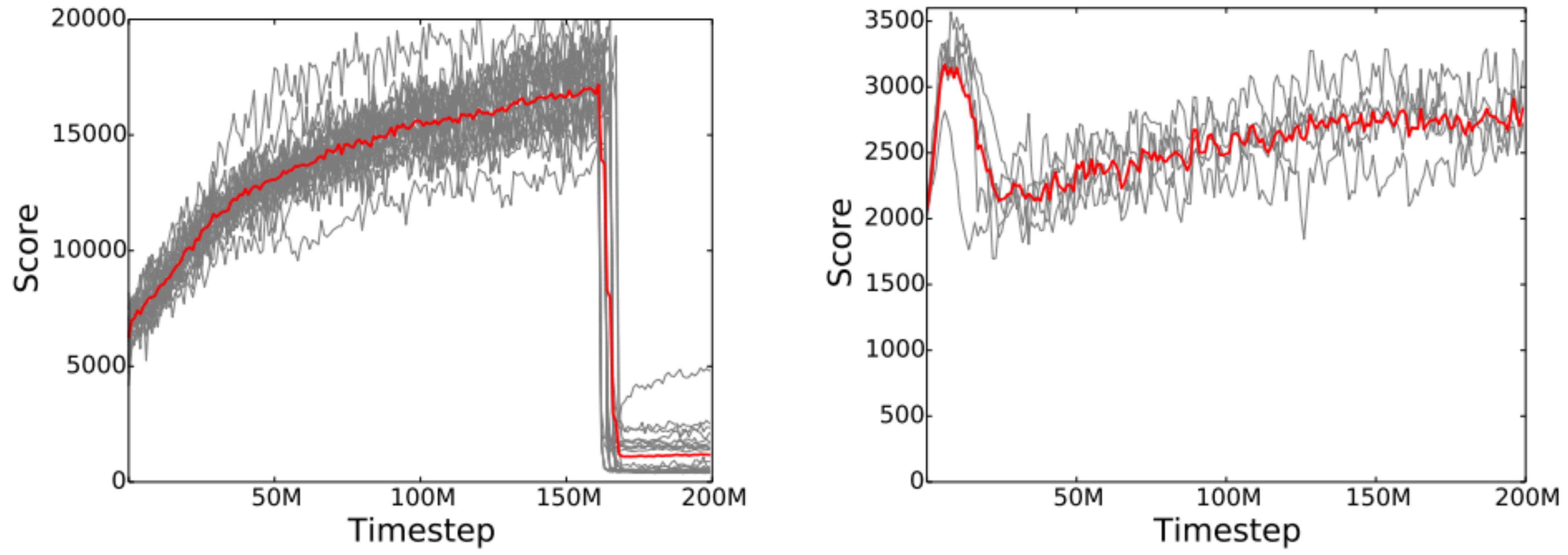
Deep dive

- Overall: more data->better policies
- But what do we want: learning speed, peak performance, stability, never-ending improvement?
- Many possible options:
    - Evaluation after learning
    - Evaluating the best policy
    - Area under the learning curve

Test performance

How good is the greedy policy

- Train, freeze learning, then average score over test episodes
  - Obscures sample efficiency
  - Agent could mostly explore during “training” then switch to eval. policy
- Nature DQN periodically evaluated the greedy policy, only reporting best evaluation results over all training
  - stability problems



