Reinforcement learning experiments that matter!

Adam White
University of Alberta, Dept of Computing Science
Alberta Machine Intelligence Institute (Amii)
RL Core Technologies
CIFAR Canada AI Chair

Reinforcement learning experiments that matter!

Machine Learning that Matters

Kiri L. Wagstaff KIRI.L.WAGSTAFF@JPL.NASA.GOV
Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109 USA

Abstract

Much of current machine learning (ML) research has lost its connection to problems of import to the larger world of science and society. From this perspective, there exist glaring limitations in the data sets we investigate, the metrics we employ for evaluation, and the degree to which results are communicated back to their originating domains. What changes are needed to how we conduct research to increase the impact that ML has? We present six Impact Challenges to explicitly focus the field's energy and attention, and we discuss existing obstacles that must be addressed. We aim to inspire ongoing discussion and focus on ML that matters.

tively solved spam email detection (Zdziarski, 2005) and machine translation (Kochn et al., 2003), two problems of global import. And so on.

And yet we still observe a proliferation of published ML papers that evaluate new algorithms on a handful of isolated benchmark data sets. Their "real world" experiments may operate on data that originated in the real world, but the results are rarely communicated back to the origin. Quantitative improvements in performance are rarely accompanied by an assessment of whether those gains matter to the world outside of machine learning research.

This phenomenon occurs because there is no widespread emphasis, in the training of graduate student researchers or in the review process for submitted papers, on connecting ML advances back to the larger

Deep Reinforcement Learning that Matters

Peter Henderson[1*], Riashat Islam[1,2*], Philip Bachman[2]
Joelle Pineau[1], Doina Precup[1], David Meger[1]
[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

Abstract

In recent years, significant progress has been made in solving challenging problems across various domains using deep reinforcement learning (RL). Reproducing existing work and accurately judging the improvements offered by novel methods is vital to sustaining this progress. Unfortunately, reproducing results for state-of-the-art deep RL methods is seldom straightforward. In particular, non-determinism in standard benchmark environments, combined with variance intrinsic to the methods, can make reported results tough to interpret. Without significance metrics and tighter standardization of experimental reporting, it is difficult to determine whether improvements over the prior state-of-the-art are meaningful. In this paper, we investigate challenges posed by reproducibility, proper experimental techniques, and reporting procedures. We illustrate the variability in reported metrics and results when comparing against common baselines and suggest guidelines to make future results in deep RL more reproducible. We aim

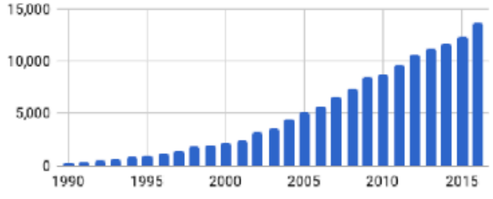
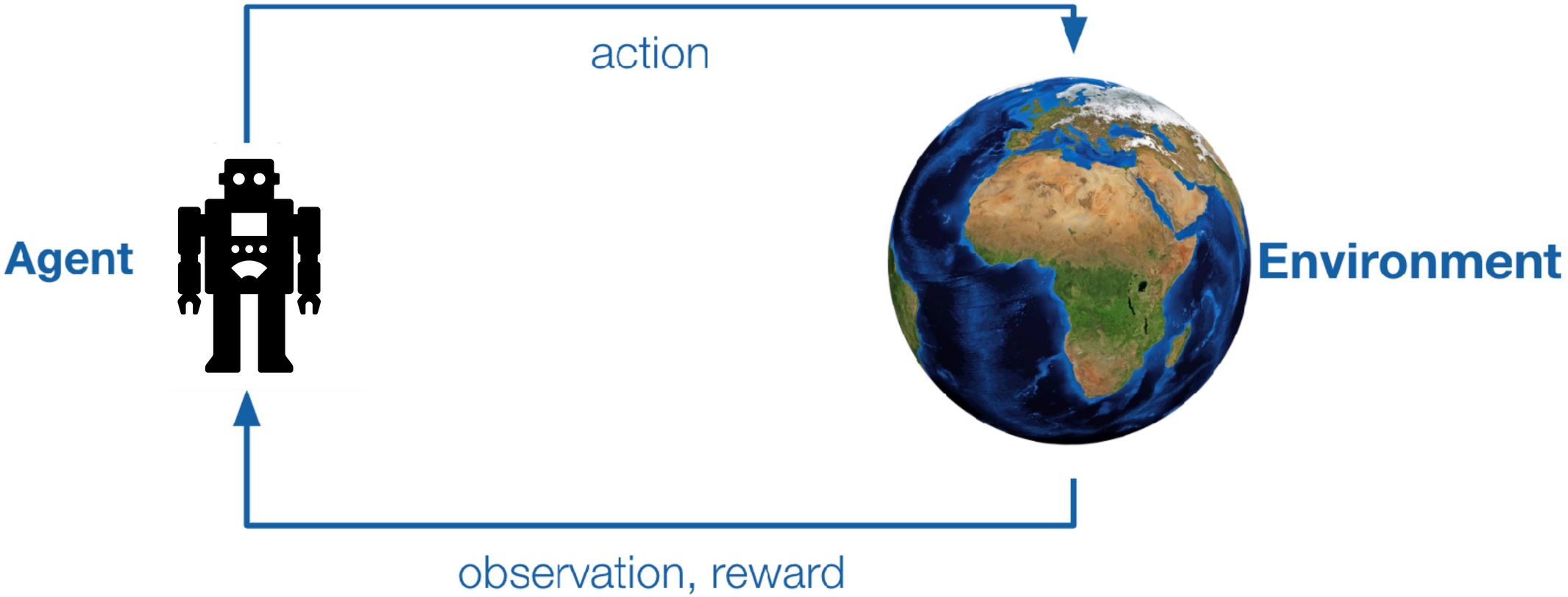


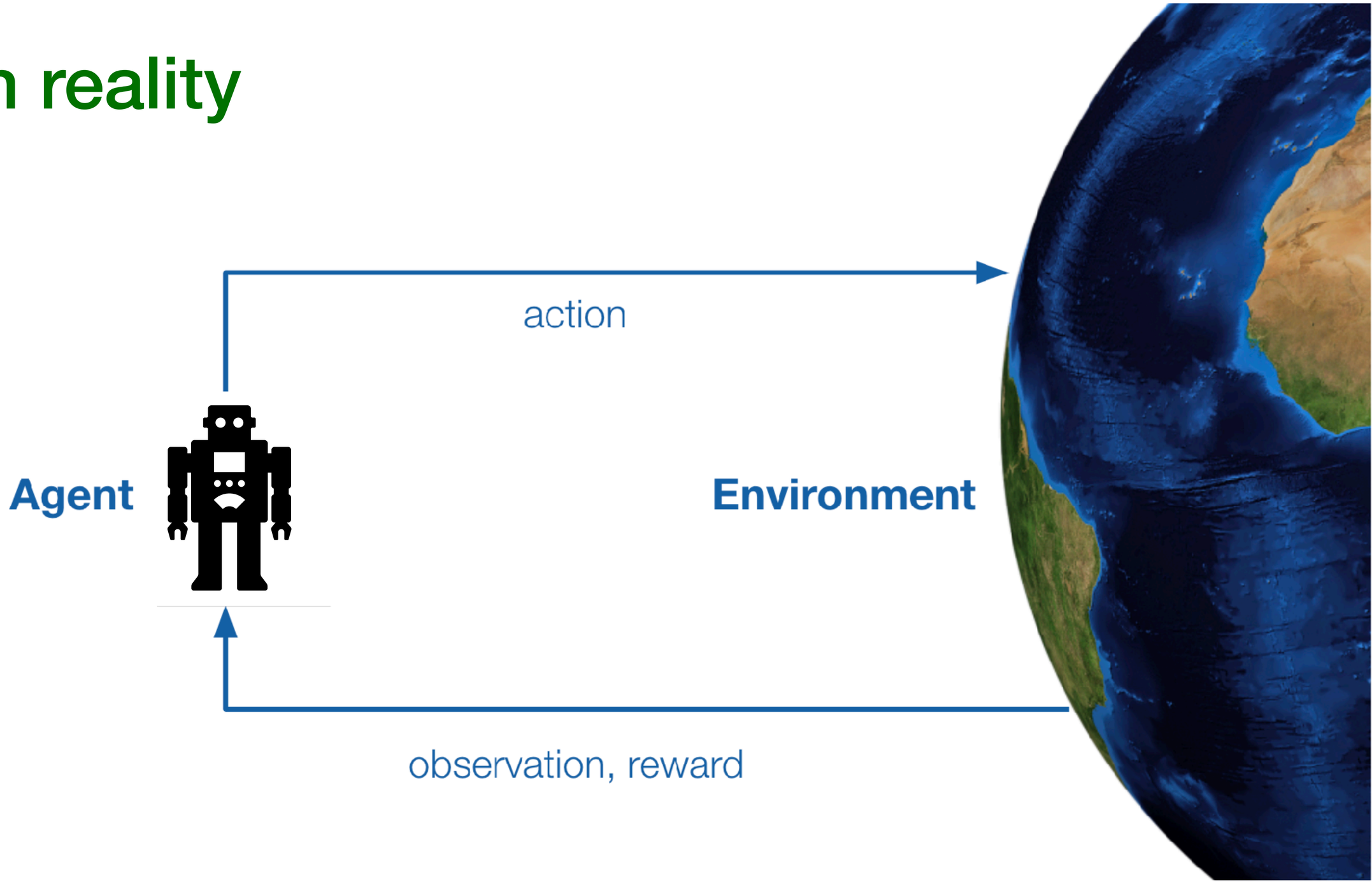
Figure 1: Growth of published reinforcement learning papers. Shown are the number of RL-related publications (y-axis) per year (x-axis) scraped from Google Scholar searches.

fects of random seeds or environment properties). We investigate these sources of variance in reported results through a representative set of experiments. For clarity, we focus

Reinforcement Learning



But in reality



Most progress in RL requires empirical validation

What are your **goals** when running a reinforcement learning experiment?

1. **solve a simulator** (e.g., chip design), where the **optimal policy** is a valuable artefact; unlike *pong*!

2. **SOTA on a benchmark**, where the benchmark crystallizes some key unresolved challenges & requires new algorithmic innovation!

3. Understanding **intelligence/mind**

4. Towards **deployment in the real-world**

Regardless, we want to compare algorithms carefully and scientifically as possible.

Outline

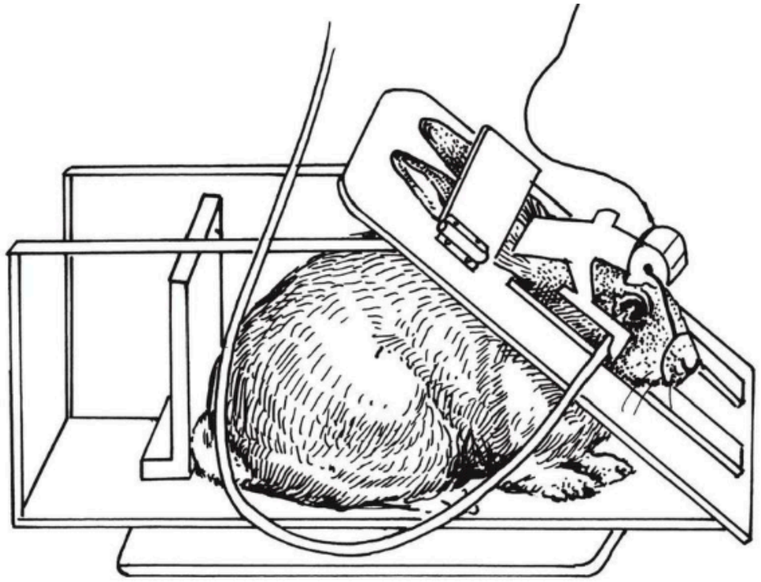
- **Part 1: don't waste your time**
  - a case study
  - you need more runs than you think!
  - better statistical tools
- Part 2: pesky hypers
  - the hyperparameter crisis in RL
  - dealing with hyperparameter when you don't have a simulator

RL is a branch of science

- Science is the study of the natural world; posing **questions** and seeking **answers**
- In RL we study computing, not the natural world!
- We invent the problem setting, environment (MDP), the agent (algorithm), and the experiment protocol
  - This produces a dynamical system that we seek to understand
- At the end of the day empiricists, theorists, algorithm designers are all seeking the same thing: **new insights & understanding — *knowledge***

High-stakes empirical research: animal learning

- Consider conducting eye-blink experiments with rabbits (i.e., classical conditioning)
    - The goal is to understand how rabbits come to predict stimuli
- Each day the researcher must run several rabbits through repeated trials of the experiment
- Researchers are cognizant of many important details: lighting, how they handle the animals, not to wear strong scents, temperature, etc



RL is easier than animal learning!

- We completely construct the **environment** the **agent** operates in and perfectly control the experiment **protocol** (robotics is different)
- We can control and isolate sources of variation
  - We can control “genetic” differences in individual agents
- We can instrument our experiments, collecting whatever stats we deem relevant
- We can run repeated independent trials, in parallel no less!!
- We do all of this on computers many times faster than realtime—spinning up a new experiment takes minutes

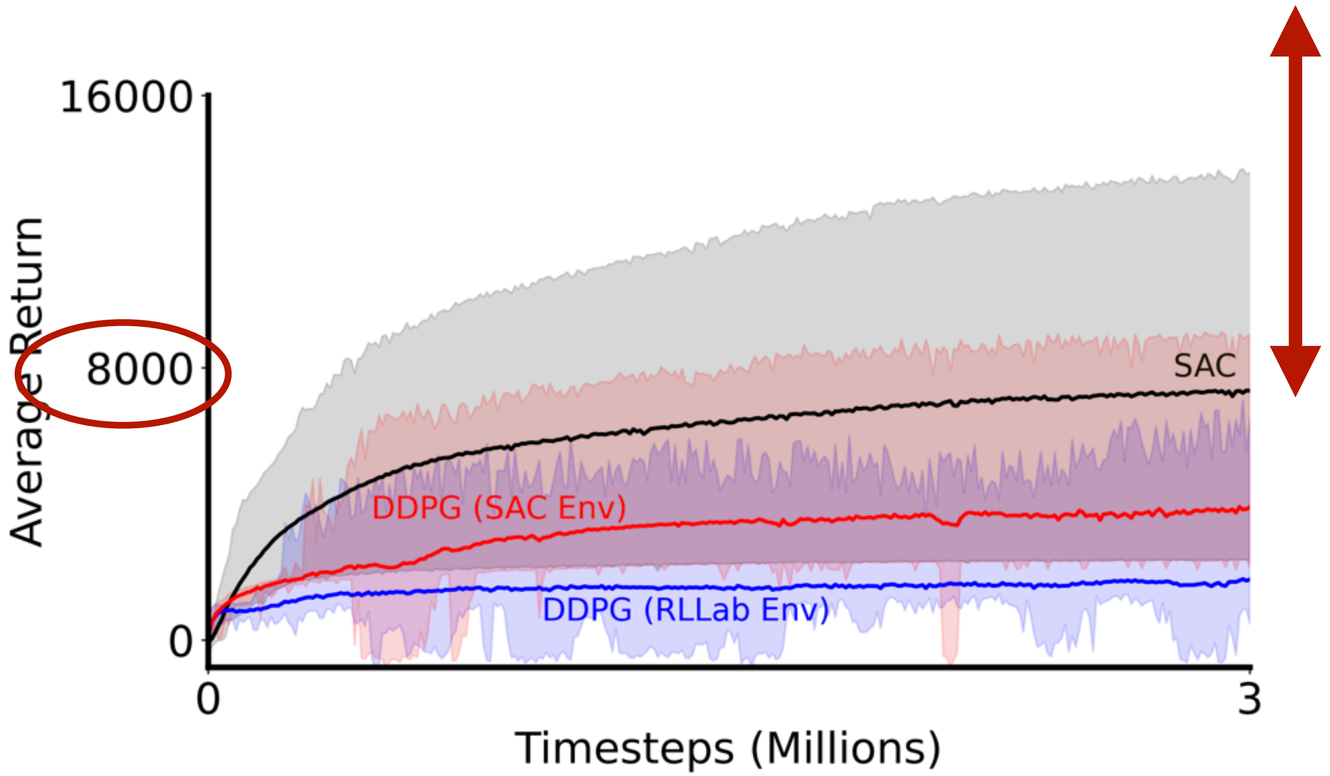
And yet poor empirical practice is common

A case study

- *Objective*: reproduce a result from the literature with two well-known policy gradient algorithms

A case study

- *Objective*: reproduce a result from the literature with two well-known policy gradient algorithms



