

Reinforcement Learning with Ray Rllib



Dean Wampler
Data Data Texas, Jan. 28, 2023
dean@deanwampler.com
[@deanwampler](https://twitter.com/deanwampler)
@discuss.systems@deanwampler
deanwampler.com/talks
ray.io



Dean Wampler

Engineering Director,
Accelerated Discovery Platform
dean.wampler@ibm.com

<https://research.ibm.com/blog/what-is-accelerated-discovery>

<https://time.com/6249784/quantum-computing-revolution/>

TIME 2030
[← BACK TO HOME](#)

Quantum Computers Could Solve Countless Problems—And Create a Lot of New Ones



Research

Focus areas

Publications

Collaborate

Careers

Events

About

01 Feb 2022

Explainer

10 minute read

What's next in computing: The era of accelerated discovery

To meet the growing challenges of an ever-shifting world, the ways we have discovered new ideas in the past won't cut it moving forward. A convergence of computing revolutions taking place right now will help accelerate the rate of scientific discovery like nothing before.

erated discovery aim to develop, validate, and incubate technologies that accelerate the the scientific method, harnessing the capabilities of AI, hybrid cloud, and quantum computing. We-edge technologies, working together, to address society's greatest challenges — to find novel als able to help us fight challenges from pandemics to climate change and so much more.

Outline

- Why Reinforcement Learning?
- Ray RLlib
 - Aside: Why Ray?
- More Reinforcement Learning Concepts and Challenges
- Reinforcement Learning for Recommendations
- To Learn More...

A wide-angle photograph of a mountainous landscape. In the foreground, there's a large body of water, possibly a lake or a wide river, with shallow, sandy areas where small pools of water reflect the sky. The middle ground is dominated by a range of mountains with sharp peaks, some of which are partially covered in snow or ice. The lighting suggests it's either sunrise or sunset, with warm, golden light illuminating the peaks and casting long shadows. The background shows a dense forest of coniferous trees. The overall scene is serene and majestic.

<https://www.youtube.com/watch?v=Lu56xVIZ40M>

Why Reinforcement Learning?



TWO MINUTE PAPERS

WITH KÁROLY ZSOLNAI-FEHÉR



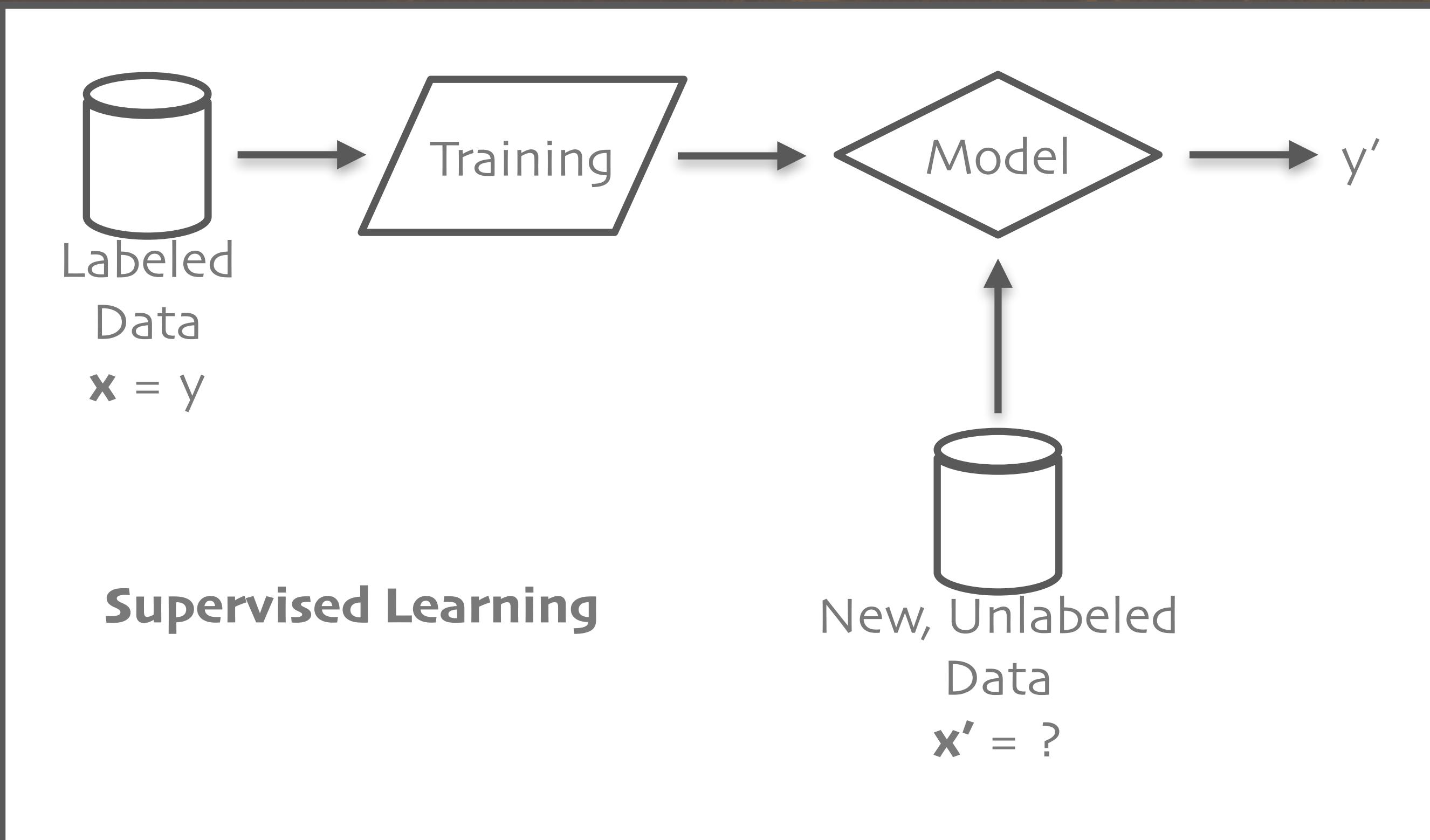
The Agent chooses an Action, then Observes any changes to the Environment and a Reward received, if any.

Through repeated steps like this, the Agent learns a Policy for maximizing the cumulative Reward.

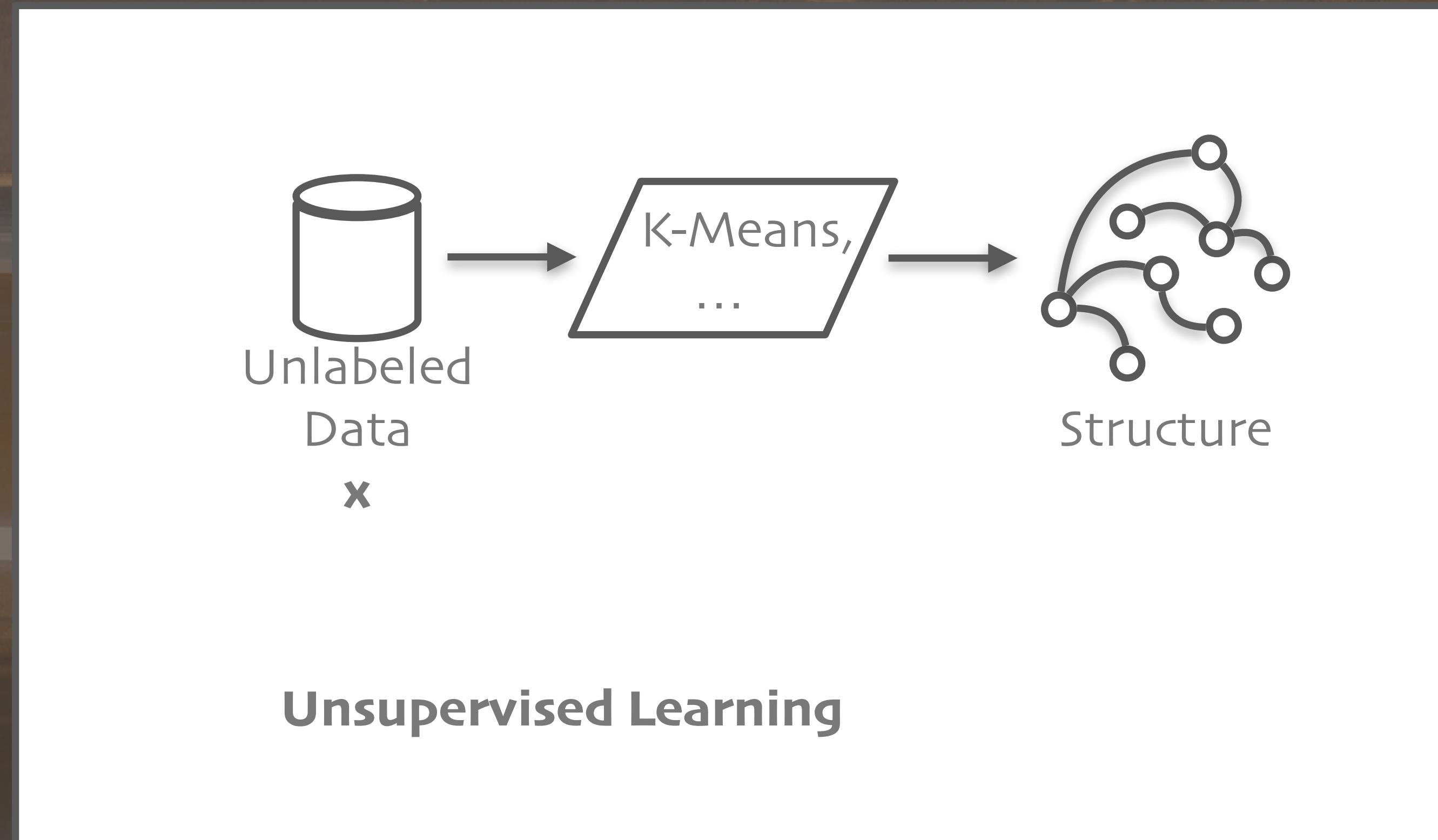
Each sequence is an Episode. It takes many Episodes to learn a good Policy.



Compared to Supervised Learning



Compared to Unsupervised Learning



RL Applications

AlphaGo, Atari, OpenAI Gym/
Gymnasium, ...

Games

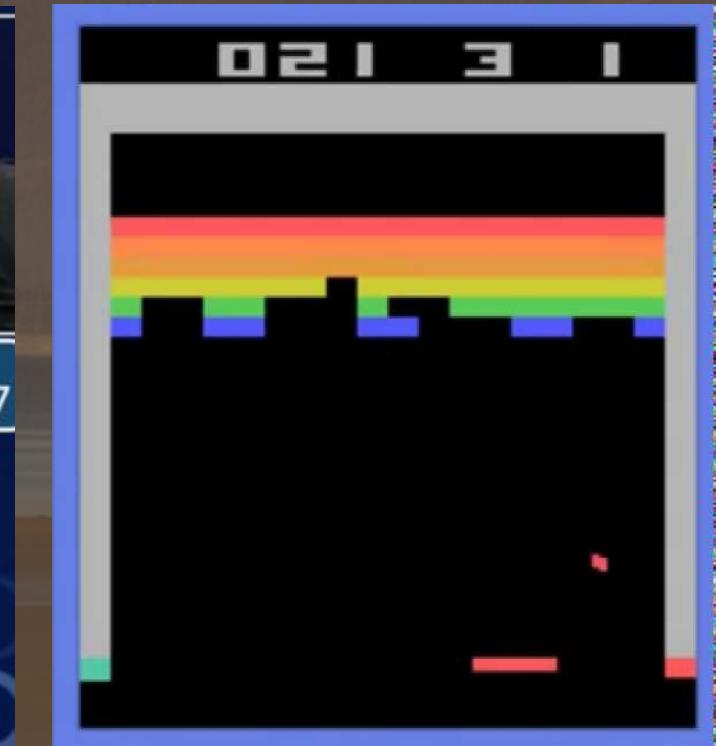
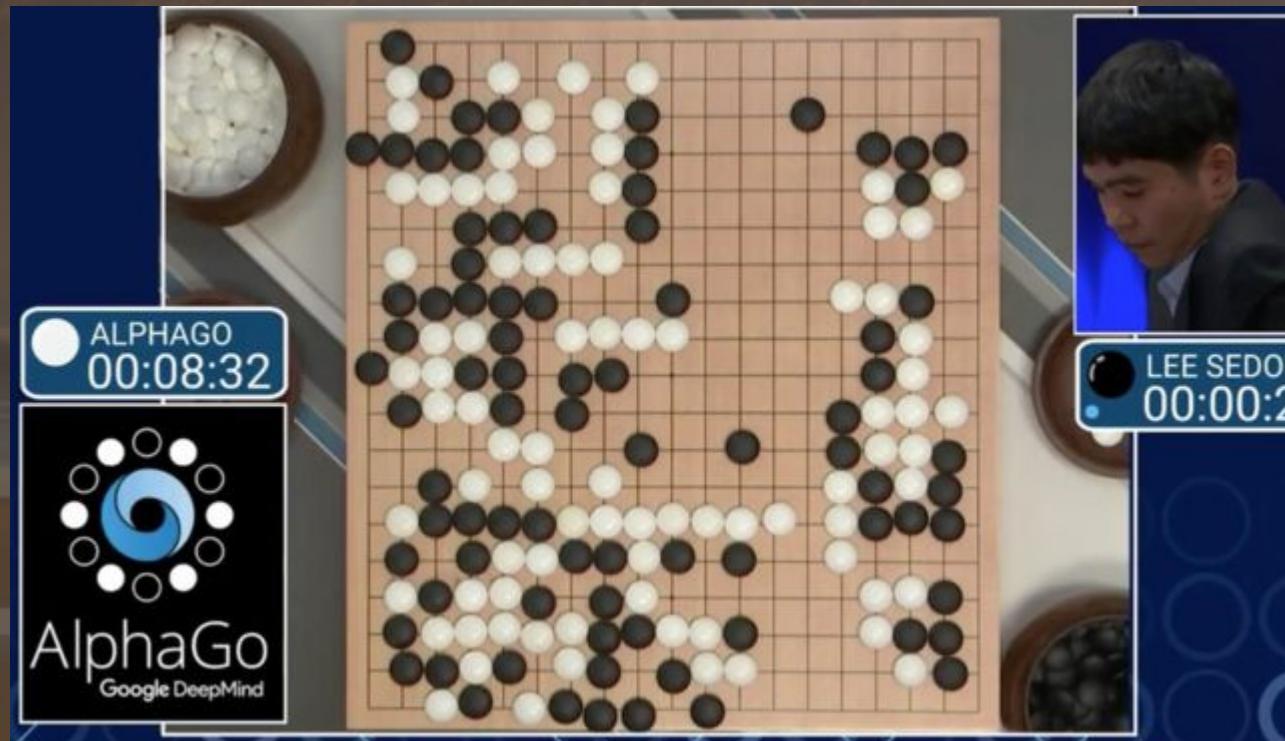
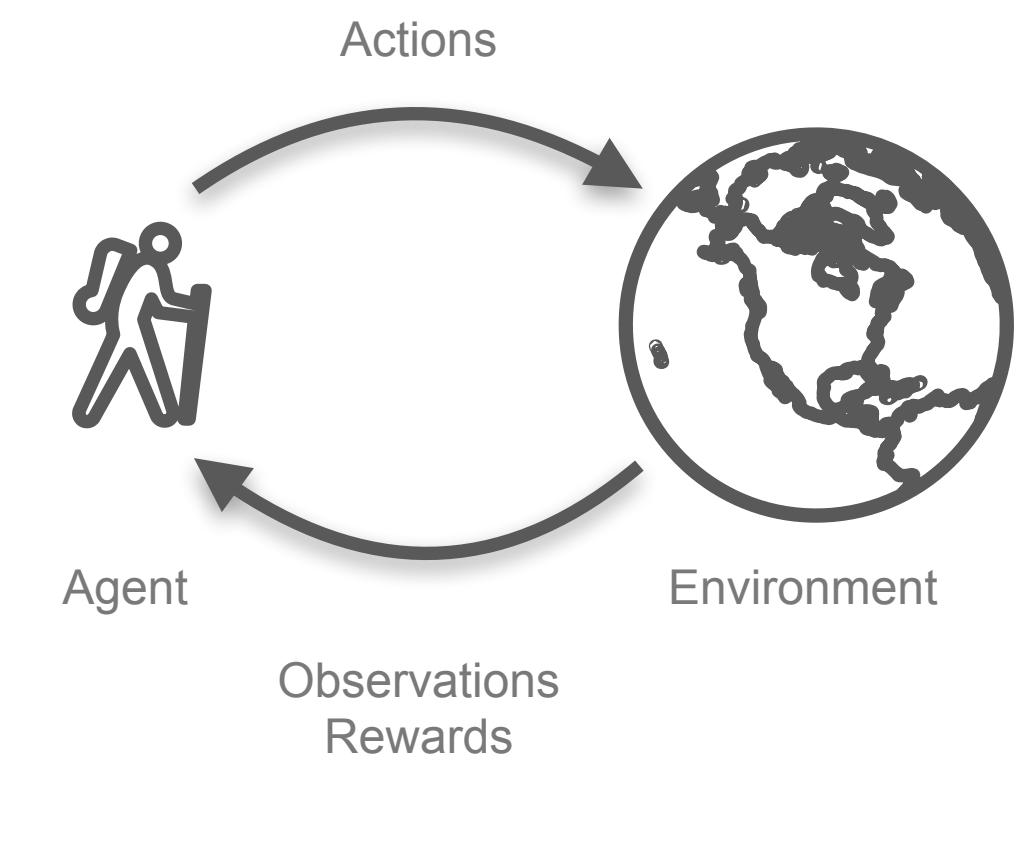
Robotics,
Autonomous
Vehicles

Industrial
Processes

System
Optimization

Advertising,
Recommendations

Finance



RL Applications

Autonomous vehicles, N-pedal robots, pick and place robots, ...

Games

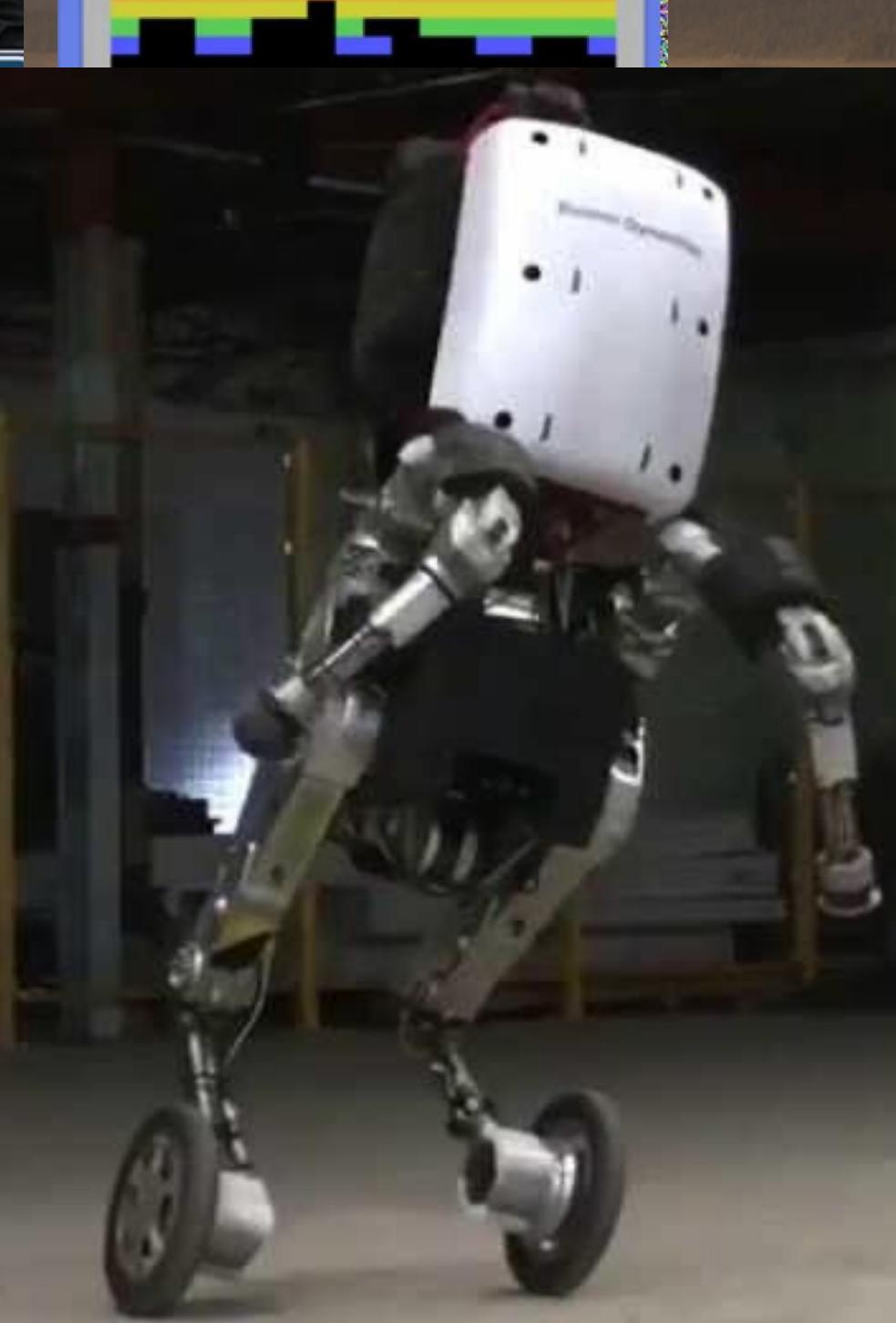
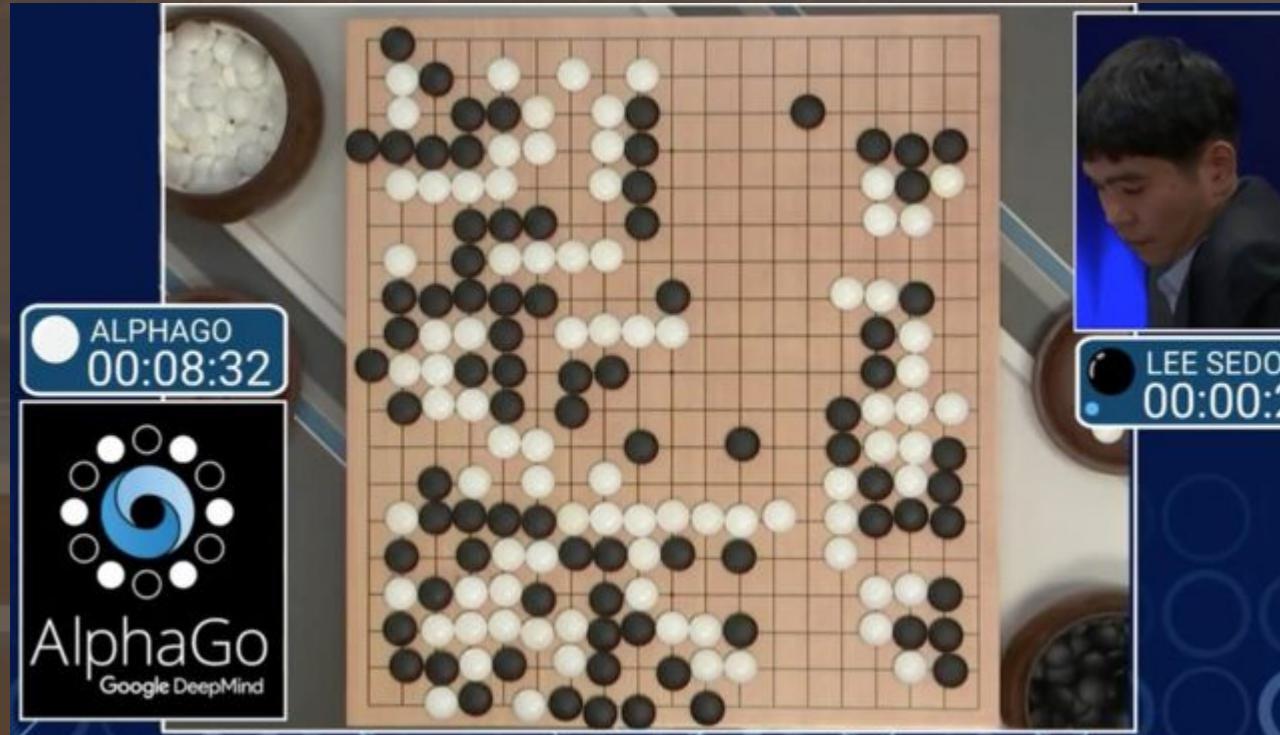
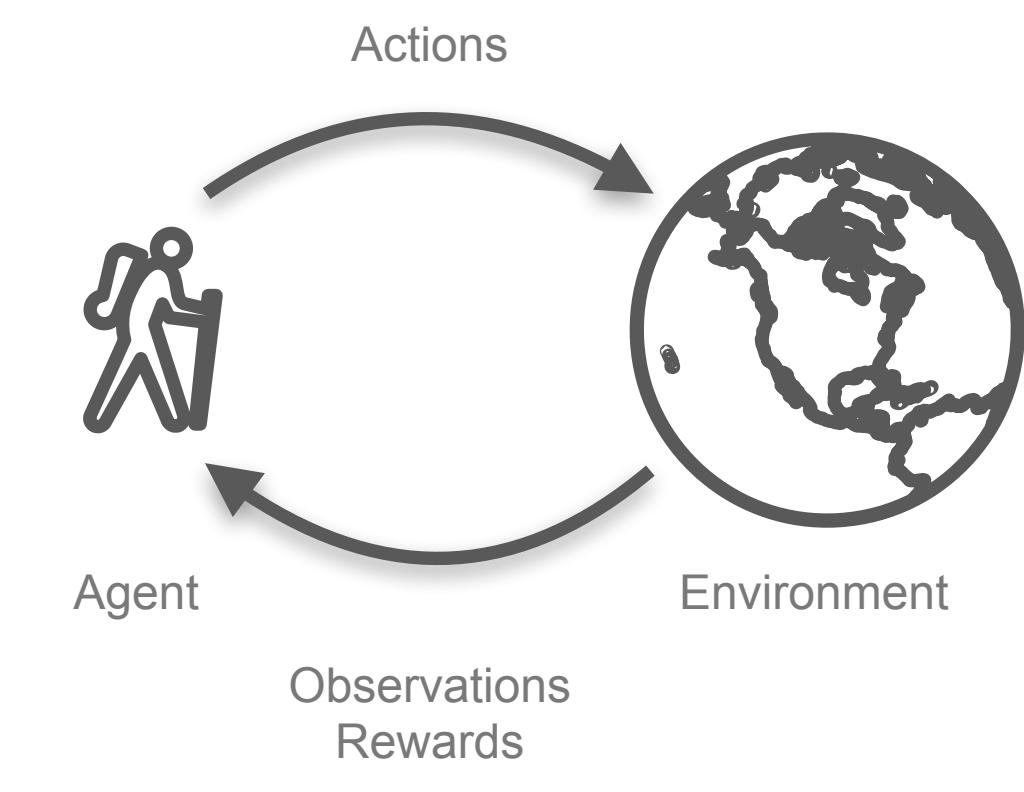
Robotics,
Autonomous
Vehicles

Industrial
Processes

System
Optimization

Advertising,
Recommendations

Finance



RL Applications

Assembly lines, warehouse and delivery routing, ...