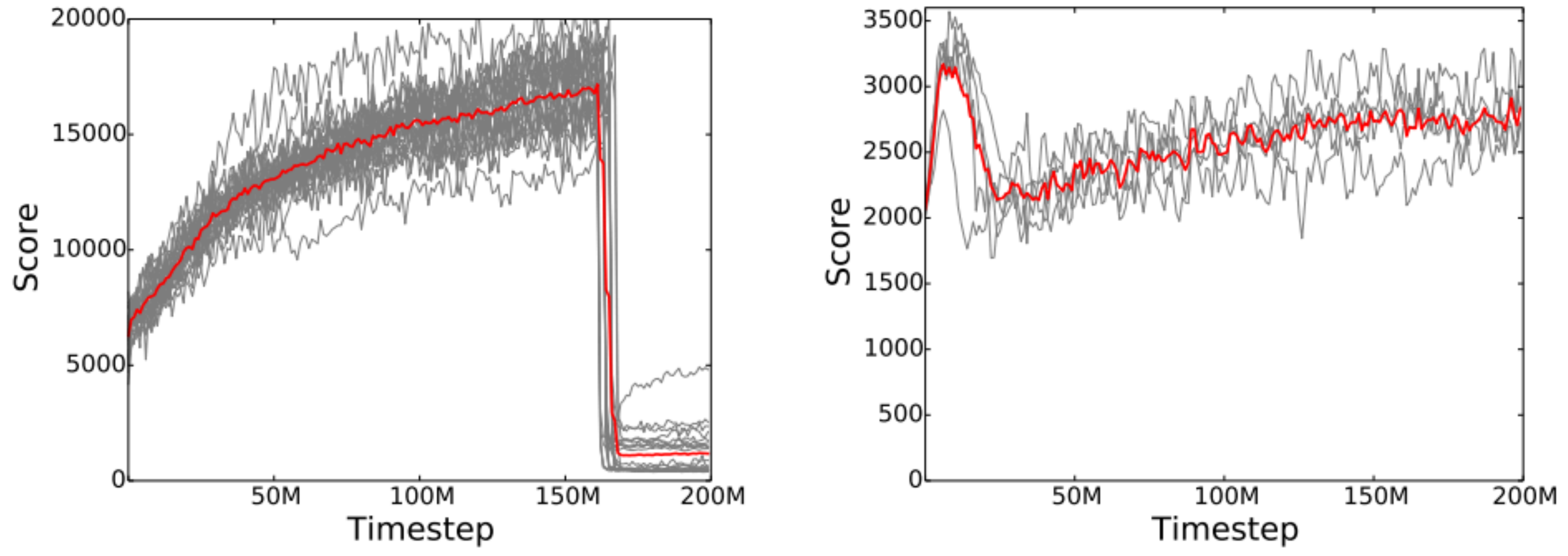
(a) Sarsa(λ) + Blob-PROST

(b) DQN

Performance during training

Return to good-old-fashion RL

- Reward/score plotted against training time… on **test episodes**?
  - Then take the area under the curve
- Short-run good performance de-emphasized
- Plummeting might not be well captured
- High-variance not always well captured… more in coming lectures
- Reward periodically during training…**no tests**
  - Average of last k episodes (continual learning)



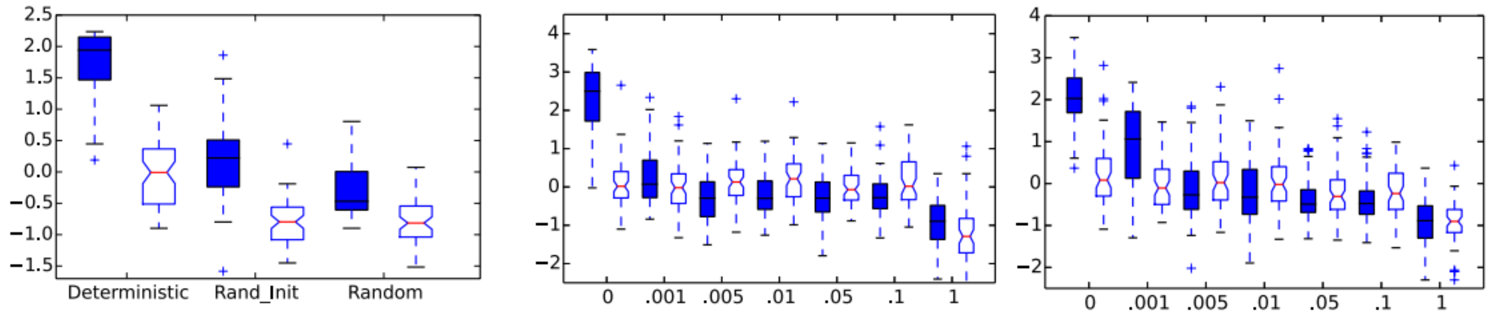
(a) Sarsa(λ) + Blob-PROST

(b) DQN

Determinism & stochasticity

Atari is totally deterministic

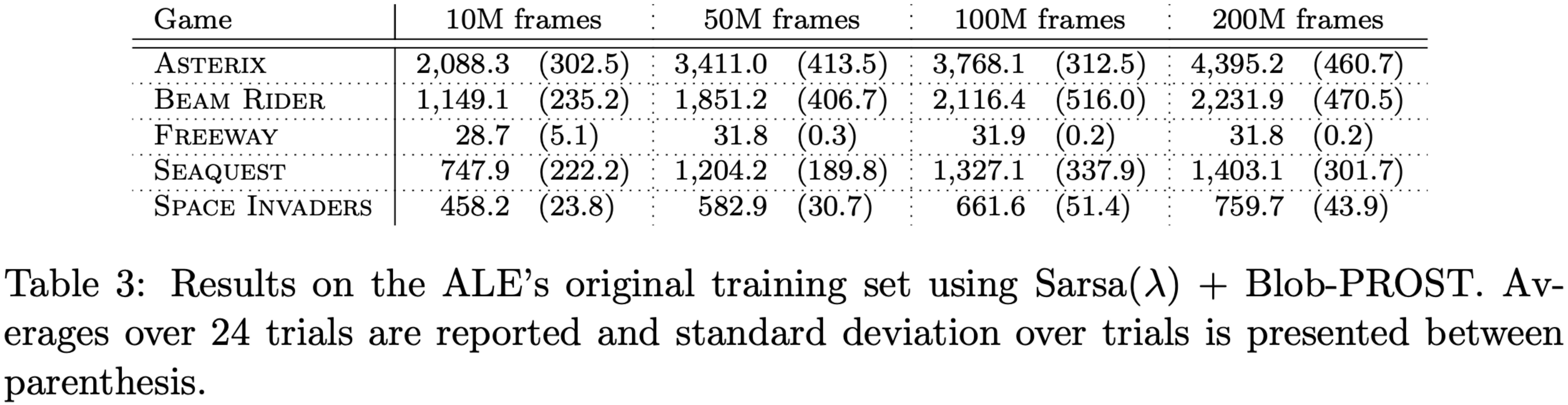
- Deterministic start states, deterministic outcomes given a sequence of actions
- Agent’s can exploit this…do we care? (e.g., same is true for Mountain Car)
    - Hydra architecture and GoExplore exploit this
    - “The Brute” maintains a partial history of trajectories to compute models and does Dynamic programming to find the best trajectory so far
    - Such deterministic agents are often not robust



- The real problem is non-standard injection of stochasticity:
    - e.g., human starts, random # of no-ops (no effect or huge), forced random actions, etc ..
    - Paper recommends sticky actions: small prob that prior action is executed again