Results compiled by Samuel Neumann

Case study findings

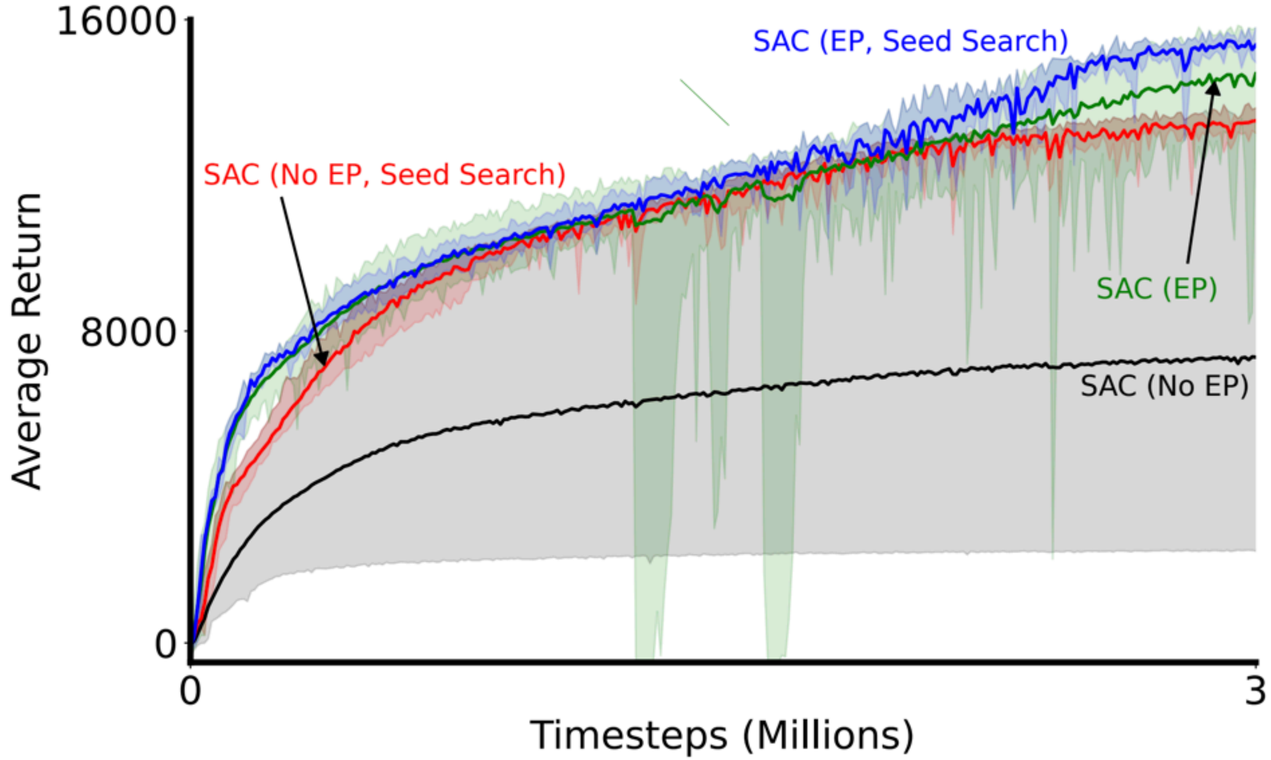
We could not reproduce the results using 30 runs and extensive hyperparameter sweeping and ablation of architecture/implementation choices

Case study findings

- **We could not reproduce the results using 30 runs and extensive hyperparameter sweeping and ablation of architecture/implementation choices**
  - Repo contains wrappers for two different code bases; unclear which implementation of baseline was used
  - Neither the paper nor repo specify the hyperparameters for baseline methods
  - Repo code includes algorithmic components not mentioned in the paper

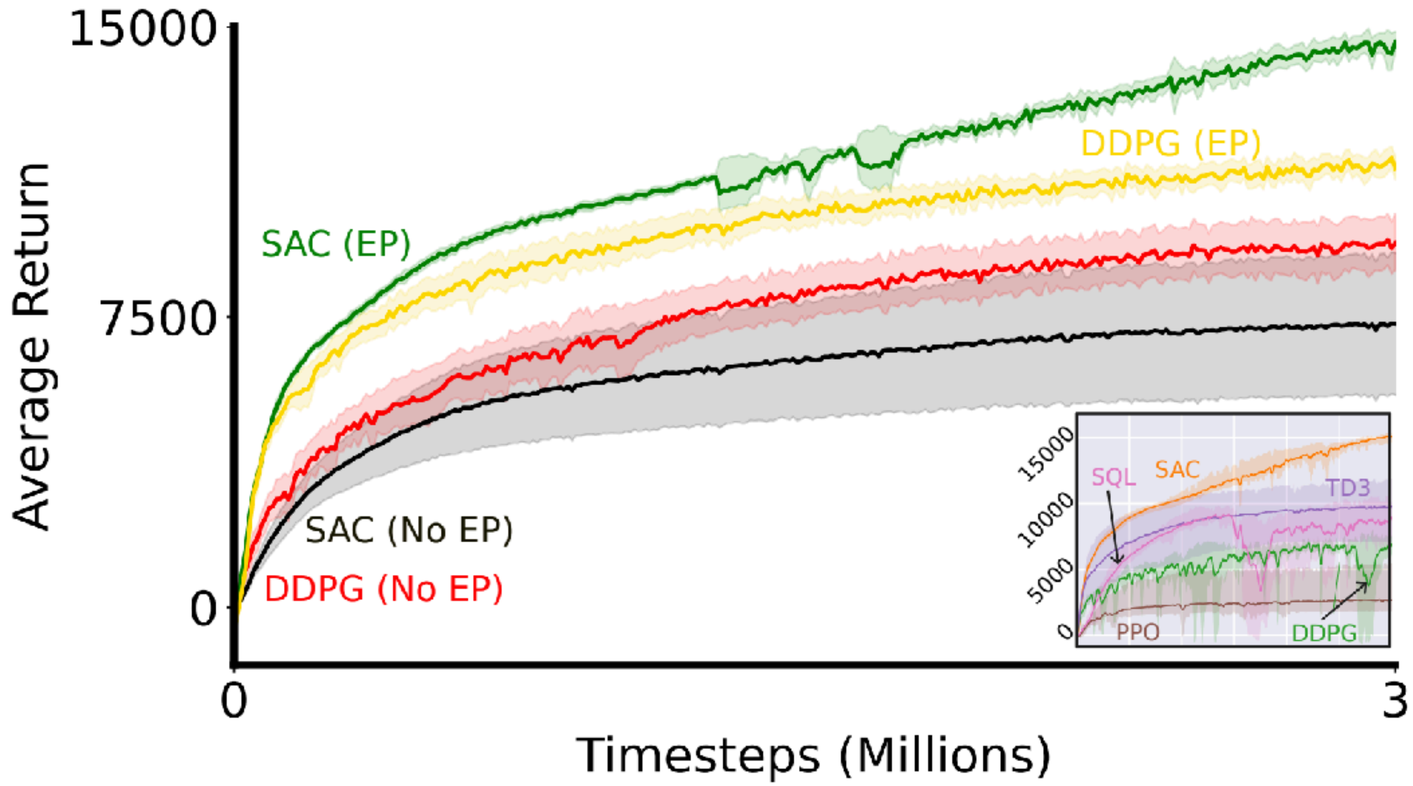
The MythBusters approach

- We noticed some runs were good, some terrible
- We noticed the repo includes code to search for good random seeds
  … or maybe they got 5 lucky seeds
- Another undocumented choice: initial exploration period



Tuning the baseline

- Better hypers & better action selection noise
- Possible error in updating on episode cutoff—not an environment termination



Our troubles are not for a lack of guidance

- **Crash course in statistics for RL:**
  - *A hitchhiker's guide to statistical comparisons of reinforcement learning algorithms* ~Colas et al
- **Dealing with hyperparameters** when comparing algorithms:
  - *Evaluating the Performance of Reinforcement Learning Algorithms* ~Jordan et al
- **Insights from small scale experiments:**
  - *Revisiting Rainbow: Promoting more insightful and inclusive deep reinforcement learning research* ~ Ceron & Castro
- **Aggregating performance over multiple environments (IQM):**
  - *Deep Reinforcement Learning at the Edge of the Statistical Precipice* ~ Agarwal et al

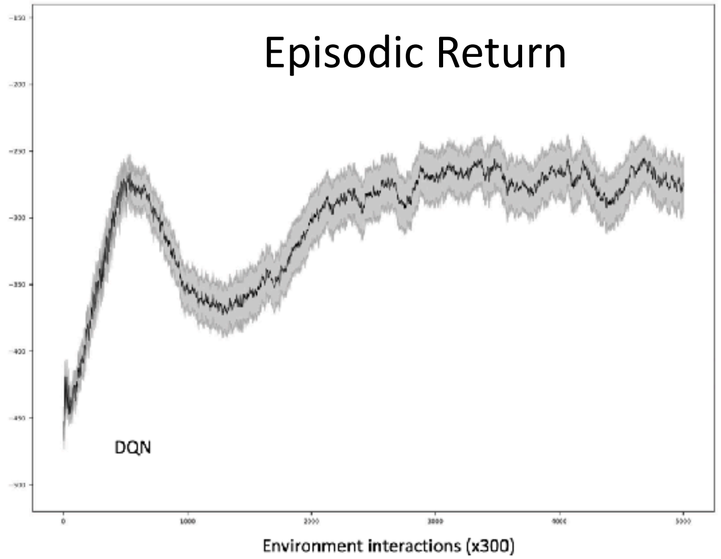
Running good experiments is possible,
but you have to care about not wasting your own time

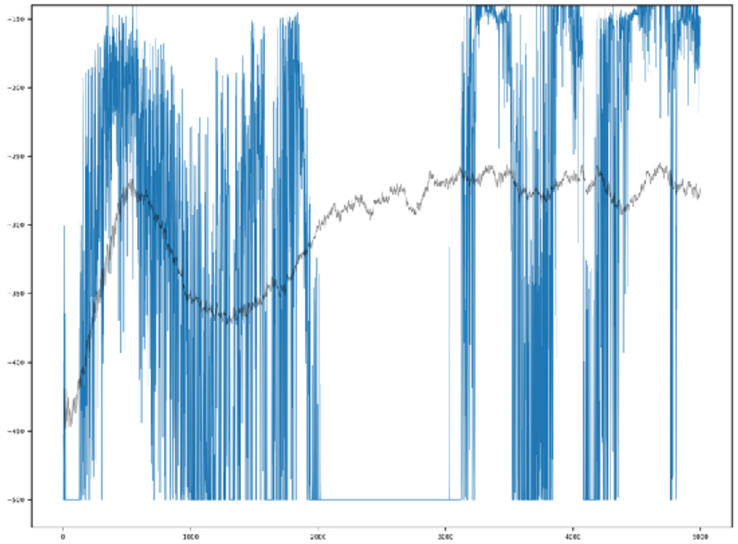
Outline

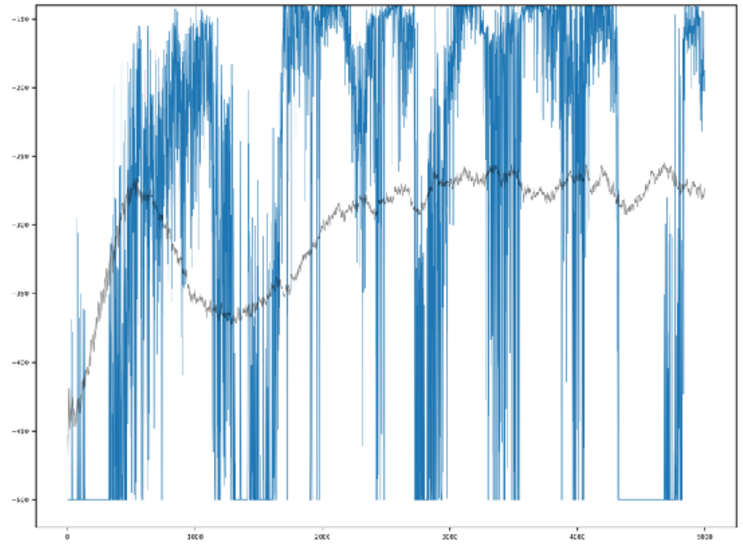
- Part 1: don't waste your time
  - a case study
  - **you need more runs than you think!**
  - better statistical tools
- Part 2: pesky hypers
  - the hyperparameter crisis in RL
  - dealing with hyperparameter when you don't have a simulator

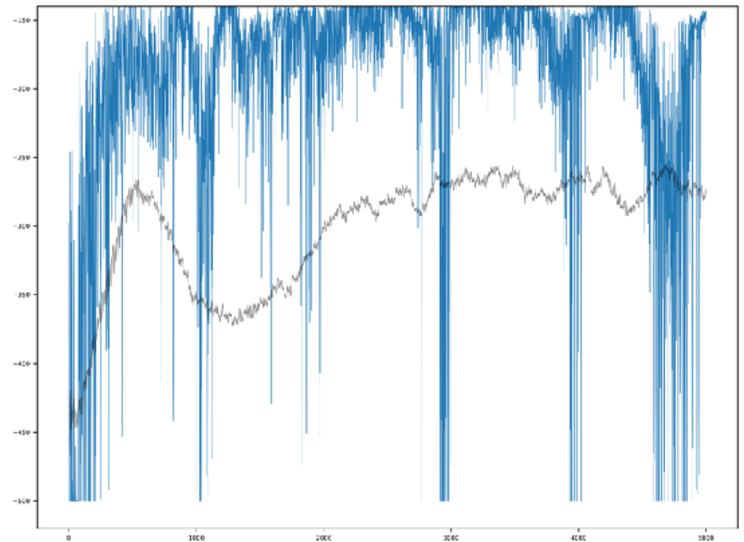
Let's look at some data

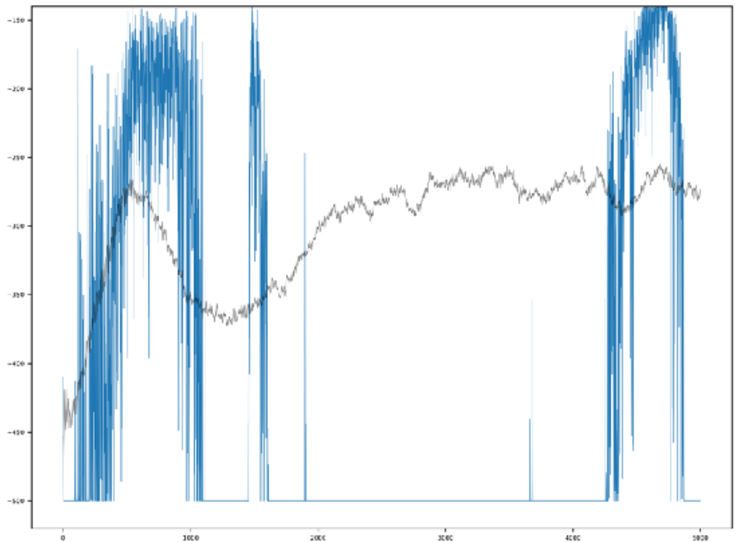
- DQN on Mountain Car, averaged over 200 runs

