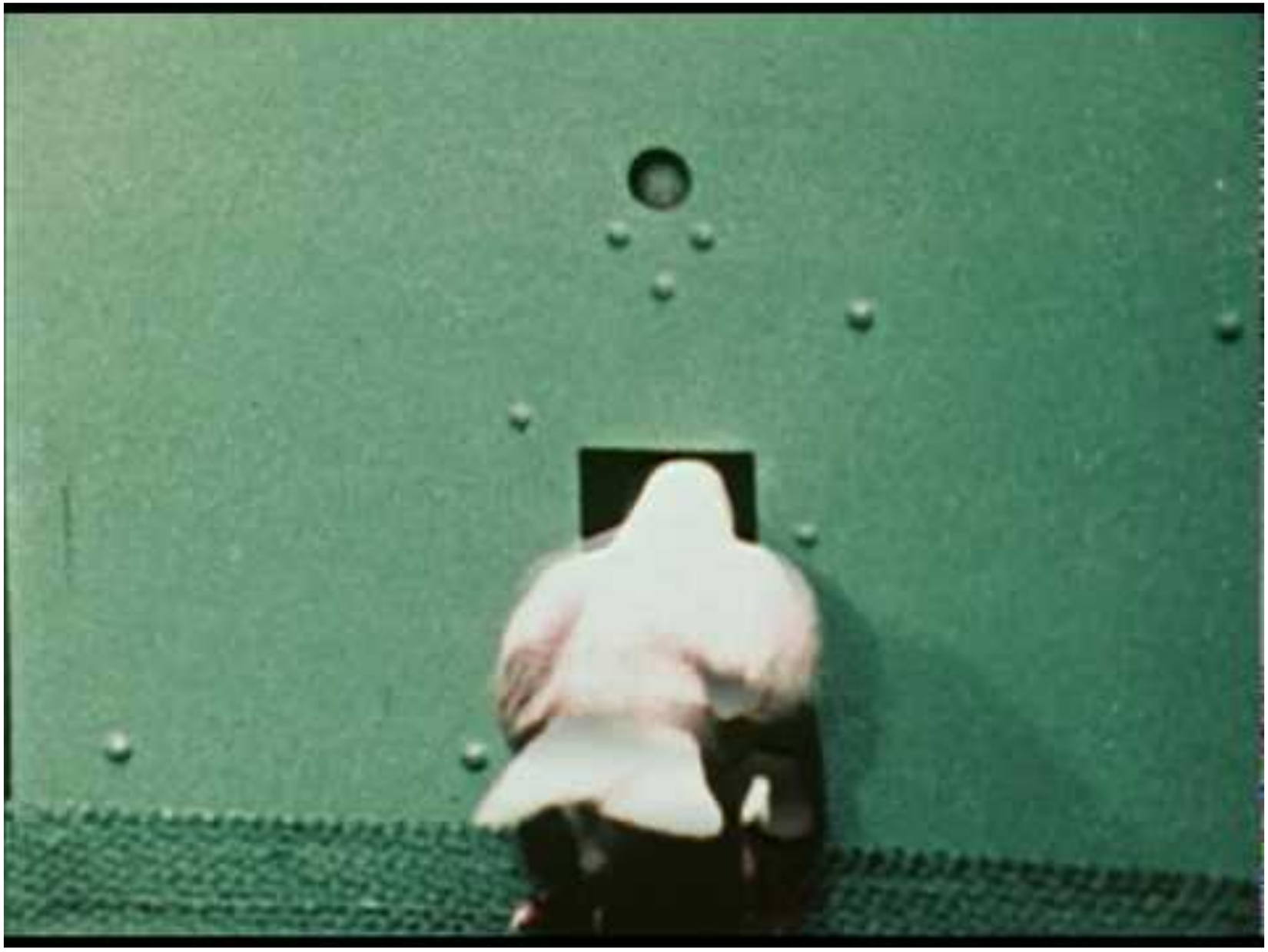
Better Experiments in RL



Projects

- Sign-up deadline TODAY
- The focus is empirical projects, NOT algorithm development or solving open problems
- The main objective is to learn to do RL experiments well, thus
  - Reward prediction or control must be involved
  - If you find yourself trying to create an algorithm—DANGER
  - If you find yourself implementing a poorly understood algorithm from a recent paper—Danger

Projects

- READ the project names/descriptions carefully. Scope is implied
- Don’t do too much. If you submit a report with some good experiments and some bad ones—DANGER

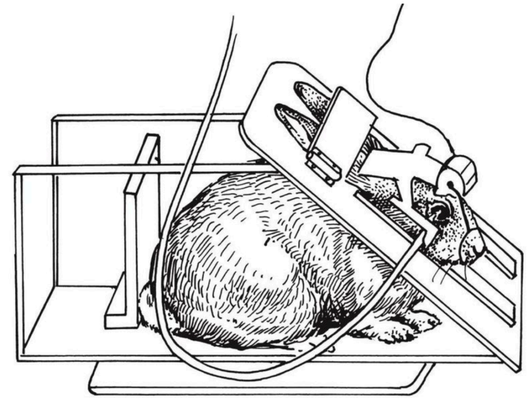
Key Messages

- You need more runs/experiment replications (seeds) than you think
- Take a statistical point of view: think of distributions of performance, how you want to characterize that distribution and what assumptions you are making
- Sensitivity to hyperparameters is important …but sweeping is not an algorithmic answer
- 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

RL as an empirical science

- Science is the study of the natural world
- In computing science we study computing, not the natural world
  - We pose questions and seek answers, typically by inventing problems
- In RL 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***

Empirical research in animal learning



- Consider a researcher conducting eye-blink experiments with rabbits
  - 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