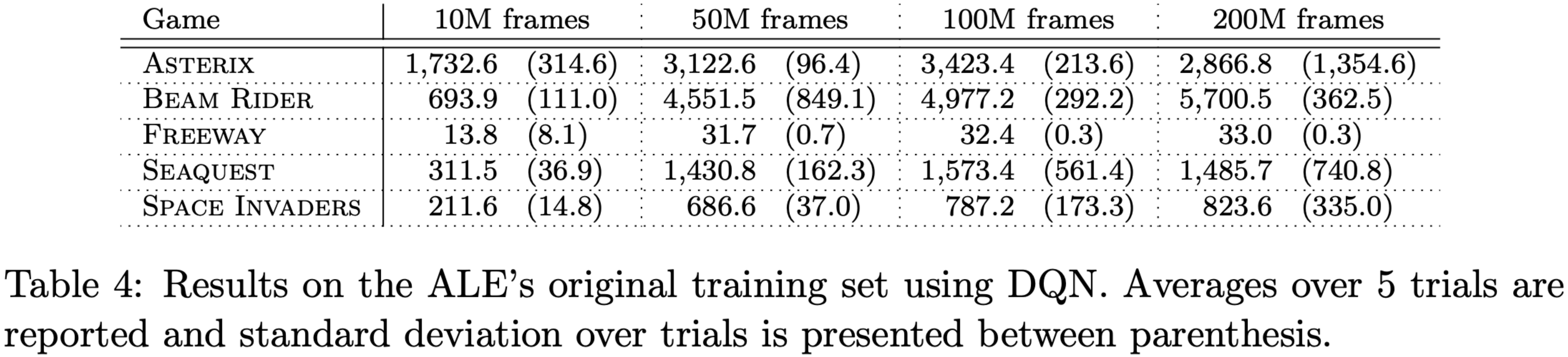
A new set of benchmark results

Blob what?



Game	10M frames		50M frames		100M frames		200M frames	
Asterix	2,088.3	(302.5)	3,411.0	(413.5)	3,768.1	(312.5)	4,395.2	(460.7)
Beam Rider	1,149.1	(235.2)	1,851.2	(406.7)	2,116.4	(516.0)	2,231.9	(470.5)
Freeway	28.7	(5.1)	31.8	(0.3)	31.9	(0.2)	31.8	(0.2)
Seaquest	747.9	(222.2)	1,204.2	(189.8)	1,327.1	(337.9)	1,403.1	(301.7)
Space Invaders	458.2	(23.8)	582.9	(30.7)	661.6	(51.4)	759.7	(43.9)

Table 3: Results on the ALE's original training set using Sarsa(λ) + Blob-PROST. Averages over 24 trials are reported and standard deviation over trials is presented between parenthesis.



Game	10M frames		50M frames		100M frames		200M frames	
Asterix	1,732.6	(314.6)	3,122.6	(96.4)	3,423.4	(213.6)	2,866.8	(1,354.6)
Beam Rider	693.9	(111.0)	4,551.5	(849.1)	4,977.2	(292.2)	5,700.5	(362.5)
Freeway	13.8	(8.1)	31.7	(0.7)	32.4	(0.3)	33.0	(0.3)
Seaquest	311.5	(36.9)	1,430.8	(162.3)	1,573.4	(561.4)	1,485.7	(740.8)
Space Invaders	211.6	(14.8)	686.6	(37.0)	787.2	(173.3)	823.6	(335.0)

Table 4: Results on the ALE's original training set using DQN. Averages over 5 trials are reported and standard deviation over trials is presented between parenthesis.