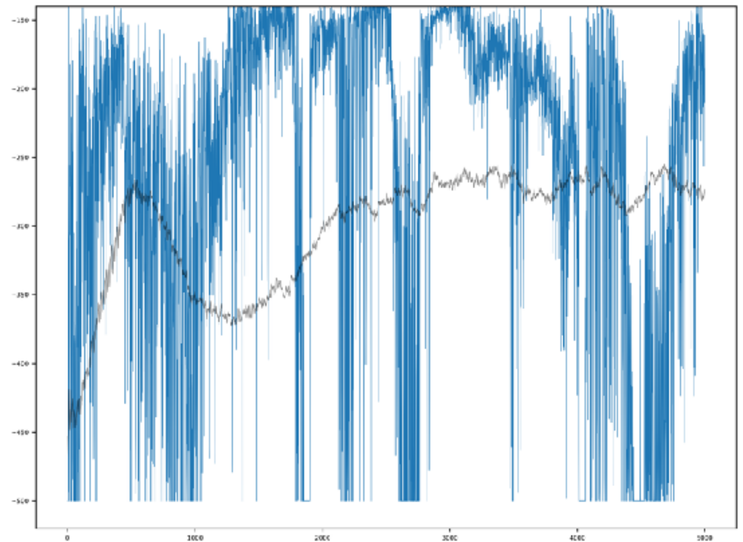




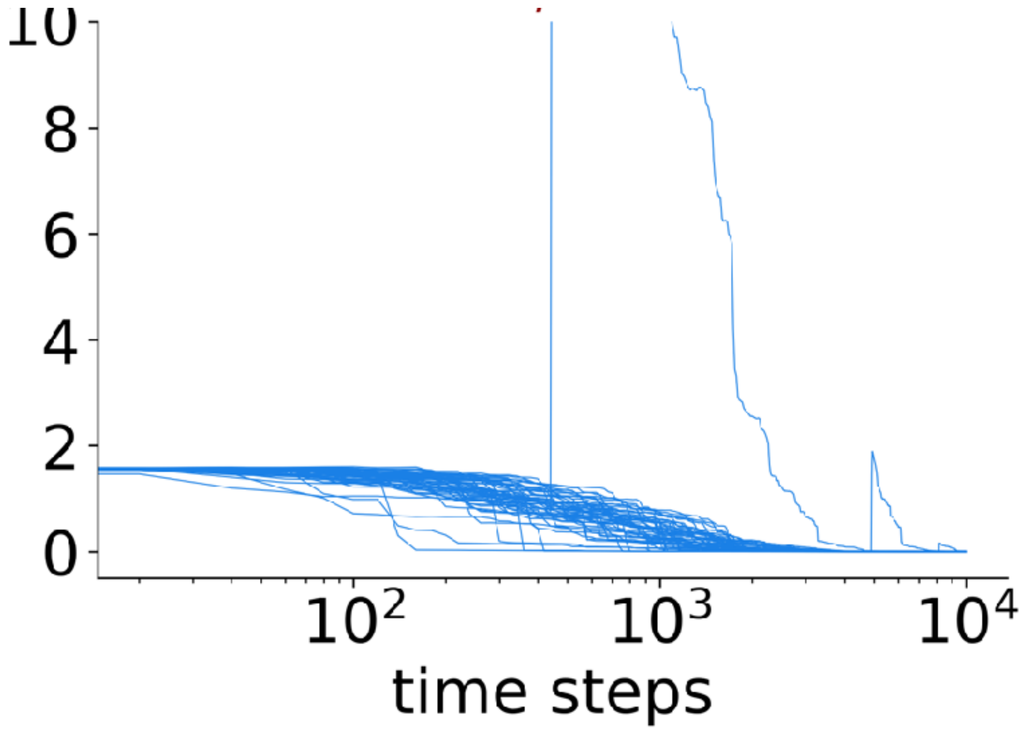








More data



N-step off-policy TD on Baird's star-counterexample

It's much worse than you think

- Imagine 10,000 labs ran the same RL experiment
- Goal: **rank four algorithms** using as **few seeds as possible**
- How many seeds do you think they would need to be statistically confident that you correctly identified the best algorithm?
  - on toy problems

It's much worse than you think

Table 1: Chance of incorrect claims

	3 runs
Acrobot	47%
Cartpole	7%
CliffWorld	54%
LunarLander	16%
MountainCar	22%
PuddleWorld	18%

Andrew Patterson, Samuel Neumann, Raksha Kumaraswamy, Martha White, and Adam White (2024). The Cross-environment Hyperparameter Setting Benchmark for Reinforcement Learning. RL Journal.

Do it for science!

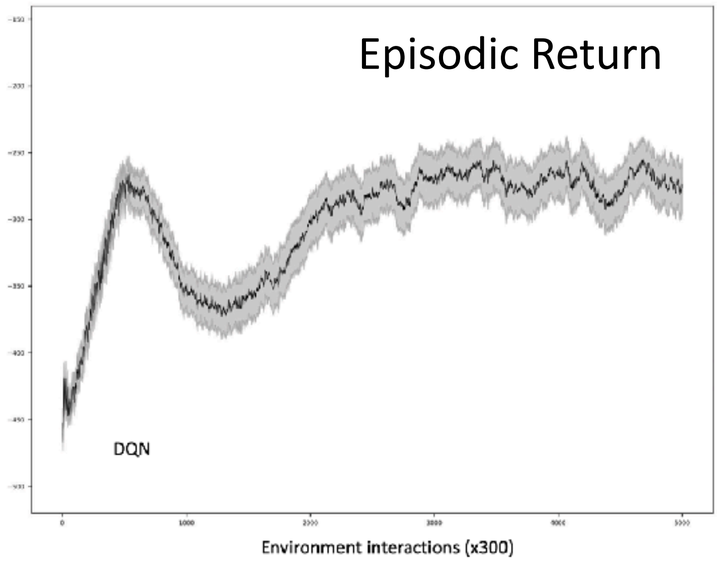
- Scientists don't report results from a study of 3 rats! **Neither should you!!**
- In the natural sciences the **robustness** and **generality** of a result is established via independent replications in followup research:
  - Slightly different apparatus, individuals (rats), protocols, etc
  - Eventually researchers explore variations: e.g., subspecies, tasks, …
- We can ensure **reproducibility** and explore **generality** much faster!

Outline

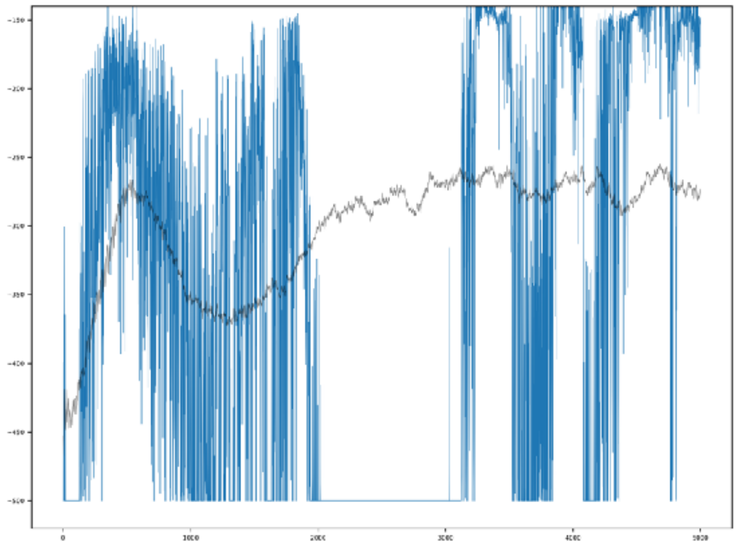
- Part 1: don't waste your time
  - a case study
  - you need more runs than you think!
  - **better statistical tools**
- Part 2: pesky hypers
  - the hyperparameter crisis in RL
  - dealing with hyperparameter when you don't have a simulator

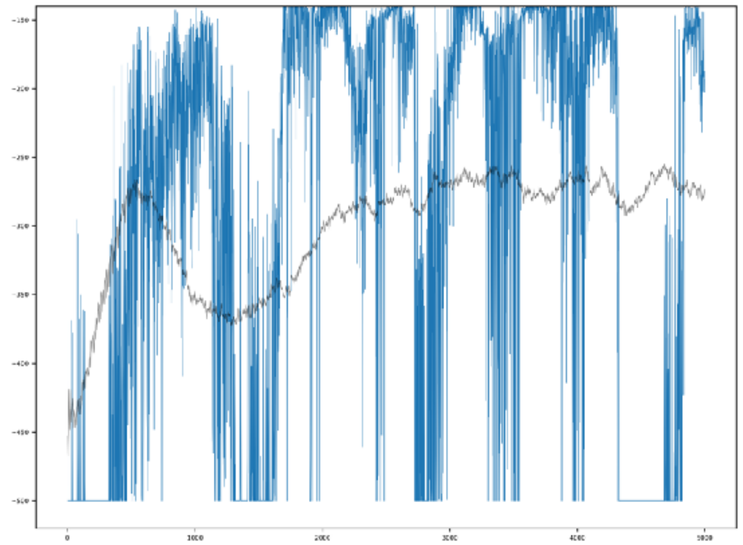
Is this a useful summary of individual agent performance?

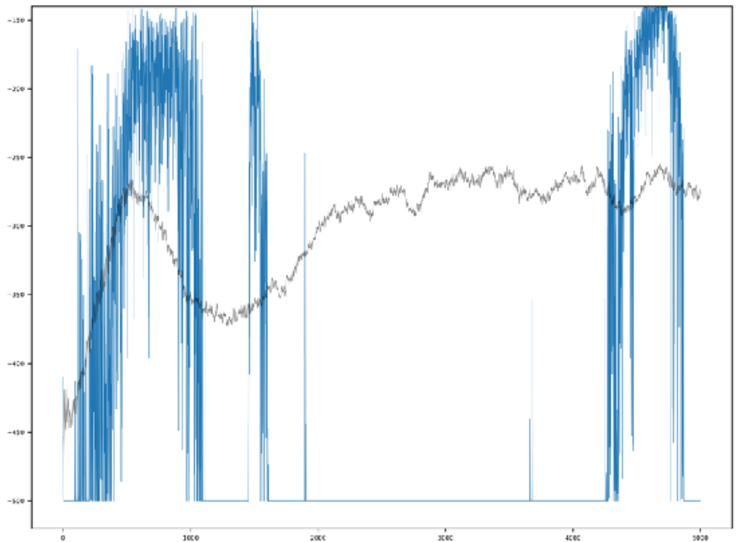
DQN on Mountain Car

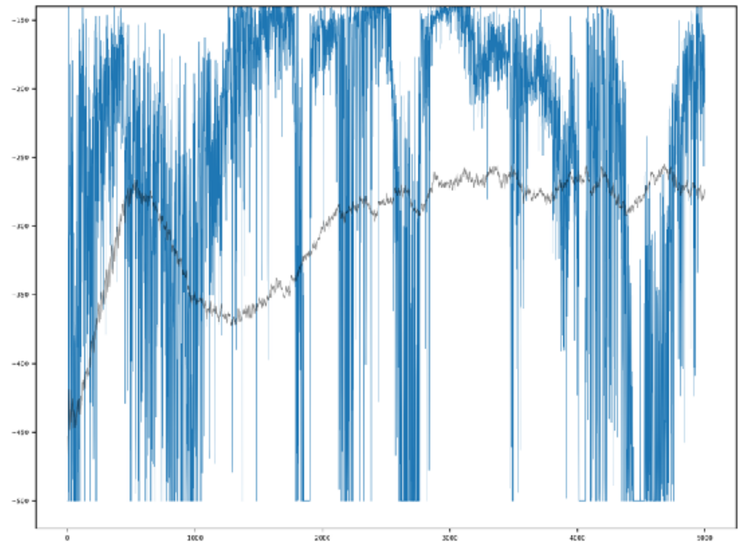


Mean with shaded region corresponding to 95% CI





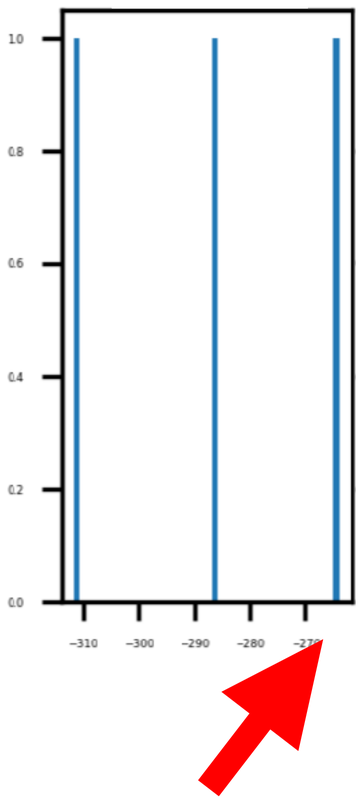




Consider the distribution of performance

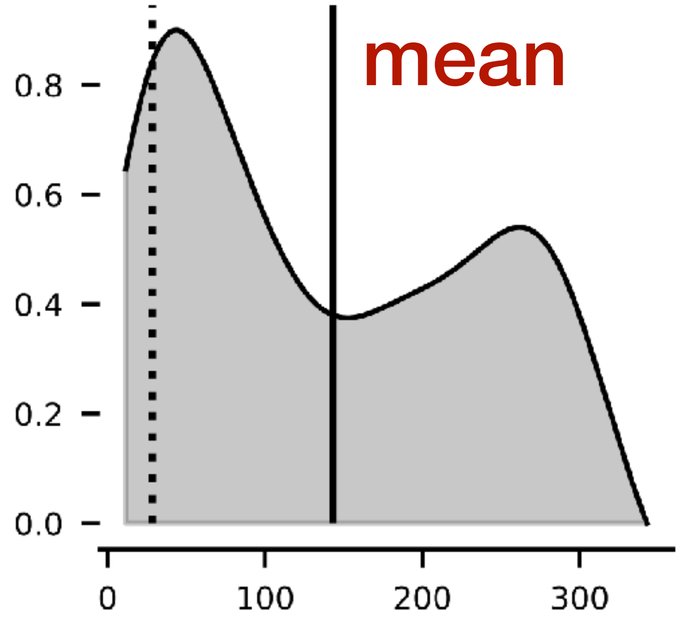
- Sarsa(λ) with tile coding, on Mountain Car

3 runs



Results compiled by Raksha Kumaraswamy

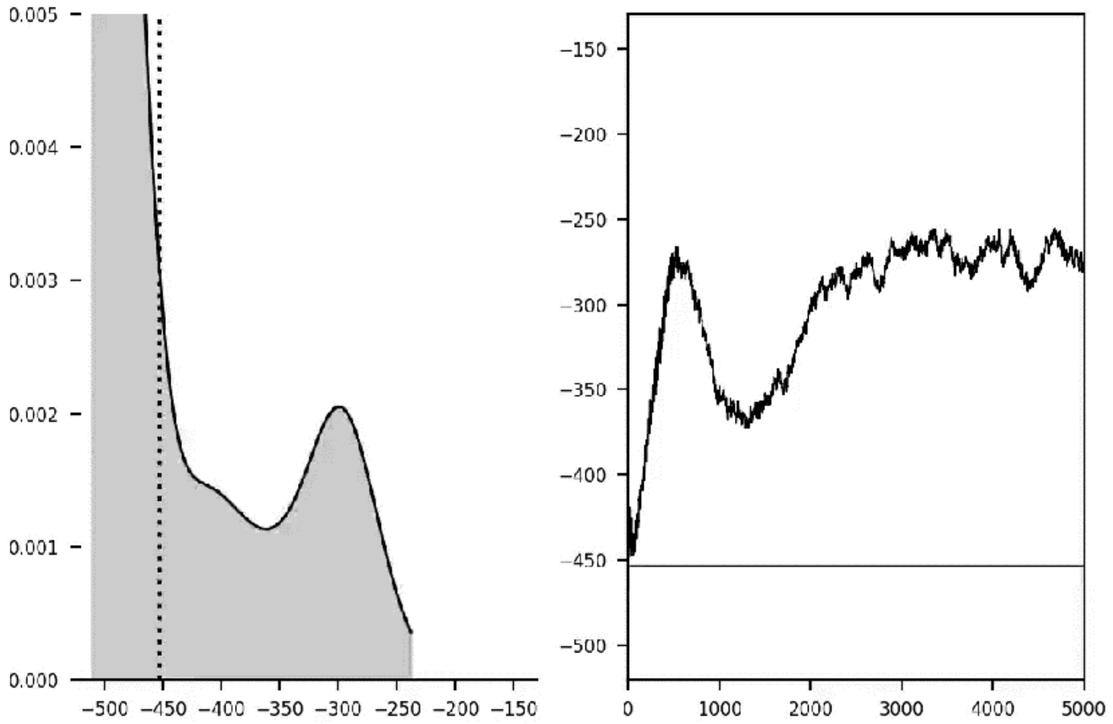
Performance distribution of DQN in Cartpole



1. An agent is likely to balance the pole for only 35 steps
2. Less likely to balance the pole for over 250 steps
3. Very few agents balance the pole for 125 steps.

These non-normal distributions appear in the wild

- In fact, the distribution changes continually throughout learning!

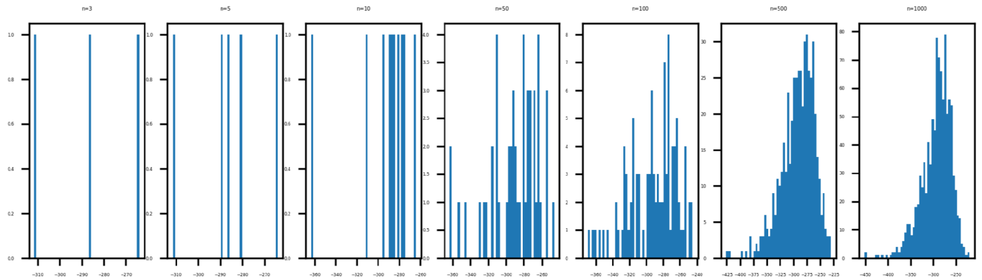


Mountain Car

Results & animation due to Andy Patterson

Check your assumptions!