Games

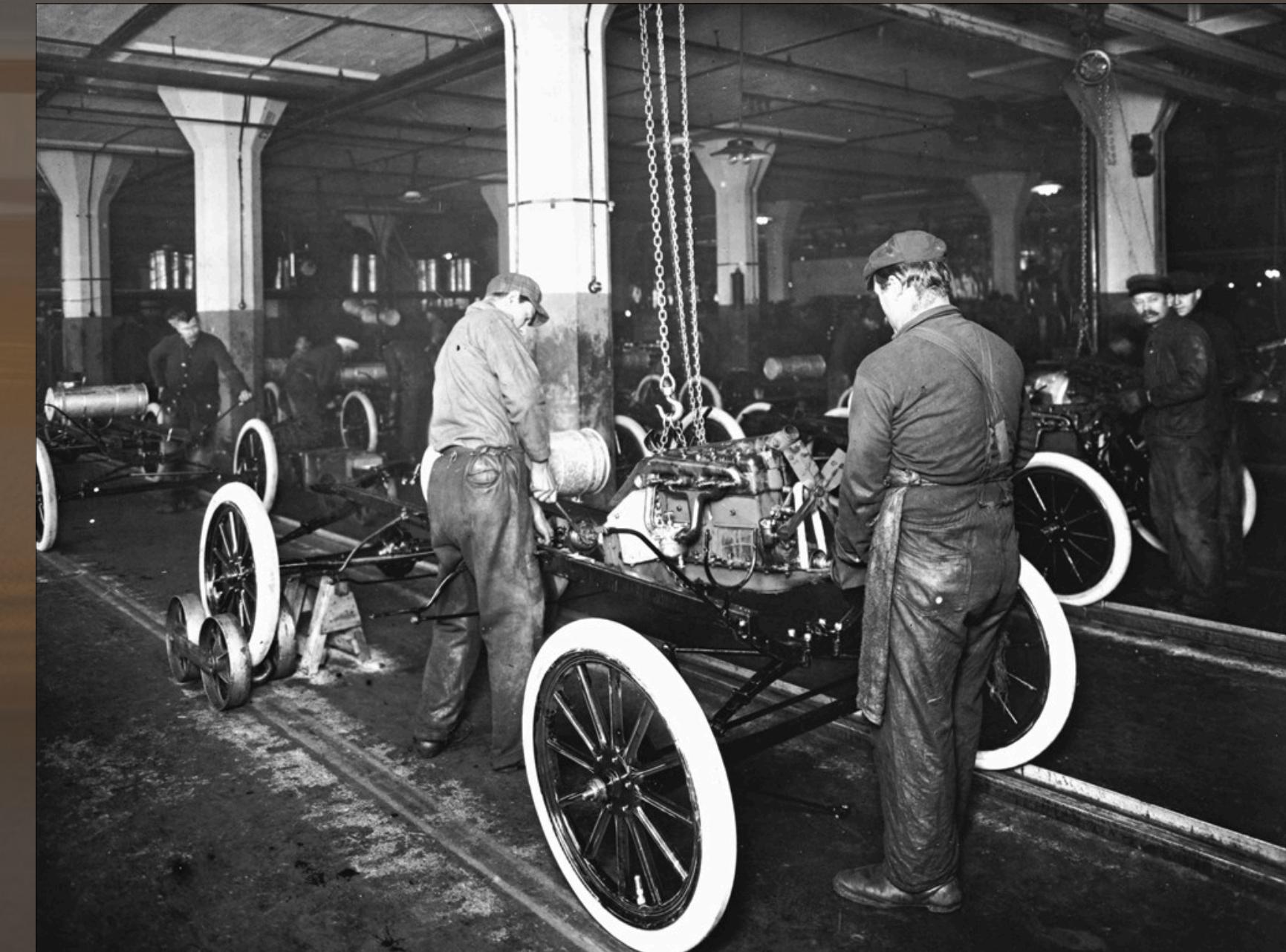
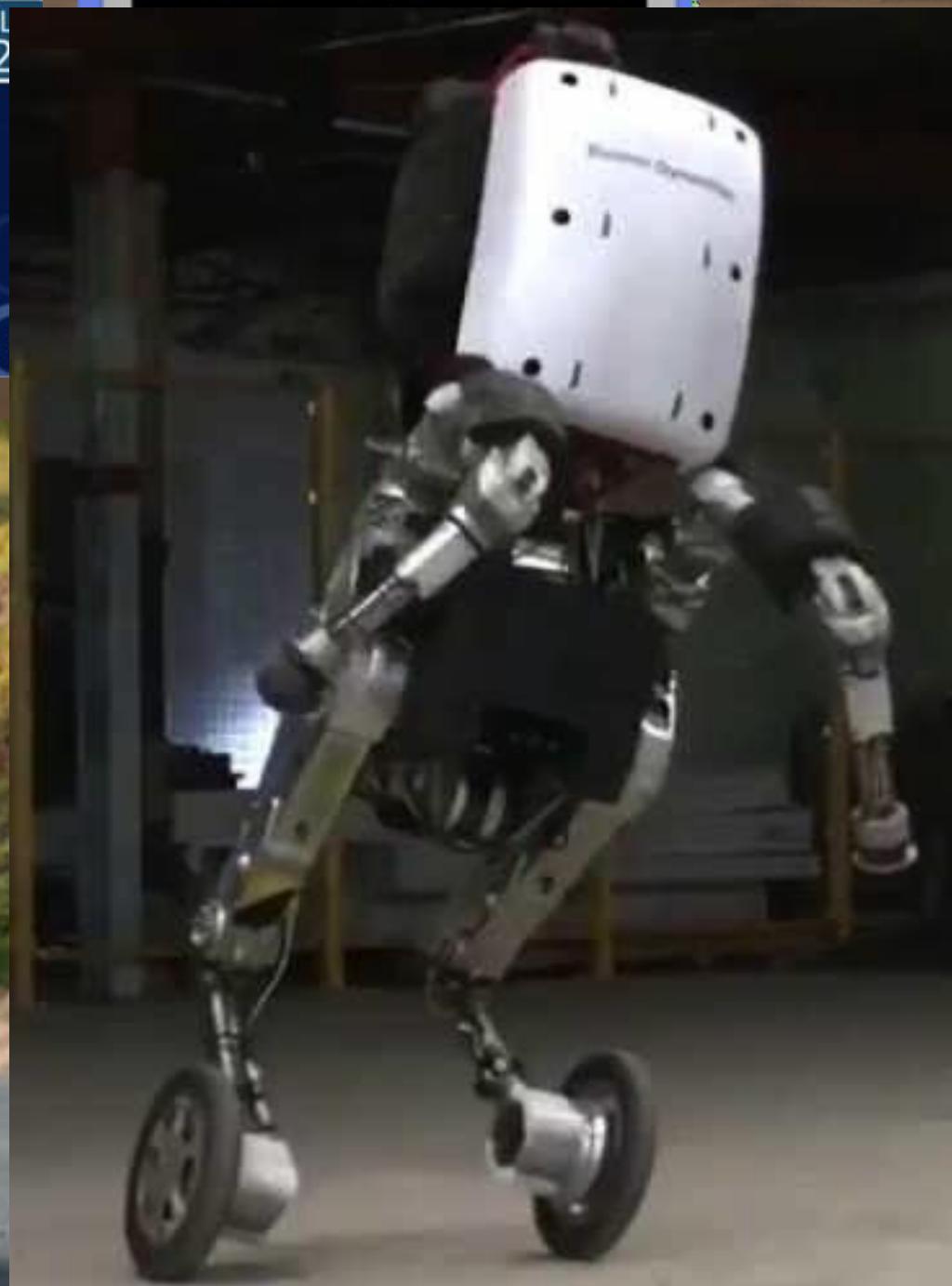
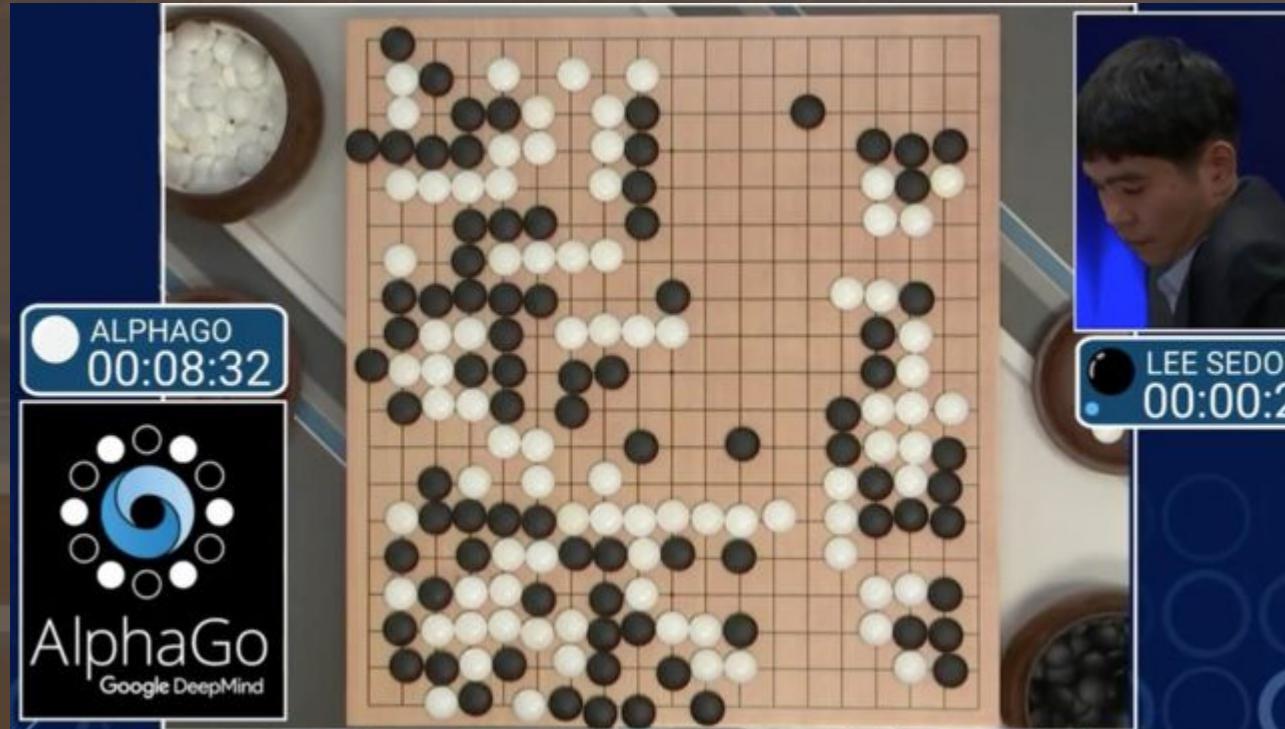
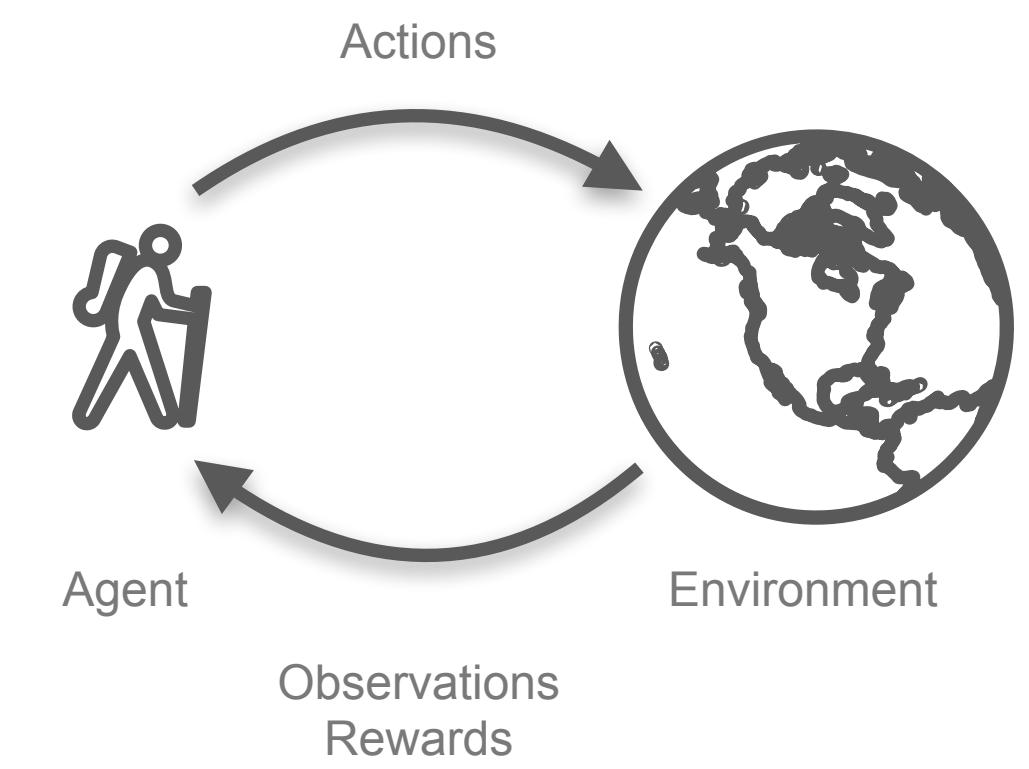
Robotics,
Autonomous
Vehicles

Industrial
Processes

System
Optimization

Advertising,
Recommendations

Finance



RL Applications

HVAC optimization, networks,
business processes, ...

Games

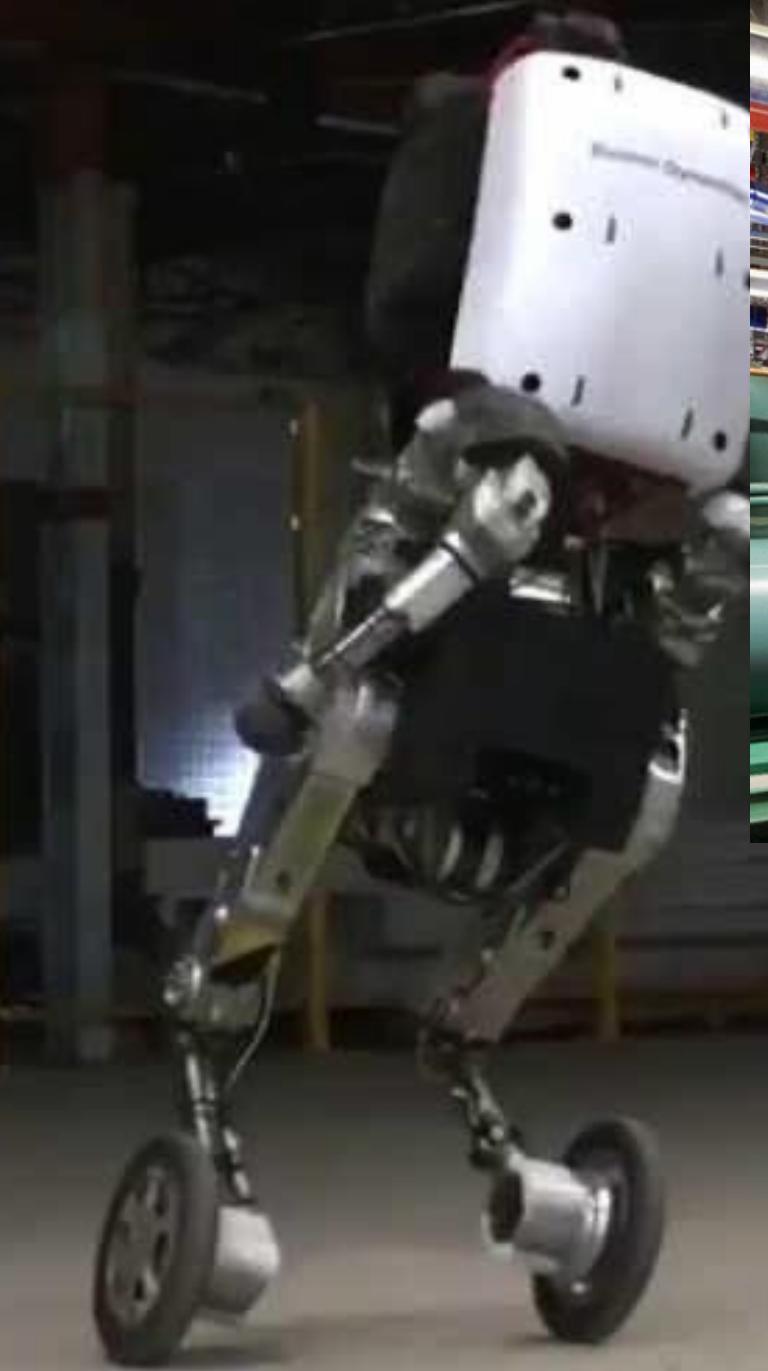
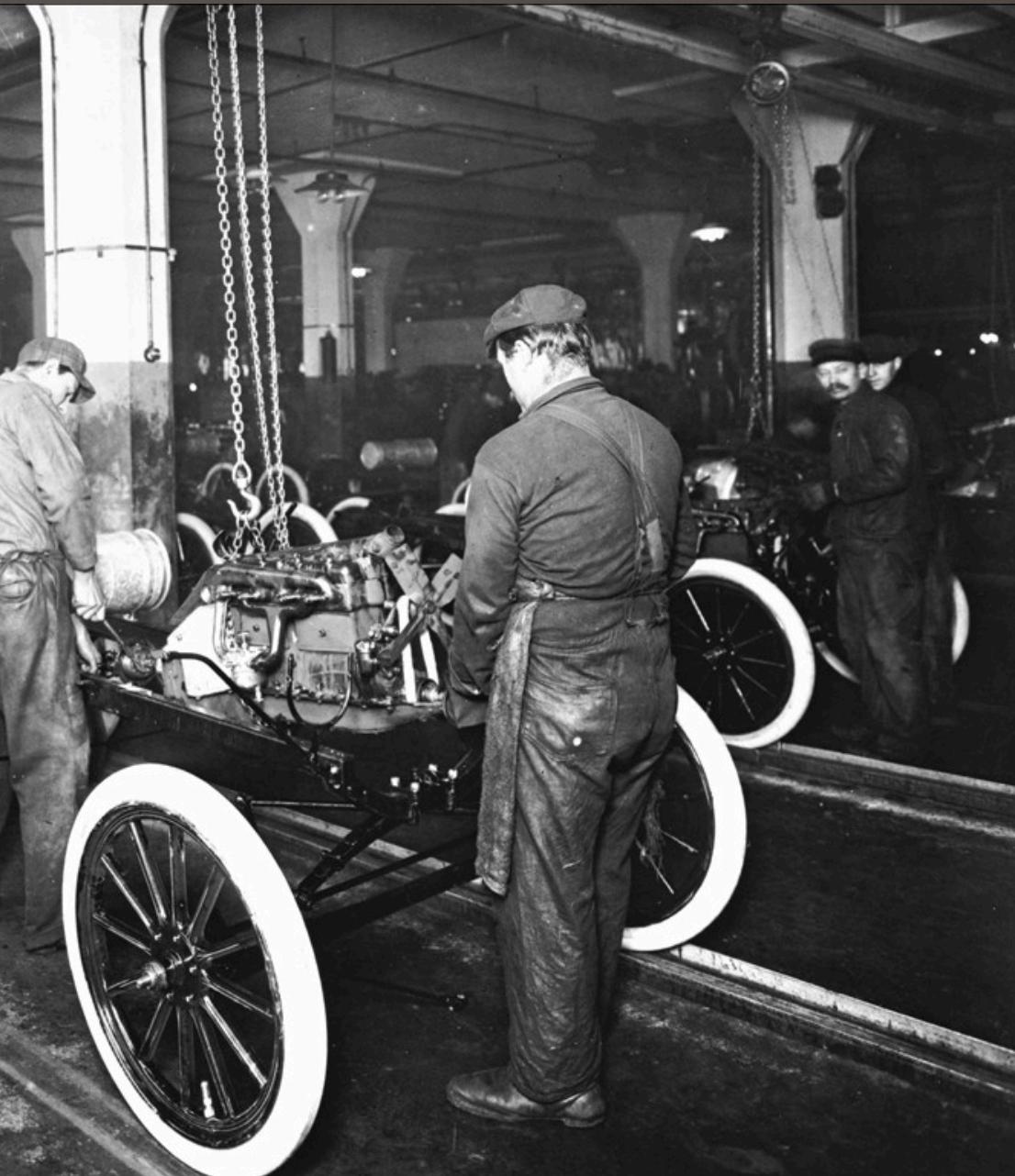
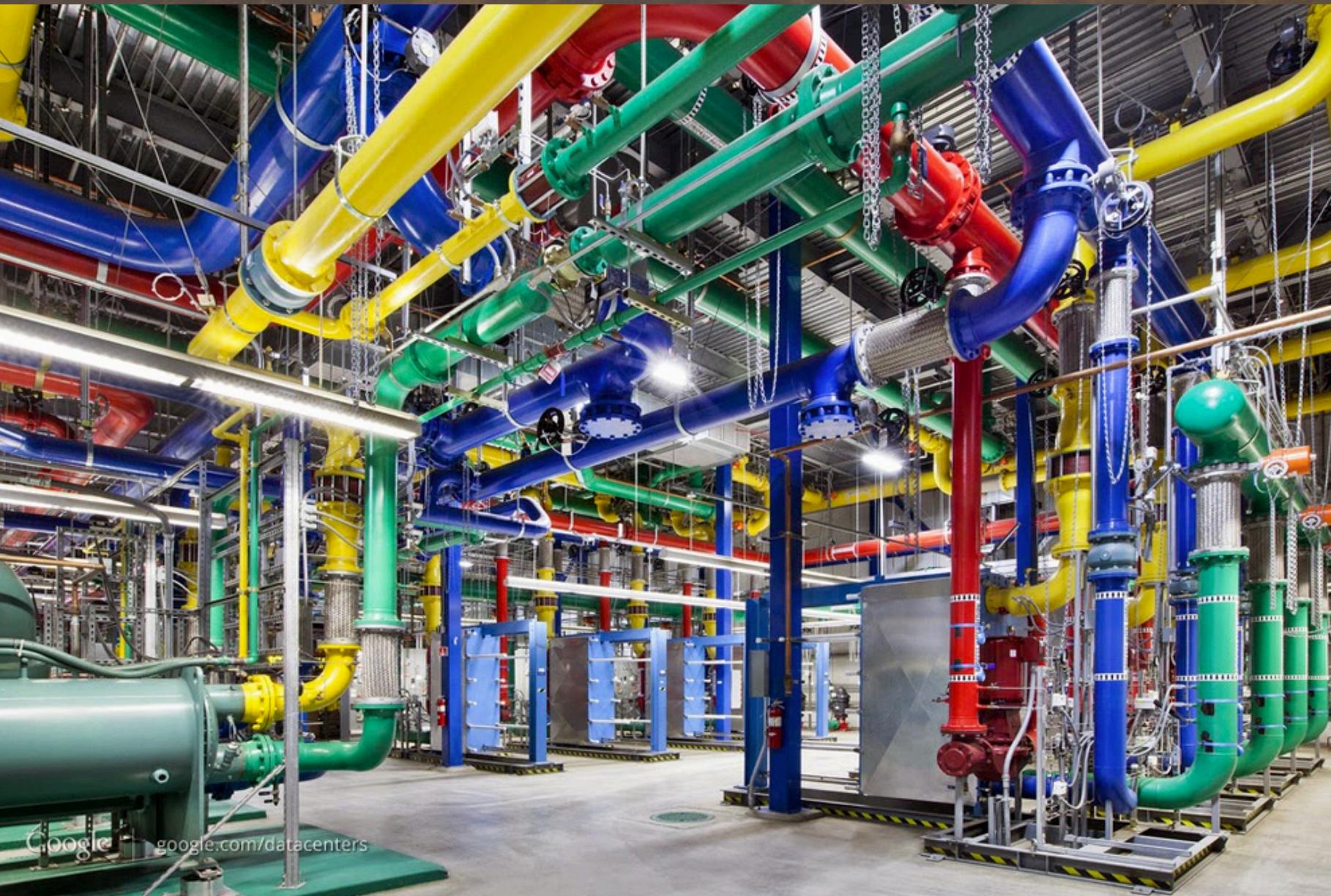
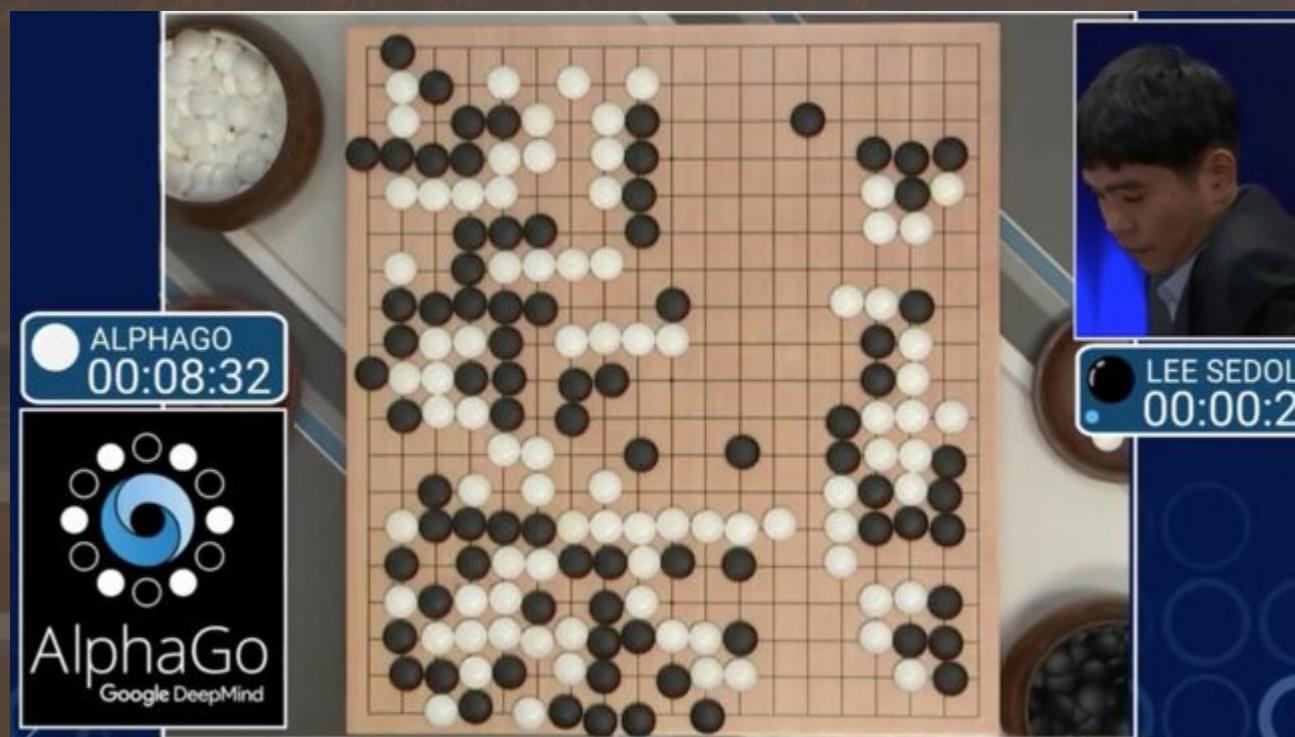
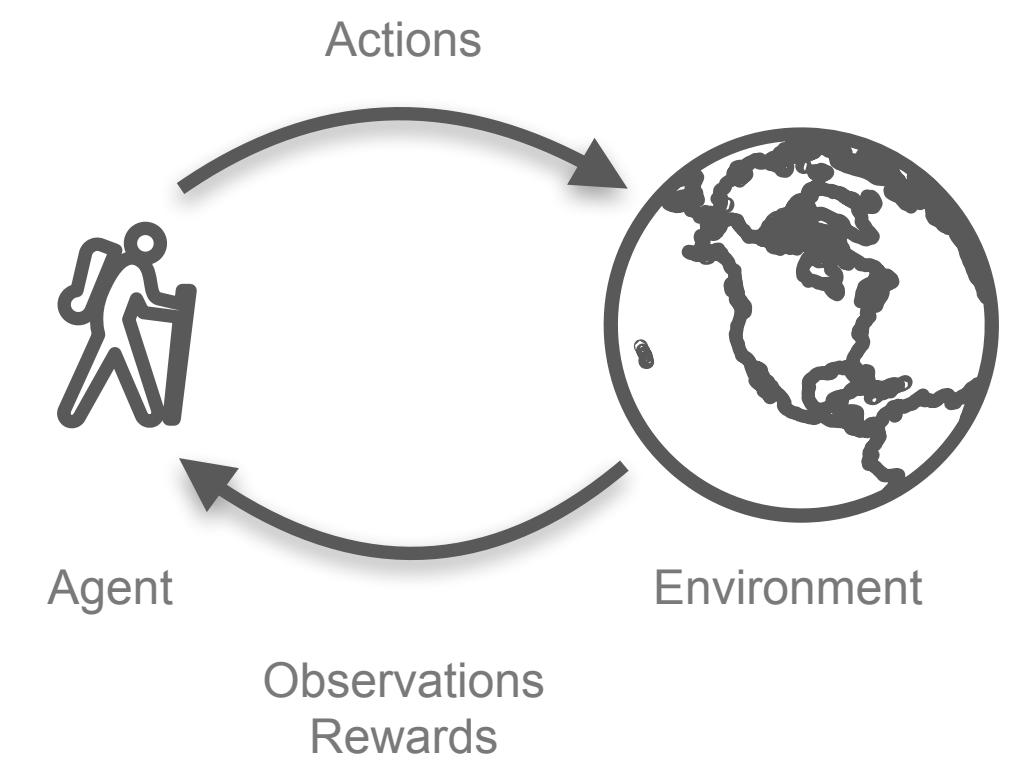
Robotics,
Autonomous
Vehicles

Industrial
Processes

System
Optimization

Advertising,
Recommendations

Finance



RL Applications

Better recommendations, ad placements, ...