A new set of benchmark results

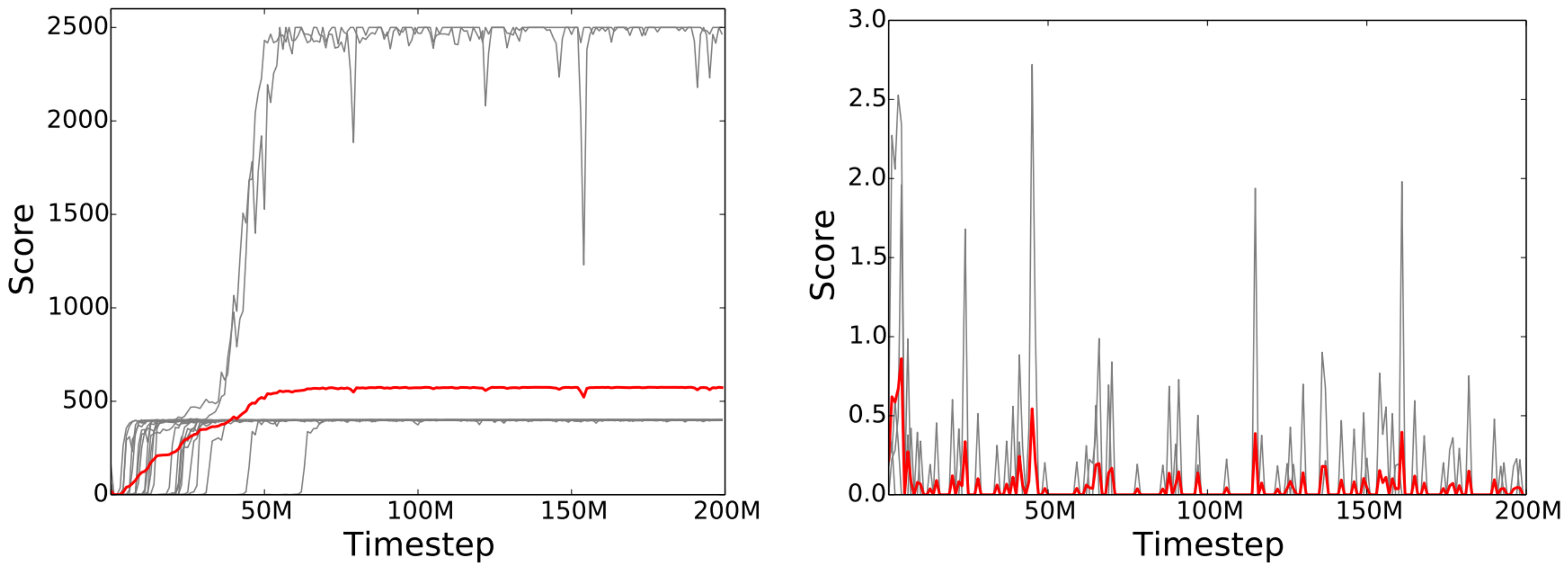
Main results

- Sarsa's perf steadily increases (on in 10% of games is perf worse after 200m)
  - In 38 games no stat. sig. improvement after 100m frames
- DQN: higher variability and did not benefit as much from more data
- But results have low significance overall due to limitations of 5 runs
  - Original DQN results were one run!
- Performance drops were algorithm dependent not game dependent
- Overall perf is very close & sticky actions didn't hurt either agent

Open problems in ALE

Things Atari should be really good for investigating

- **Representation learning:**
  - Neural-net agents have high sample complexity; especially in sparse reward tasks
  - Blob-Prost Sarsa is generally faster
  - Prioritized replay, Auxiliary task learning, and intrinsic rewards help
  - Many fiddly parameters and specialized compute needed
  - More fresh ideas like Generate and Test (http://www.incompleteideas.net/papers/MS-AAAIws-2013.pdf) and Co-agent networks (http://proceedings.mlr.press/v119/kostas20a/kostas20a.pdf) are needed



(a) Sarsa(λ) + Blob-PROST

(b) DQN

Open problems

Models and planning

- Most successes in ALE use the emulator—a perfect model

- Unrolling learned, imperfect models is hard

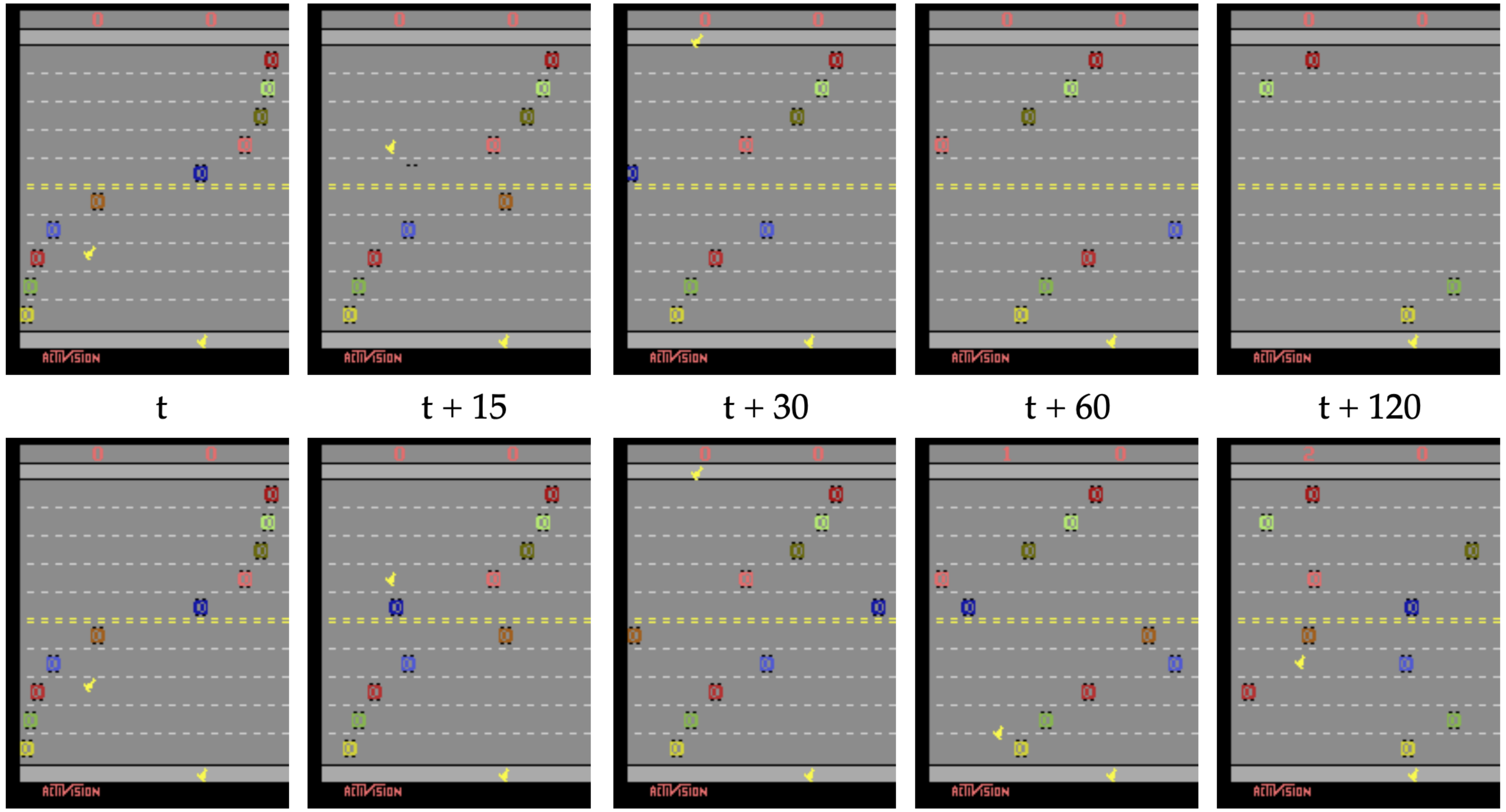
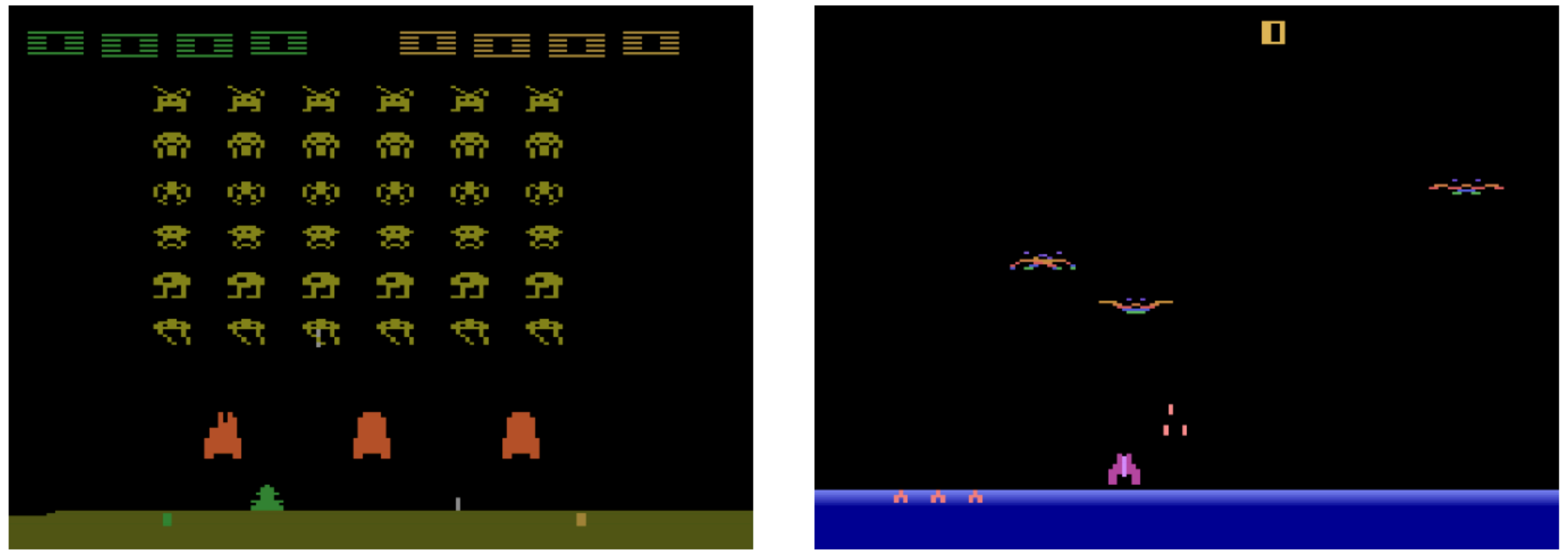