- **Confidence intervals**: reflect our confidence that the true performance is within some range
  - e.g., plotting the mean learning curve with standard error shading
  - Given enough data, confidence range goes to zero
- Often there are assumptions attached to a CI, such as known variance or that the data comes from a particular distribution (e.g., normal)
- Does your experiment ***data*** match the assumptions? Did you even think of that? Is it valid to estimate the variance from 3 samples?

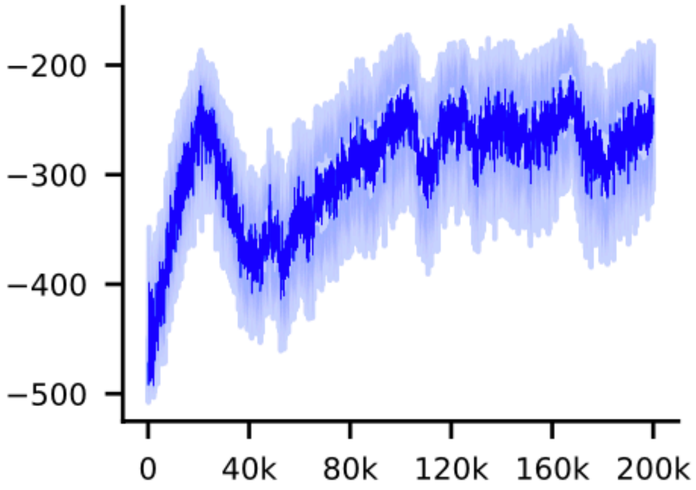


Performance distribution

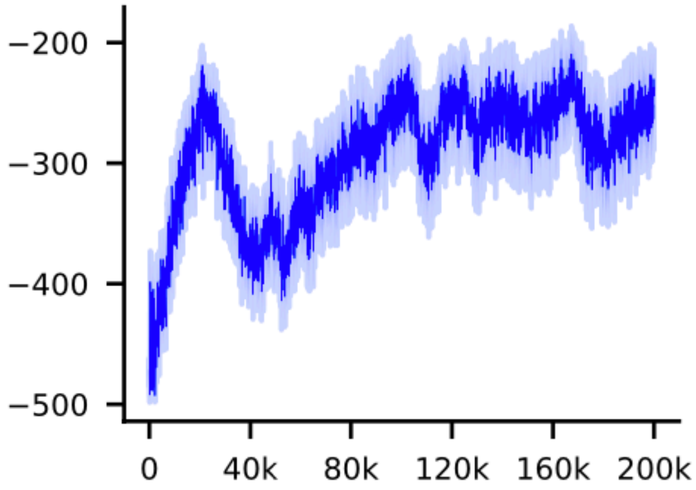
There are other tools you can use!

- Use a ***Student's t*** interval that explicitly multiples the confidence band by a scalar based on the number of runs

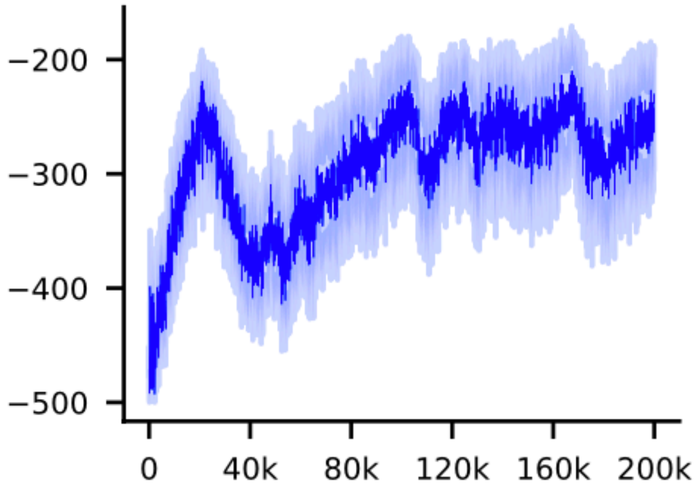
- Use ***bootstrap*** confidence intervals based on resampling your performance data repeatedly & building an empirical distribution



(a) $\alpha = 0.05$ with Student's t

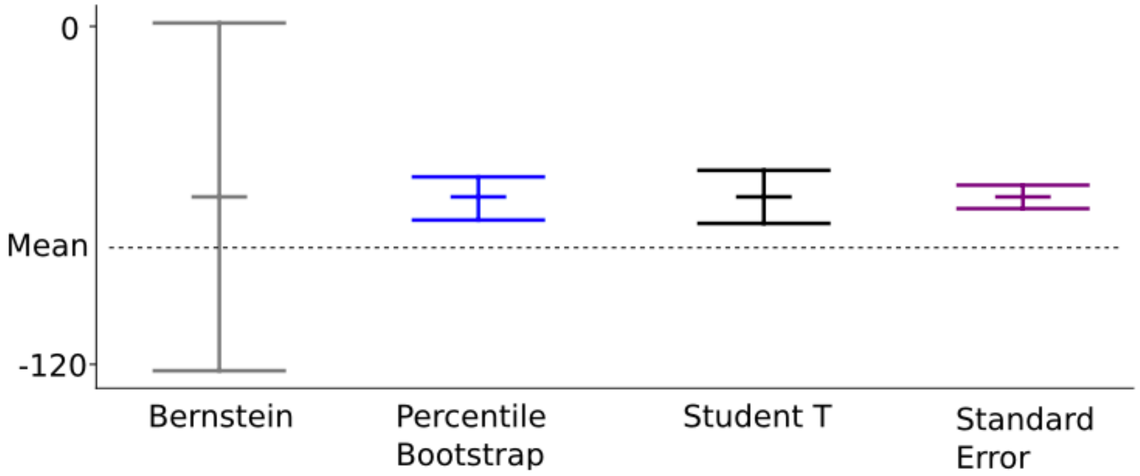


(b) $\alpha = 0.3$ with Student's t

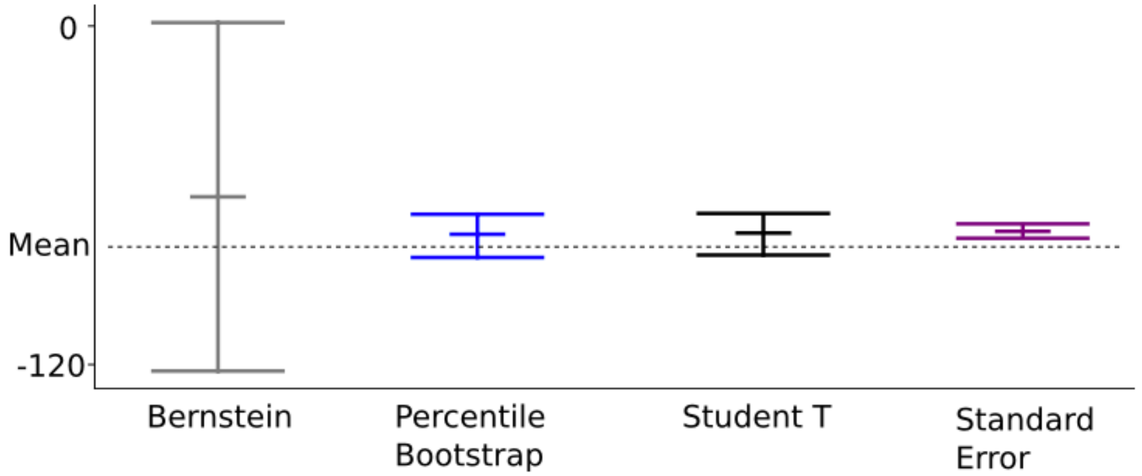


(c) $\alpha = 0.05$ with bootstrap

But! You can be confidently incorrect!



(a) Confidence intervals with 10 runs.

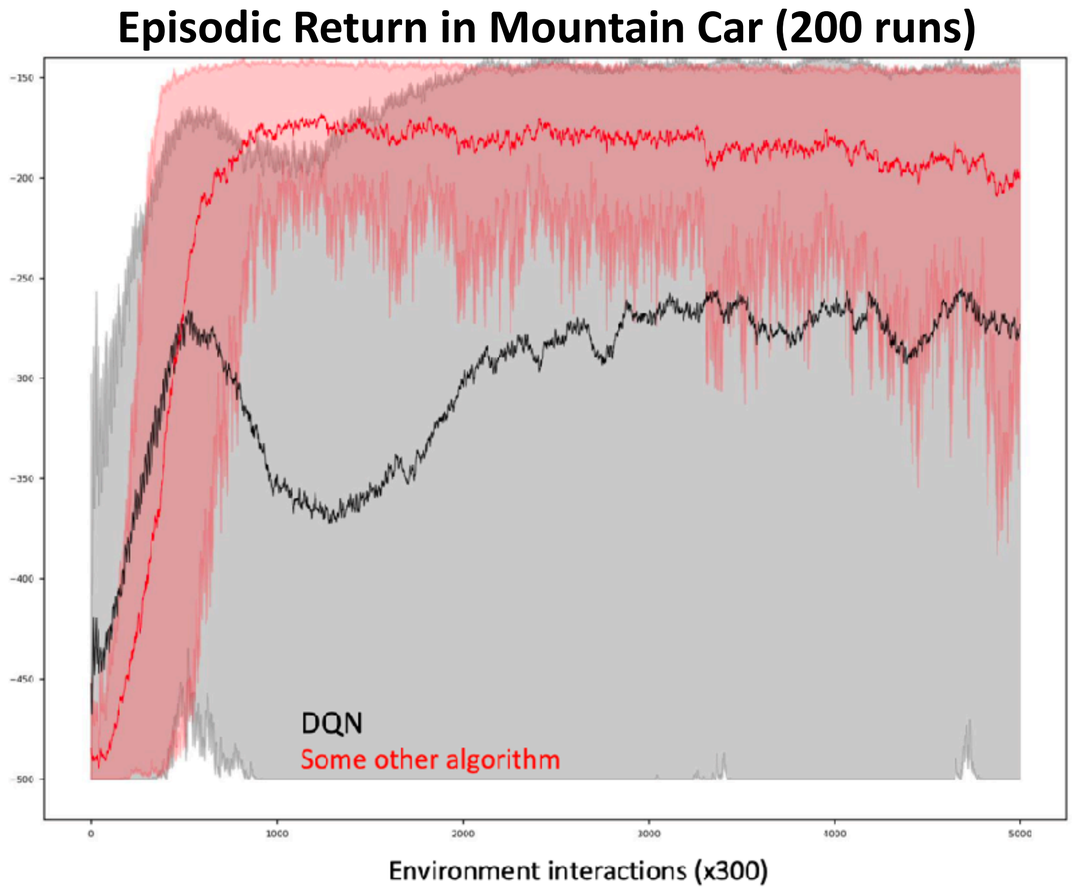


(b) Confidence intervals with 50 runs.

DQN on Puddleworld

Consider using measures that characterize variability

- **Tolerance intervals**: percentiles of agent performance + uncertainty factor based on the number of runs
- Not to be confused with confidence intervals



Do I really need more runs???

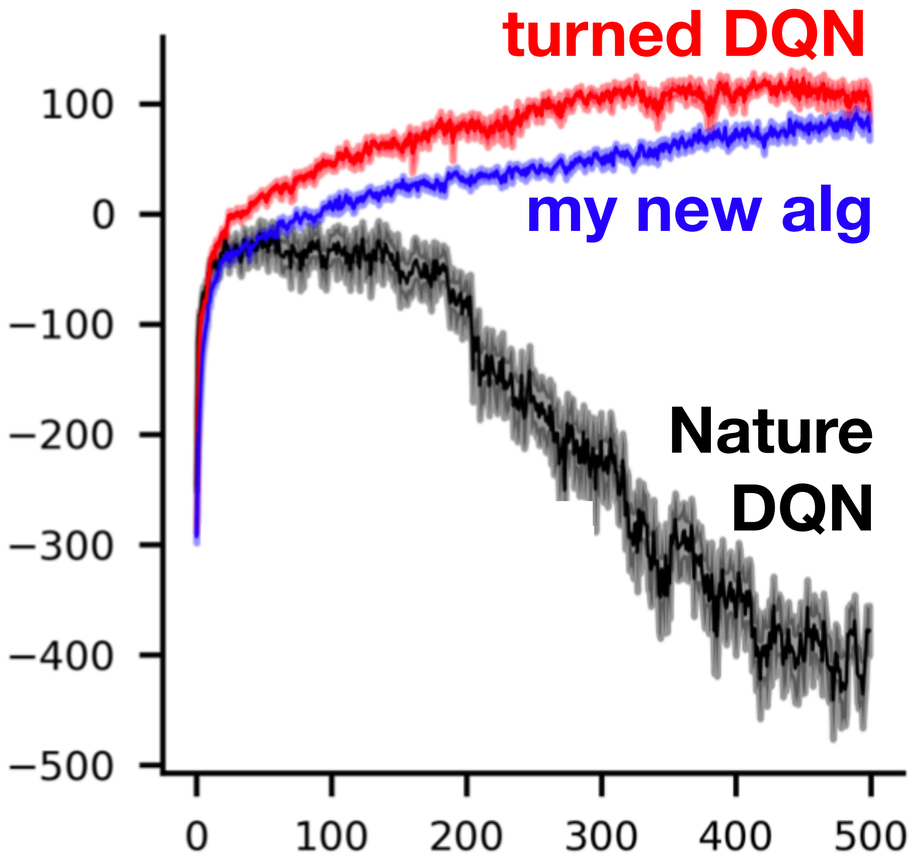
- **Common reaction**: “but my agent/environment is huge and doing more runs requires too much compute”
- **Translation**: I want to run an experiment that I don’t have enough resources for, so can you just pretend with me that this result means what I claim it does?
- **Solution**: only ask empirical questions for which you have the data, time (deadlines), and compute to answer

Everything becomes more complex when hyperparameters are involved

Untuned baselines

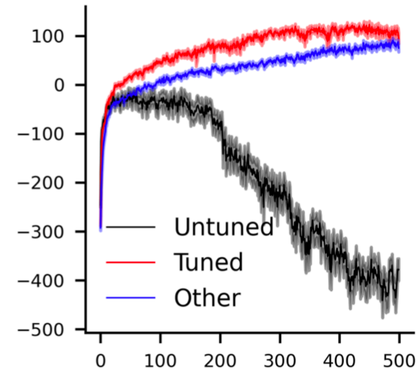
Misrepresenting the performance of other methods

- **Common practice**: you test your agent on a **new environment (gameX)** you invented. You compare **your agent** to **DQN** on gameX. You simply use Nature DQN's settings for the hyper-parameters and network architecture.
- **Problem**: DQN is tuned for Atari!! It will likely do much better on gameX if you retune the hyper-parameters



Untuned baselines

Misrepresenting the performance of other methods



- Your new algorithm is highly tuned for gameX!
- **Common reaction:** “DQN works on everything”, “nobody does that”, “it’s too much compute to do that”
- **One solution:** Use an environment for which DQN’s hyper-parameters have been tuned

What are other solutions to this challenge?

Key Messages

- You need more runs/replications than you think
- Take a statistical point of view! Think about:
  - the distributions involved, how to summarize the distribution of performance, and what assumptions you are making
- Watch out for untuned baselines, best to avoid the situation altogether!
- ***Appeal to authority fallacy***: just because you saw it in a highly cited paper doesn't make it OK

An RL Cookbook

- My lab has written a comprehensive guide to experiments in RL
- Best practices, common pitfalls, and plenty of examples & open-source code!!