Empirical research in RL is easier in many ways

- 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 observe and understand “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

Mounting evidence of poor scholarship

- ***Troubling trends in machine learning scholarship (2018)***
  ~Lipton & Steinhardt

  - Failure to distinguish between speculation and explanation
  - Failure to identify sources of gain (improvement) in experiments
  - Using math to impress and confuse the reader
  - Using language poorly: overloading established terms or using fancy words with particular English meanings to suggest something about your algorithm

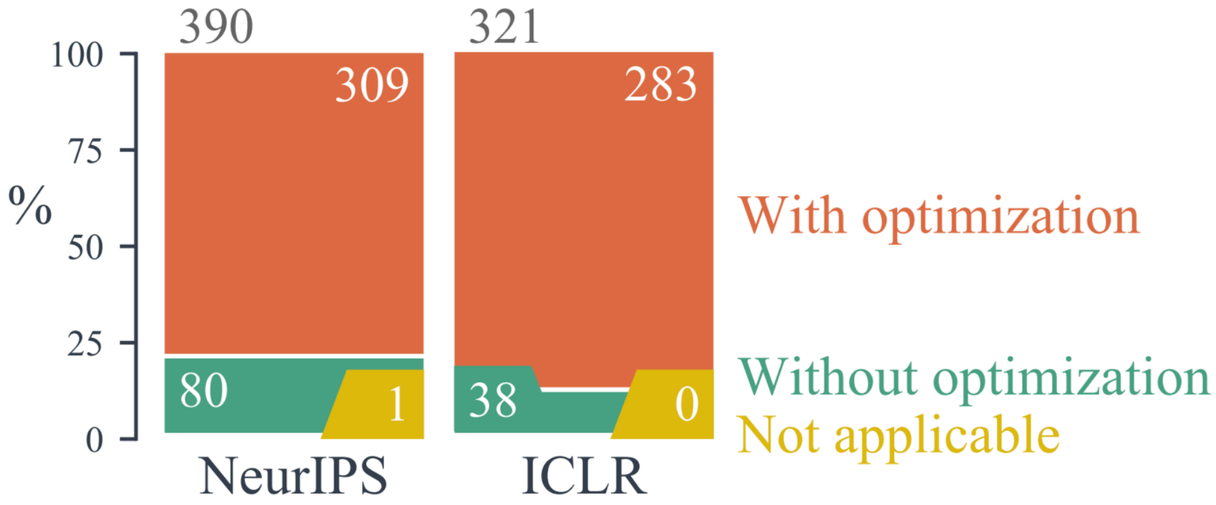
    - e.g., “The *XX* agent is *curious* about its world …”

Troubling empirical trends in RL

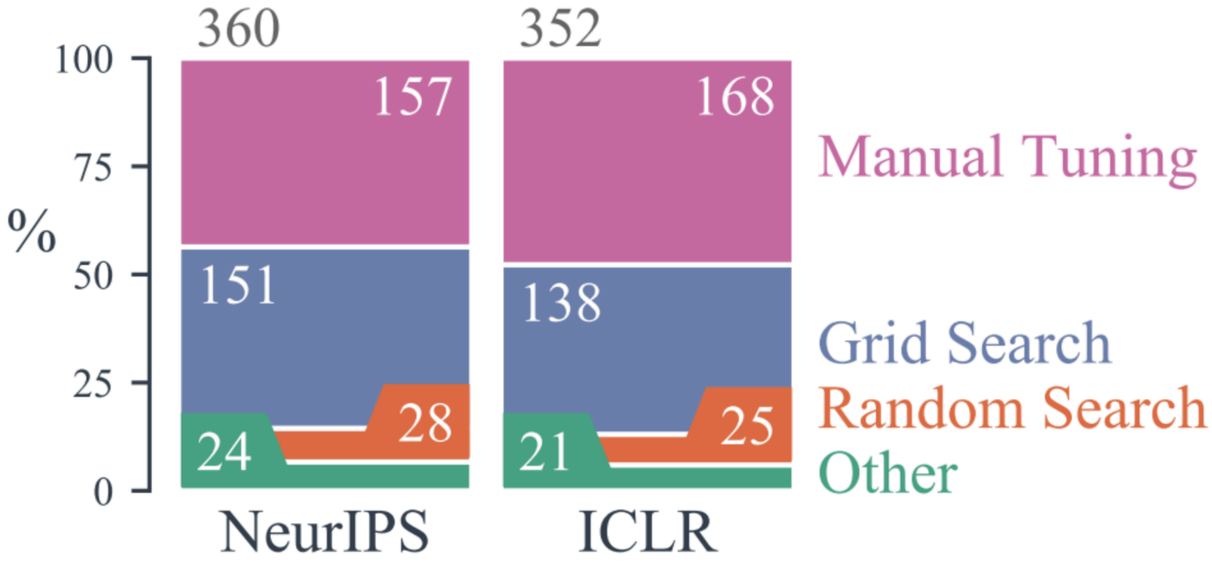
- ***Deep RL that Matters (2018)*** ~Henderson et al
  - Highlighted conflicting empirical results found in the literature
  - Advocated for increased focus on reproducibility
  - Advocated for using more independent runs (“seeds”)
  - Provided a critical evaluation of current empirical practices, showing:
    - Network architectures, activation functions, reward scaling, code bases all matter!
    - Dramatic differences in performance from environment to environment