Games

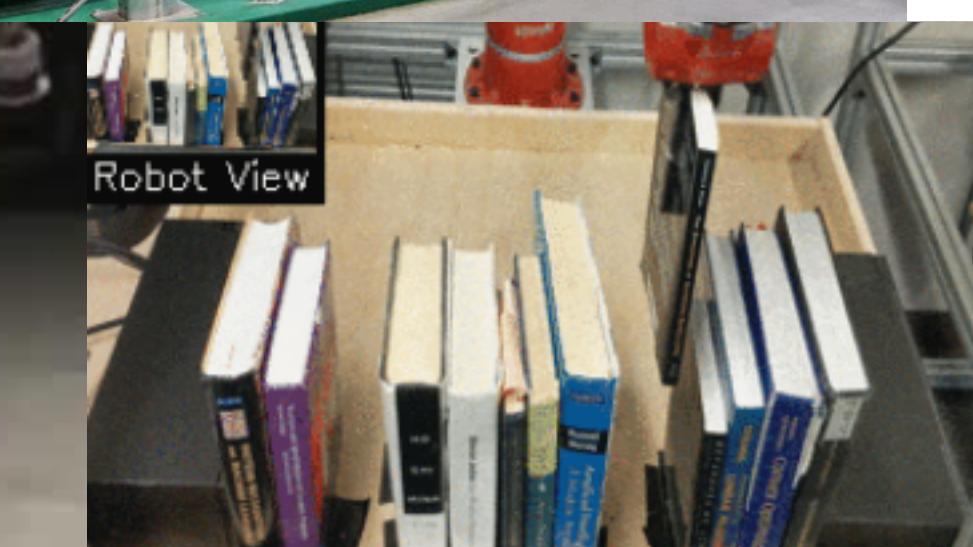
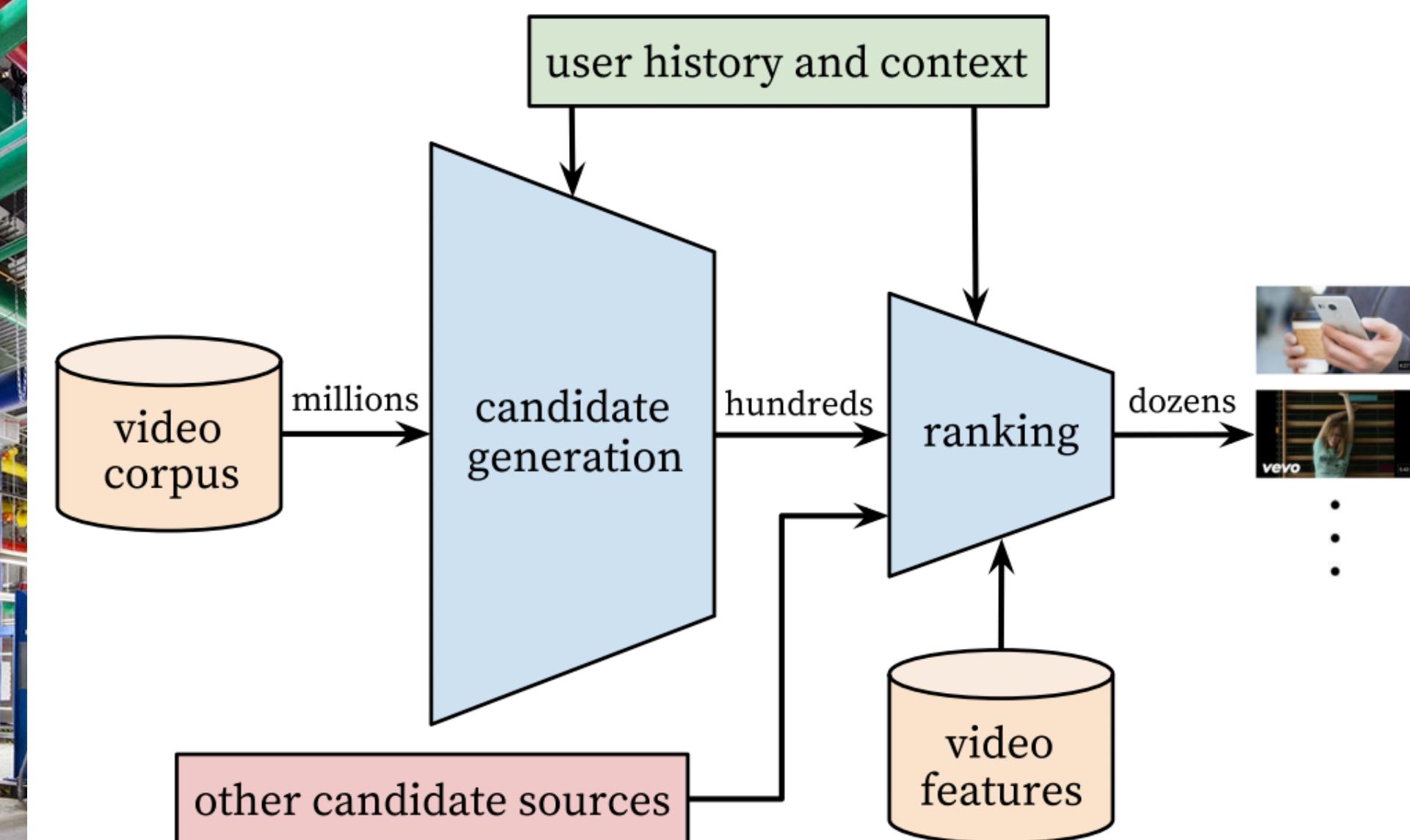
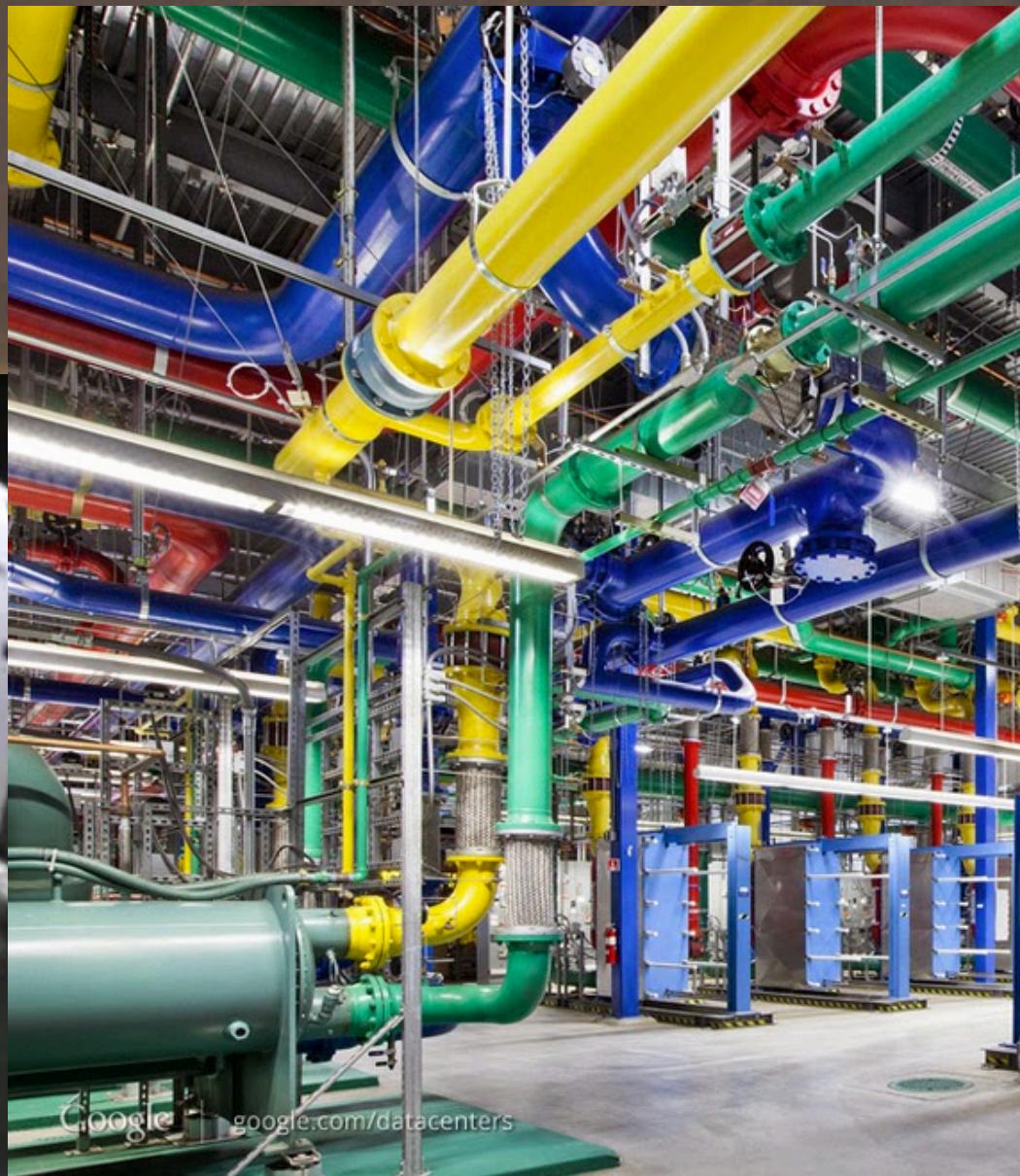
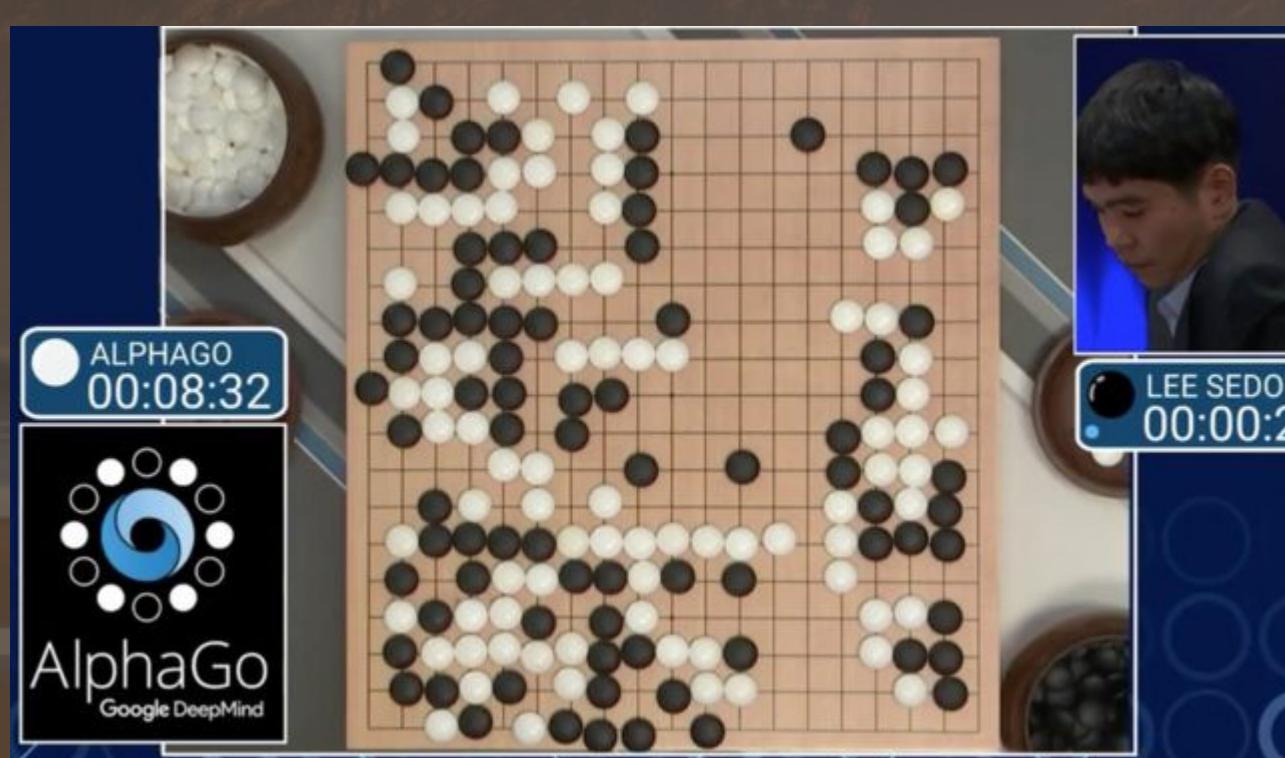
Robotics,
Autonomous
Vehicles

Industrial
Processes

System
Optimization

Advertising,
Recommendations

Finance



RL Applications

Market trends, timing of trades,

...

Games

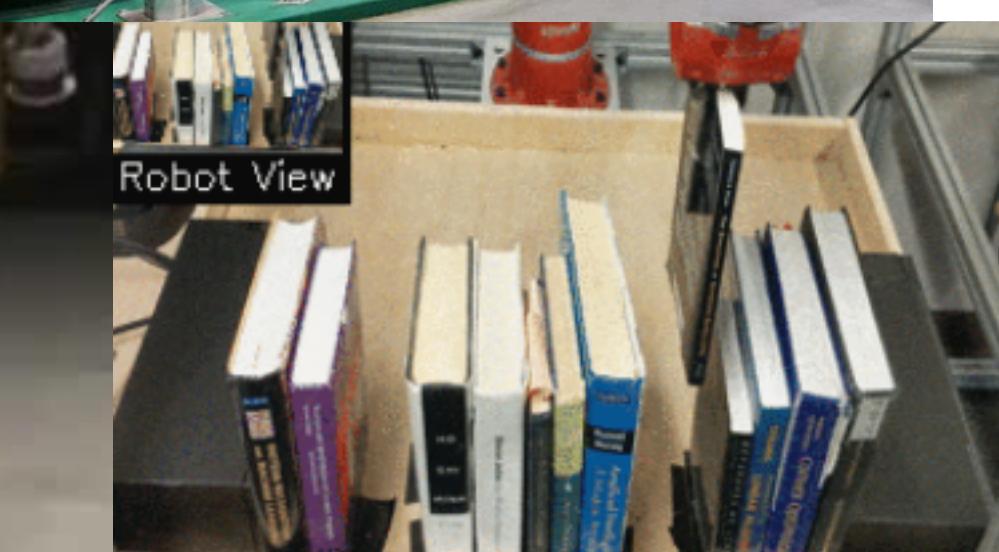
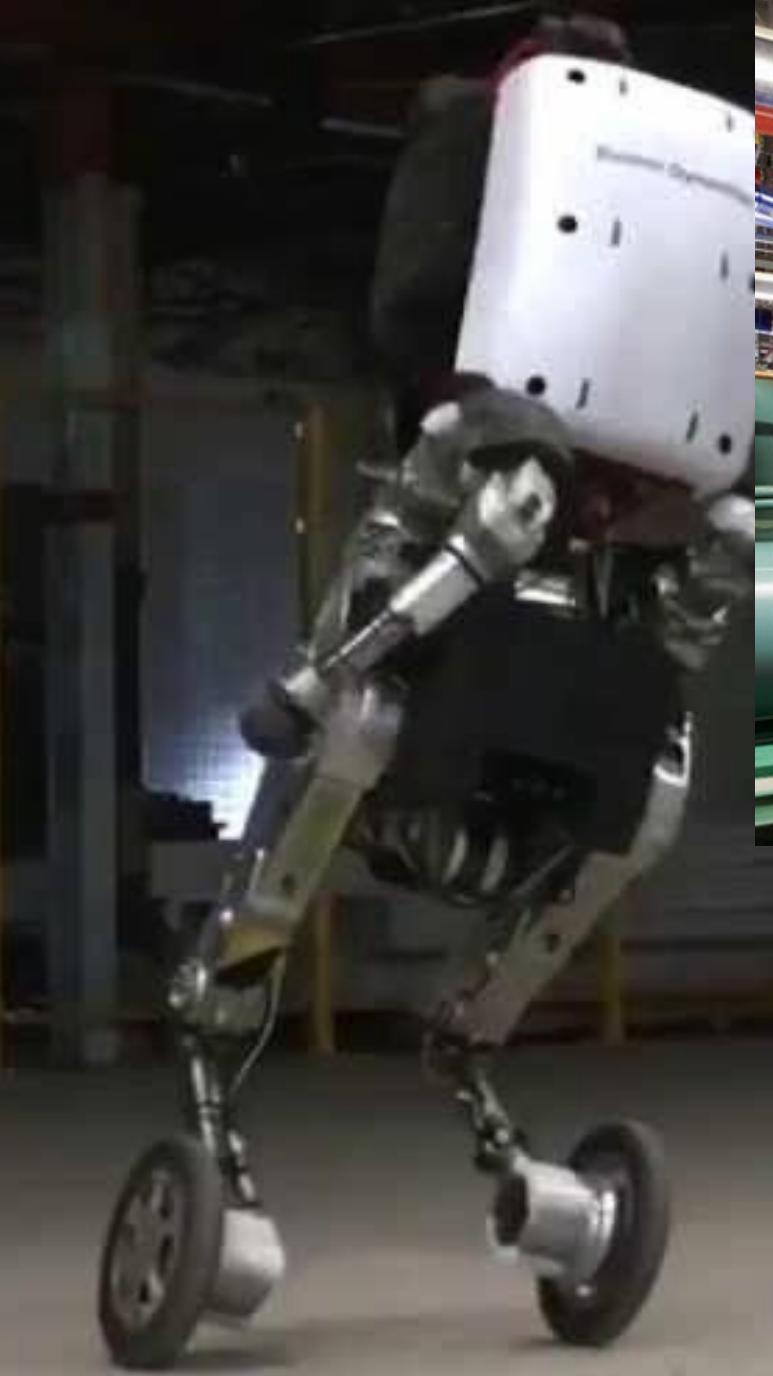
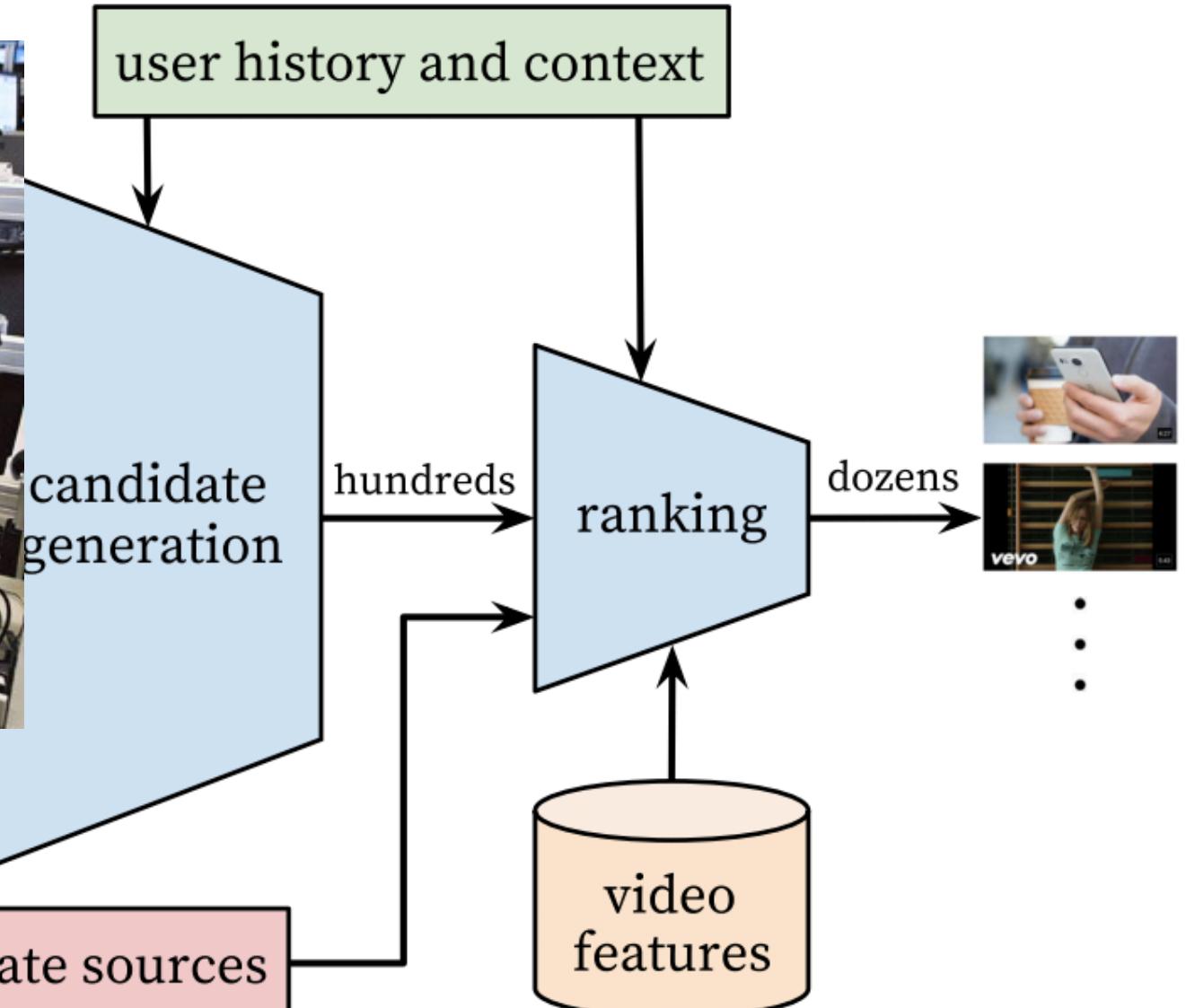
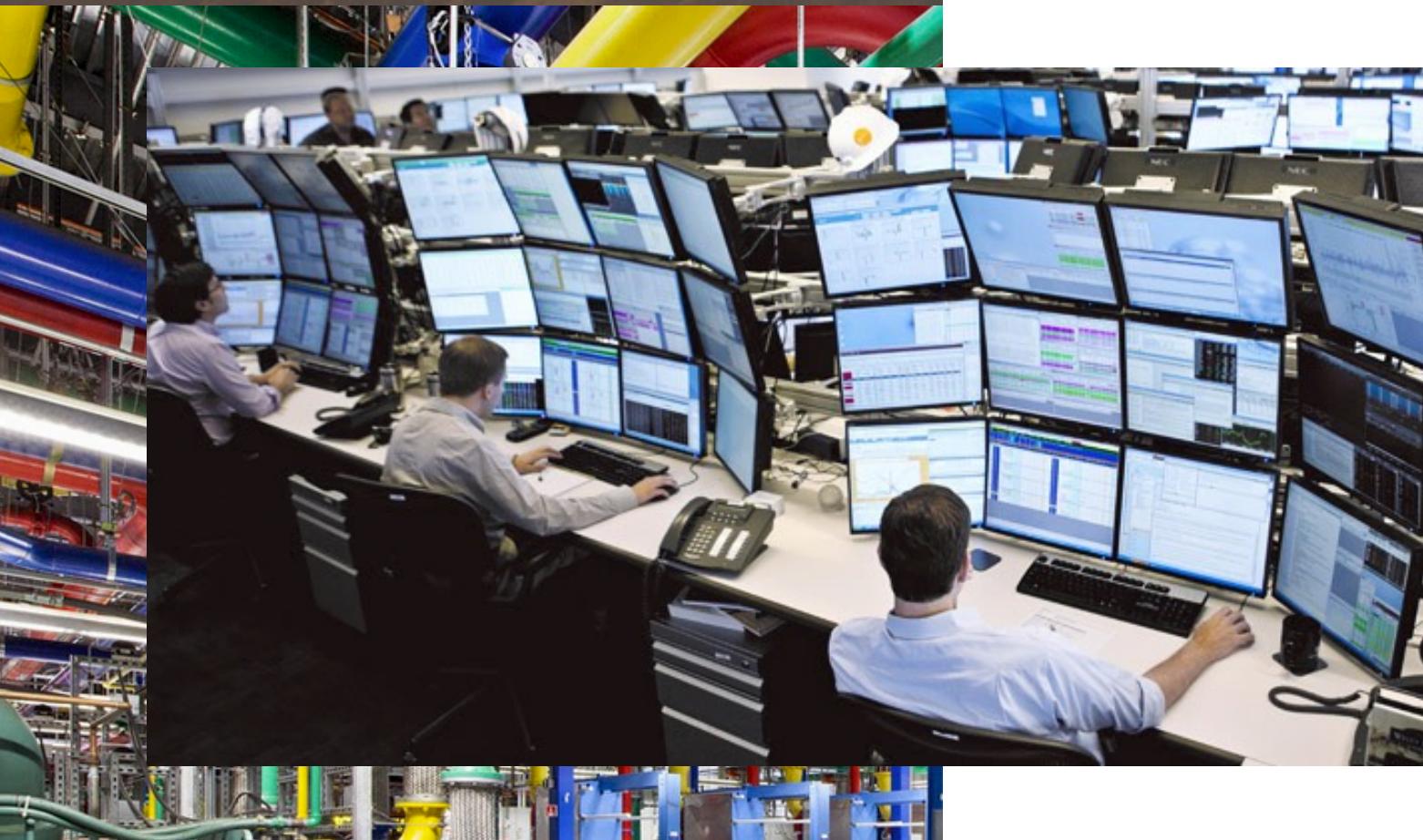
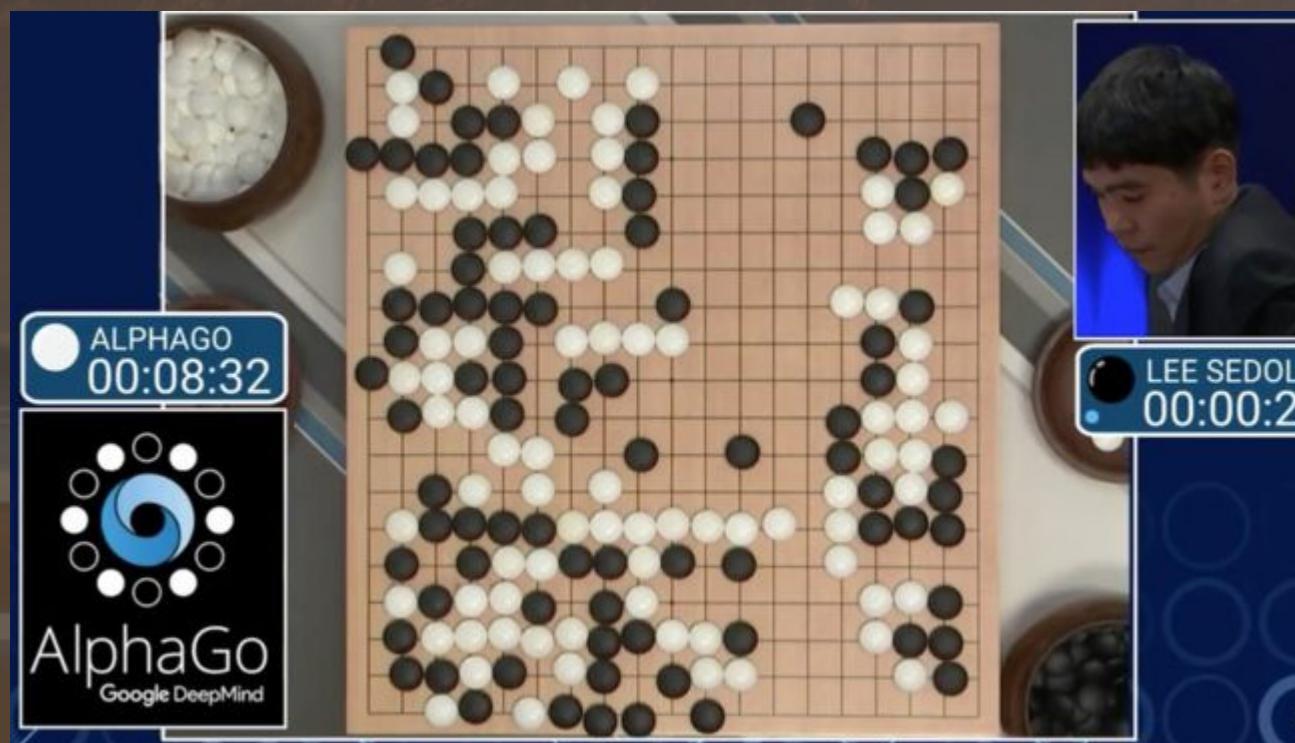
Robotics,
Autonomous
Vehicles