Figure 5: **Top row:** Rollout obtained with a learned model of the game FREEWAY. **Bottom row:** Ground truth. Small errors can be noticed ($t + 15$) but major errors are observed

Open problems

The rest …



(a) Space Invaders (b) Demon Attack

- **Exploration**: Thompson sampling, Optimistic Exp for NNs, model error, counts
- **Transfer**: we can transfer between similar games in a general way:
    - Not the policy, the value function, the representation …
    - Promising empirical study: https://arxiv.org/pdf/1810.00123.pdf
- **Options and temporal abstraction**: learning macro actions
- **Off-policy learning**: we do it (Q-learning & replay), but potential, is unrealized due to stability, SOTA focus, and biases of researchers

A long-time ago …

When nobody used GPUs, Neural nets were unpopular and people still used Mountain Car

- Empirical work can yield inflated results & specialization to particular problems
  - Can you imagine spending all your time on puddle world?
- Consider the usual setup: say MC, average reward averaged over runs
  - We can engineer the features, value func. init. & learning rate for MC
  - The same setting could be poor in puddle world, or something else (see why ALE is great)
- Single environment evaluation in RL can be problematic
- In supervised learning *overfitting* can be an issue
- In RL we have **environment and evaluation overfitting**
- Can be worse in RL, because environments can generate unlimited training data

Historical context

The RL Competition

- Started in 2005, called the NeurIPS Benchmarks and Bakeoffs
- We built software to standardized evaluation in RL: called RL-Glue
- Each year we improved the software to make it more robust to cheating:
  - Environments ran of a server
  - Limited interaction with the environment (few trial runs—one a week), then your final performance runs (limited)
  - Signed jar files and code to detect modification and other forms of cheating
- Focus of the competition was online, model-free RL & most importantly research progress
  - Problems were selected to represent fundamental, open questions

Admin

- No office hours this week
- Project team list sheet:
    - https://docs.google.com/spreadsheets/d/1f-QybvJk5V5dilsHOL9f6eNx5fAR0gFEuUekHS94MhE
- Session moderator for today: **Mihucz, Gabor**
    - https://docs.google.com/spreadsheets/d/1dbmlvduupZUCDjxU4HW2_350OVrVG-g1FoEAG-uWhMk

They still cheated

Wanted to win Tetris; figured out the model and did Dynamic programming

Our focus back then was general algorithms

Polyathlon competition track

- 5 Continuous state, discrete action problems
- 3 revealed to competitors and available for development
- 2 hidden
- Overall performance across all 5 determined best, most general agent
- People cheated:
  - if statements to detect which problem to deploy pre-trained agents
  - Guessed possible hidden problems and pre-trained agents
- Very similar goals to the ALE!