Troubling trends

Question: *Did you optimize your hyperparameters?*

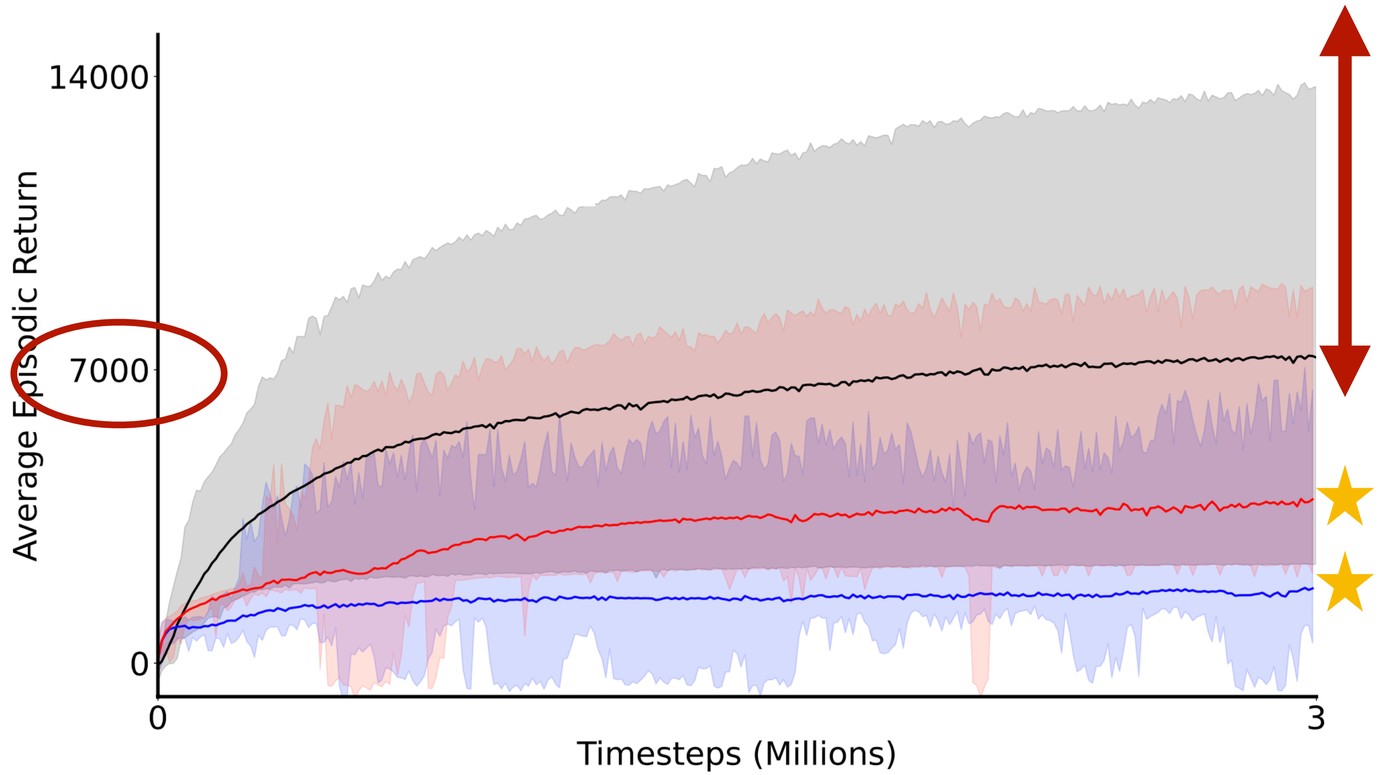


Question: *If yes, you did optimize, how did you tune them?*



A case study

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



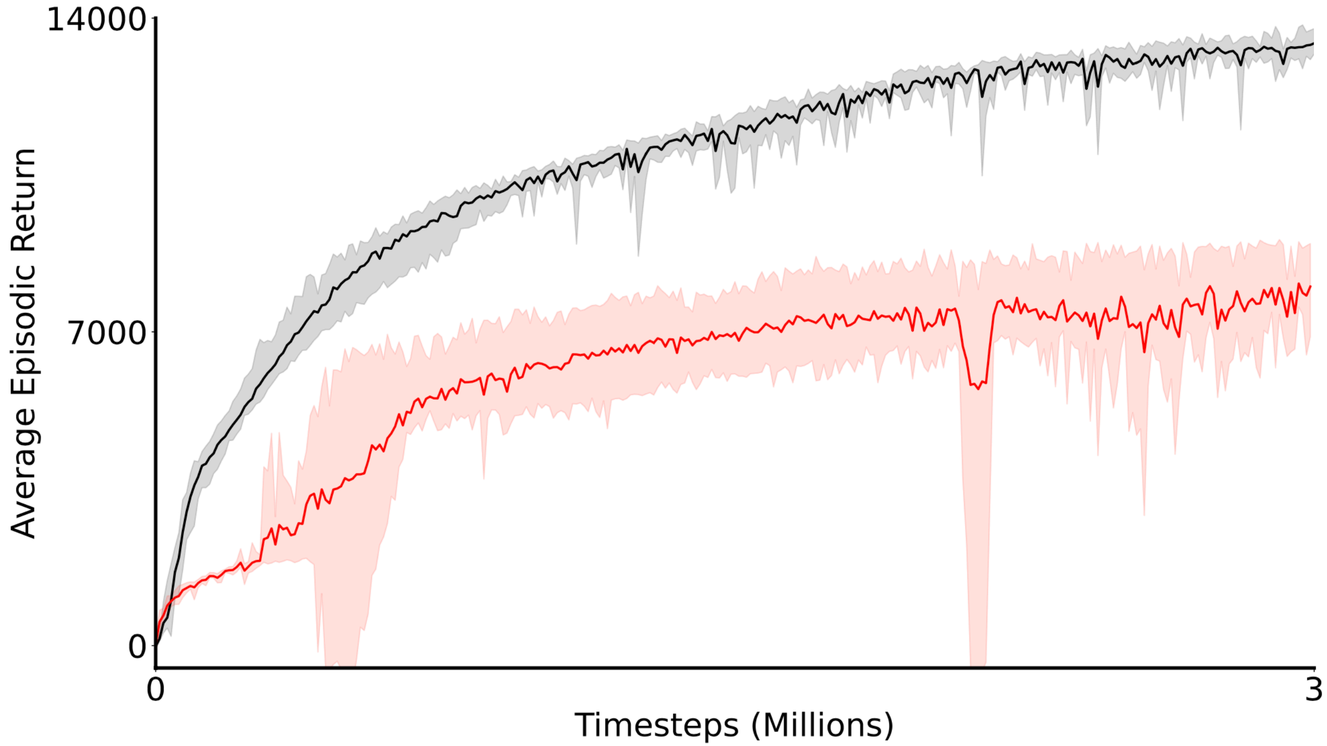
Results compiled by Samuel Neumann

Case study findings

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

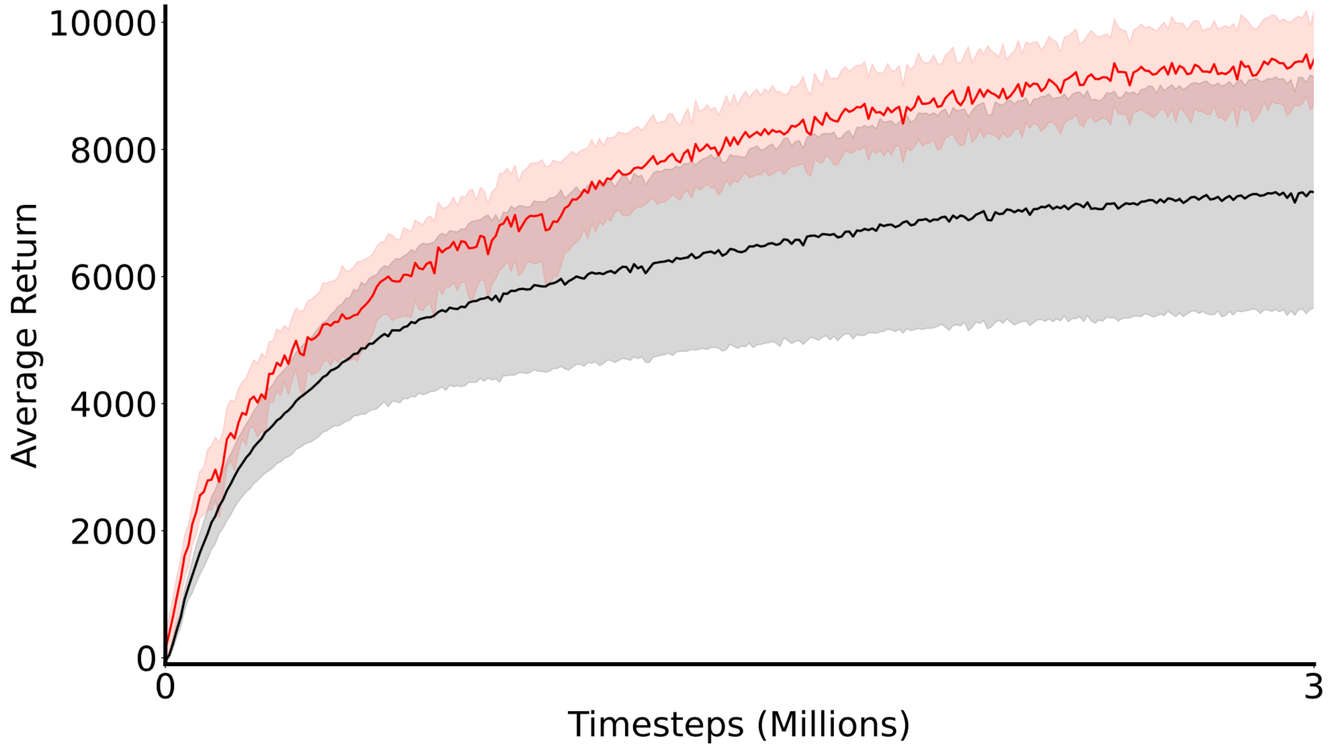
The MythBusters approach

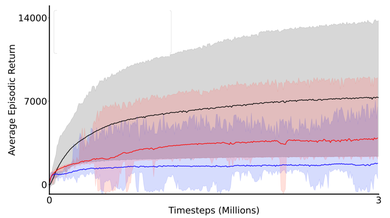
- We noticed some runs were good, some terrible
- We noticed the repo includes code to search for good seeds
- … or maybe they got 5 lucky seeds



Tuning the baseline

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





Many great resources out there

- **Crash course in statistics** relevant to RL:
  - *A hitchhiker's guide to statistical comparisons of reinforcement learning algorithms* ~Colas, Sigaud, Oudeyer
- How to fairly & systematically **deal with hyperparameters** when comparing algorithms:
  - *Evaluating the Performance of Reinforcement Learning Algorithms* ~Jordon et al
- Insights from **small scale experiments:**
  - *Revisiting Rainbow: Promoting more insightful and inclusive deep reinforcement learning research* ~ Ceron & Castro
- **Debunking** ideas based on flawed empirical work:
  - *The mirage of action-dependent baselines in reinforcement learning*
  - *Implementation Matters in Deep RL: A Case Study on PPO and TRPO*