Industrial
Processes

System
Optimization

Advertising,
Recommendations

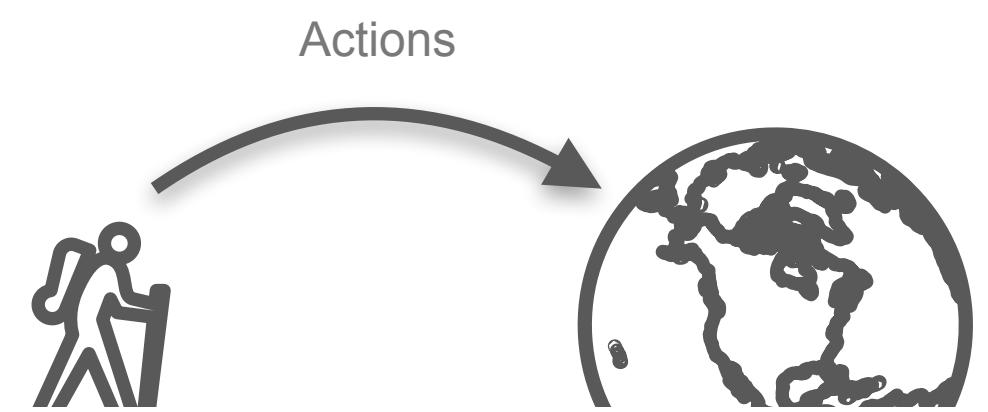
Finance



RL Applications

chatGPT!

<https://openai.com/blog/chatgpt/>



Introducing ChatGPT research release [Try ↗](#) [Learn more >](#)

OpenAI

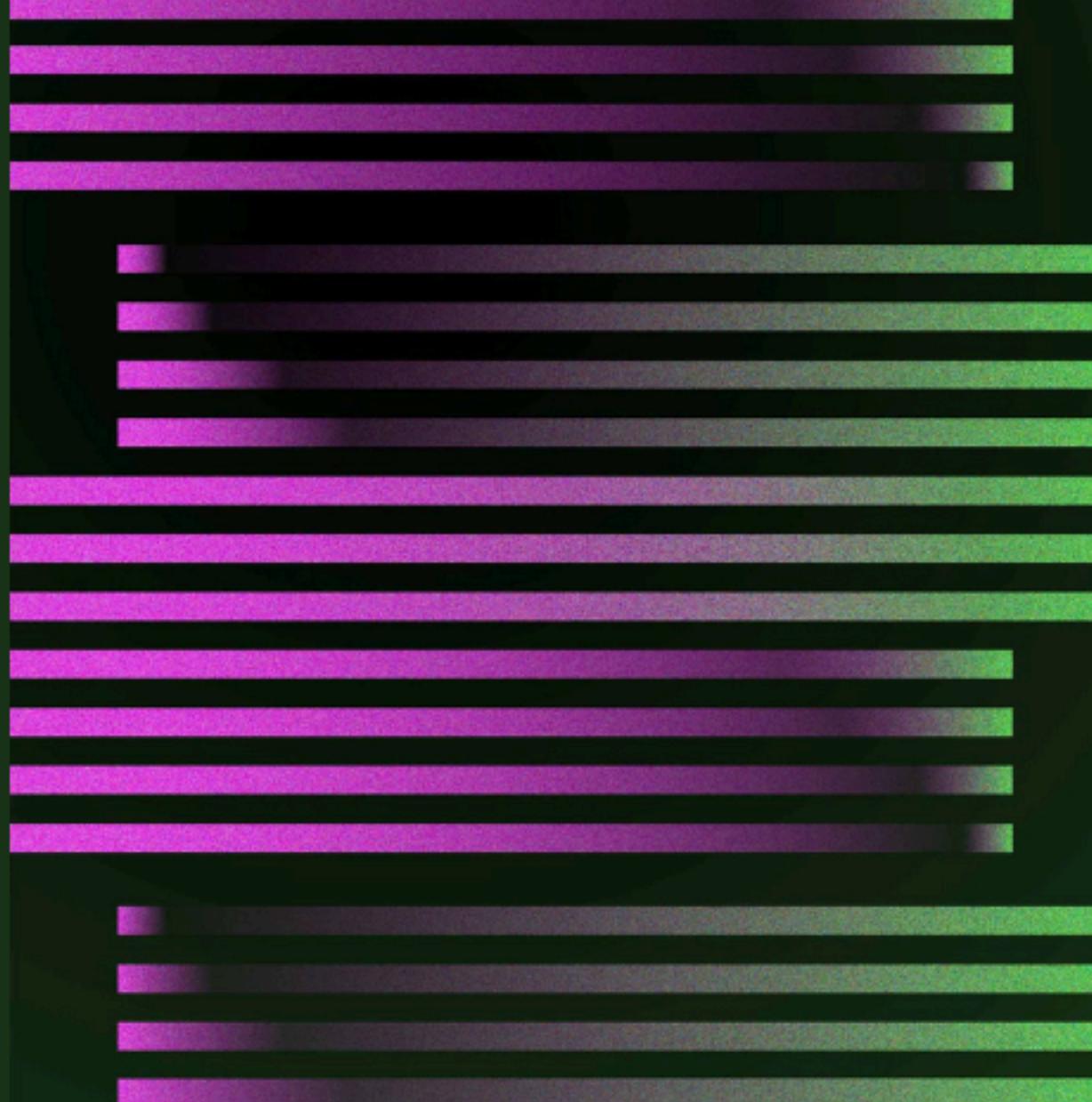
API RESEARCH BLOG ABOUT

ChatGPT: Optimizing Language Models for Dialogue

We've trained a model called ChatGPT which interacts in a conversational way. The dialogue format makes it possible for ChatGPT to answer followup questions, admit its mistakes, challenge incorrect premises, and reject inappropriate requests. ChatGPT is a sibling model to InstructGPT, which is trained to follow an instruction in a prompt and provide a detailed response.

[TRY CHATGPT ↗](#)

November 30, 2022
13 minute read



Methods

We trained this model using Reinforcement Learning from Human Feedback (RLHF), using the same methods as [InstructGPT](#), but with slight differences in the data collection setup. We trained an initial model using supervised fine-tuning: human AI trainers provided conversations in which they played both sides—the user and an AI assistant. We gave the trainers access to model-written suggestions to help them compose their responses. We mixed this new dialogue dataset with the InstructGPT dataset, which we transformed into a dialogue format.

To create a reward model for reinforcement learning, we needed to collect comparison data, which consisted of two or more model responses ranked by quality. To collect this data, we took conversations that AI trainers had with the chatbot. We randomly selected a model-written message, sampled several alternative completions, and had AI trainers rank them. Using these reward models, we can fine-tune the model using [Proximal Policy Optimization](#). We performed several iterations of this process.

Step 1
Collect demonstration data and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

Step 2
Collect comparison data and train a reward model.

A prompt and several model outputs are sampled.

A Explain reinforcement learning to a 6 year old.
B Explain rewards...
C In machine learning...
D We give treats and punishments to teach...

Step 3
Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

A new prompt is sampled from the dataset.

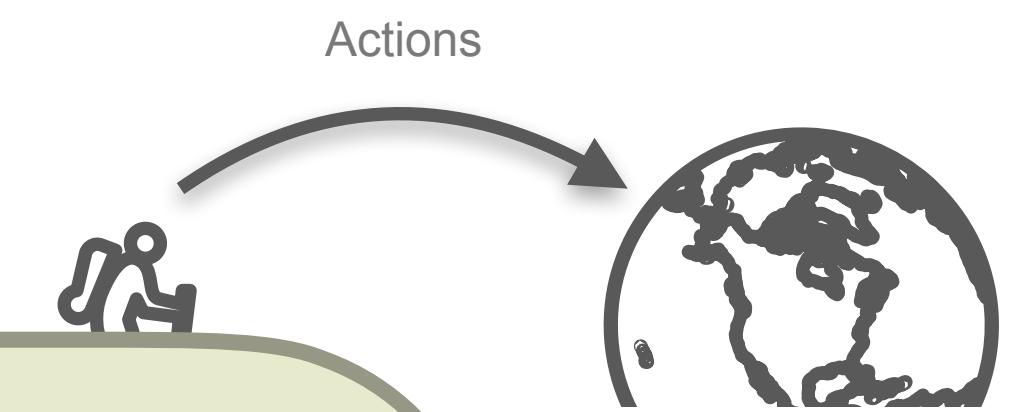
The PPO model is initialized from the supervised policy.

A Write a story about otters.
B PPO

RL Applications

chatGPT!

<https://openai.com/blog/chatgpt/>



← → C https://openai.com/blog/chatgpt/

Introducing ChatGPT



ChatGPT: Optimizing Language Models for Dialogue

We've trained a model called ChatGPT which interacts in a conversational way. The dialog format makes it possible for ChatGPT to answer followup questions, admit its mistakes, challenge incorrect premises, and reject inappropriate requests. ChatGPT is a sibling model to InstructGPT, which is trained to follow an instruction in a prompt and provide a detailed response.

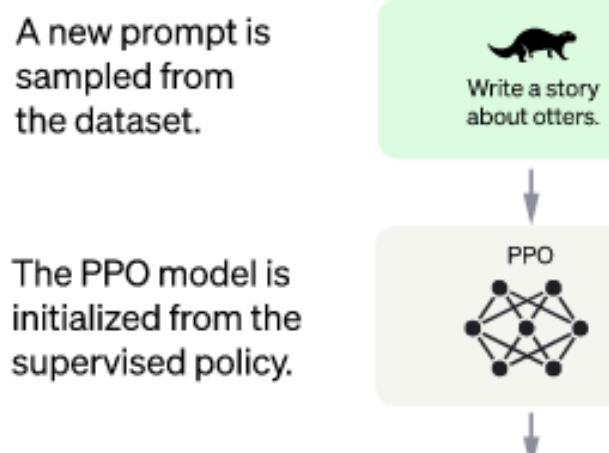
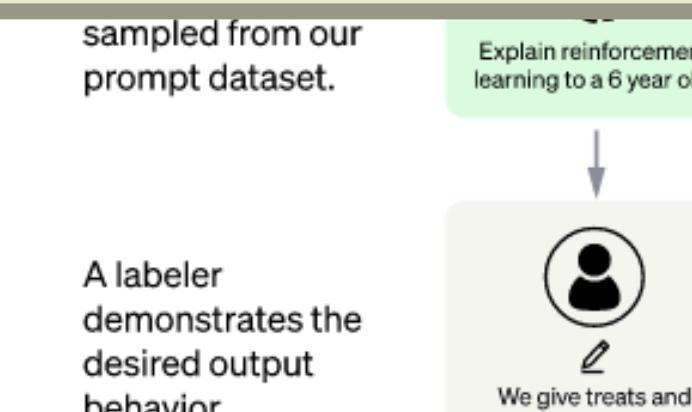
TRY CHATGPT ↗

November 30, 2022
13 minute read

Common Theme:

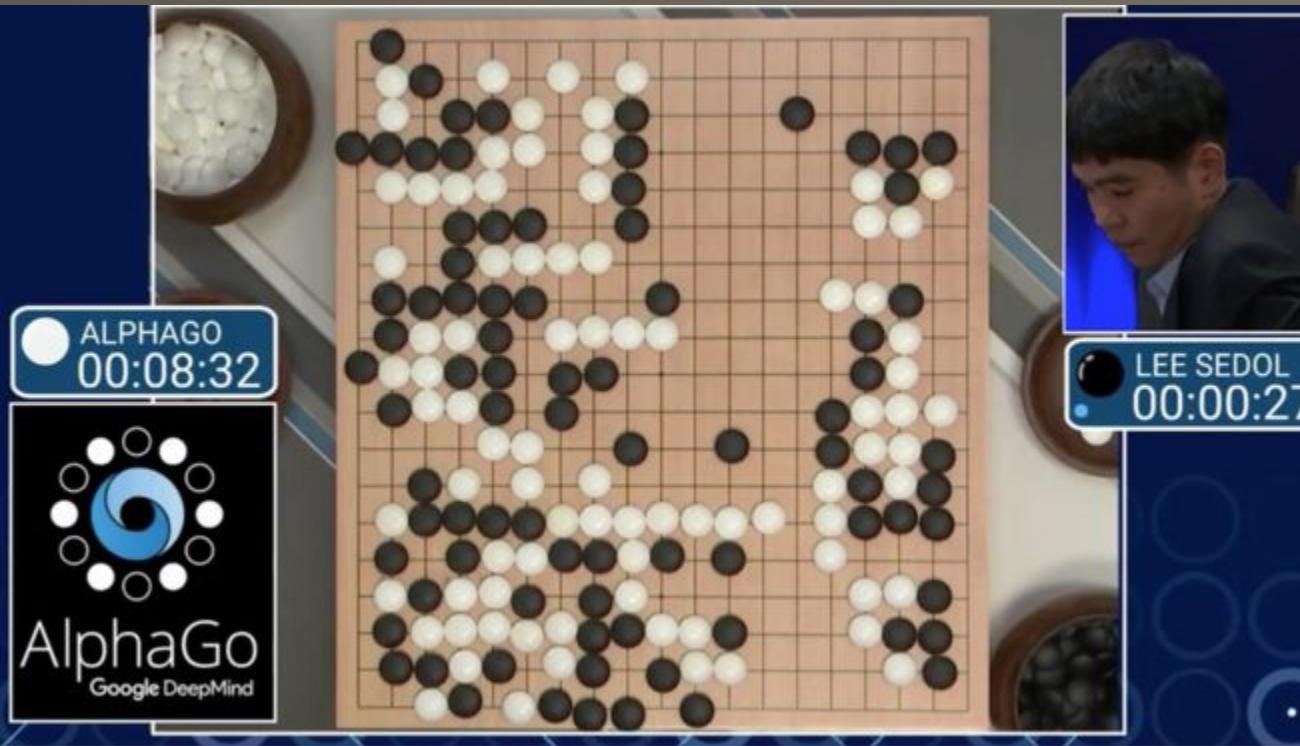
All involve sequential, evolving state in the environment + agent.

Some systems have rewards at each step, some only at the end!



Step 3
Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

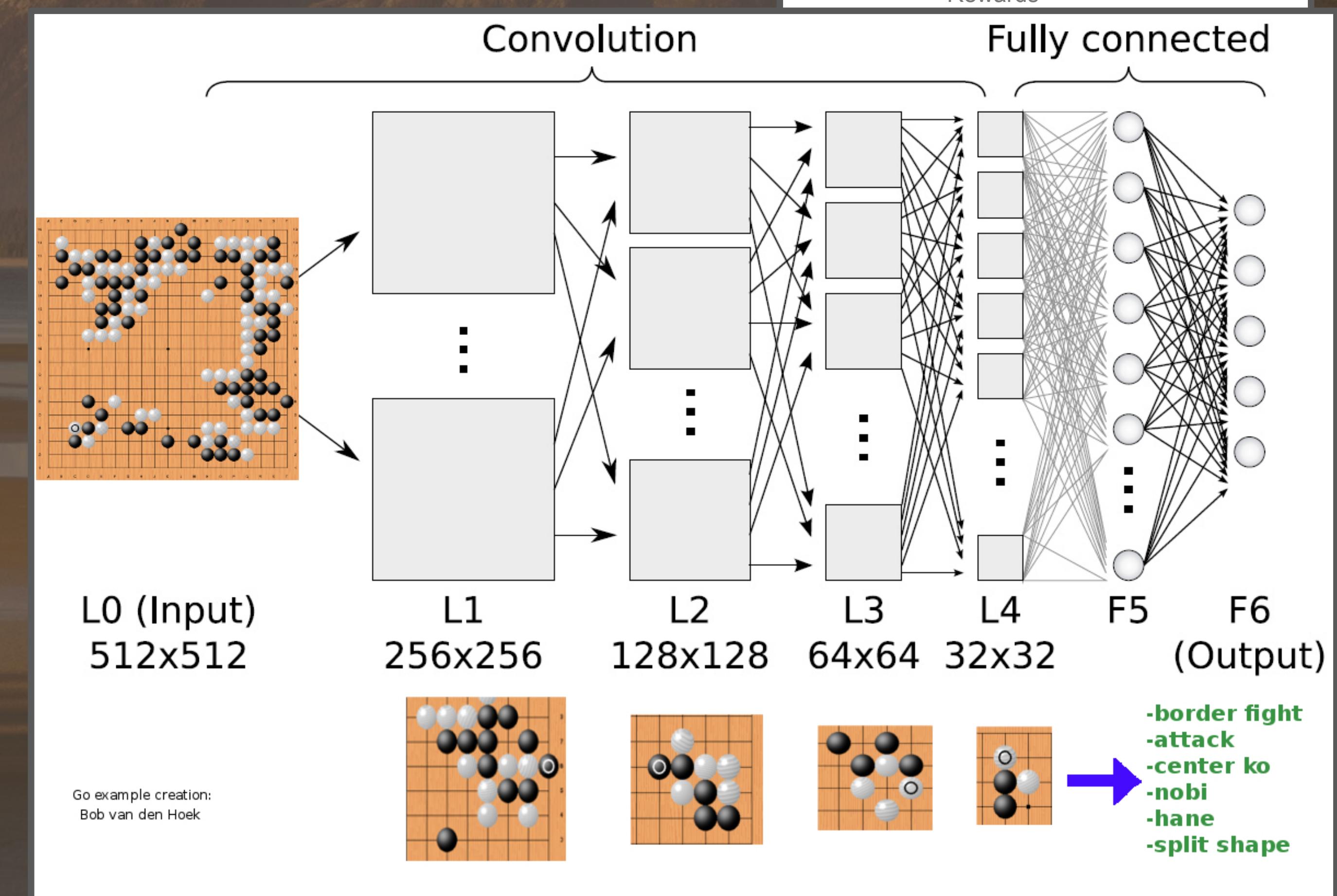
AlphaGo example



Deep Reinforcement Learning

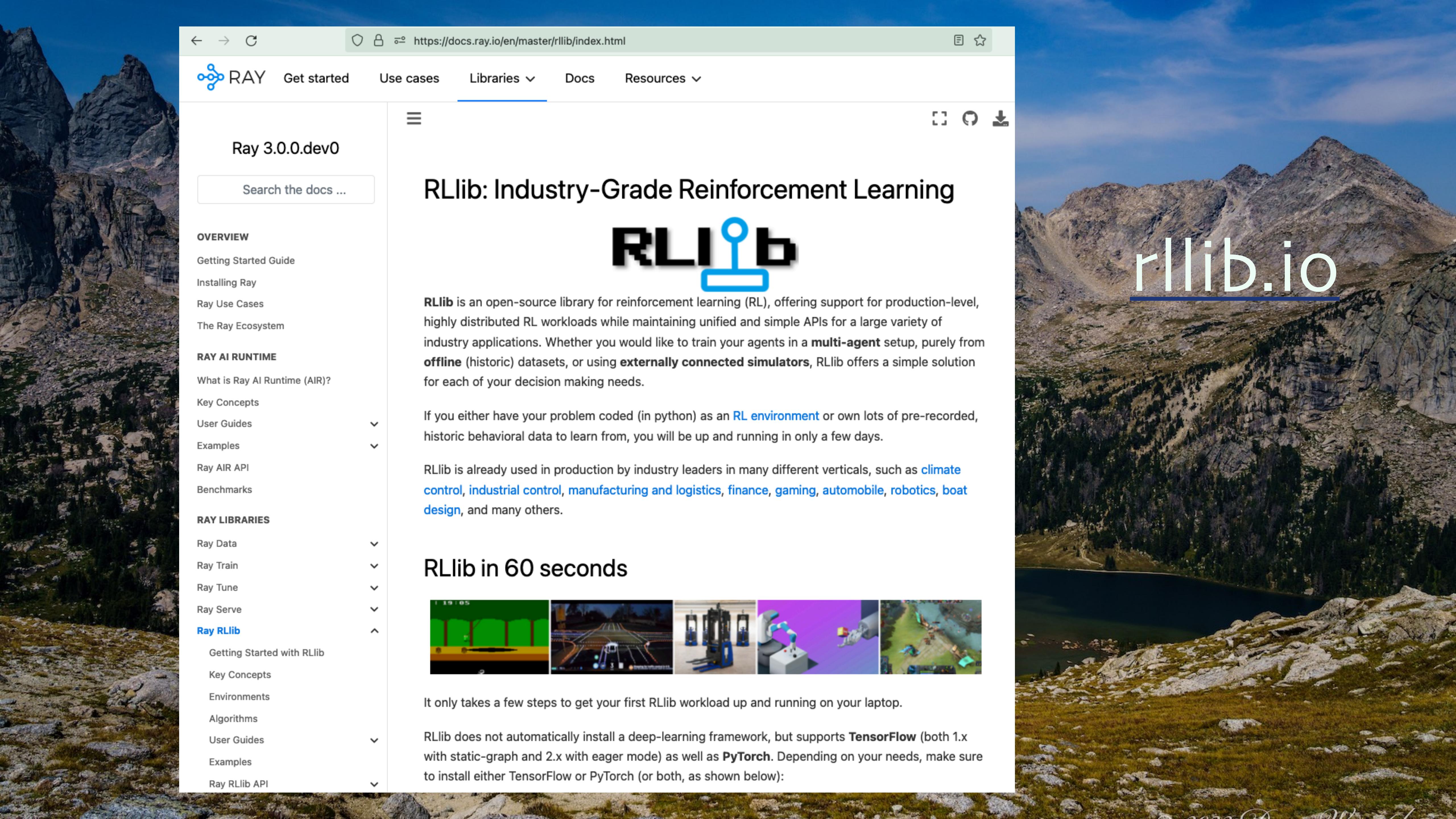
AlphaGo (Silver et al. 2016)

- **Observations:**
 - board state
- **Actions:**
 - where to place the stones
- **Rewards:**
 - 1 if you win
 - 0 otherwise



Ray Rllib





<https://docs.ray.io/en/master/rllib/index.html>

RAY Get started Use cases Libraries Docs Resources

Ray 3.0.0.dev0

Search the docs ...