Journal of Machine Learning Research 23 (2023) 1-59 Submitted 2/23; Revised X/Y; Published X/Y

Empirical Design in Reinforcement Learning

Andrew Patterson AP3@UALBERTA.CA
Samuel Neumann SFNEUMAN@UALBERTA.CA
Martha White† WHITEM@UALBERTA.CA
Adam White† AMW8@UALBERTA.CA
Department of Computing Science and Alberta Machine Intelligence Institute (Amii)
University of Alberta, Edmonton, Canada

Outline

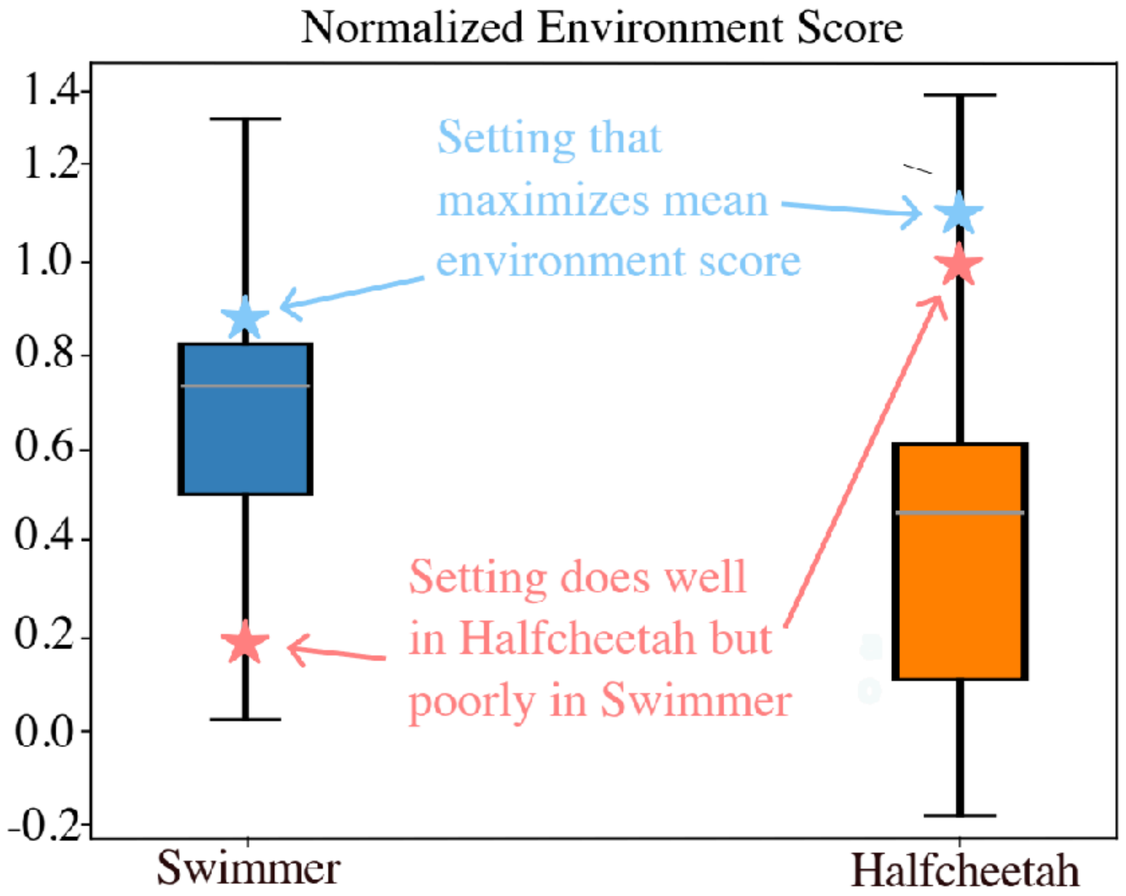
- Part 1: don't waste your time
  - a case study
  - you need more runs than you think!
  - better statistical tools
- **Part 2: pesky hypers**
  - the hyperparameter crisis in RL
  - dealing with hyperparameter when you don't have a simulator

Hyperparameters in RL

- Consider Deep Q-learning:
  - **Optimization parameters**: stepsize and momentum for Adam optimizer
  - **Exploration parameters:** initial ε, minimum ε, decay-schedule
  - **Architecture**: neural network, activation function, …
  - **Weight initialization**: can induce optimism and exploration
  - **Replay parameters**: buffer size, minimum buffer size, batch-size, …

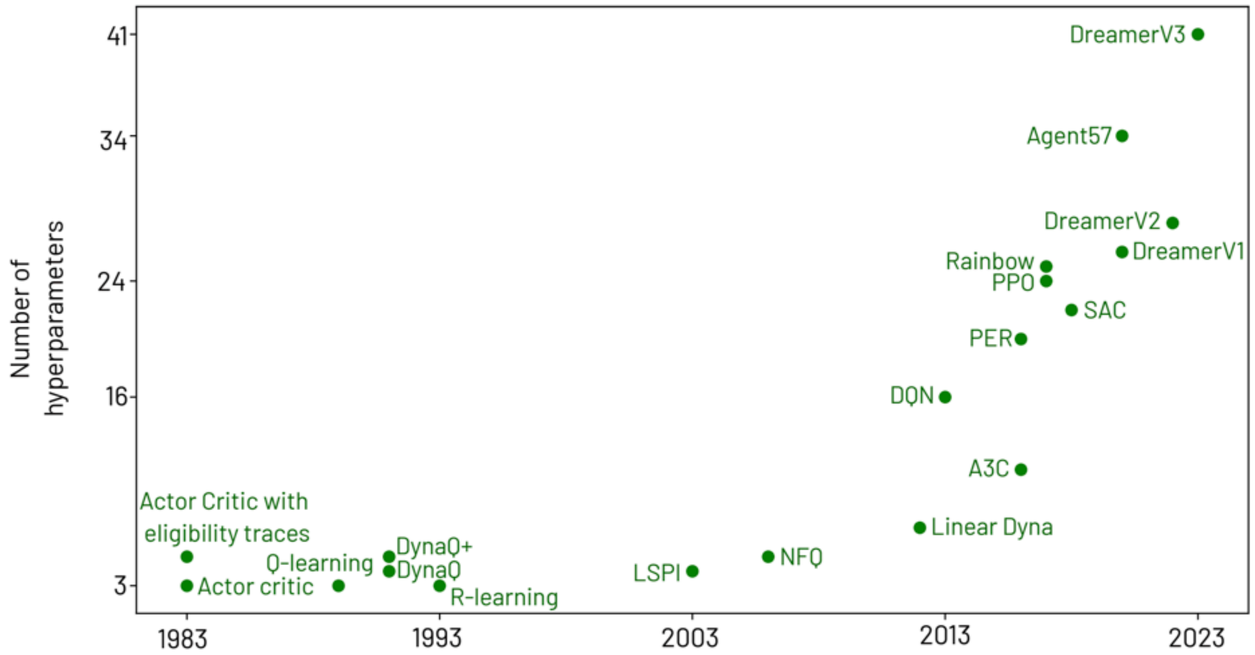
The Hyperparameter Crisis in RL

- The performance of RL agents critically depends on setting key hyperparameters



The Hyperparameter Crisis in RL

- The performance of RL agents critically depends on setting key hypers
- **The proliferation of hyperparameters in RL**: agents are becoming more complex and increasingly dependent on more and more hypers



** does not include architecture hypers*

A Method for Evaluating Hyperparameter Sensitivity in Reinforcement Learning. Jacob Adkins, Michael Bowling, Adam White. NeurIPs 2024

The Hyperparameter Crisis in RL

- The performance of RL agents critically depends on setting key hypers
- There has been a proliferation of hyperparameters in RL
- To tune these agents, designers assume:
    1. **Full knowledge/access** to deployment setting
    2. **Things won't break:** when testing bad hyperparameter settings
    3. **Tuning does not count:** time spent tuning hyperparameters is not reflected in standard evaluation pipelines
- 1 & 2 are true in simulators ...

Consider your goal

- **Goal #1**: want to **prove your algorithm is best (SOTA)**—likely impossible
- **Goal #2**: **understand** how an algorithm's performance changes as you vary key hyperparameters
- **Goal #3**: an application and you want to **deploy an agent in the wild**—tuning hypers on a water treatment plant is likely infeasible

Consider your goal

- Goal #1: want to prove your algorithm is best (SOTA)—likely impossible
- Goal #2: understand how an algorithm's performance changes as you vary key hyperparameters
- **Goal #3**: an application and you want to **deploy an agent in the wild**—tuning hypers on a water treatment plant is likely infeasible

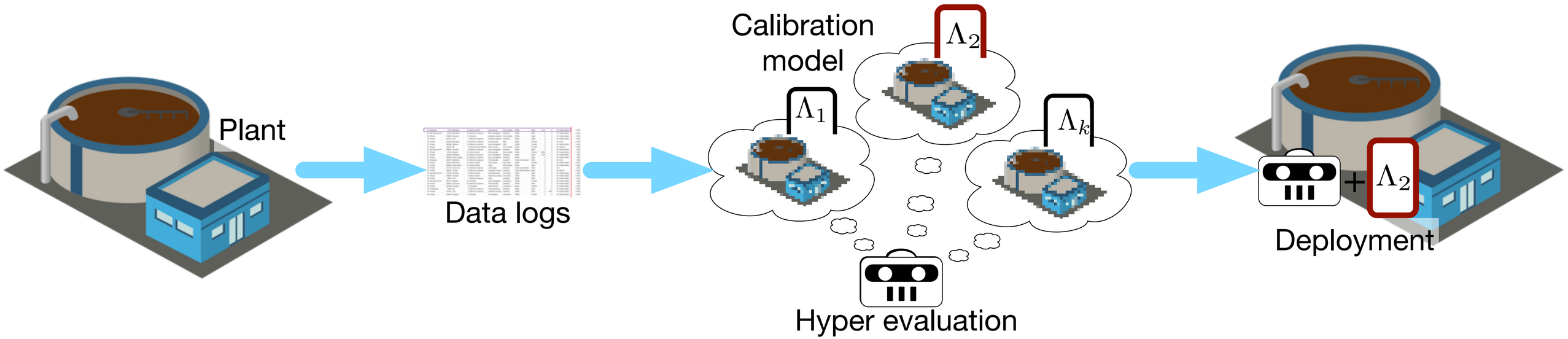
Consider A Water Treatment Plant

Many things to learn about in water treatment

Outline

- Part 1: don't waste your time
  - a case study
  - you need more runs than you think!
  - better statistical tools
- Part 2: pesky hypers
  - the hyperparameter crisis in RL
  - **dealing with hyperparameter when you don't have a simulator**

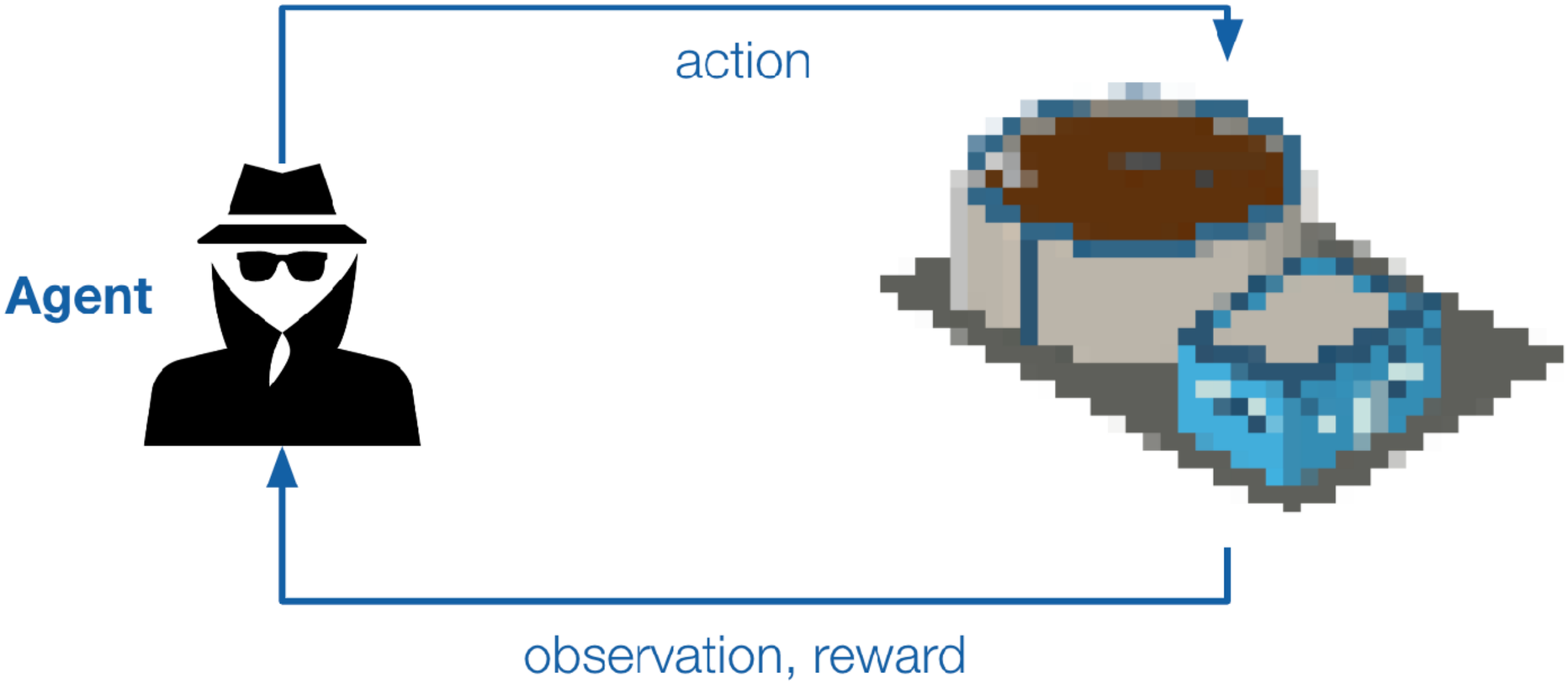
Proposal: Calibration Models



1. **Learn** a calibration model from data logs (approximate MDP model)
2. **Test** many hyperparameters in the calibration model (the agent learns in the calibration model as if its the real world)
3. Select **best** hyperparameters according to performance in calibration model
4. Deploy **continual** learning agent with selected hyperparameters

RL in the real world

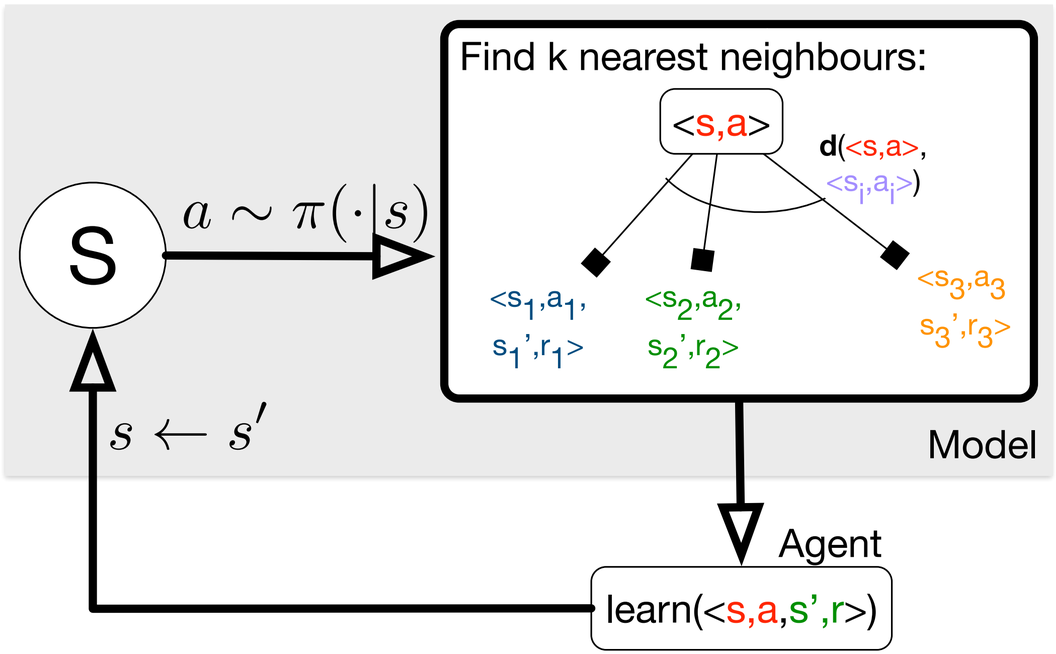
RL in a calibration model



Stability of the Model

- **The goal is to use the Calibration model as a stand-in for the real world:** conduct many experiments with different hyperparameters, pretending agent is interacting with the plant
  - **must** remain stable when iterated many steps (5000, 500k, 5m steps)
  - **should** produce realistic states
- **Proposal**: KNN model (k-nearest neighbors)