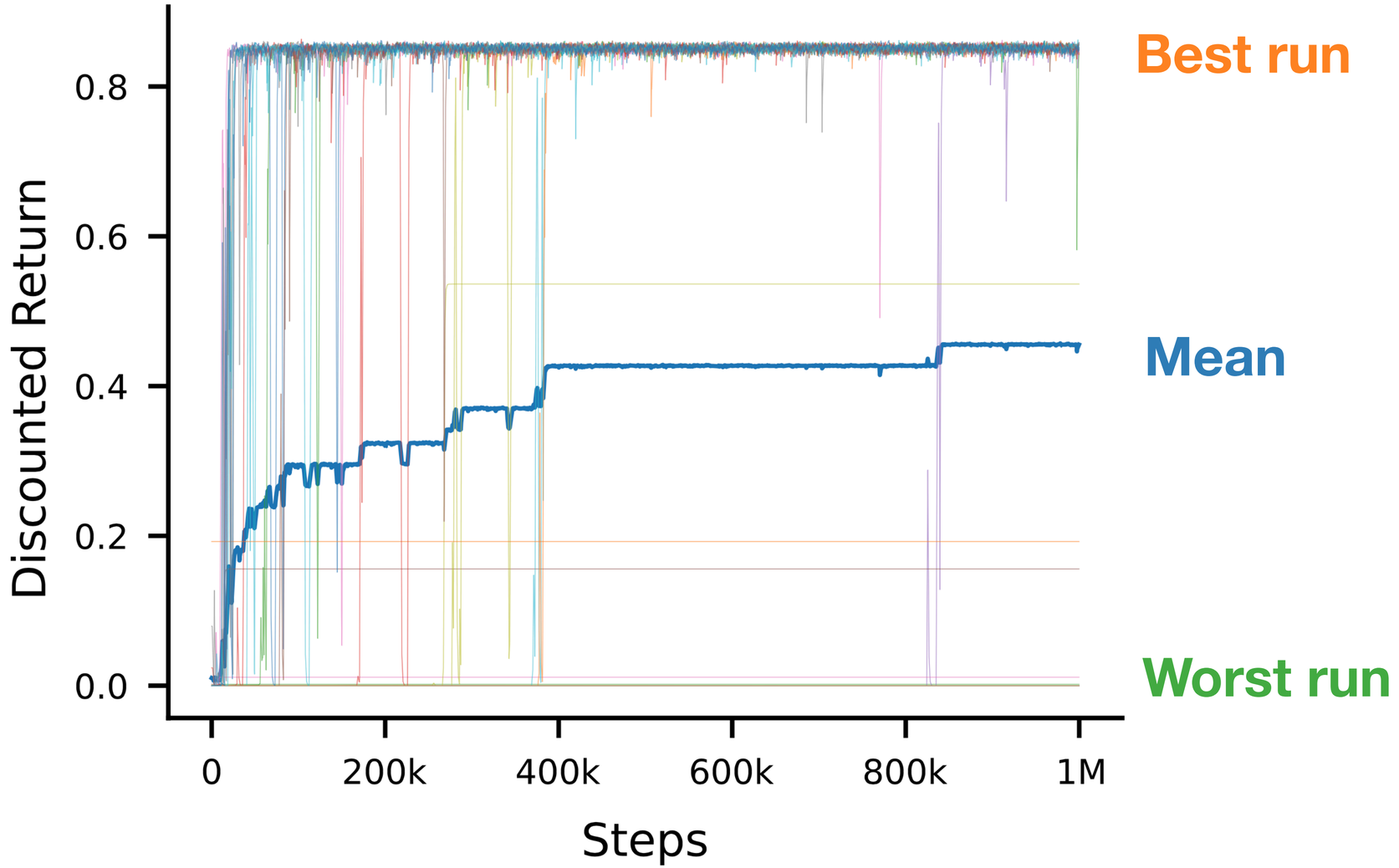
Starting simple

- Let’s start with a simple objective: understanding the performance of a single algorithm like Q-learning or Sarsa

- So many decisions to make:
  - How long should you run the experiment?
  - number of episodes or number of steps?
  - measure performance online or off-line?
  - How to aggregate performance?
  - Dealing with hyperparameters: rules of thumb or some systematic search?
  - What baselines to compare against?
  - Which environments?
  - …

- Different empirical questions demand difference design choices

Message 1: you need more runs than you think



E-Sarsa w. epsilon-greedy on a Maze

Thinking about the distribution of agent performance

- Two sources of variation in agent performance:

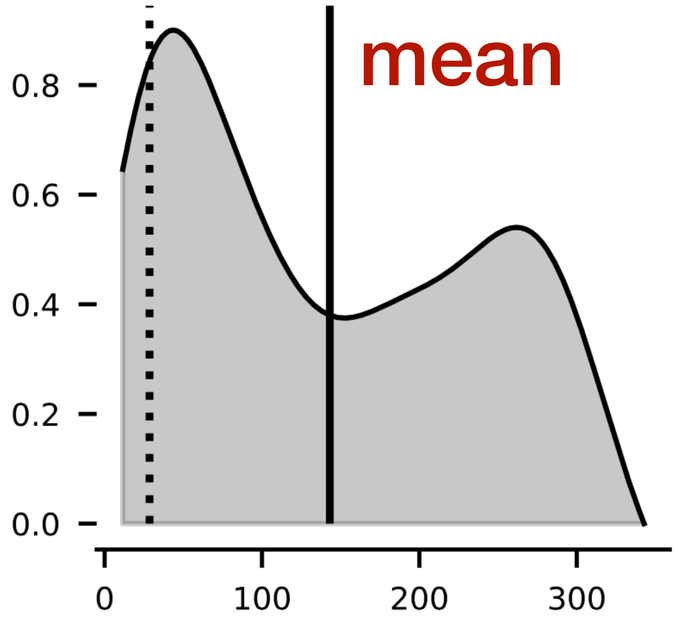
  A. Stochasticity due to agent initialization (e.g., NN weights) and during learning (e.g., exploration)