In the end little innovation and research came out of the competitions

Protecting Against Evaluation Overfitting in Empirical Reinforcement Learning

Whiteson et al (2018)

- Assume the algorithm designer is self interested!
  - We want evaluations that free designers from worry about their biases impacting evaluation
- In most cases the numerical results on a particular domain have no actual external utility:
  - nobody cares about scores in Pitfall!

Types of overfitting

Evaluation, environment, data

- Experiments suggest utility of the algorithms in some target deployment
- **Evaluation overfitting**: performing well in evaluation but not on the target distribution
- **Data overfitting**: func. learned is customized to the training data, fails to generalize to data sampled from the same environment
    - —not an issue in RL because evaluation is online from environments which are infinite data generators
- **Environment overfitting**: agent performs well on environment, but poorly on others
    - The learning algorithm is overfit rather than the function produced (policy/value-func)
    - Target distribution is the distribution over environments
    - Access to simulators actually makes this worse

Target distributions

- Typically involve multiple environments
- *Fitting*: customizing the algorithm to the target dist. at the expense of dist. outside it
- *Overfitting*: customizing the algorithm to the evaluation at the expense of the target dist.

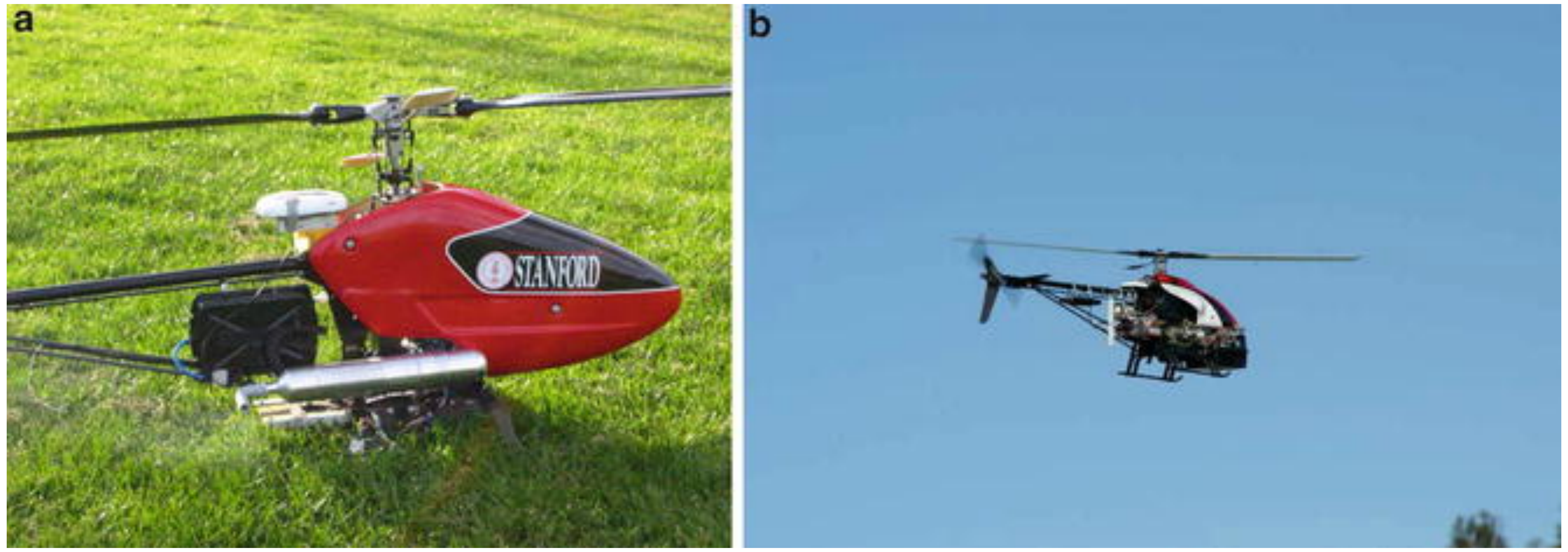
Beyond single environments

Can be good or bad

- Single using environments can lead to overfitting
- We can create a distribution over environments—starting with a single simple environment like Mountain Car
- We can then sample from this distribution evaluating an agent on multiple environments
- ***Generalized Environments***
- *G = <Θ, μ>*
- *Θ is the set of environments and μ is the distribution over the set*

Example: Helicopter hovering

- Each trial/run the agent faces a new environment:
  - Unknown wind velocity



Environment used in the RL competition

Generalized environments

Saves us from ourselves, clear experiments

- No amount of tuning can eliminate the uncertainty represented by ***G***.
  - =>fair comparisons between agents
- Makes explicit what generality is desired:
  - choice of ***G*** specifies both a target class & importance of environments in that class

How this might work

Saves us from ourselves, clear experiments

- Agent designer knows ***G***
- They can sample freely from G and tune their agent
- Then the test phase:
    - For N trials, sample an environment from ***G***
    - Train and evaluate agent
- Designer does not know exactly which environments the agent will face

Secret Generalized Environments

Don’t know *G*

- Perhaps your are thinking about deployment: flying a helicopter
    - We don’t really know the wind conditions well
- If the designer knows ***G*** they may overfit to the uncertainty it encodes
- G can be hidden and access to it limited during tuning

Meta-Generalized Environments

It’s distributions of environments all the way down

- We could have a distribution over a set generalized environments

Summarizing Performance Generalized Environments

Simple averaging won’t due

- Environments could have different reward scales
    - Because some environments are harder, run longer, or have different reward scales
- We could do a sign test
- This question comes up in lots of places: competitions, time series forecasting
    - Paper notes people don’t do this in SL

A simple experiment

To evaluate this evaluation scheme

- Compare usual tile coding, tuned by hand, with an adaptive one
- Conventional tile coding: need to know the ranges of the state variables
- The Range-adaptive TC:
  - We need to stretch the tile coder, whenever we get a point outside the range
  - So we need to adjust the weights associated with each tile on the fly
- What are the challenges in showing RA-TC is better than TC on just one environment?