Calibration Model using a KNN approach



Essentially chaining together observed transitions

\+ Maintains **stability** because each transition stays within observed data

\+ Produces only **real** states and rewards

\- **Limited** generalization

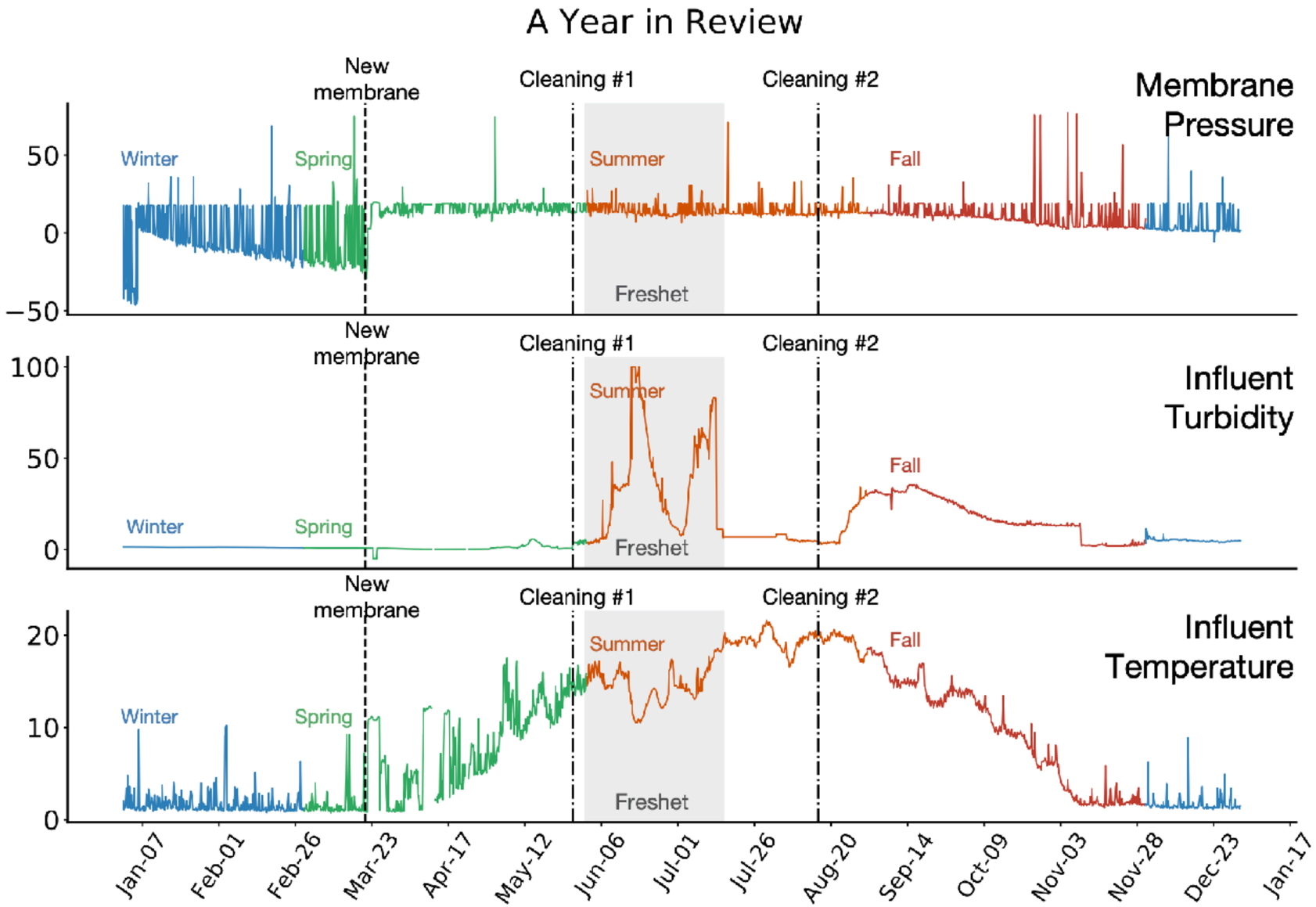
Insights from small experiments

- In low-dimensional, small simulation problems (Acrobot, Puddleworld, Cartpole)
- Our approach is **more effective** than offline RL (fixed pre-trained policy)
- NN-based calibration worse than KNN-based
- Approach was effective with relatively **small** datasets, and **different** data gathering **policies**
- Robust to some non-**stationarity** (i.e., small change to dynamics)

No More Pesky Hyperparameters: Offline Hyperparameter Tuning for RL. Transactions on Machine Learning Research 2022

Does the approach work on a real system?

The data of water treatment



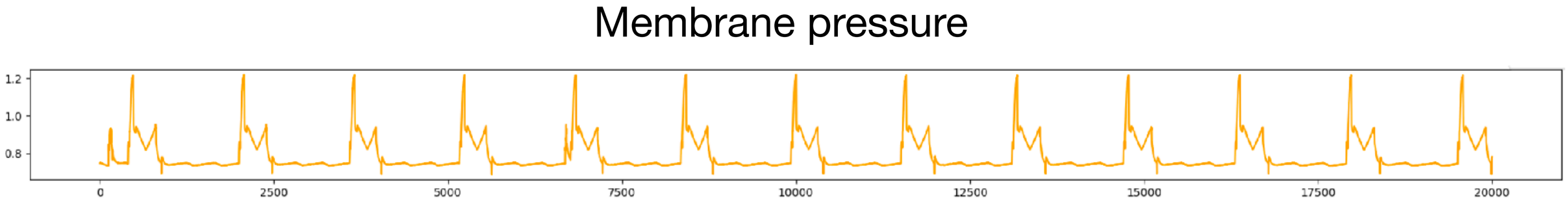
Janjua, M. K., Shah, H., White, M., Miahi, E., Machado, M. C., & White, A. (2023). GVFs in the Real World: Making Predictions Online for Water Treatment. Machine Learning Journal.

Building a Calibration model for water treatment

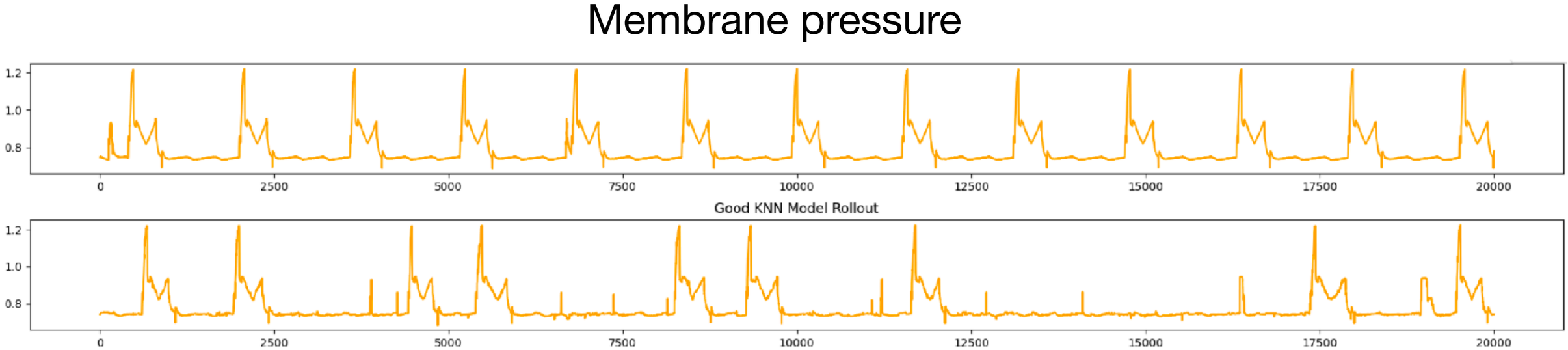
1. Take logdata from operators controlling the plant
    - 1 week (350k samples), 480 sensor readings
      (also works with 1 year of data)
2. Build a KNN-calibration model
3. Treat CM like the real world: try each HP combination many times and return best performing

- **Deployment goal:** set step-size of Adam for a prediction agent

Membrane pressure timeseries



Simulating membrane pressure with a CM



☑ **Deployment goal:** set step-size of Adam for a prediction agent

Can we do better than CM for real-world RL?

Yes! But we need to stop assuming tuning is free!

We need better algorithms: fewer hyperparamters or at least meta learning

Better empirical practices and better test problems are critical!

Thanks for listening

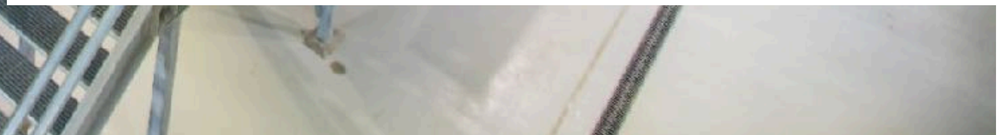
UNIVERSITY OF ALBERTA

Appendix

Water treatment is all about change and adaption

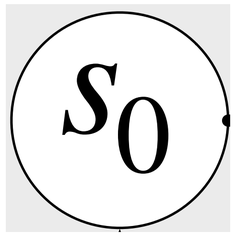
- Weather & climate
- Activities along the river
- Mechanical wear-n-tear
- Changing operator behaviours

- Operator data logs will not reflect all these situations
- We can’t anticipate/engineer for all these situations



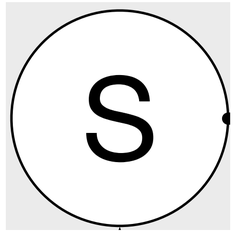
Calibration Model using a KNN approach

Agent-environment interaction: $s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, \ldots$



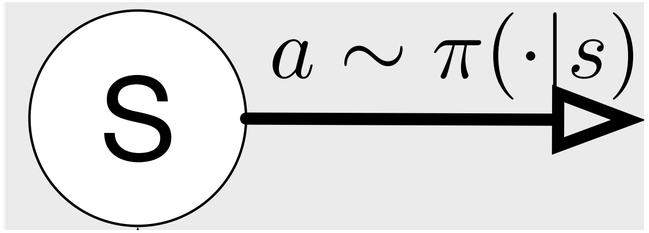
Calibration Model using a KNN approach

Agent-environment interaction: $s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, \ldots$



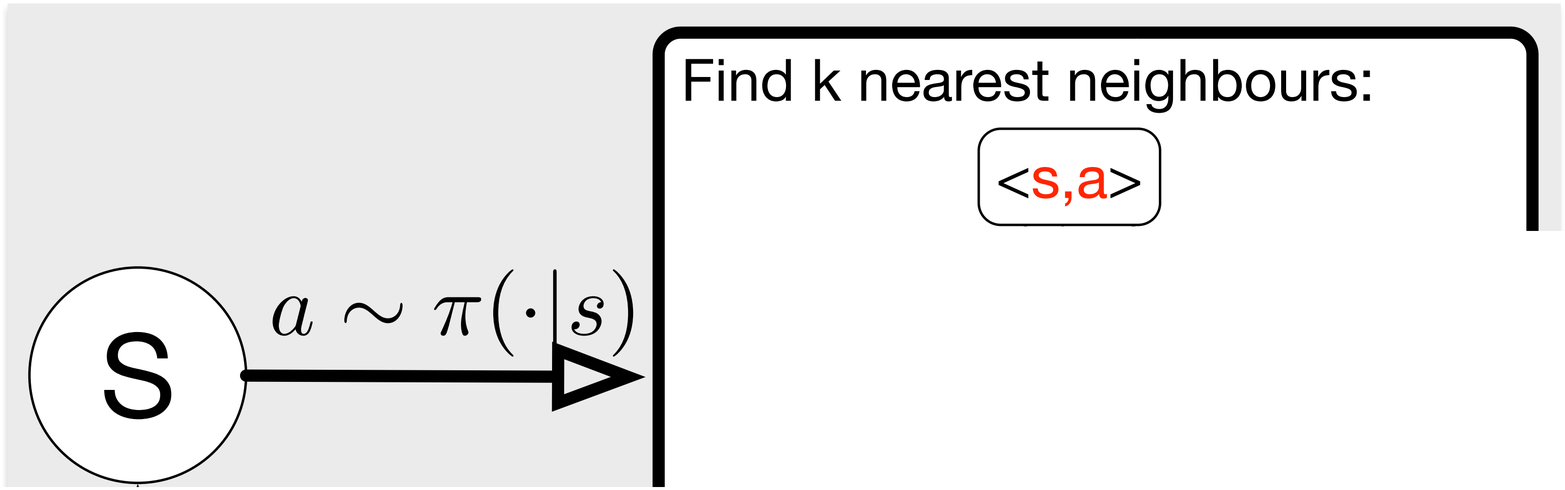
Calibration Model using a KNN approach

Agent-environment interaction: $s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, \ldots$



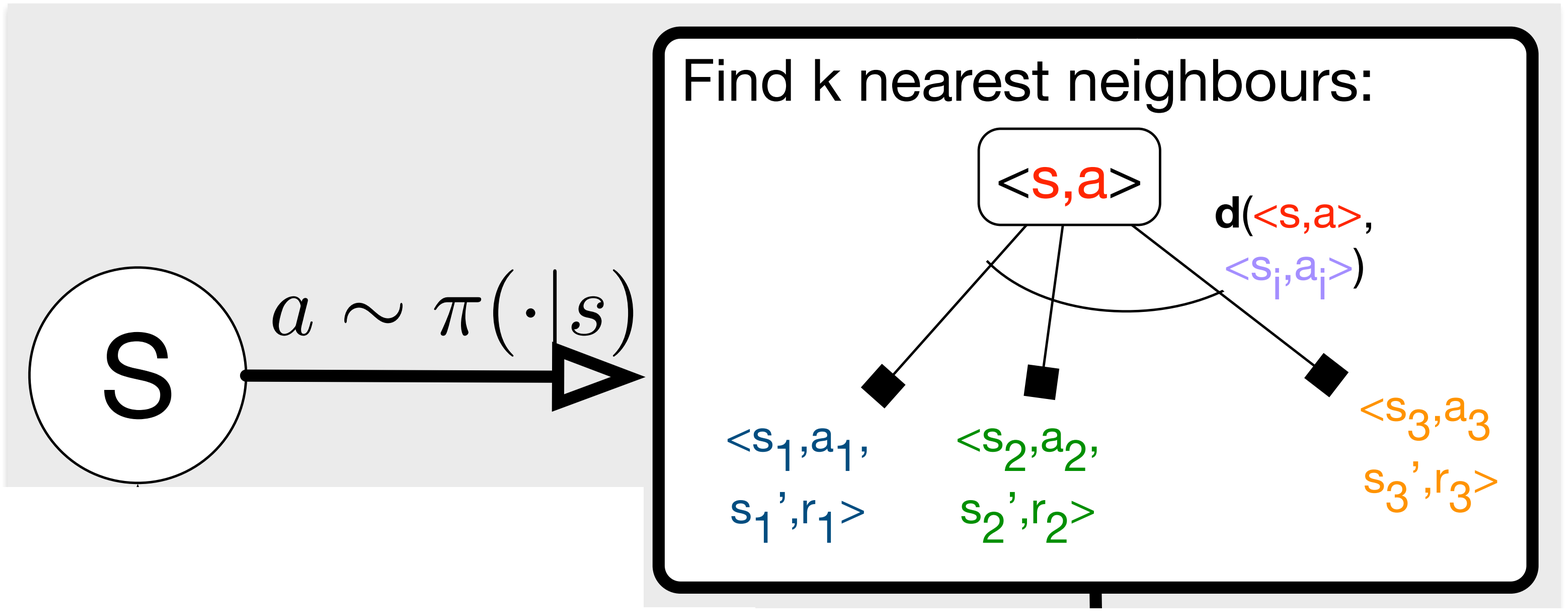
Calibration Model using a KNN approach

Agent-environment interaction: $s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, \ldots$



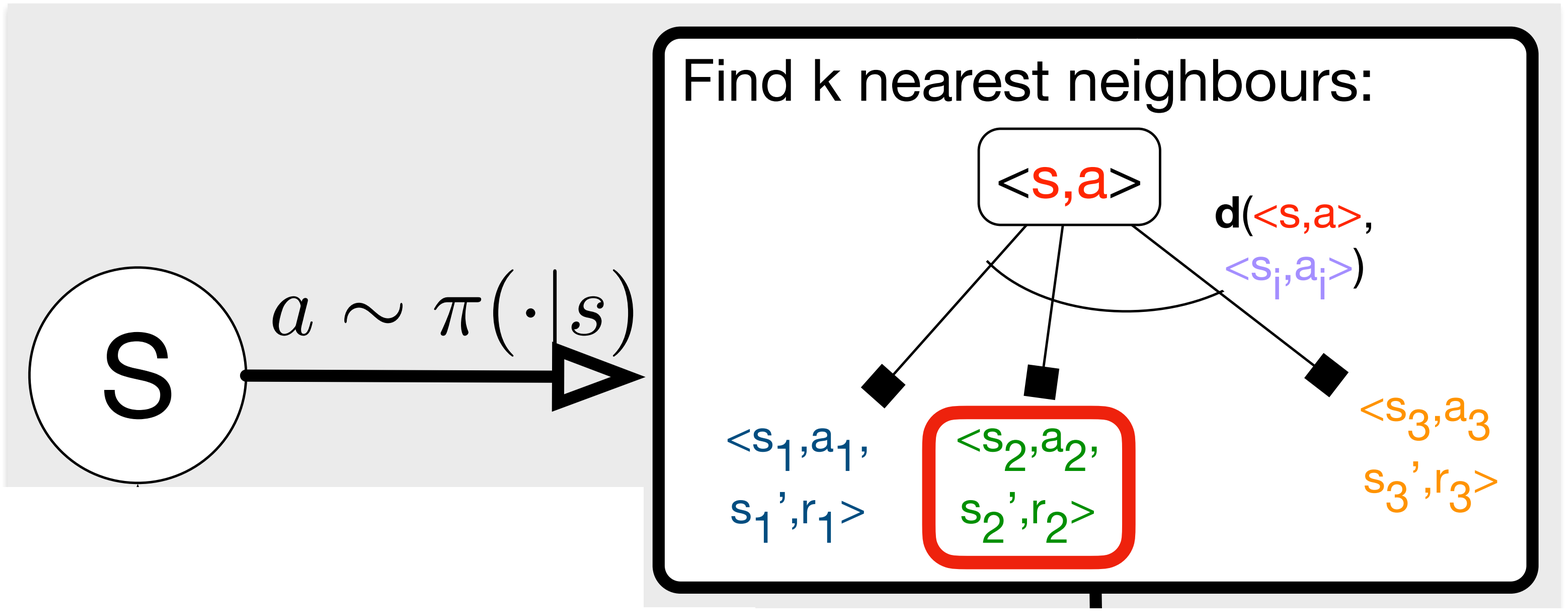
Calibration Model using a KNN approach

Agent-environment interaction: $s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, \ldots$