  B. Stochasticity in the **environment** such as random start states, transition noise etc.

- We want this variation because: (a) our algorithms benefit from randomization & (b) we want to evaluate the robustness of our algorithms to environmental conditions

- But this means every trial can exhibit very different performance

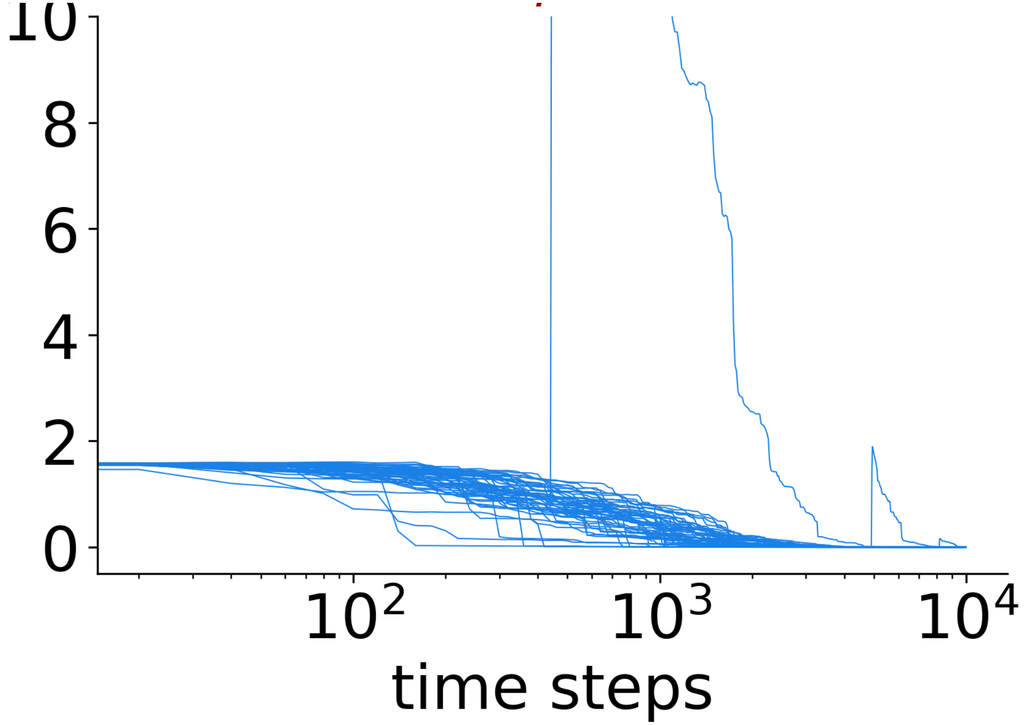
Performance distribution of DQN in Cartpole



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.

Results compiled by Andy Patterson

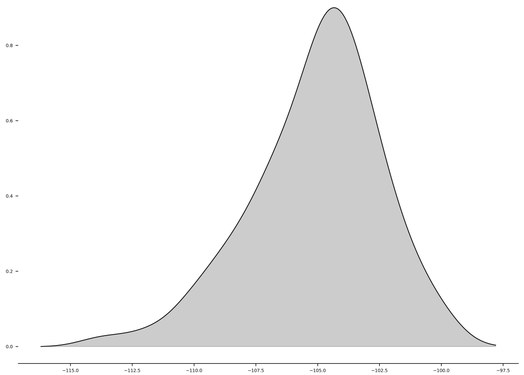
Advice: looking at all the data for additional insights



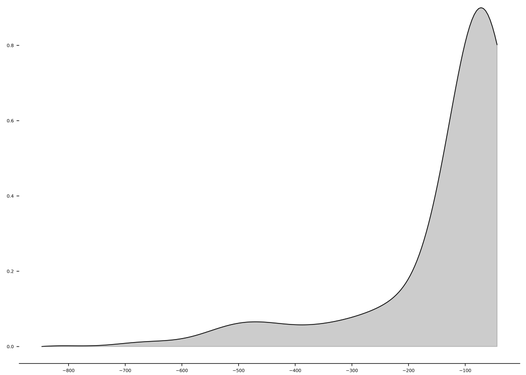
N-step off-policy TD on Baird's star-counterexample

Message 2: carefully consider how you aggregate data

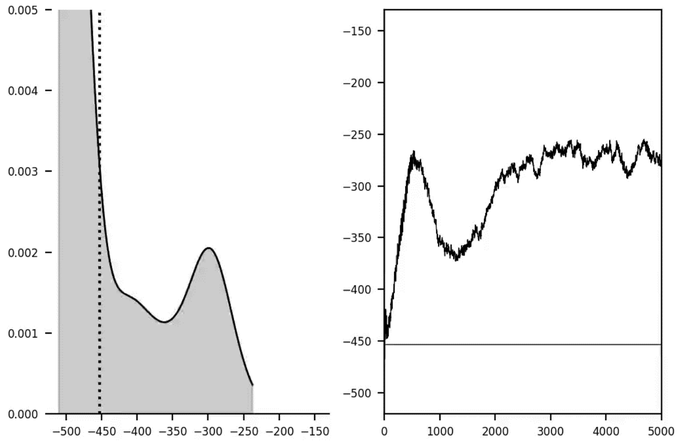
- We can characterize the variability in the distribution of performance
- Reporting the mean performance +/- variance <—this never goes to 0
  - … might not reflect performance well …