OVERVIEW

- Getting Started Guide
- Installing Ray
- Ray Use Cases
- The Ray Ecosystem

RAY AI RUNTIME

- What is Ray AI Runtime (AIR)?
- Key Concepts
- User Guides
- Examples
- Ray AIR API
- Benchmarks

RAY LIBRARIES

- Ray Data
- Ray Train
- Ray Tune
- Ray Serve
- Ray RLlib**
- Getting Started with RLlib
- Key Concepts
- Environments
- Algorithms
- User Guides
- Examples
- Ray RLlib API

RLlib: Industry-Grade Reinforcement Learning



RLlib is an open-source library for reinforcement learning (RL), offering support for production-level, highly distributed RL workloads while maintaining unified and simple APIs for a large variety of industry applications. Whether you would like to train your agents in a **multi-agent** setup, purely from **offline** (historic) datasets, or using **externally connected simulators**, RLlib offers a simple solution for each of your decision making needs.

If you either have your problem coded (in python) as an **RL environment** or own lots of pre-recorded, historic behavioral data to learn from, you will be up and running in only a few days.

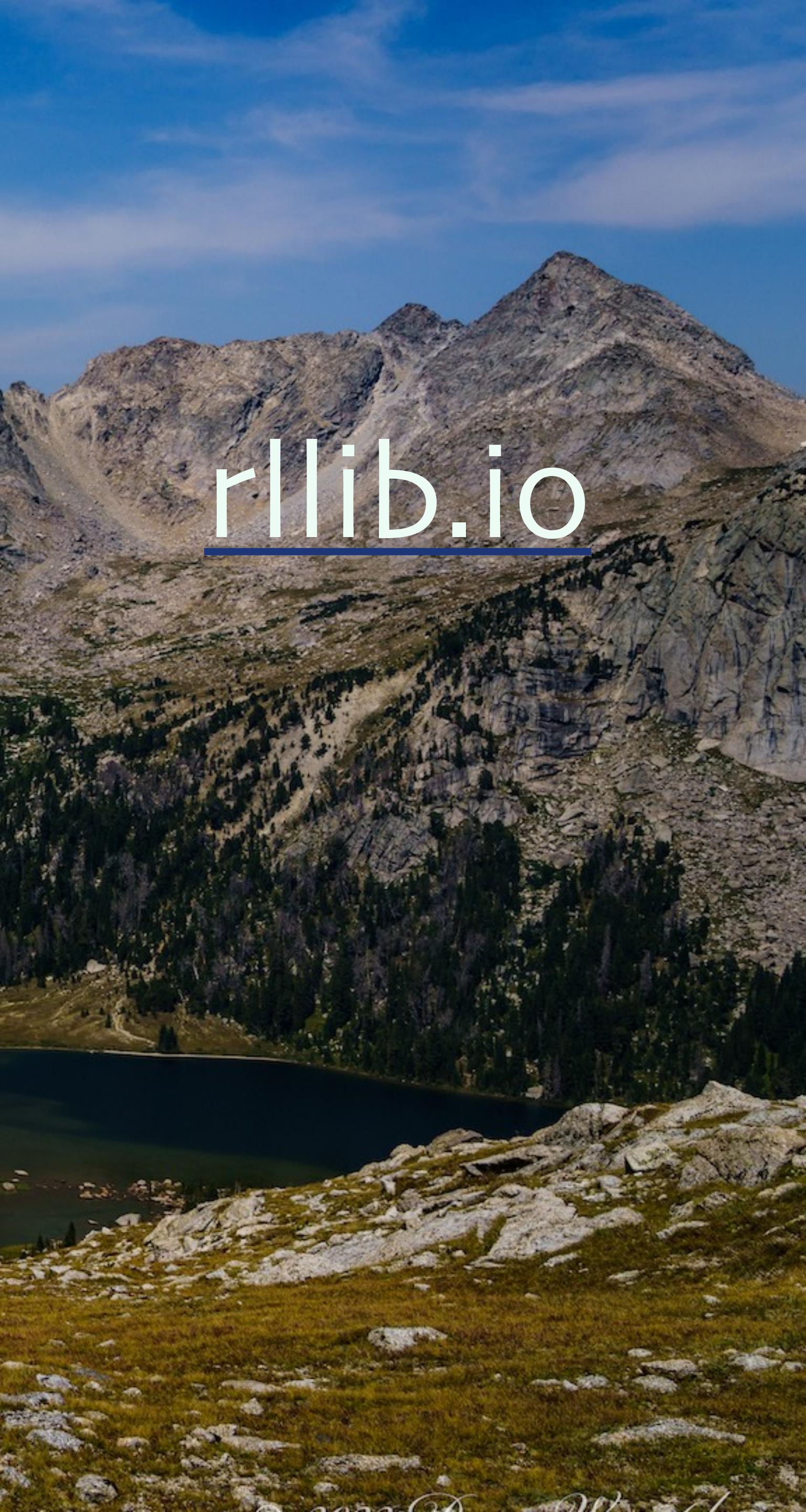
RLlib is already used in production by industry leaders in many different verticals, such as **climate control, industrial control, manufacturing and logistics, finance, gaming, automobile, robotics, boat design**, and many others.

RLlib in 60 seconds



It only takes a few steps to get your first RLlib workload up and running on your laptop.

RLlib does not automatically install a deep-learning framework, but supports **TensorFlow** (both 1.x with static-graph and 2.x with eager mode) as well as **PyTorch**. Depending on your needs, make sure to install either TensorFlow or PyTorch (or both, as shown below):



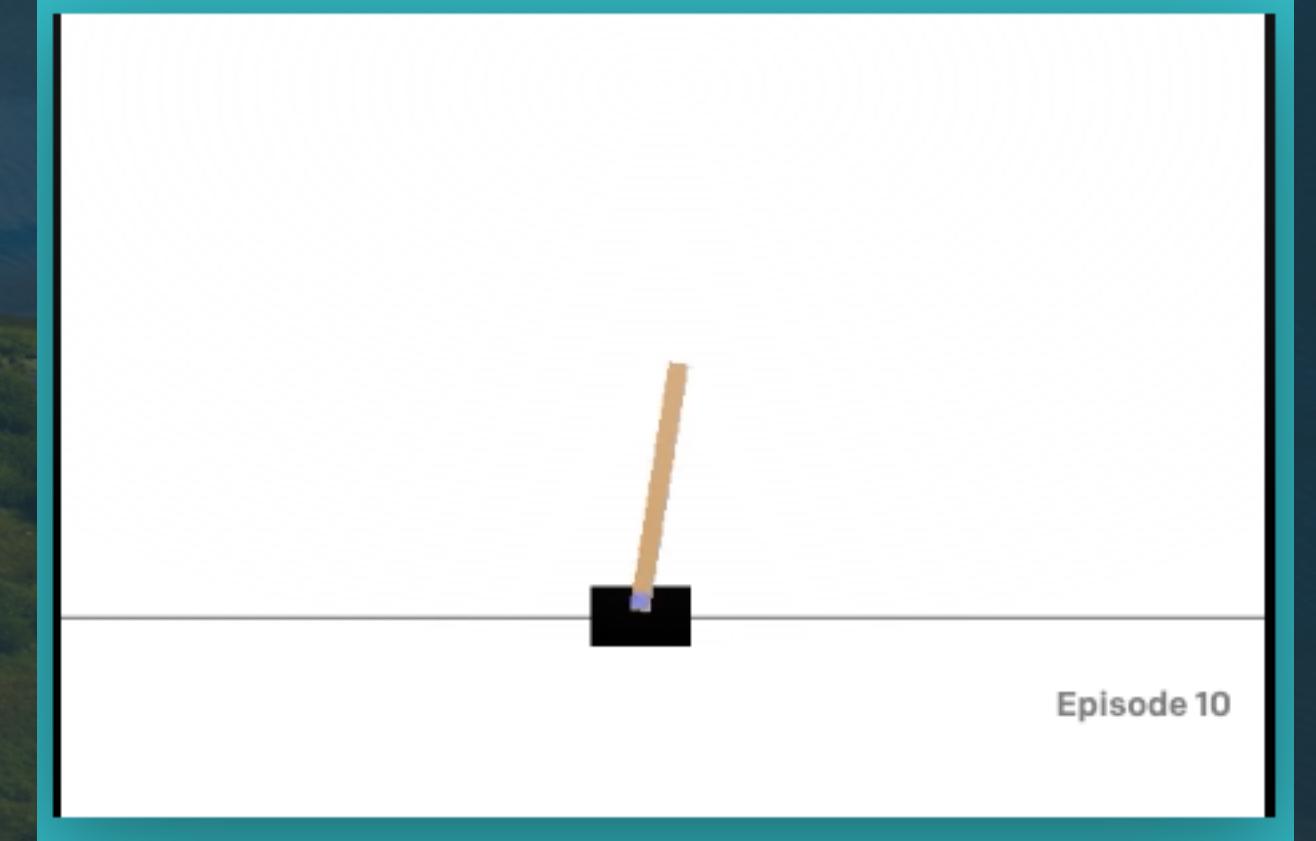


To Try It Out...

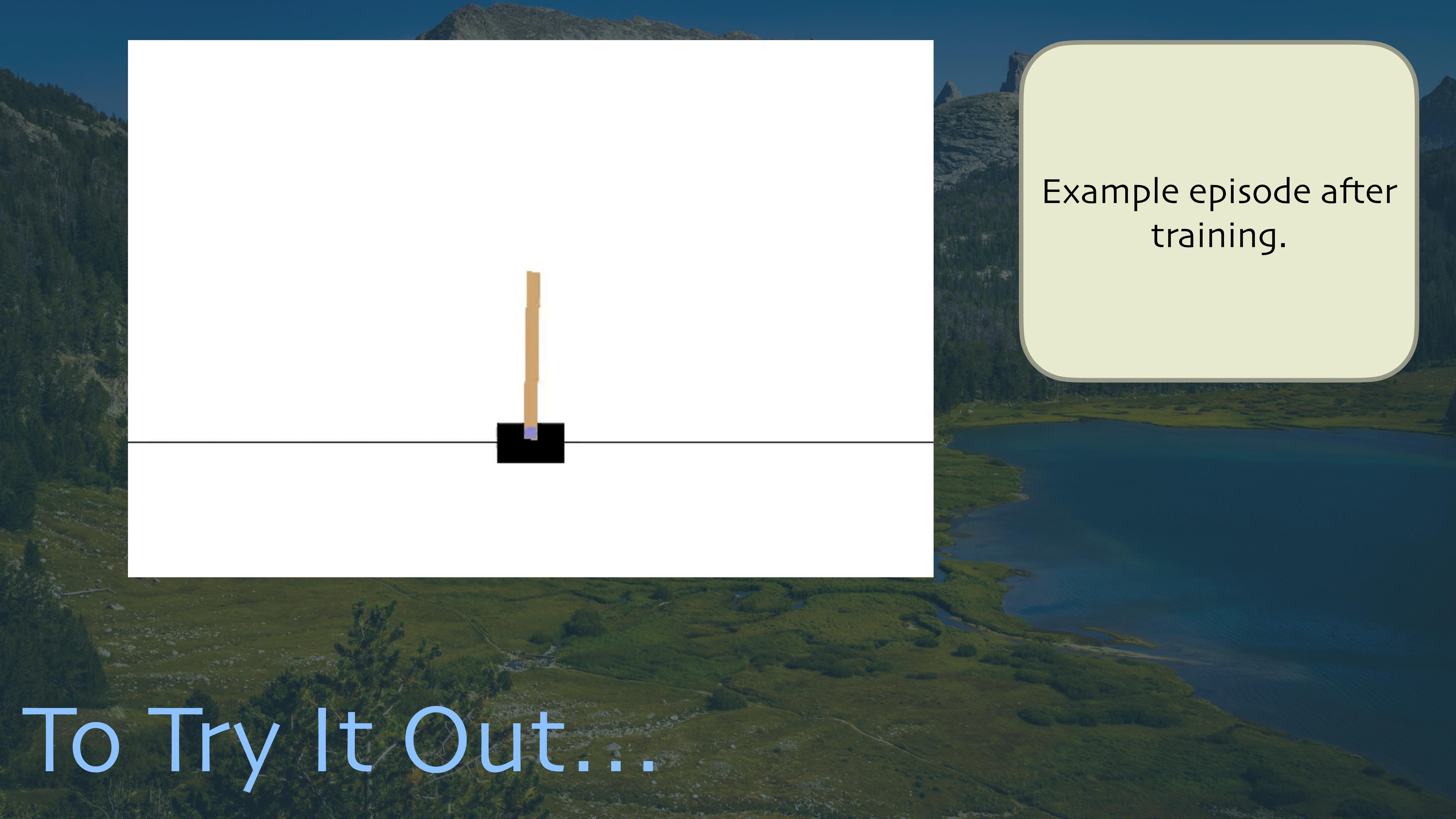
```
$ pip install "ray[rllib]" tensorflow \
tensorflow-probability pygame
$ rllib train --algo PPO --env CartPole-v1
# prints /path/to/checkpoint
```

```
$ rllib evaluate /path/to/checkpoint \
--algo PPO --env CartPole-v1
```

To Try It Out...



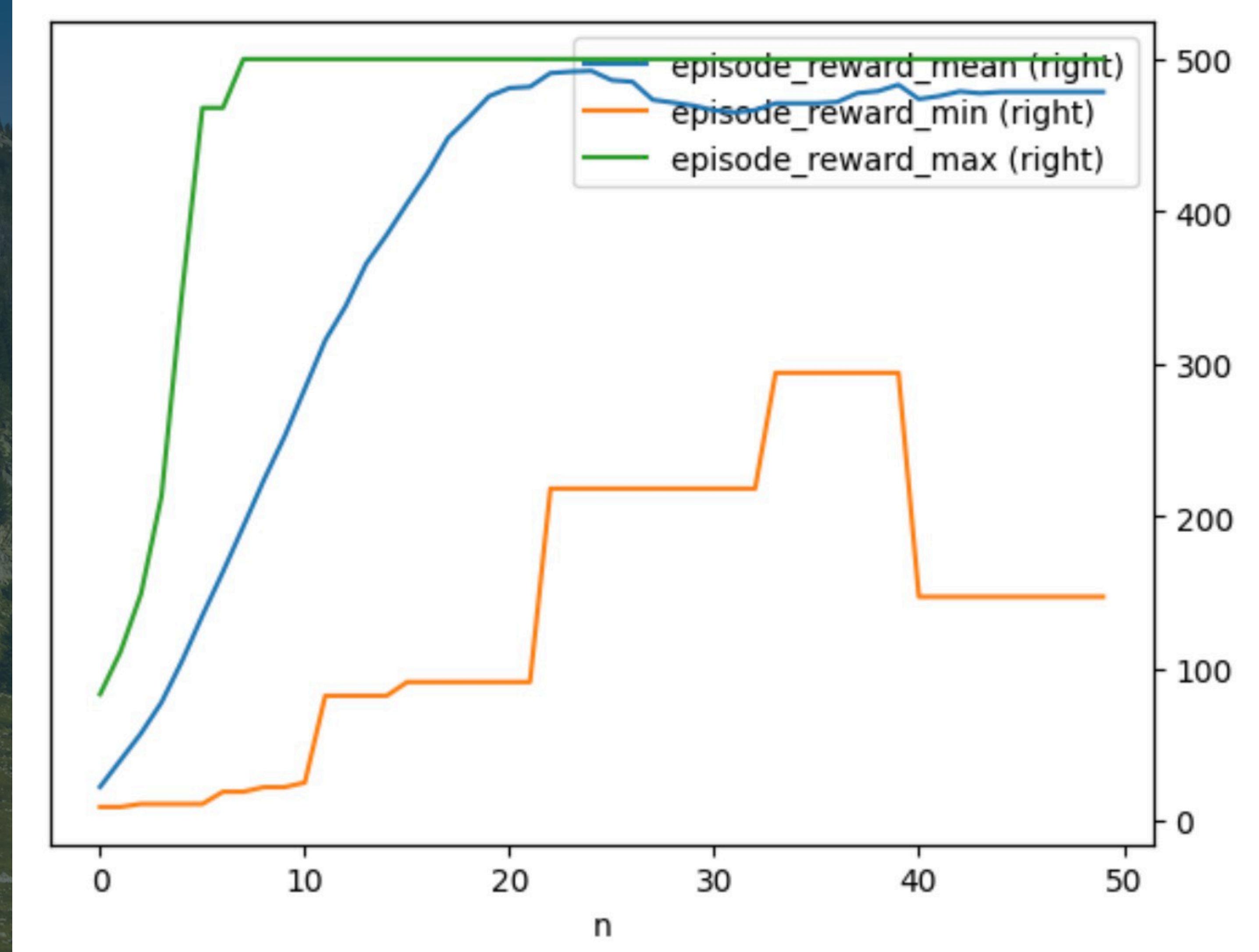
Episode 10



Example episode after
training.



To Try It Out...



Training n=50 episodes with PPO. Max score is 500. Note that the average actually dips above 20 episodes. Probably overfitting?

To Try It Out...

RLlib Takeaways

- Rich set of RL algorithms
 - ... and features for building your own.
- Integrated with OpenAI Gym/Gymnasium
 - ... and you can build your own environments.
- Integrated with PyTorch and TensorFlow.
- Excellent performance... from Ray!



Aside: Why Ray??

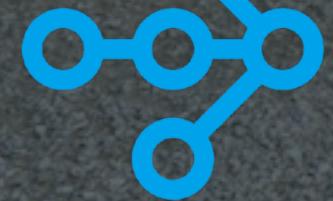
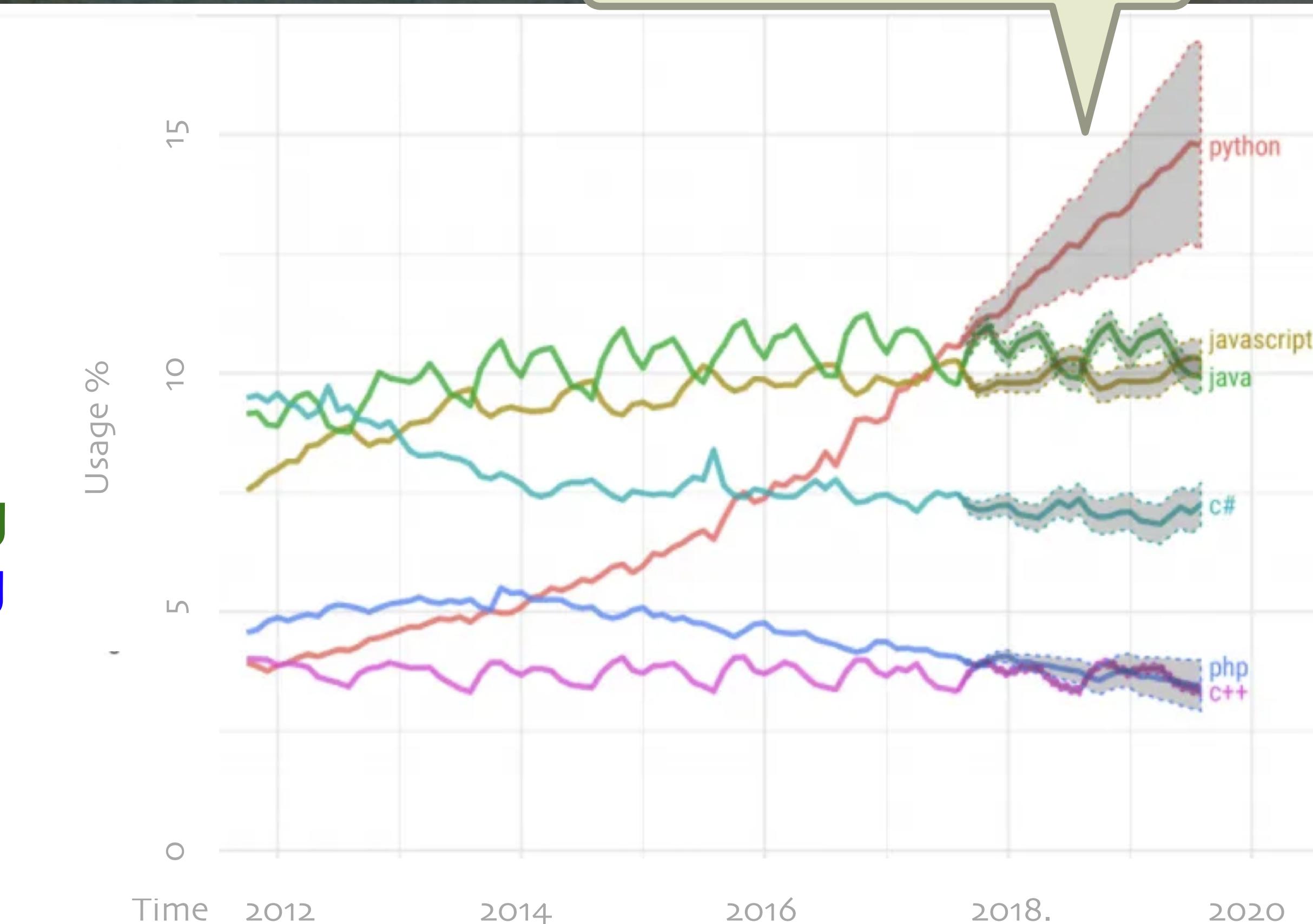
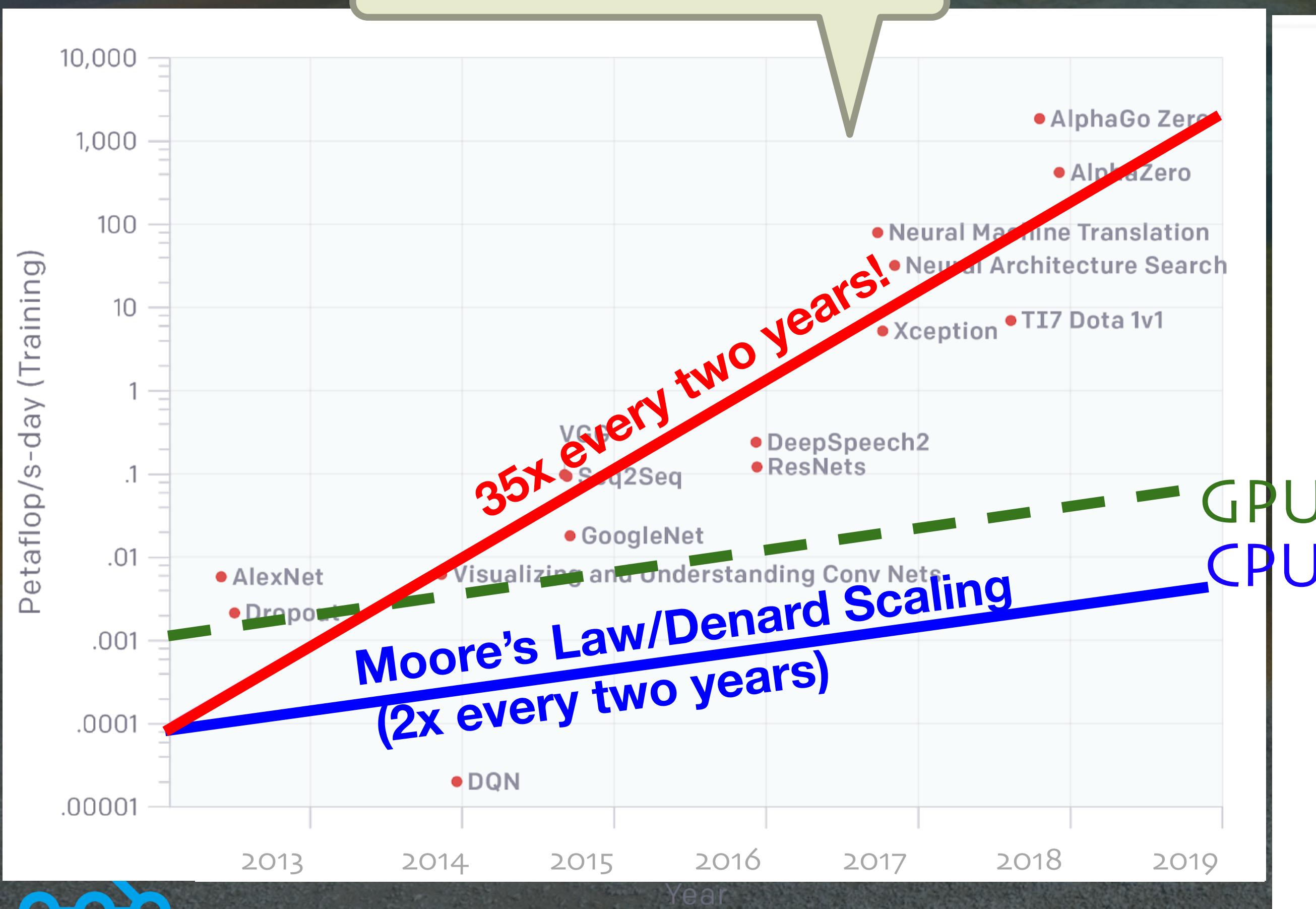


To Major Trends

Model sizes and therefore compute requirements outstripping Moore's Law

Hence, there is a pressing need for a robust, easy to use Python-centric distributed computing system

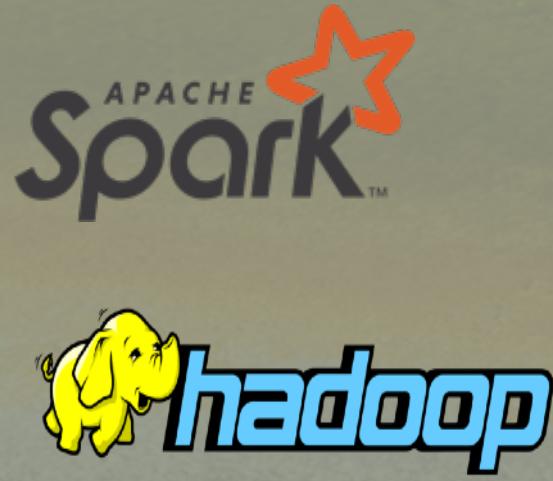
Python growth driven by ML/AI and other data science workloads



The Data & ML Landscape Today

All require distributed implementations to scale

ETL



Streaming



HPO Tuning



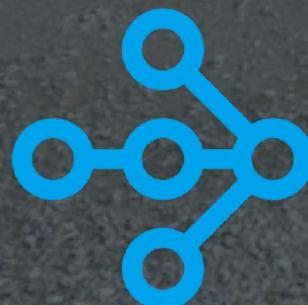
Training



Simulation

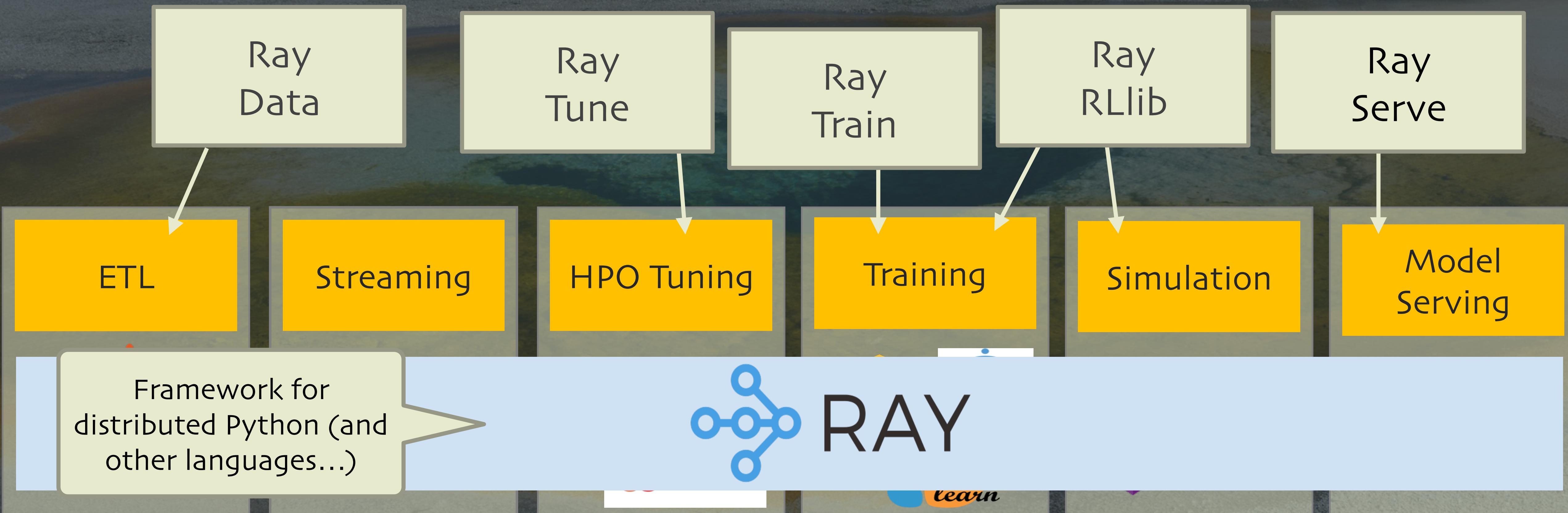


Model
Serving



The Ray Vision: Sharing a Common Framework

Domain-specific
libraries for each
subsystem



Plus a growing list of 3rd-party libraries

Diverse Compute Requirements Motivated Creation of Ray!