Generalized classic control

- Mountain Car, Acrobot, and Puddle World
- Each is a generalized environment:
    - uniform(-n,n) applied to action, sampled from uniform(0,.5)
    - Transformation function applied to observation (scale, invert, translate,…)
    - Random starts or not
- Compare fixed-range TC, adaptive-range TC, and cheater TC

Tuning agent procedure

So much compute

- Sample 25 generalized environments
- Test each type of agent for one trial on each environment, sweeping:
    - Alpha \in {2.0, 1.0, .75, .5, .25, .125, 0.06125}
    - Lambda \in {.99, .95, .9, .75, .5, .25, .125, .06125, 0}
    - Tiles \in {4,8,16,32}
    - Tilings \in {4,8,16,32}
    - And some other stuff …
    - 3024 configurations, 70k time steps
- Candidate tuned agent was the best across generalized environments

Results

Adaption matters

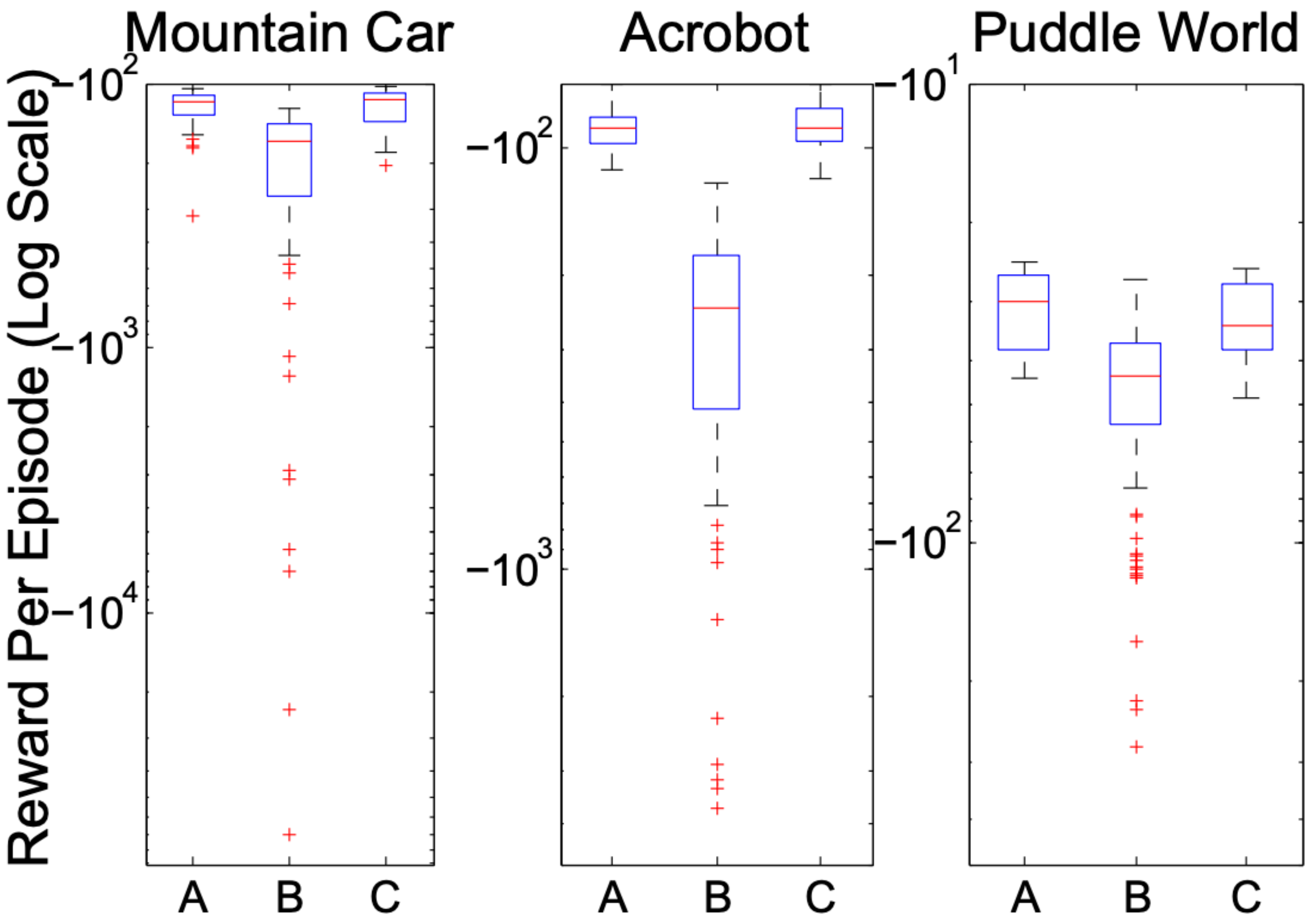


Fig. 2. Generalized methodology results for the adaptive (A), baseline (B), and cheater (C) approaches on three generalized environments.