DQN Acrobot



Cliff World

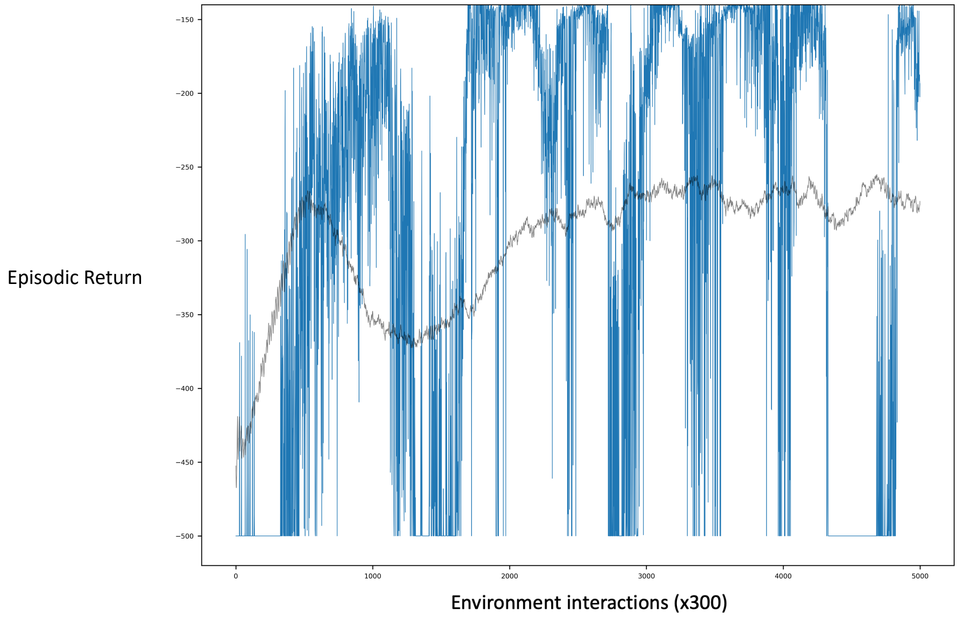


Mountain Car

Results & animation due to Andy Patterson

Message 2: carefully consider how you aggregate data

- **Tolerance intervals**: basically percentiles of agent performance + fudge factor based on the number of runs
- Not to be confused with confidence intervals
- DQN on Mountain Car averaged over 200 runs



Confidence interval

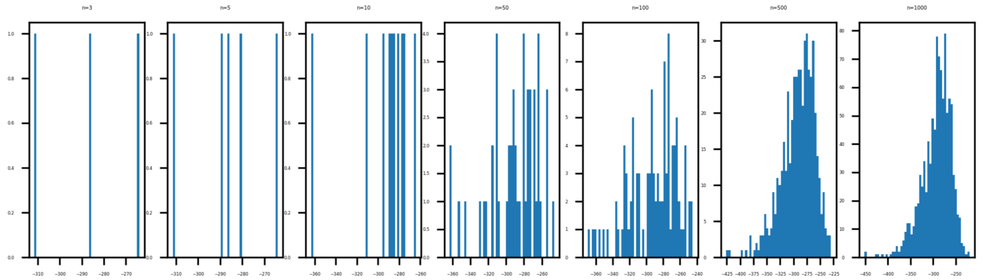
Mean with shaded region corresponding to 95% CI

Is this a good summary of variation in agent performance?

Tolerance interval

Message 2: carefully consider how you aggregate data

- **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

Mountain Car,
Sarsa(lambda) with tile coding

Results compiled by Raksha Kumaraswamy

Advice: do more runs AND choose aggregation techniques that take uncertainty into account

- Use a ***student-t*** interval that explicitly multiples the confidence band by a scalar based on the number of runs
  - **bigger multiplier** with fewer runs —> **wider confidence interval** due to uncertainty in variance estimate
- Use ***bootstrap*** confidence intervals that make no distributional assumptions:
  - Based on resampling your performance data repeatedly & building an empirical distribution
  - Typically requires more runs

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

Message 3: carefully think about hyperparameters

- Our algorithms are not yet as general as we like
- An **agent** (an algorithm + a full specification of its hyperparameters) can perform well in one environment and completely fail to learn in another environment
- How do we deal with these pesky parameters?