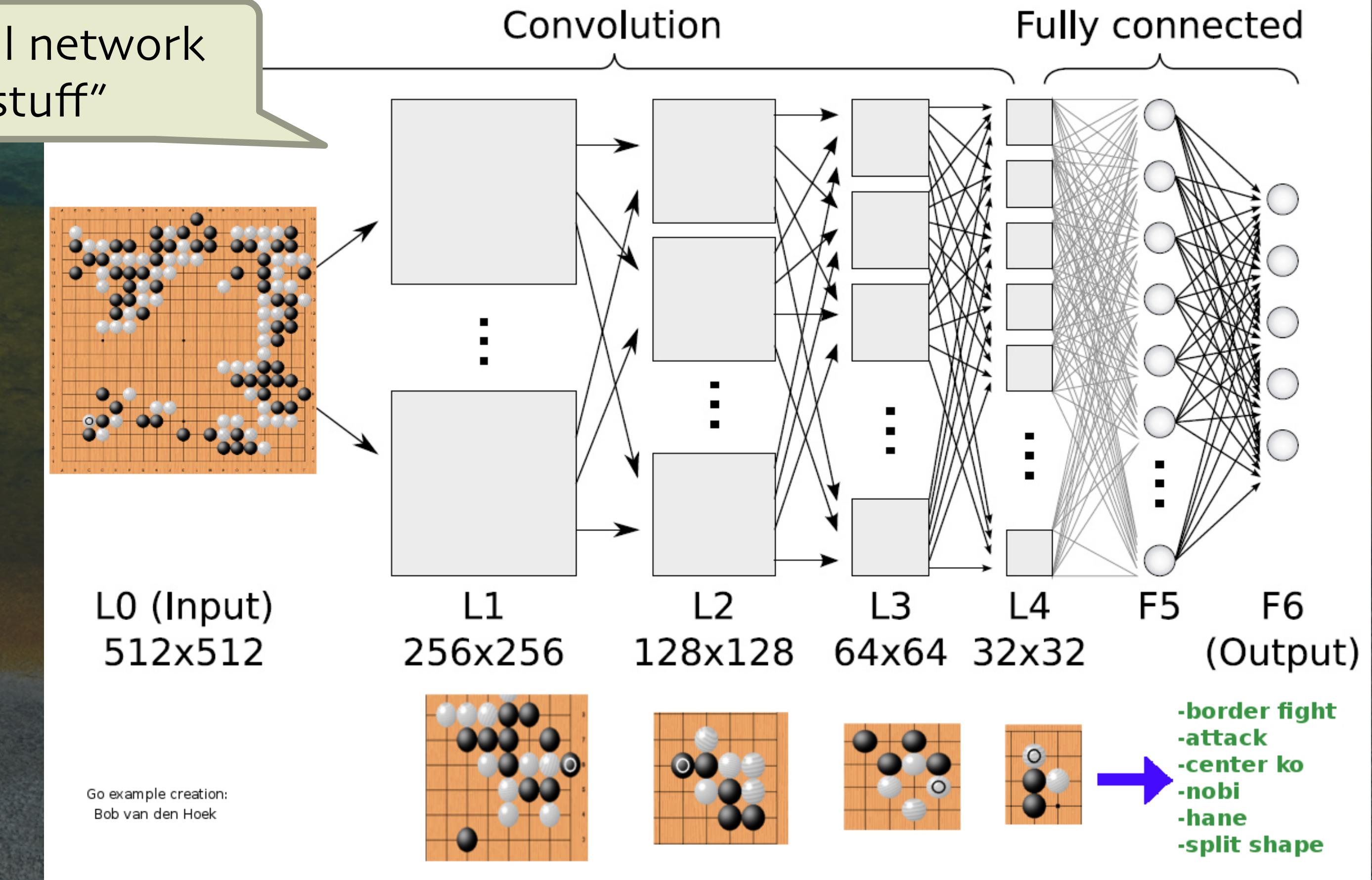
And repeated play,
over and over again,
to train for achieving
the best reward

Simulator (game
engine, robot sim,
factory floor sim...)

Complex agent?



Neural network
“stuff”



More Reinforcement Learning Concepts and Challenges



Exploitation vs. Exploration

What if the agent finds an action with a good short-term reward? Should it keep exploiting it?
Or, should it explore other actions, in case even better options exist?

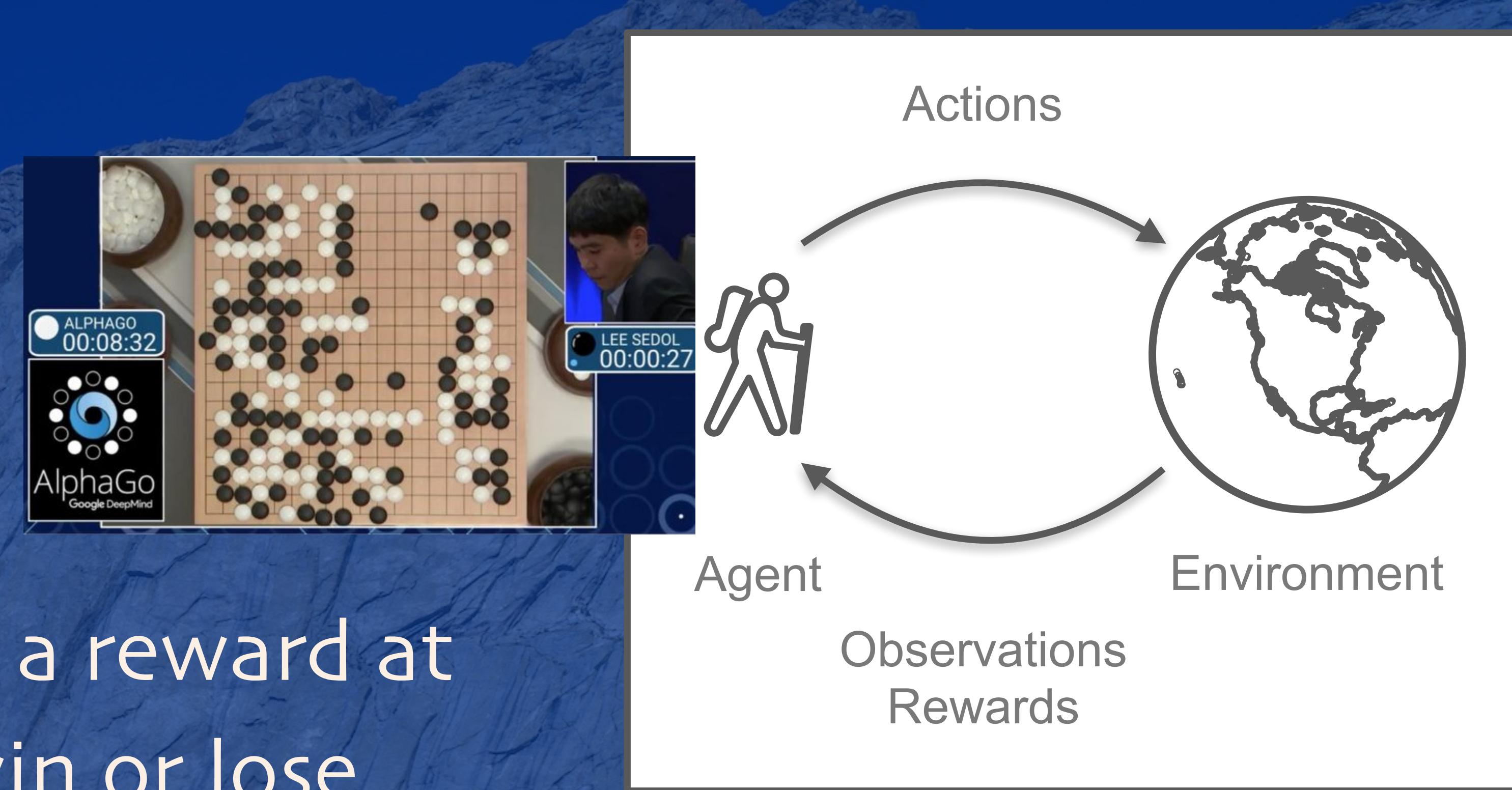


The “Exploitation vs. Exploration Tradeoff”

What Makes a Good Reward?

Games often only provide a reward at the end of the episode - win or lose.

What about intermediate rewards?



Crafting rewards is hard. Intermediate rewards can lead to greedy optimization and local optima rather than the desired global optima - the cumulative reward.

Environments and Offline RL

What if you want to train a system for optimizing a chemical plant?

You can't let a naïve policy drive your plant while it learns!! The plant might be too complex to simulate, too. The higher the stakes, the greater the fidelity required.

However, since the environment “generates” data in normal RL, what about using historical data, instead?



Offline RL works with historical data instead of interacting with the environment.

A wide-angle photograph of a mountainous landscape. In the foreground, a calm, blue lake reflects the surrounding peaks. The middle ground is dominated by a range of rugged, greyish-brown mountains with sharp, rocky ridges. Sparse green vegetation, including small trees and shrubs, is scattered across the mountain slopes. The sky is a clear, pale blue.

Reinforcement Learning for Recommendations and Ad Placements

Preferences Change...

- You bought a toilet brush.
 - Do you want to keep seeing ads for toilet brushes?
- You've watched five action movies in a row.
 - Do you want to watch a sixth action movie or maybe something else for a change?



Preferences Change...

- How have your interests changed because of:
 - the weather
 - the economy
 - local, national, or world affairs
 - ???



RL for recommendations/ads
helps with evolving
preferences.

Considerations

- RL is less able to scale to large state spaces (e.g., all available movies catalog items).
- Traditional supervised learning methods are more scalable.



Real recommendation and ad systems must combine approaches; use RL once a subset of the state space is identified using a “classic” supervised learning approach.

Considerations

- A simulator is used to model real user behavior. (Training with real users doesn't scale well, etc.)



Or use offline RL with historical data about user behavior!

Considerations

- What is the reward? Some combination of user happiness measures?
- Could be very specific to the sub-genre of entertainment or product category.



Reward calculation balances mixed preferences & tradeoffs as they evolve in response to use actions.

To Learn More...

- [rllib.io](#) & [ray.io](#)
- Anyscale RL & RLLib course:
<https://applied-rl-course.netlify.app/en>
- More resources in the extra slides!

Dean Wampler
January 28, 2023
dean@deanwampler.com
[@deanwampler](https://twitter.com/deanwampler)
@discuss.systems@deanwampler
deanwampler.com/talks

Extra Material



To Learn More...

- Courses

- Hugging Face RL course <https://huggingface.co/deep-rl-course/>

- Delta Academy <https://delta-academy.xyz/>

- Fast Deep RL <https://courses.dibya.online/p/fastdeeprl>

- Coursera RL Specialization from U of A <https://www.coursera.org/specializations/reinforcement-learning>

- Udacity RL course <https://www.udacity.com/course/reinforcement-learning--ud600>

- Video lectures

- David Silver's lectures <https://www.davidsilver.uk/teaching/>

- Sergey Levine's lectures <http://rail.eecs.berkeley.edu/deeprlcourse/>

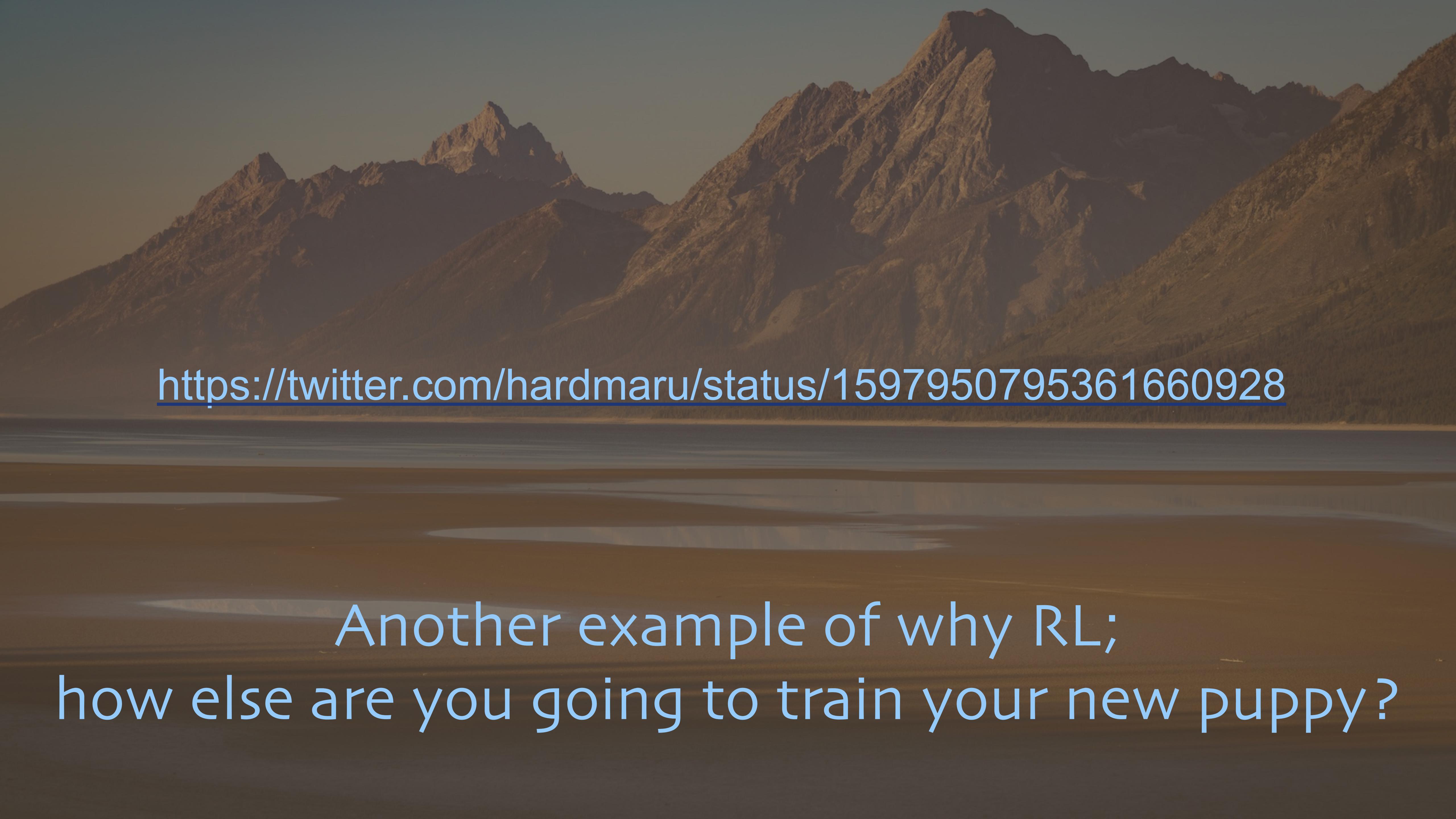
- Books

- Sutton & Barto <http://incompleteideas.net/book/the-book-2nd.html> (considered the definitive RL book)

- Deep RL Hands-On <https://www.packtpub.com/product/deep-reinforcement-learning-hands-on-second-edition/9781838826994>

- Other

- Spinning Up <https://spinningup.openai.com/en/latest/> (a well-known resource for RL)

A wide-angle photograph of a mountainous landscape during sunset or sunrise. The sky is a deep orange and yellow. In the foreground, there's a body of water with some ripples. The middle ground features several mountain peaks, their slopes covered with dense forests. The lighting creates strong shadows and highlights on the mountain faces, emphasizing their rugged texture.

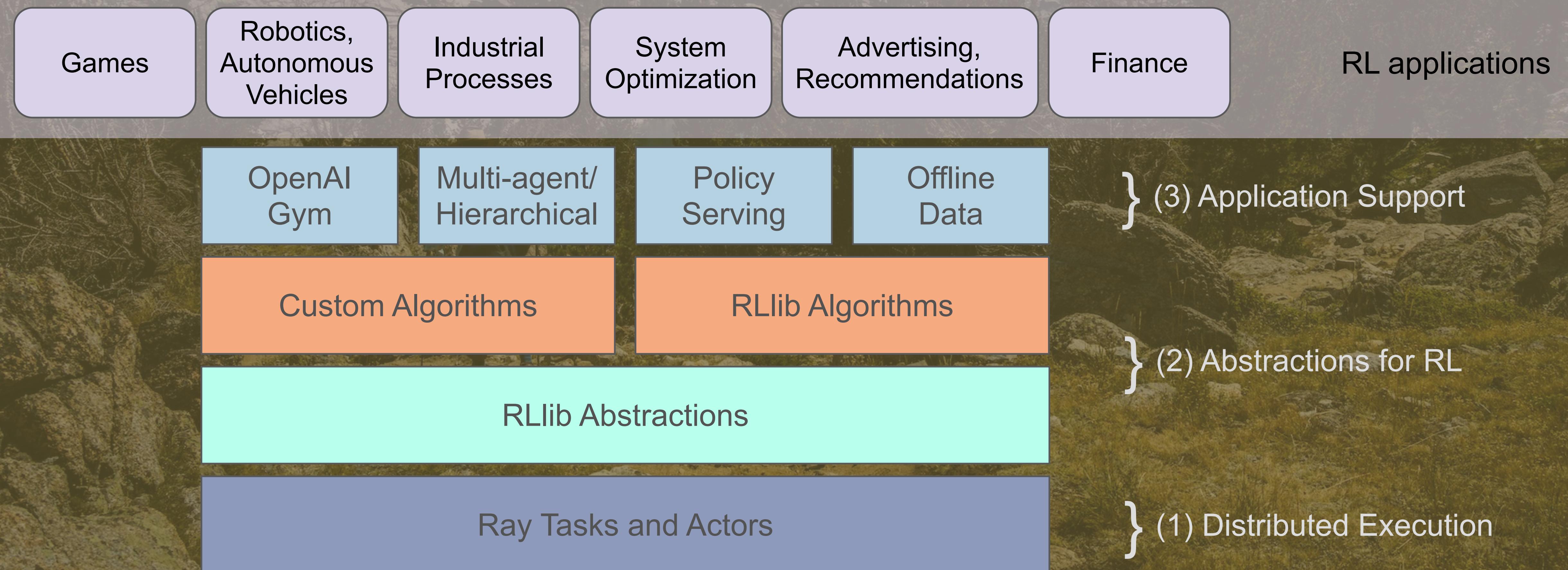
<https://twitter.com/hardmaru/status/1597950795361660928>

Another example of why RL;
how else are you going to train your new puppy?

A photograph of a hiker from behind, standing on a rocky, grassy slope. The hiker is wearing a blue and white plaid shirt, brown pants, a tan hat, and a blue backpack. They are holding two trekking poles. In the background, there are large, rugged mountains with patches of snow and green forests. A small, blue lake is visible in a valley between the mountains. The sky is clear and blue.

More about RLLib

Architecture of RLlib



Some Algorithms in RLLib

- High-throughput architectures
 - [Distributed Prioritized Experience Replay \(Ape-X\)](#)
 - [Importance Weighted Actor-Learner Architecture \(IMPALA\)](#)
 - [Asynchronous Proximal Policy Optimization \(APPO\)](#)
- Gradient-based
 - [Soft Actor-Critic \(SAC\)](#)
 - [Advantage Actor-Critic \(A₂C, A₃C\)](#)
 - [Deep Deterministic Policy Gradients \(DDPG, TD3\)](#)
 - [Deep Q Networks \(DQN, Rainbow, Parametric DQN\)](#)
 - [Policy Gradients](#)
 - [Proximal Policy Optimization \(PPO\)](#)
- gradient-free
 - [Augmented Random Search \(ARS\)](#)
 - [Evolution Strategies](#)
- Multi-agent specific
 - [QMIX Monotonic Value Factorisation \(QMIX, VDN, IQN\)](#)
- Offline
 - [Advantage Re-Weighted Imitation Learning \(MARWIL\)](#)

Available in AWS and Azure

Amazon SageMaker RL

Reinforcement learning for every developer and data scientist



-  Fully managed reinforcement learning algorithms
-  TensorFlow, MXNet, Intel Coach, and Ray RL
-  2D and 3D simulation environments via OpenGym
-  Simulate environments

 Microsoft | Docs [Documentation](#) Learn Q&A Code Samples

Azure Product documentation ▾ Architecture ▾ Learn Azure ▾ Develop ▾ Resources ▾

Azure / Machine Learning Bookmark

Azure Machine Learning Documentation

Overview

- What is Azure Machine Learning?
- Azure Machine Learning vs Studio (classic)
- Architecture & terms

Tutorials

- > Studio
- > Python SDK
- > R SDK
- > Machine Learning CLI
- > Visual Studio Code
- > Samples
- > Concepts

Reinforcement learning (preview) with Azure Machine Learning

05/05/2020 • 11 minutes to read •   

APPLIES TO:  Basic edition  Enterprise edition [\(Upgrade to Enterprise edition\)](#)

Note

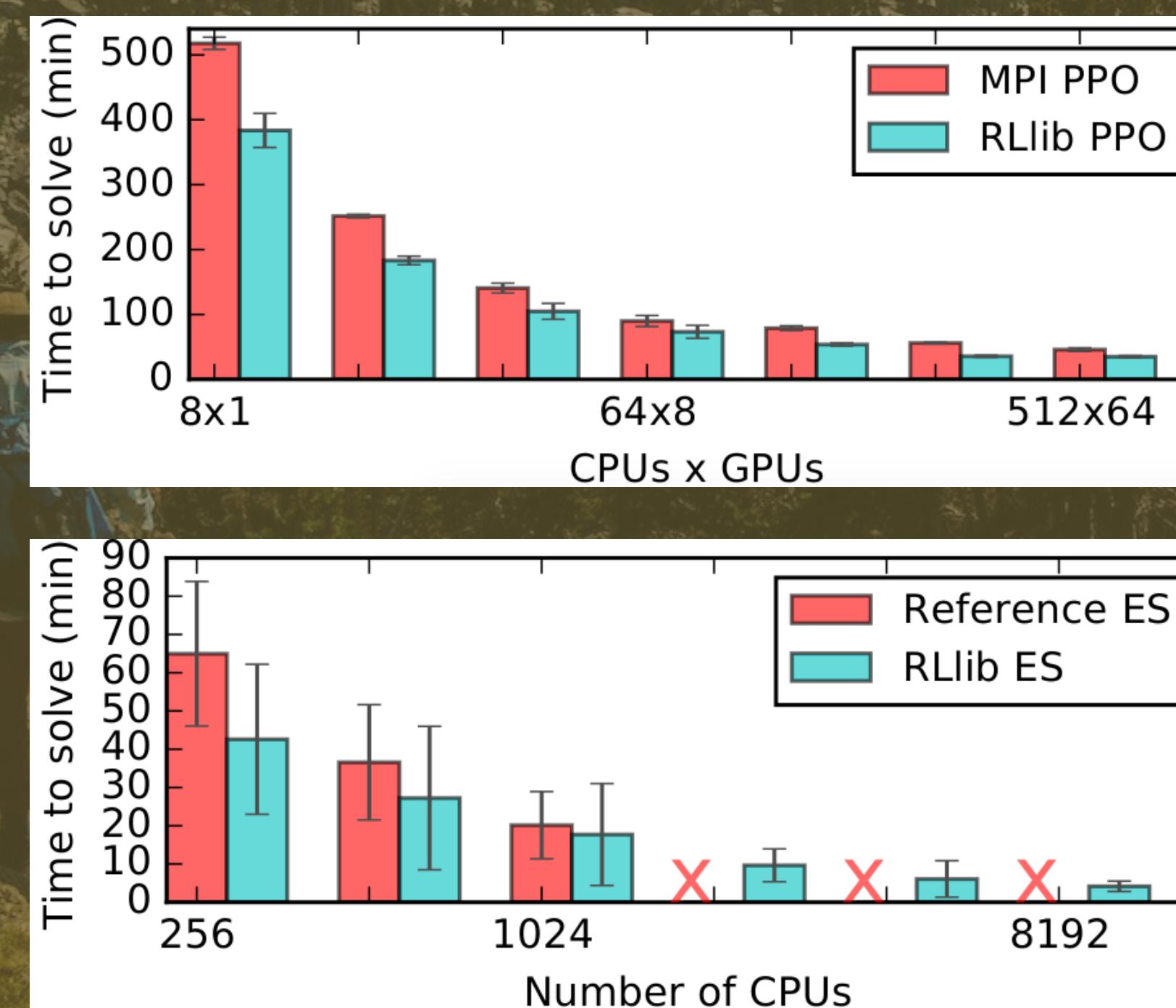
Azure Machine Learning Reinforcement Learning is currently a preview feature. Only Ray and RLLib frameworks are supported at this time.