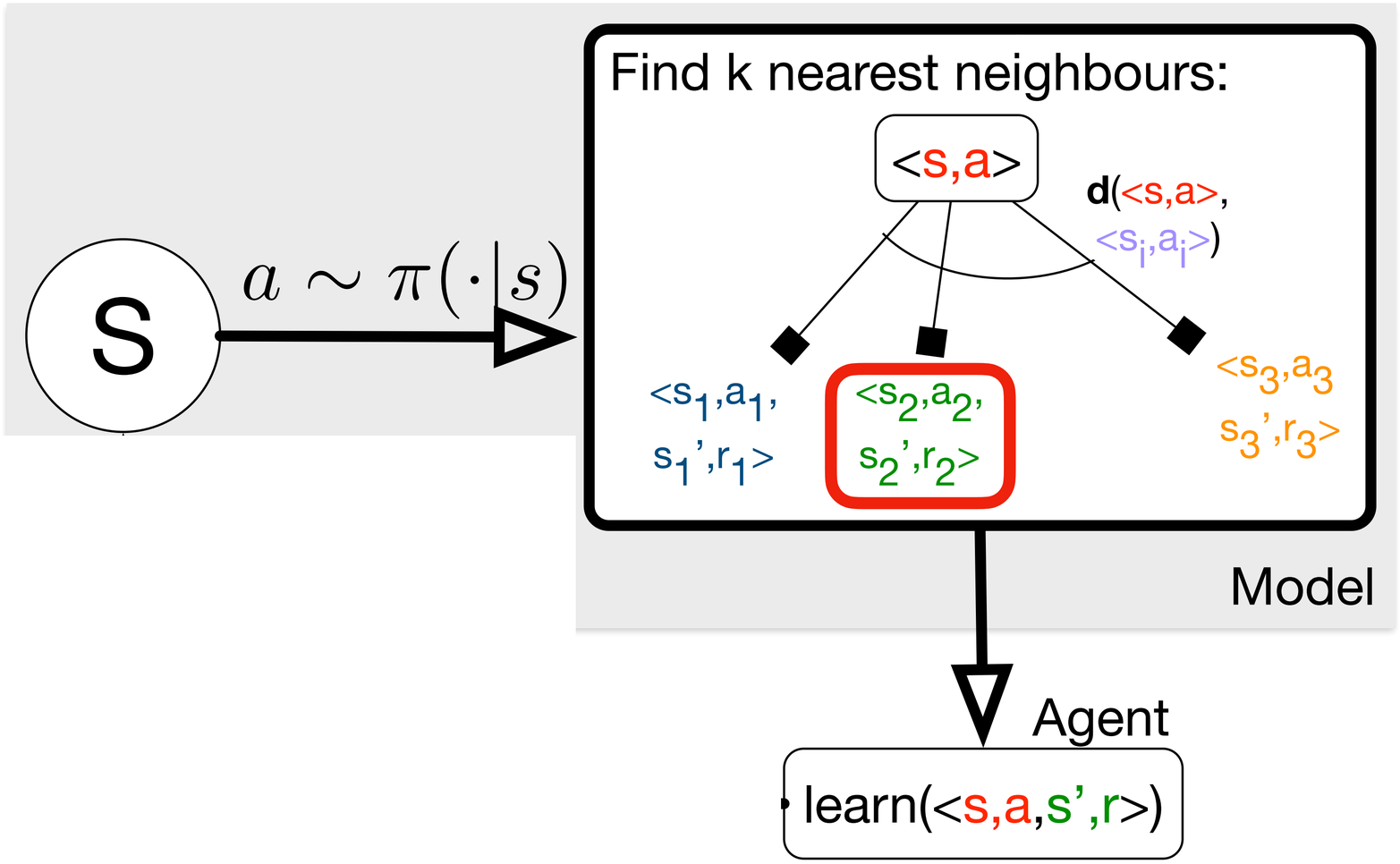
Calibration Model using a KNN approach

Agent-environment interaction: $s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, \ldots$



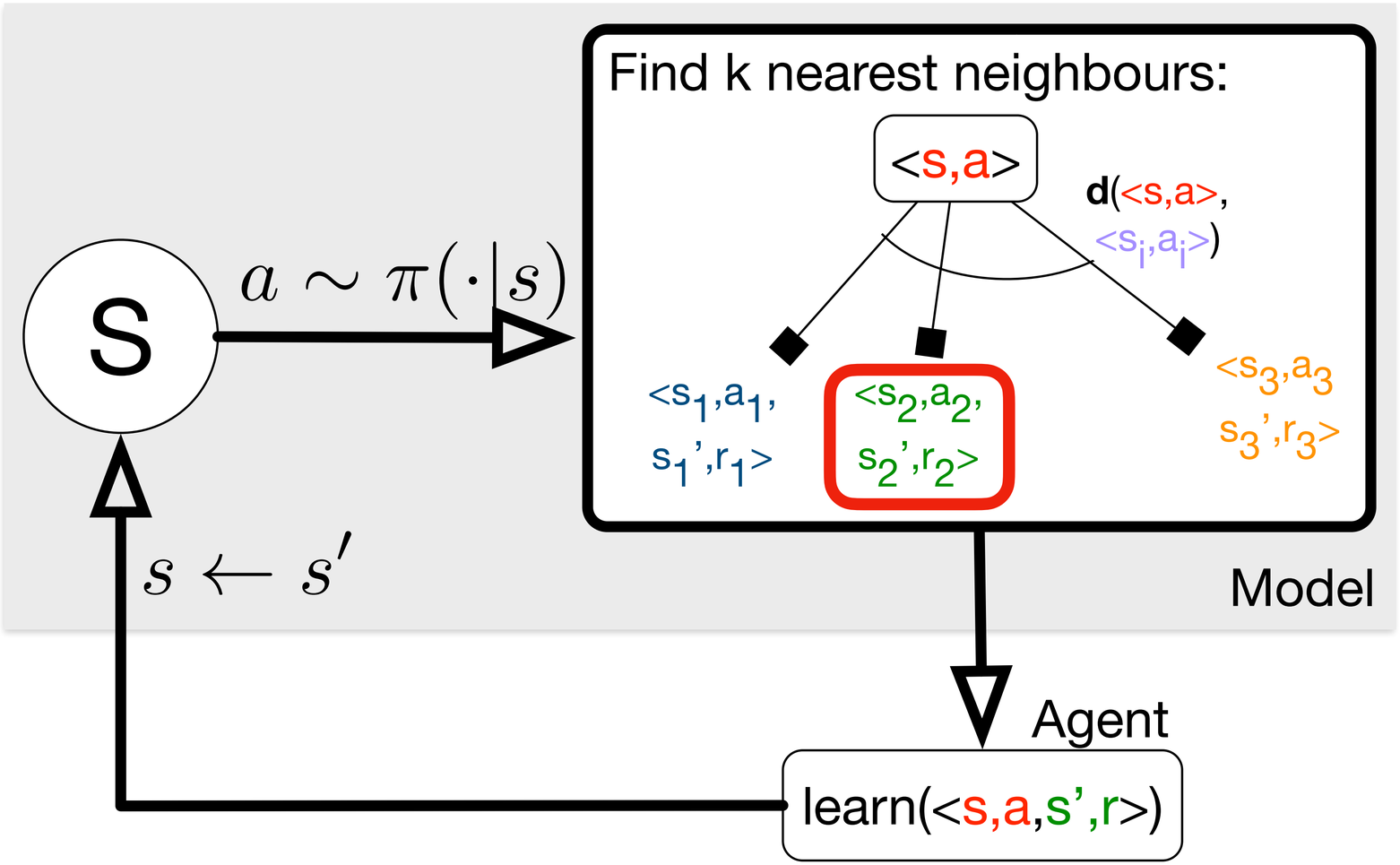
Calibration Model using a KNN approach

Agent-environment interaction: $s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, \ldots$



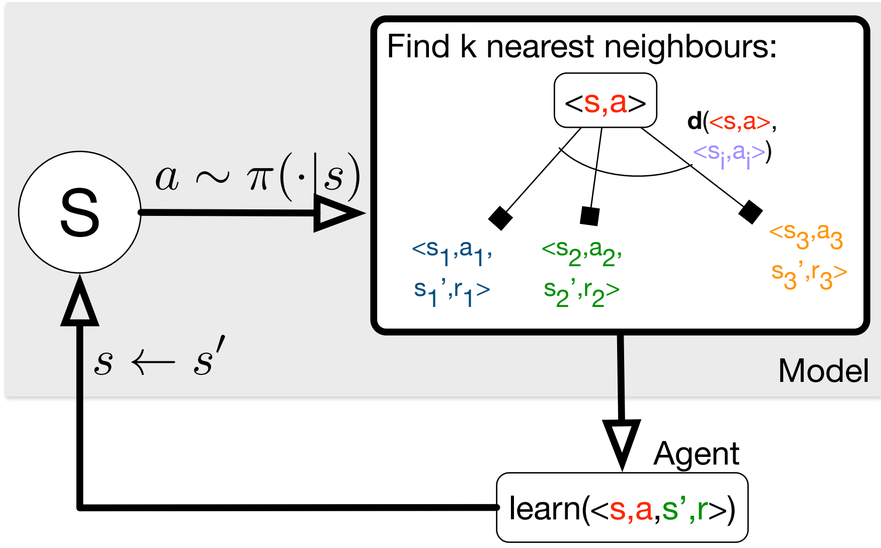
Calibration Model using a KNN approach

Agent-environment interaction: $s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, \ldots$



Additions to KNN Calibration Model: Learning the Distance

- Relies **heavily** on the **distance** function
- We learn a Laplacian distance that respects the **dynamics** (e.g., walls)
  - Related to using successor features as targets: $\phi(s), \phi(s')$ are encouraged to be similar if discounted sum of future features is similar
- Provides a transition-aware distance metric



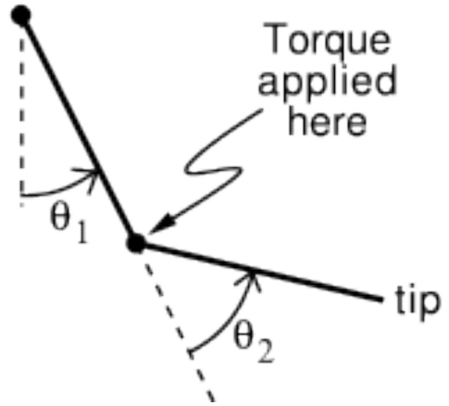
Empirical validation

1. Test on simple problems:
    - vary data collection (simulating different operators)
    - try tuning different hyper-parameters
    - compare against relevant baselines (random hypers, learn policy from dataset and deploy)
    - explore limitations and counter-examples

2. Try on real plant data

A Simple Example: Acrobot

Goal: Raise tip above line



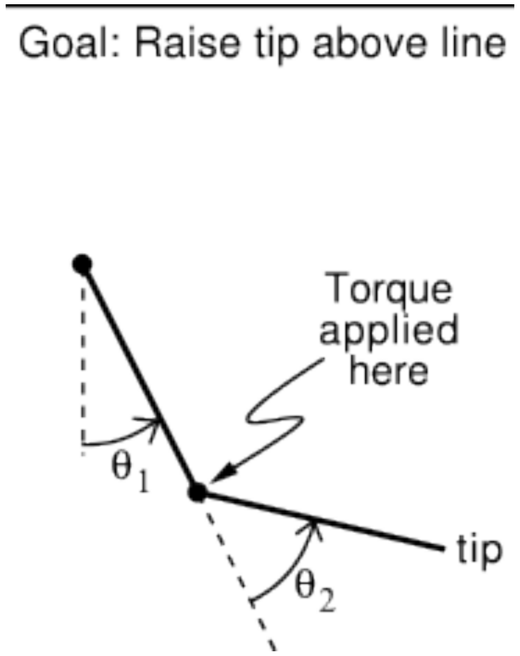
Data: 5000 transitions under near-optimal policy

Agent: Sarsa with softmax, tile-coding features

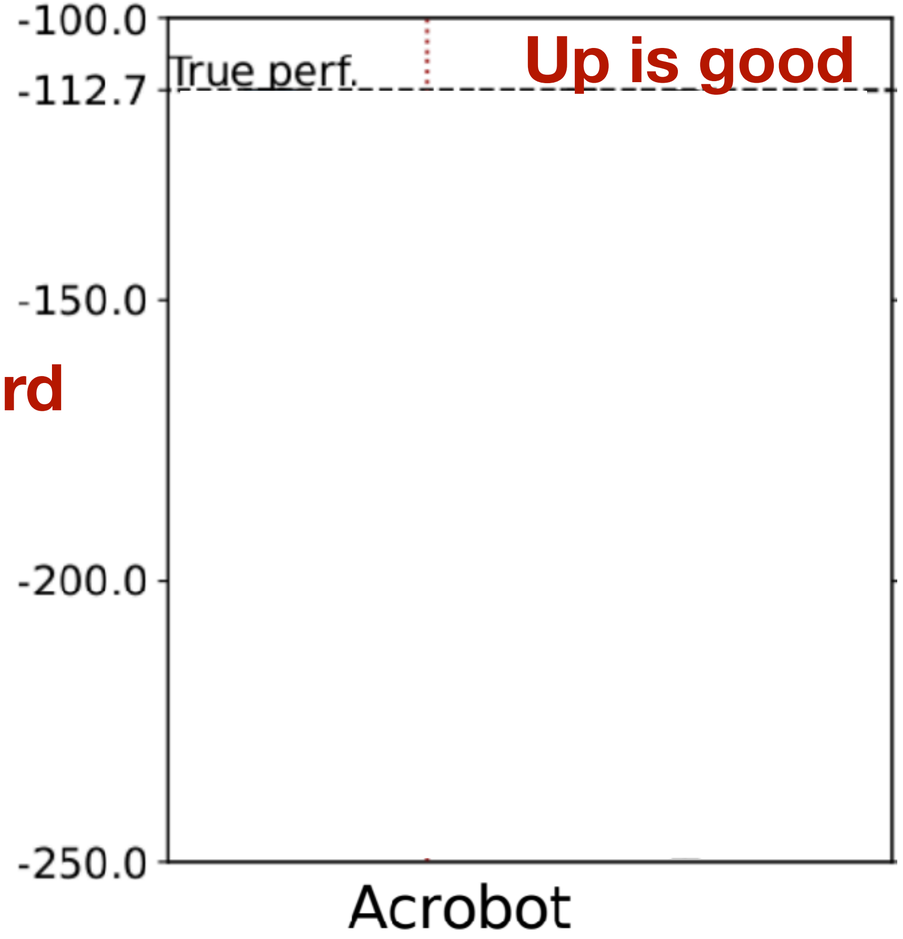
Hyperparameters to tune: stepsize and momentum in Adam algorithm (optimizer), temperature in softmax, weight initialization (optimism)

Treat CM like the real world: try each HP combination many times in CM and return best performing