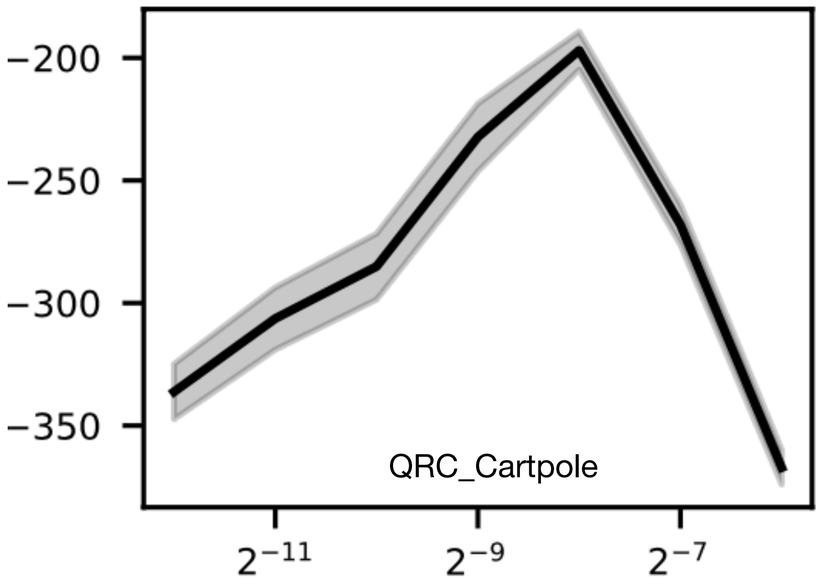
Consider your goal

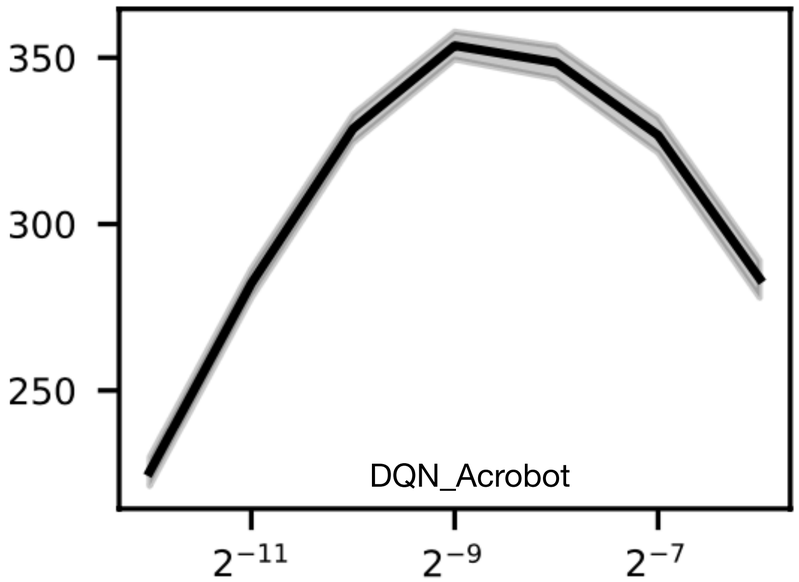
- **Goal #1**: we want to understand how an algorithm's performance changes as you vary key hyperparameters
- **Goal #2**: we have an application and we want to deploy an agent in the wild—tuning hypers on a water treatment plant is likely infeasible
- **Goal #3**: we want to prove one algorithm is best (SOTA)—likely impossible
  - Better but difficult goal: show a setting where an agent is better than others
    - Want to ensure result is fair and reproducible
    - Want to ensure hyperparameters of baselines are ***reasonably tuned***

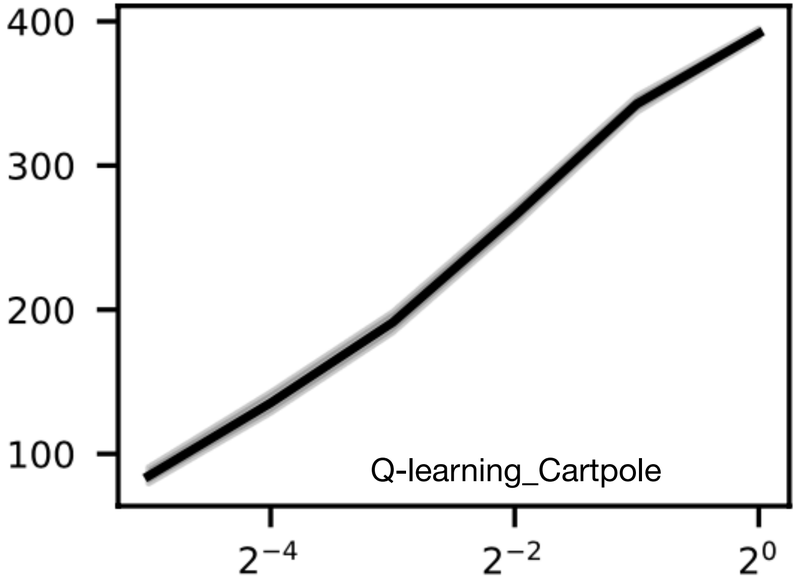
Understanding hyperparameter sensitivity

- Recipe:
  - Select a hyper that is important for performance (e.g., step-size alpha)
  - Decide on a range of values to test: e.g., alpha in {…, .0625, 0.125, .25, .5}
  - Test the algorithm for each alpha in the set, ***k*** times aggregating perf.
- Requires a lot of compute
- But provides significant insight into algorithm performance
  => path to parameter-free algorithms

Good and Bad sensitivity curves







α

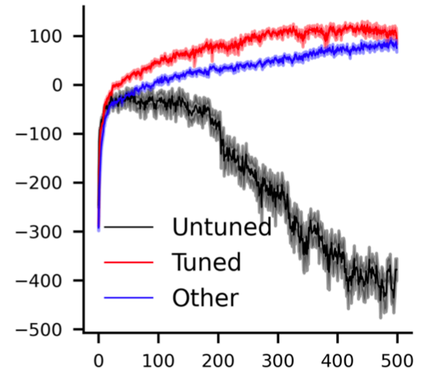
Results compiled by Andy Patterson

Advice: don’t think of hyperparameter sweeping as an algorithm

- It is not feasible in many deployment scenarios
- Sweeping is only feasible in **simulators**, but still not the best option
  - Bayesian methods and black box optimization schemes are better choices if you actually want to optimize hyperparameters
- Best to think of sweeping as a tool for insight

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
  - **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"
- **Solution**: Use an environment for which DQN's hyper-parameters have been tuned

This lecture is just tip of the iceberg

- There is no general recipe to follow
- **Be a skeptic**: try to break your own ideas and your own results
- Good empiricist are rare! The job market is competitive! Good empiricism is a competitive advantage!
  - Certainly matters in applications
- Inflated expectations and unscientific claims lead to AI-winters
- **There is hope:** many good papers on the topic, leaders of the field are doing the right things, conferences are cracking down

Coming soon: A Cookbook for Better RL

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

Selecting baselines

Measuring stability

Diagnostic MDPs

Living draft available on my website!

-vals

What to report

Hypothesis testing

Benchmarks vs challenge problems

Common errors

Multiple comparisons

Using a set of MDPs

Key Messages

- You need more runs/experiment replications (seeds) than you think
- Take a statistical point of view: think of distributions of performance, how you want to characterize that distribution and what assumptions you are making
- Sensitivity to hyperparameters is important, but sweeping is not an algorithmic answer
- 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

Thanks for listening!