In this article, you learn how to train a reinforcement learning (RL) agent to play the video game Pong. You will use the open-source Python library [Ray.RLLib](#) with Azure Machine Learning to manage the complexity of distributed RL jobs.

In this article you will learn how to:

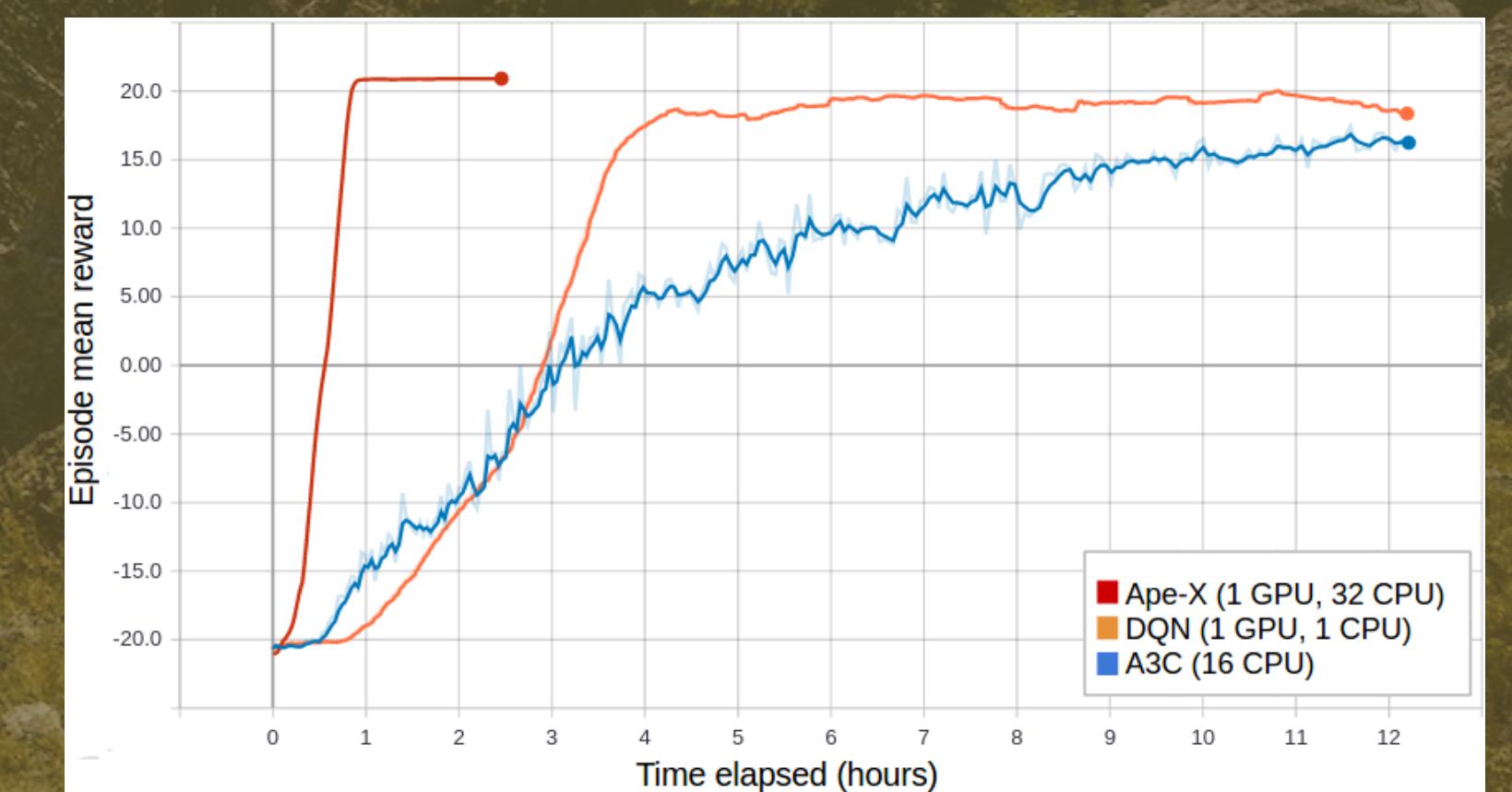
Excellent Performance vs. “Hand-tuned” Implementations

Distributed PPO



Evolution Strategies

Ape-X Distributed
DQN, DDPG





Quick Intro to the Ray API



API - Designed to Be Intuitive and Concise

Functions -> Tasks

```
def make_array(...):  
    a = ... # Construct a NumPy array  
    return a
```

```
def add_arrays(a, b):  
    return np.add(a, b)
```

The Python you
already know...



API - Designed to Be Intuitive and Concise

Functions -> Tasks

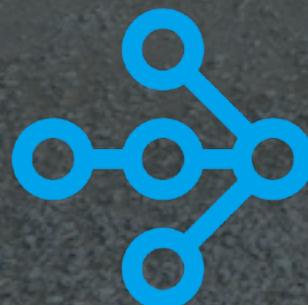
For completeness, add these first:

```
@ray.remote  
def make_array(...):  
    a = ... # Construct a NumPy array  
    return a
```

```
@ray.remote  
def add_arrays(a, b):  
    return np.add(a, b)
```

```
import ray  
import numpy as np  
ray.init()
```

Now these functions
are remote "tasks"



API - Designed to Be Intuitive and Concise

Functions -> Tasks

```
@ray.remote  
def make_array(...):  
    a = ... # Construct a NumPy array  
    return a
```

```
@ray.remote  
def add_arrays(a, b):  
    return np.add(a, b)
```

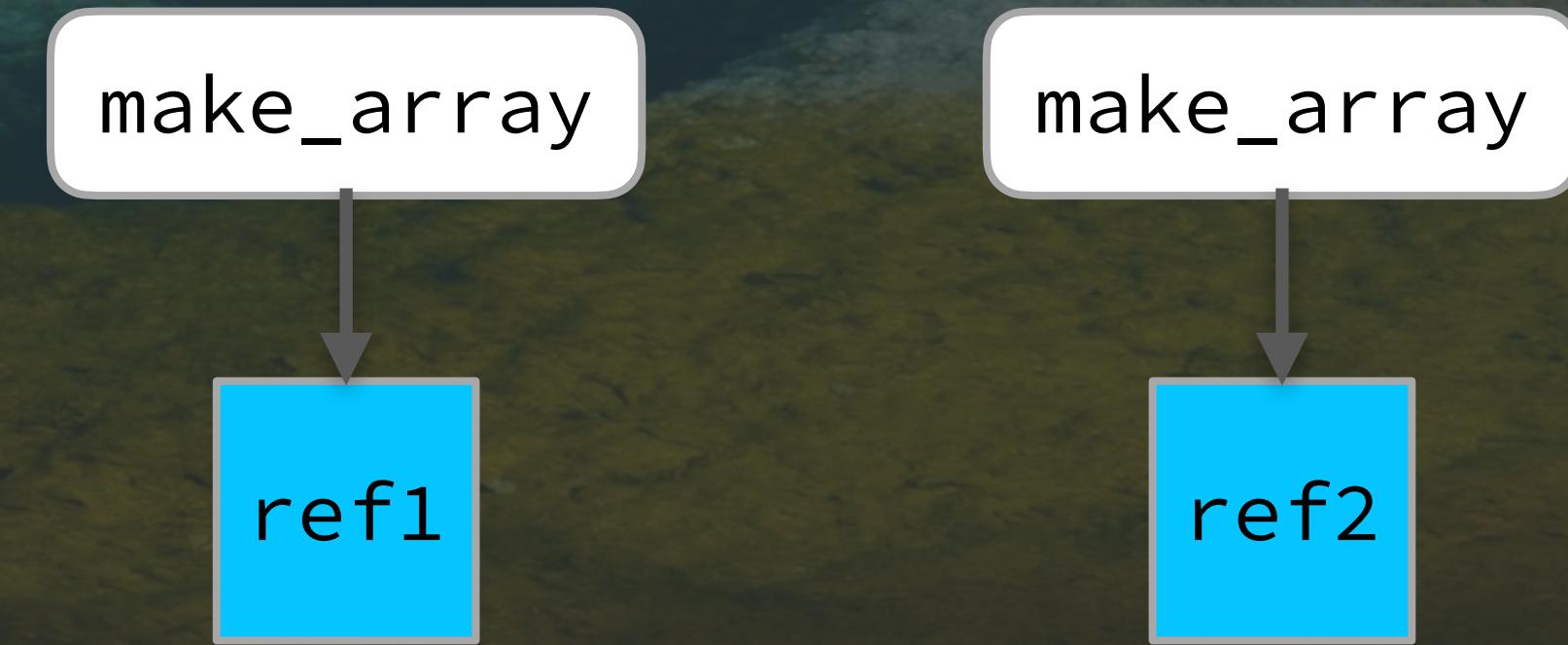
```
ref1 = make_array.remote(...)
```



API - Designed to Be Intuitive and Concise

Functions -> Tasks

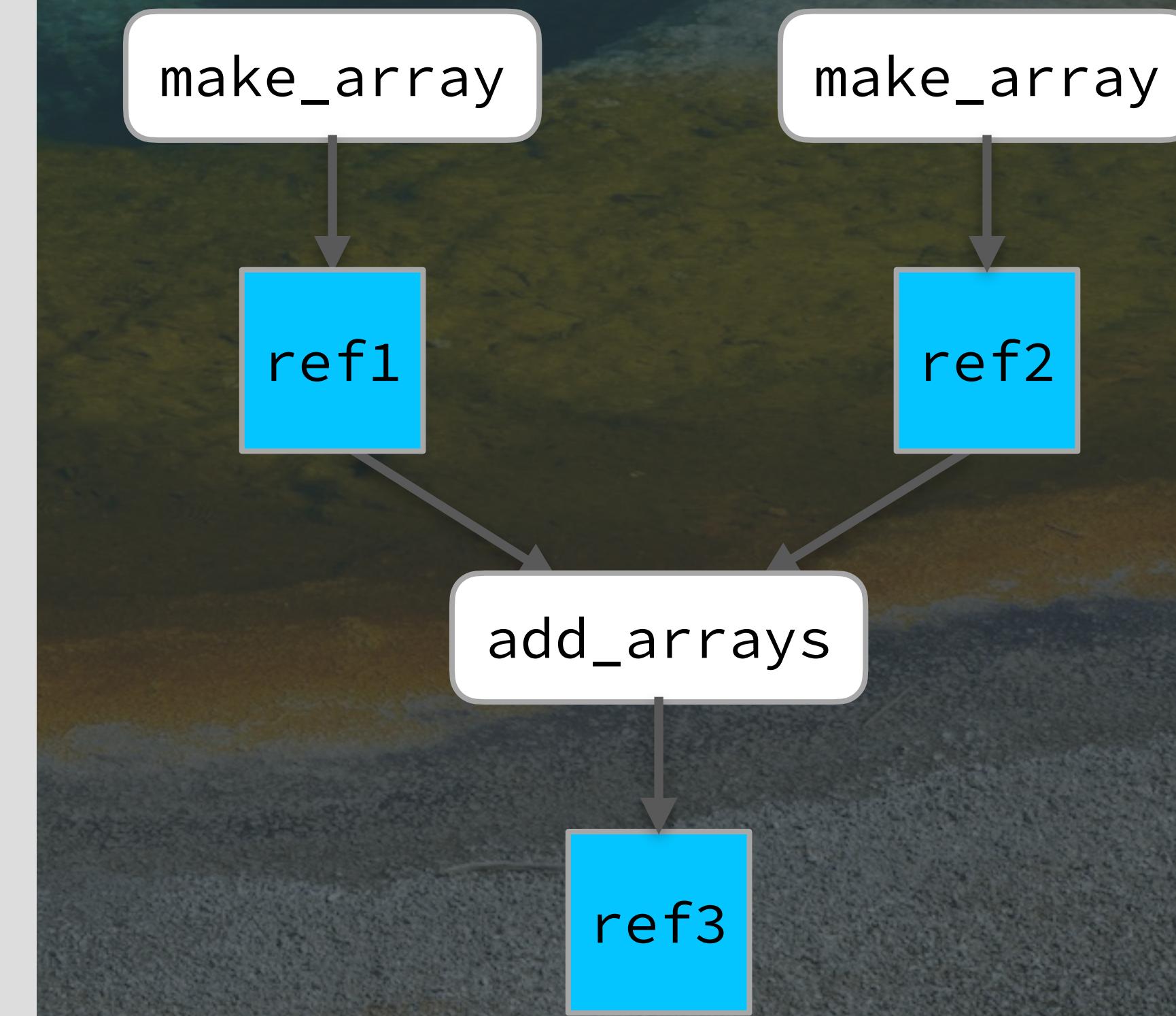
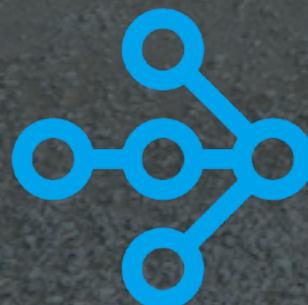
```
@ray.remote  
def make_array(...):  
    a = ... # Construct a NumPy array  
    return a  
  
@ray.remote  
def add_arrays(a, b):  
    return np.add(a, b)  
  
ref1 = make_array.remote(...)  
ref2 = make_array.remote(...)
```



API - Designed to Be Intuitive and Concise

Functions -> Tasks

```
@ray.remote  
def make_array(...):  
    a = ... # Construct a NumPy array  
    return a  
  
@ray.remote  
def add_arrays(a, b):  
    return np.add(a, b)  
  
ref1 = make_array.remote(...)  
ref2 = make_array.remote(...)  
ref3 = add_arrays.remote(ref1, ref2)
```



API - Designed to Be Intuitive and Concise

Functions → Tasks

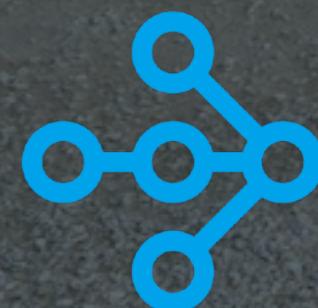
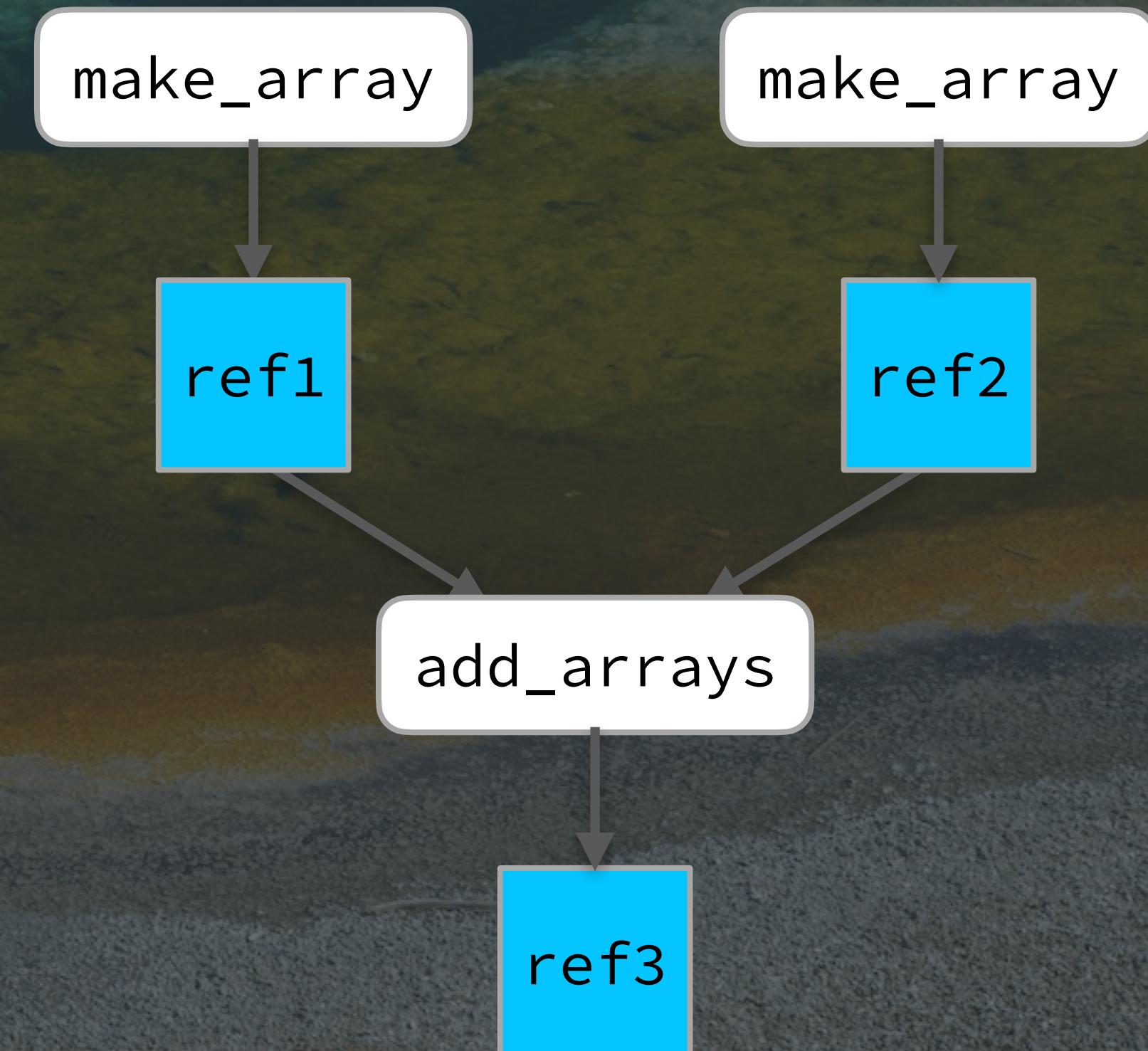
```
@ray.remote  
def make_array(...):  
    a = ... # Construct a NumPy array  
    return a
```

```
@ray.remote  
def add_arrays(a, b):  
    return np.add(a, b)
```

```
ref1 = make_array.remote(...)  
ref2 = make_array.remote(...)  
ref3 = add_arrays.remote(ref1, ref2)  
ray.get(ref3)
```

Ray handles sequencing
of async dependencies

Ray handles extracting the
arrays from the object refs



What about
distributed
state?

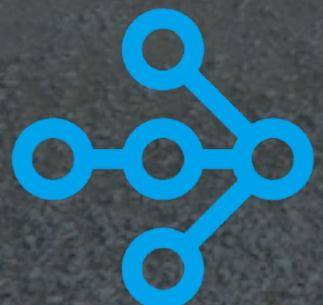
API - Designed to Be Intuitive and Concise

Functions -> Tasks

```
@ray.remote
def make_array(...):
    a = ... # Construct a NumPy array
    return a

@ray.remote
def add_arrays(a, b):
    return np.add(a, b)

ref1 = make_array.remote(...)
ref2 = make_array.remote(...)
ref3 = add_arrays.remote(ref1, ref2)
ray.get(ref3)
```



API - Designed to Be Intuitive and Concise

Functions -> Tasks

```
@ray.remote  
def make_array(...):  
    a = ... # Construct a NumPy array  
    return a  
  
@ray.remote  
def add_arrays(a, b):  
    return np.add(a, b)  
  
ref1 = make_array.remote(...)  
ref2 = make_array.remote(...)  
ref3 = add_arrays.remote(ref1, ref2)  
ray.get(ref3)
```

Classes -> Actors

```
class Counter(object):  
    def __init__(self):  
        self.value = 0  
    def increment(self):  
        self.value += 1  
    return self.value
```

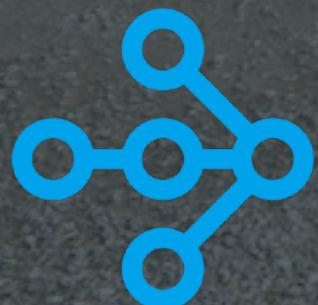
The Python
classes you
love...



API - Designed to Be Intuitive and Concise

Functions -> Tasks

```
@ray.remote  
def make_array(...):  
    a = ... # Construct a NumPy array  
    return a  
  
@ray.remote  
def add_arrays(a, b):  
    return np.add(a, b)  
  
ref1 = make_array.remote(...)  
ref2 = make_array.remote(...)  
ref3 = add_arrays.remote(ref1, ref2)  
ray.get(ref3)
```



... now a remote
“actor”

Classes -> Actors

```
@ray.remote  
class Counter(object):  
    def __init__(self):  
        self.value = 0  
    def increment(self):  
        self.value += 1  
        return self.value  
    def get_count(self):  
        return self.value
```

You need a
“getter” method
to read the state.

API - Designed to Be Intuitive and Concise

Functions -> Tasks

```
@ray.remote  
def make_array(...):  
    a = ... # Construct a NumPy array  
    return a  
  
@ray.remote  
def add_arrays(a, b):  
    return np.add(a, b)  
  
ref1 = make_array.remote(...)  
ref2 = make_array.remote(...)  
ref3 = add_arrays.remote(ref1, ref2)  
ray.get(ref3)
```



Classes -> Actors

```
@ray.remote  
class Counter(object):  
    def __init__(self):  
        self.value = 0  
    def increment(self):  
        self.value += 1  
        return self.value  
    def get_count(self):  
        return self.value  
  
c = Counter.remote()  
ref4 = c.increment.remote()  
ref5 = c.increment.remote()  
ray.get([ref4, ref5]) # [1, 2]
```

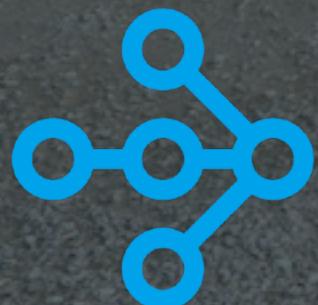
API - Designed to Be Intuitive and Concise

Functions -> Tasks

```
@ray.remote
def make_array(...):
    a = ... # Construct a NumPy array
    return a

@ray.remote
def add_arrays(a, b):
    return np.add(a, b)

ref1 = make_array.remote(...)
ref2 = make_array.remote(...)
ref3 = add_arrays.remote(ref1, ref2)
ray.get(ref3)
```



Classes -> Actors

```
@ray.remote(num_gpus=1)
class Counter(object):
    def __init__(self):
        self.value = 0
    def increment(self):
        self.value += 1
        return self.value
    def get_count(self):
        return self.value

c = Counter.remote()
ref4 = c.increment.remote()
ref5 = c.increment.remote()
ray.get([ref4, ref5]) # [1, 2]
```

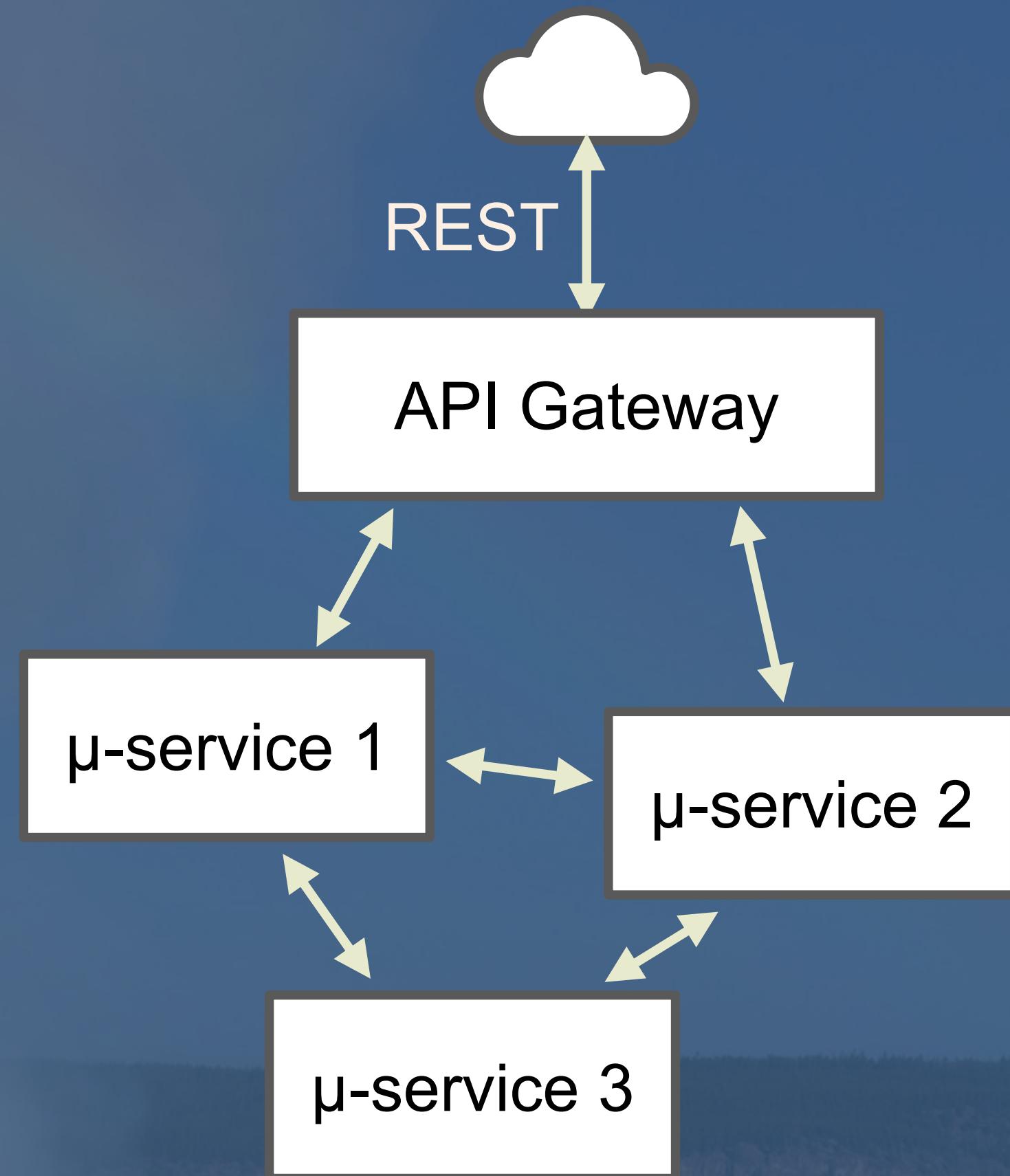
Optional configuration specifications

Other Uses of Ray: Microservices



What Are Microservices?