A Simple Example: Acrobot



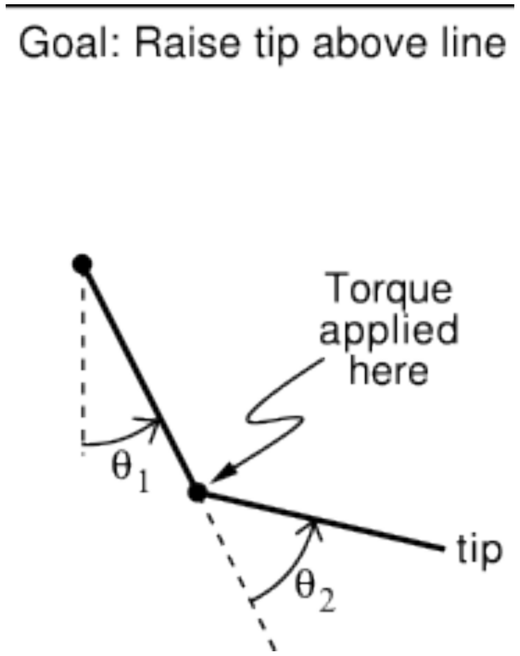
Total reward averaged over runs



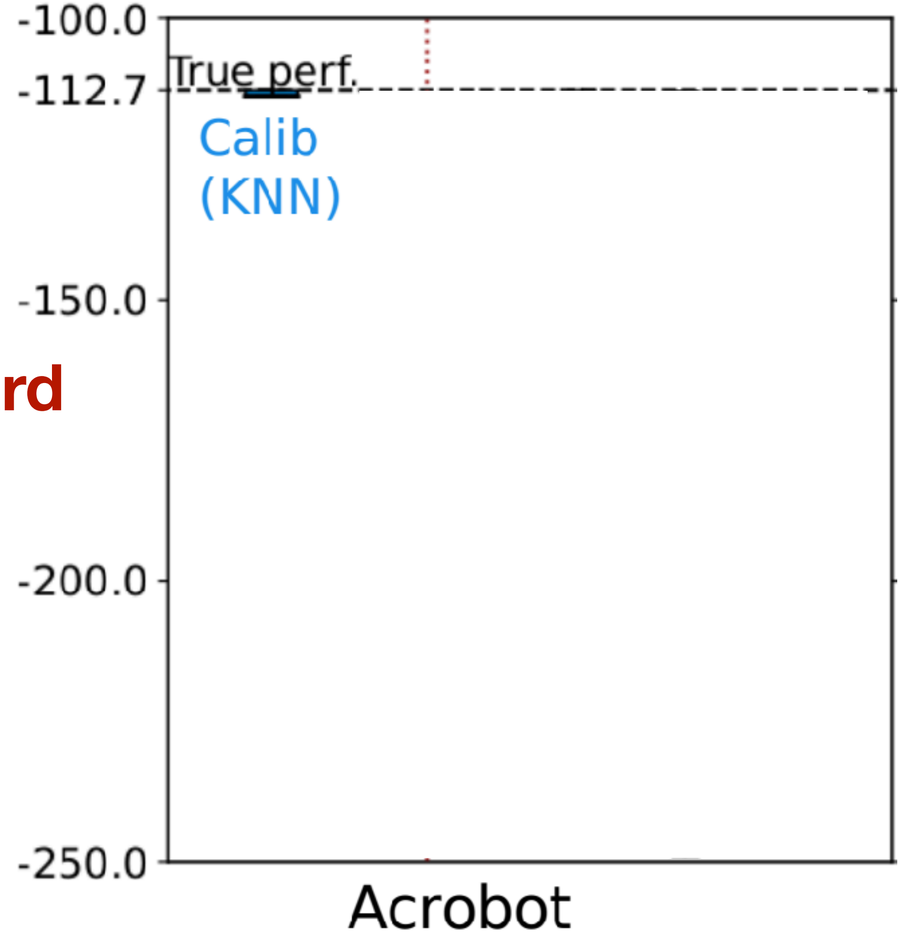
Agent l**earns** in CM for 15000 steps

Report median performance across 30 runs (30 datasets), with 25/75 percentiles

A Simple Example: Acrobot



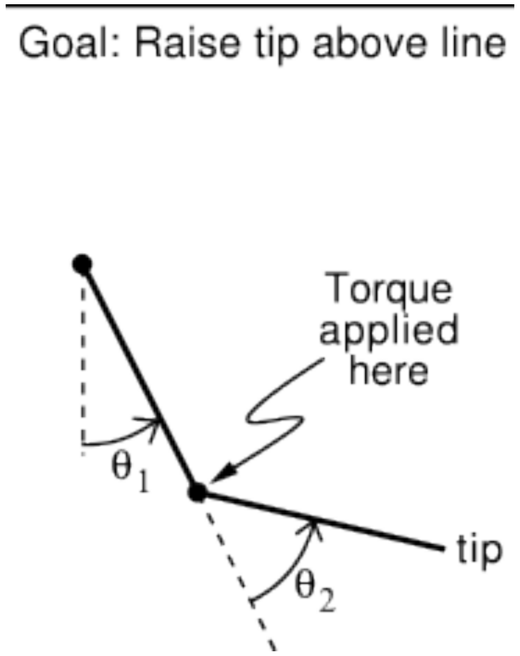
Total reward averaged over runs



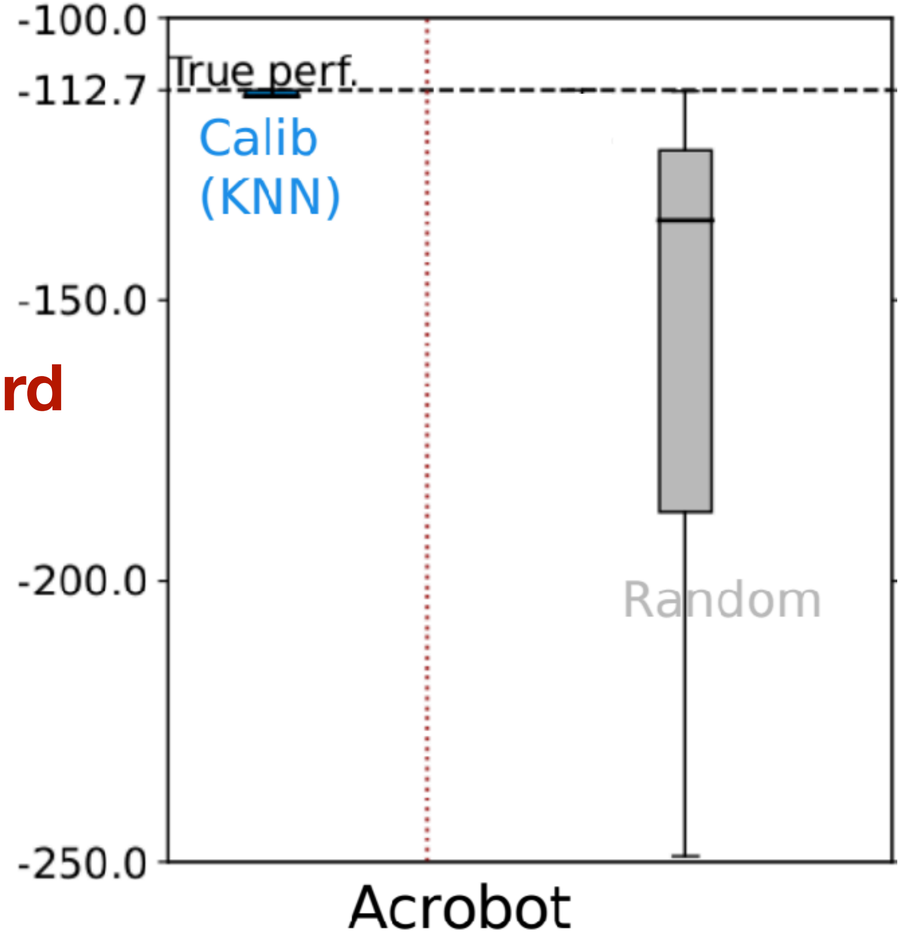
Agent l**earns** in CM for 15000 steps

Report median performance across 30 runs (30 datasets), with 25/75 percentiles

A Simple Example: Acrobot



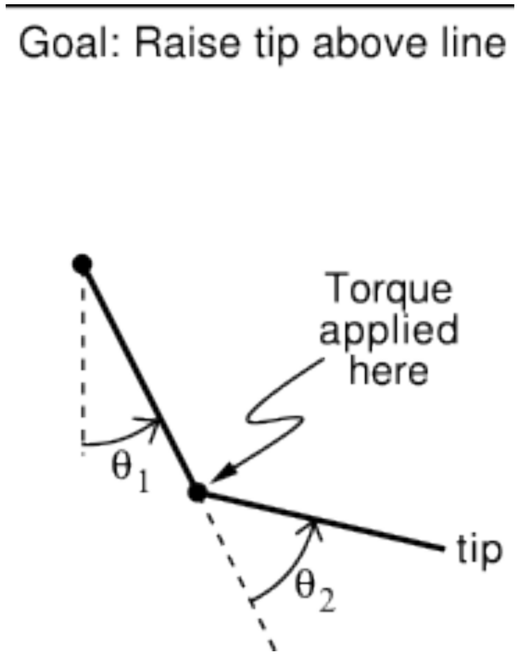
Total reward averaged over runs



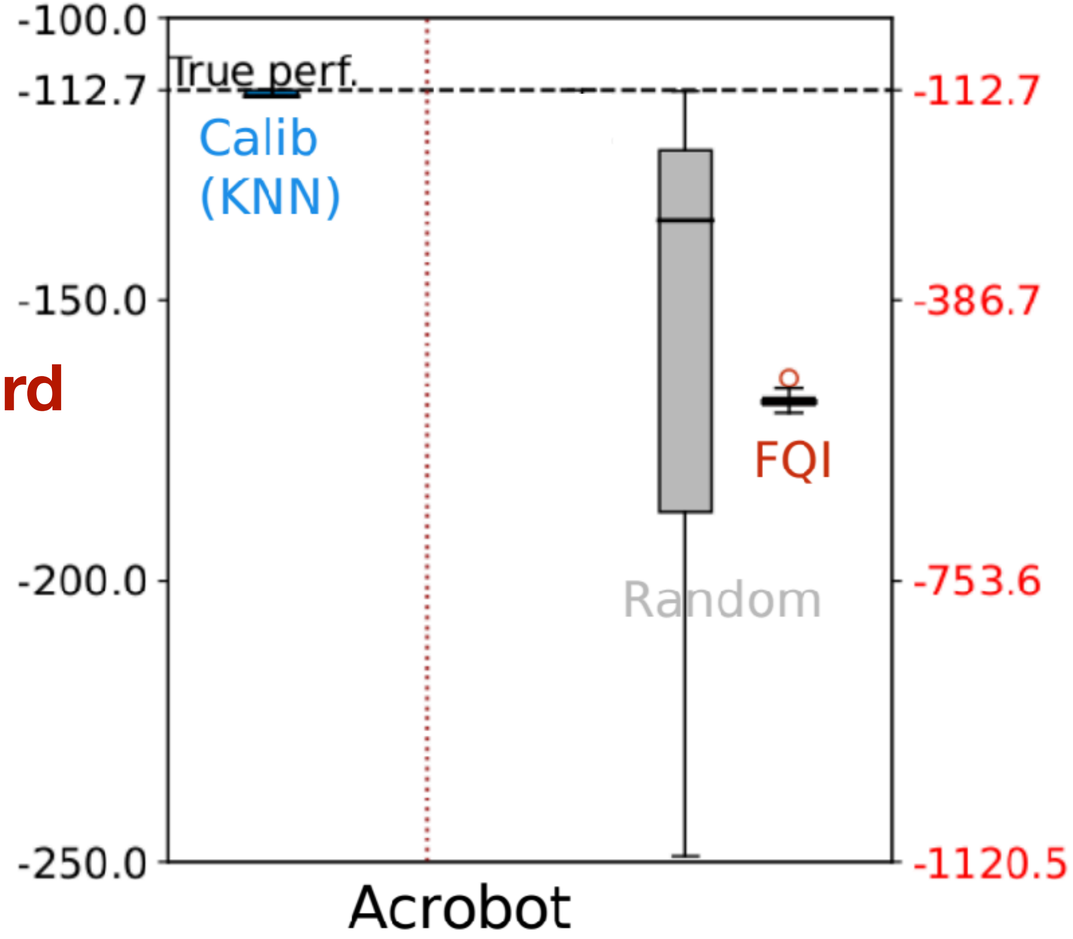
Agent l**earns** in CM for 15000 steps

Report median performance across 30 runs (30 datasets), with 25/75 percentiles

A Simple Example: Acrobot



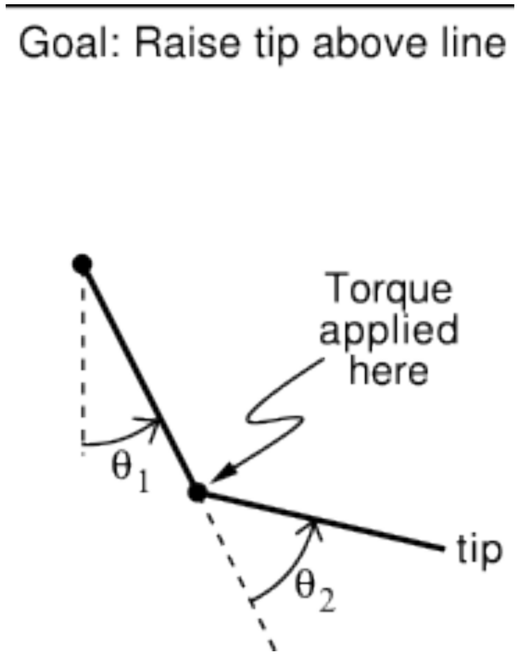
Total reward averaged over runs



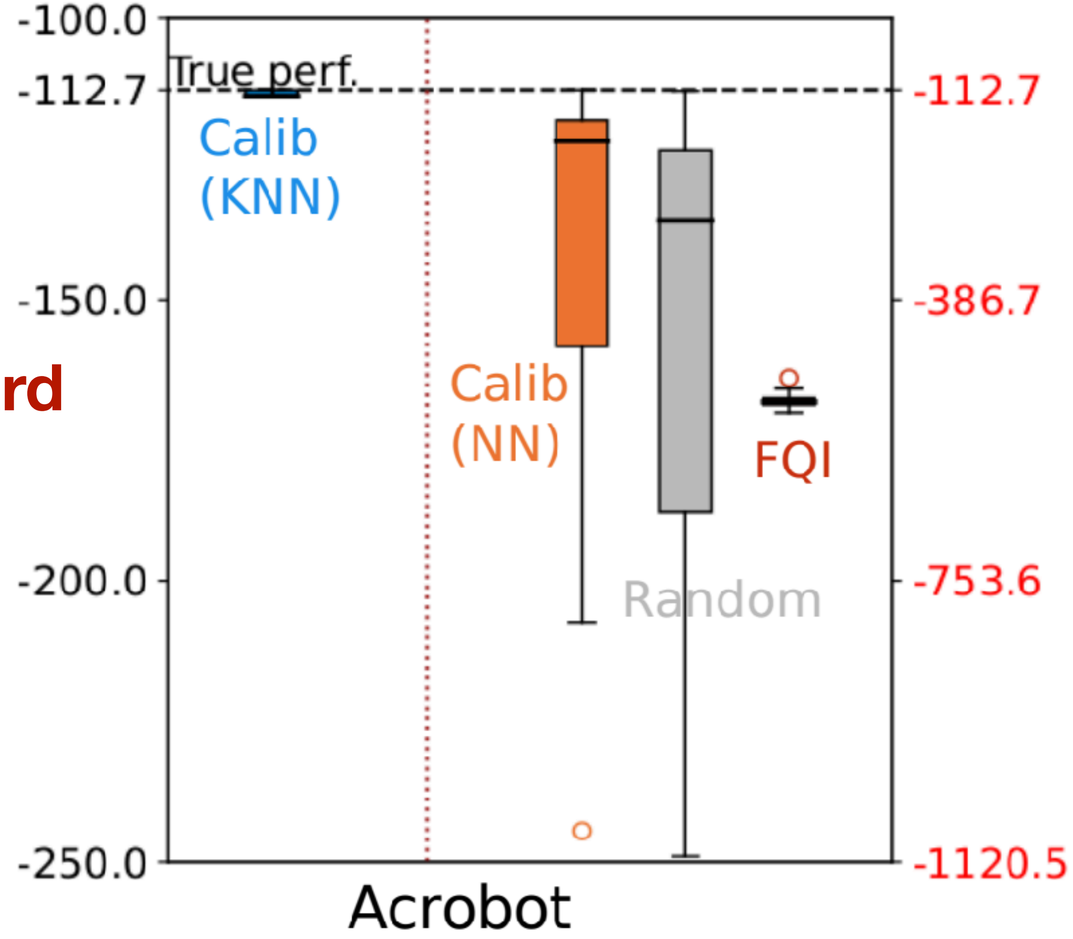
Agent l**earns** in CM for 15000 steps

Report median performance across 30 runs (30 datasets), with 25/75 percentiles

A Simple Example: Acrobot



Total reward averaged over runs



Agent l**earns** in CM for 15000 steps

Report median performance across 30 runs (30 datasets), with 25/75 percentiles