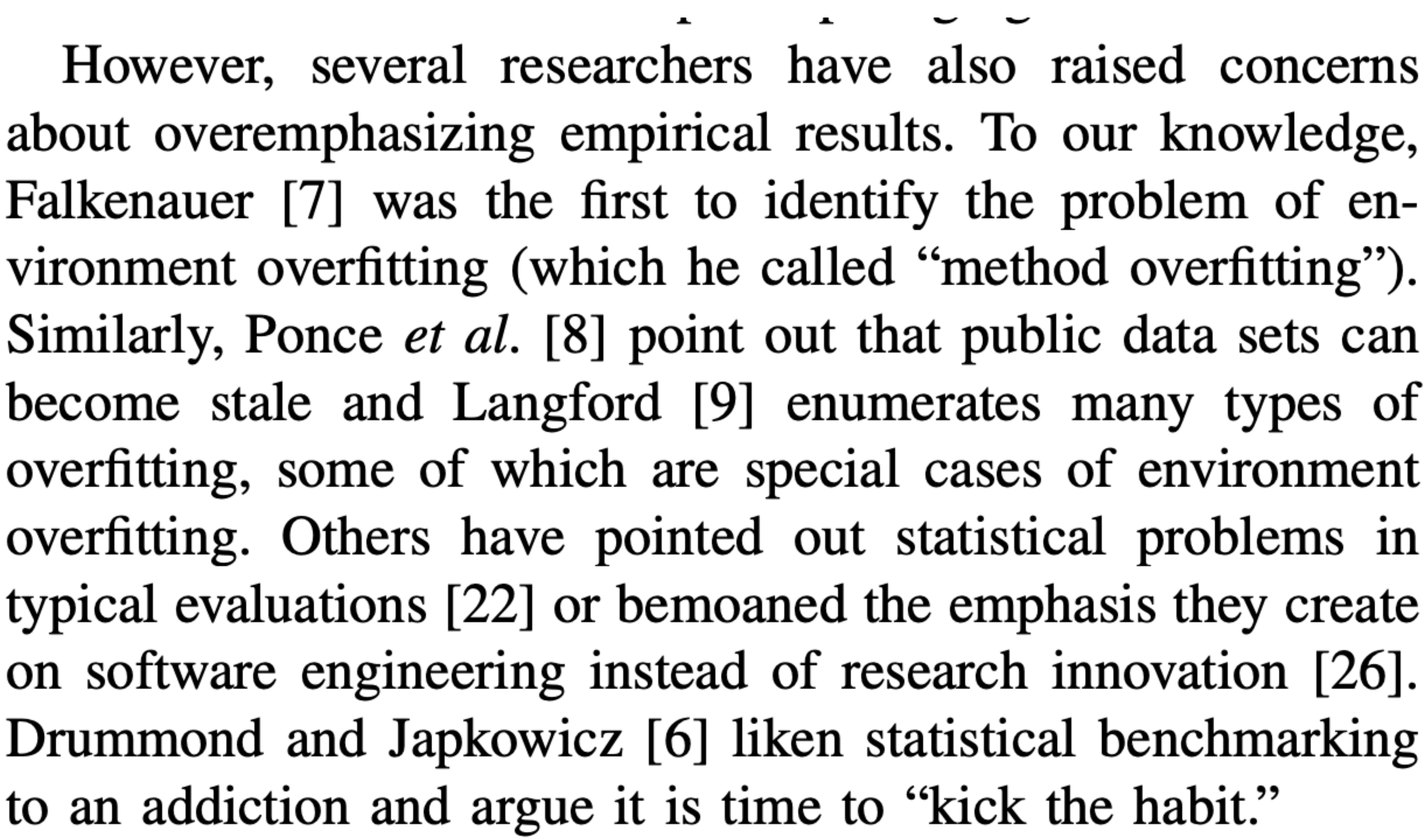
- Each tuned agent was tested on 100 additional environments

Results summary

Relevant today

- Clear experiment showing the adaptive method is useful & non-adaptive method—even with tuning—was not as effective
- Enormous compute required
- Open problem on how to exactly aggregate performance across environments
- Is this close to what we are getting with the Atari (ALE) or AIGym problem sets?
- Could we do this with modern large scale domains? Would it lead to more general agents?

Don’t over-focus on benchmarks



However, several researchers have also raised concerns about overemphasizing empirical results. To our knowledge, Falkenauer [7] was the first to identify the problem of environment overfitting (which he called “method overfitting”). Similarly, Ponce *et al*. [8] point out that public data sets can become stale and Langford [9] enumerates many types of overfitting, some of which are special cases of environment overfitting. Others have pointed out statistical problems in typical evaluations [22] or bemoaned the emphasis they create on software engineering instead of research innovation [26]. Drummond and Japkowicz [6] liken statistical benchmarking to an addiction and argue it is time to “kick the habit.”

Overfitting, target dists & ALE

- Whiteson: research **towards general agents**
    - with good evaluation methodology & Mountain Car
- ALE : a challenge problem to inspire research **towards general agents**
- ALE research fell victim to bad evaluation methodology
- Two insights here:
    - evaluation & environment overfitting in ALE can happen, just like MC!
    - empirical methodology is a constant back & forth

Your questions

- Average reward: What are good options for environments to test these algs on? (Not the 2-cycle world)
    - What’s a good performance measure?
- What topographical structure are these TD networks constraint to have?
- In the black and white world example to illustrate the need for tracking in stationary problems - How exactly is tracking incorporated in the final implementation?
- What are some pitfalls in randomly generating procedural (chain) MDPs and Gridworlds?