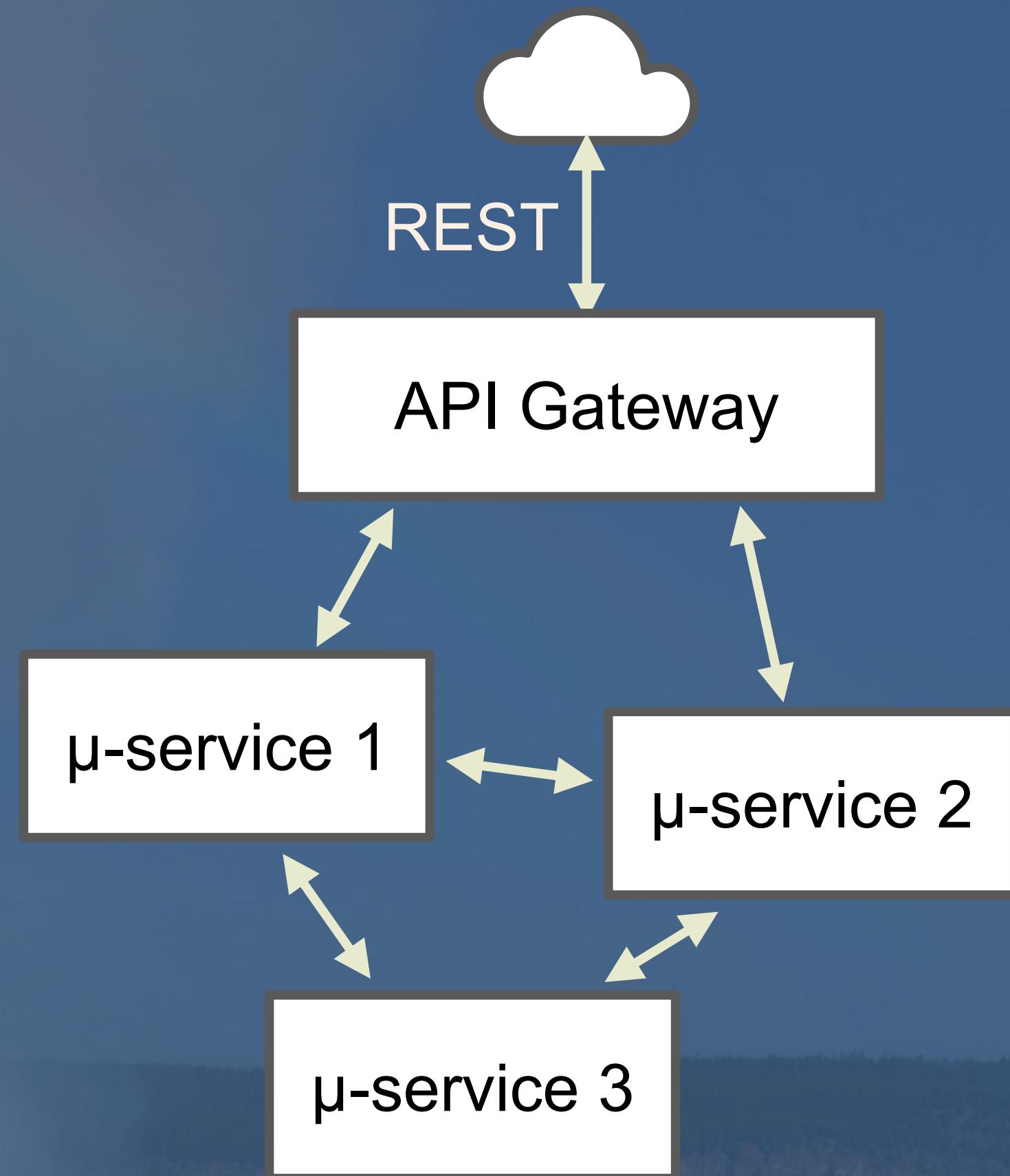
- They partition the domain
 - Conway's Law - Embraced
 - Separate responsibilities
- Separate management



What Are Microservices?

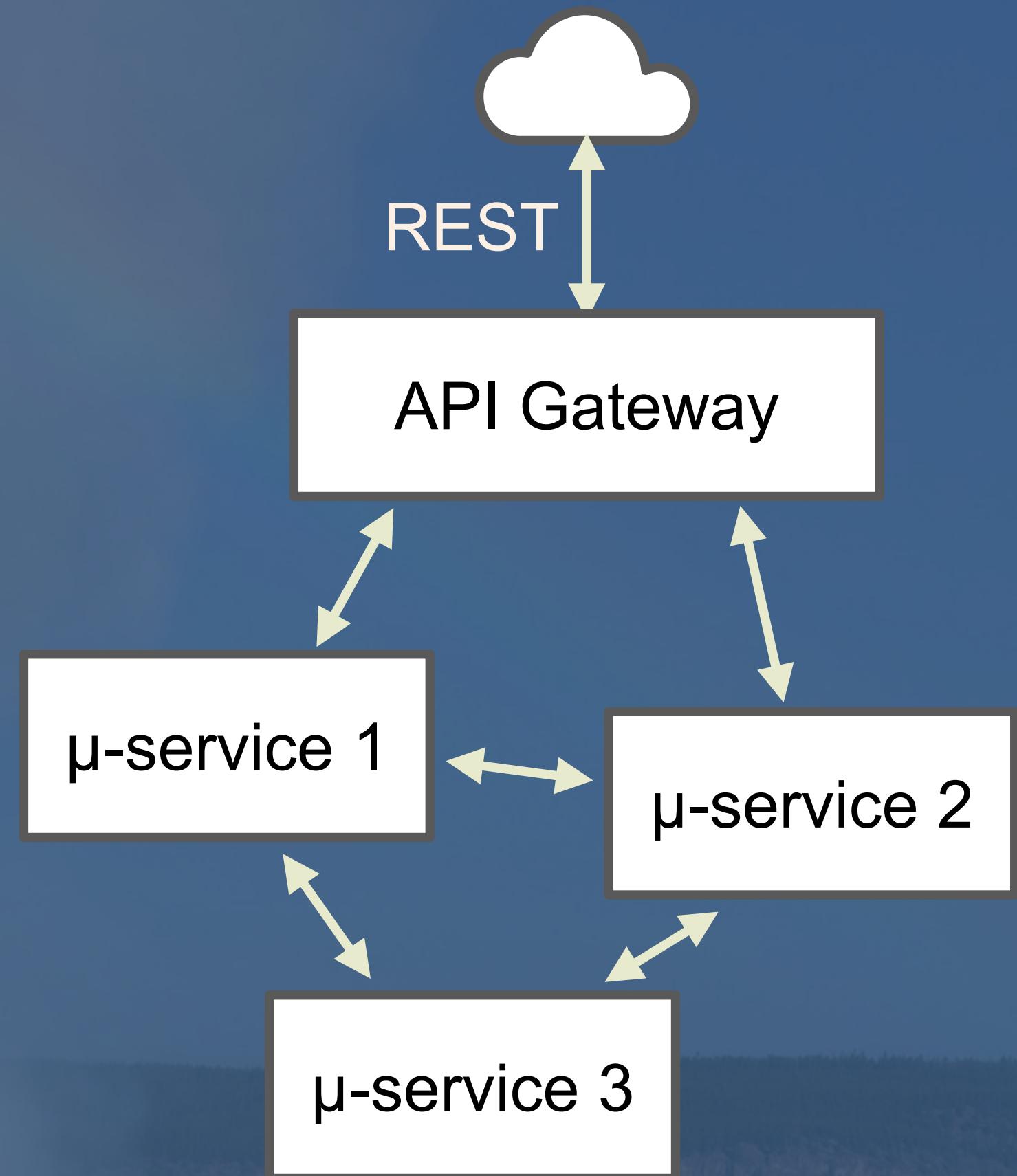
- They partition the domain
 - Conway's Law - Embraced
 - Separate responsibilities
- Separate management

What we mostly care
about for today's talk, the
“Ops in DevOps”



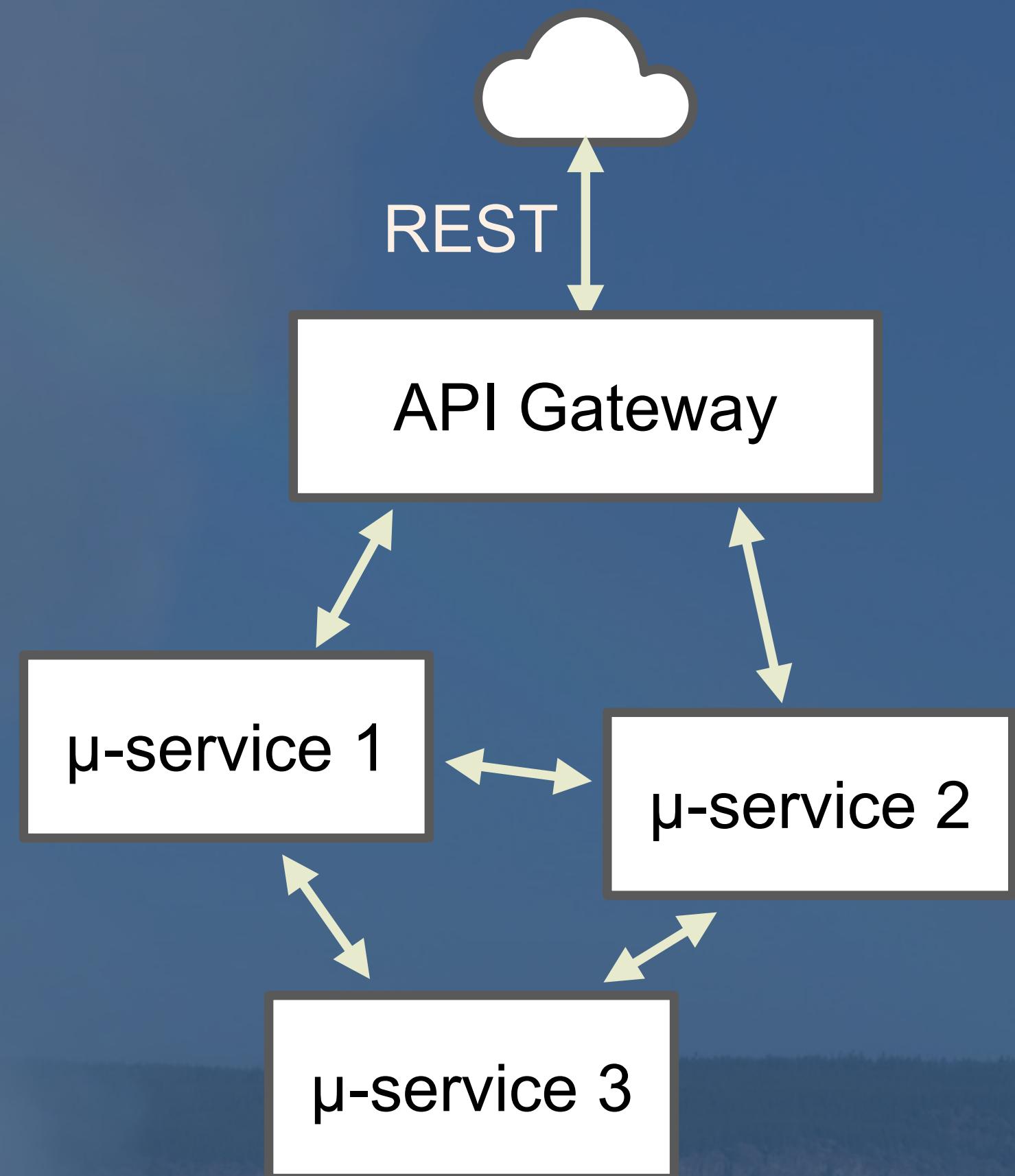
Conway's Law - Embraced

- “Any organization that designs a system will produce a design whose structure is a copy of the organization's communication structure”
- Let each team own and manage the services for its part of the domain



Separate Responsibilities

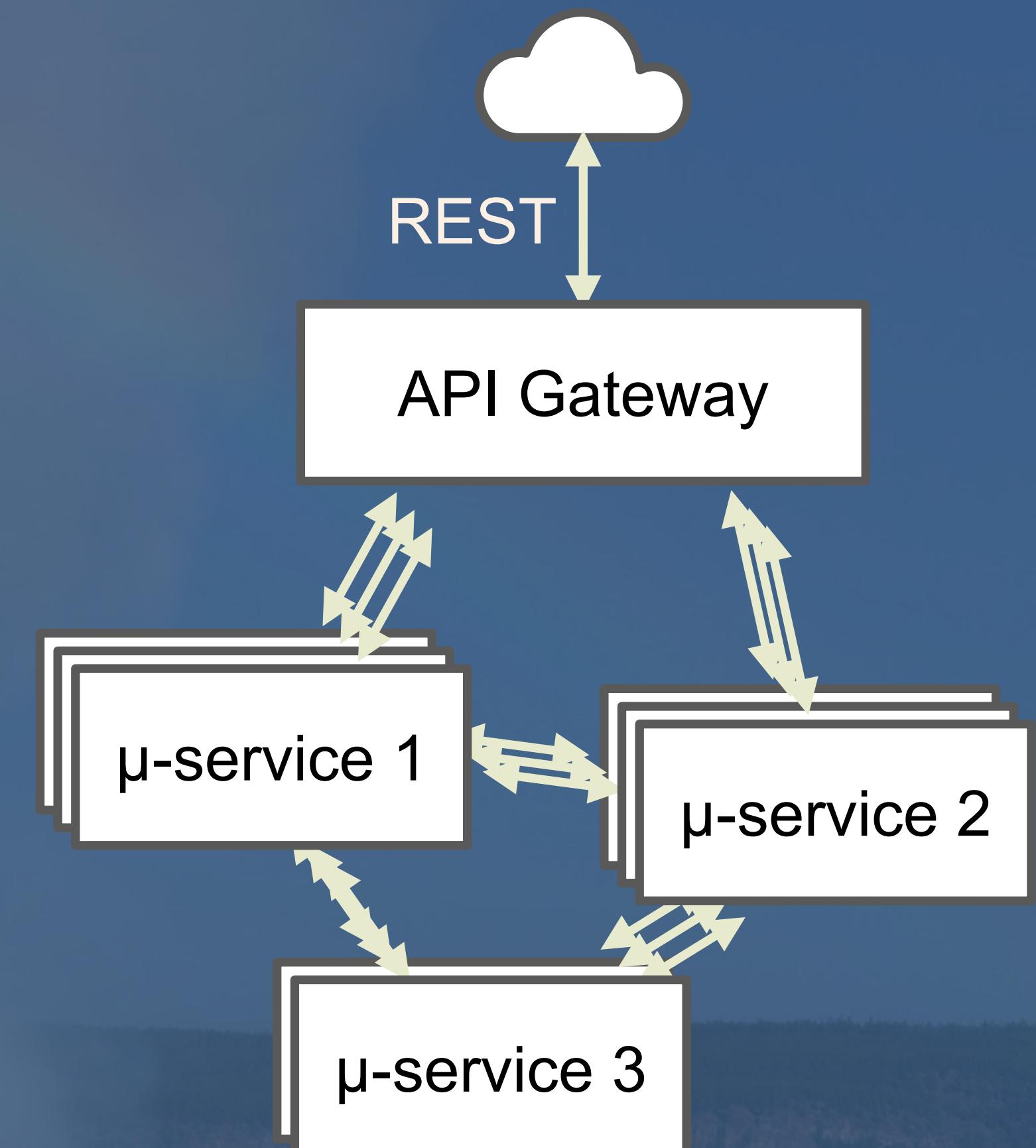
- Each microservice does “one thing”, a single responsibility with minimal coupling to the other microservices
- (Like, hopefully, the teams are organized, too...)



wikipedia.org/wiki/Single-responsibility_principle

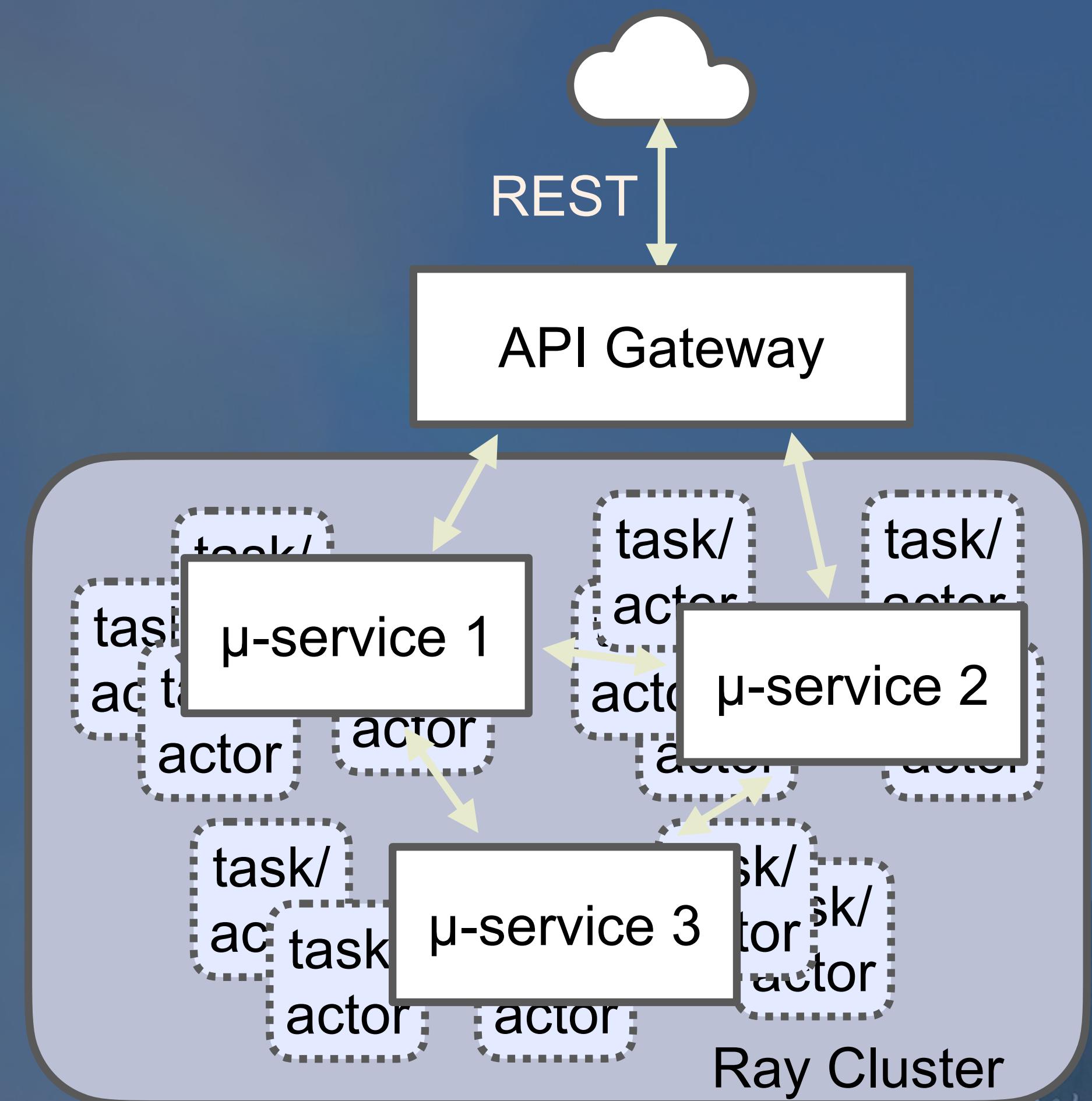
Separate Management

- Each team manages its own instances
- Each microservice has a different number of instances for scalability and resiliency
- But they have to be managed **explicitly**



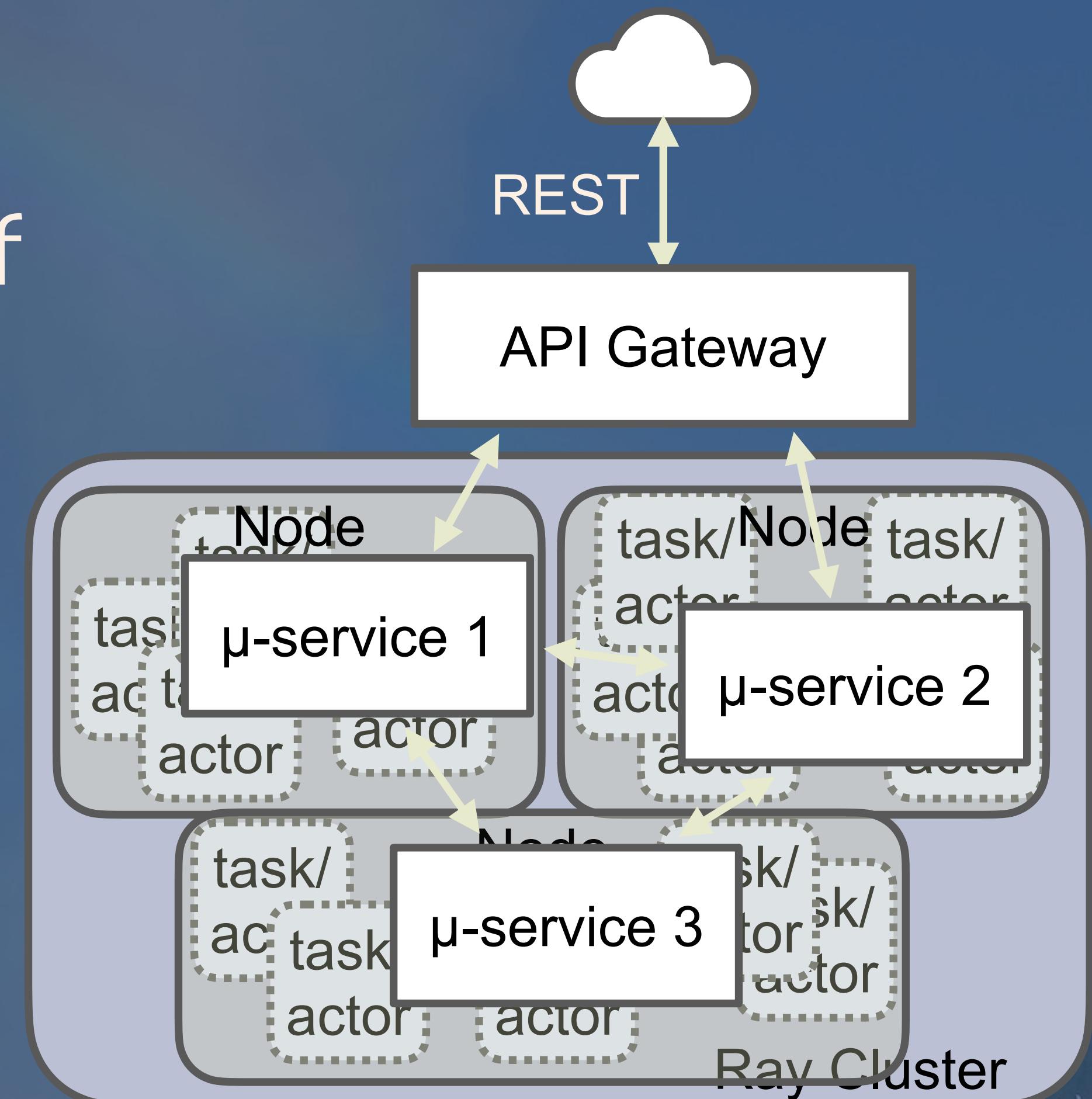
Management - Simplified

- With Ray, you have one “logical” instance to manage and Ray does the cluster-wide scaling for you.



What about Kubernetes (and others...)?

- Ray scaling is very fine grained.
- It operates within the “nodes” of coarse-grained managers
- Containers, pods, VMs, or physical machines



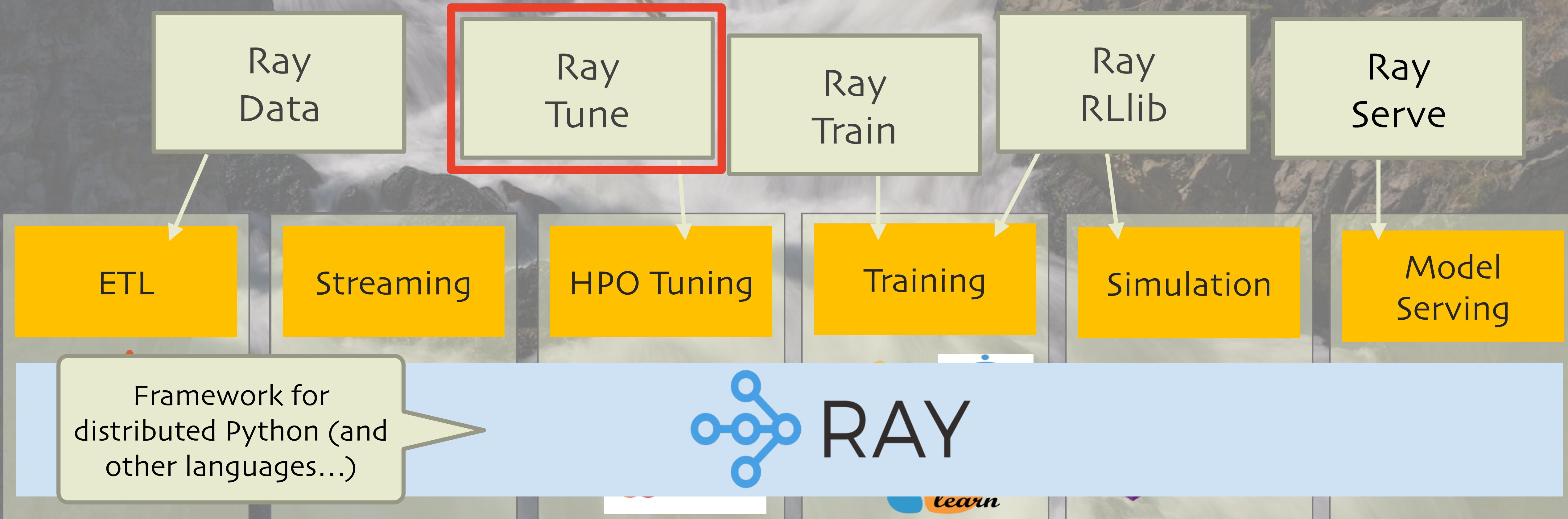


Hyper Parameter Tuning with Ray Tune



Hyperparameter Tuning - Ray Tune

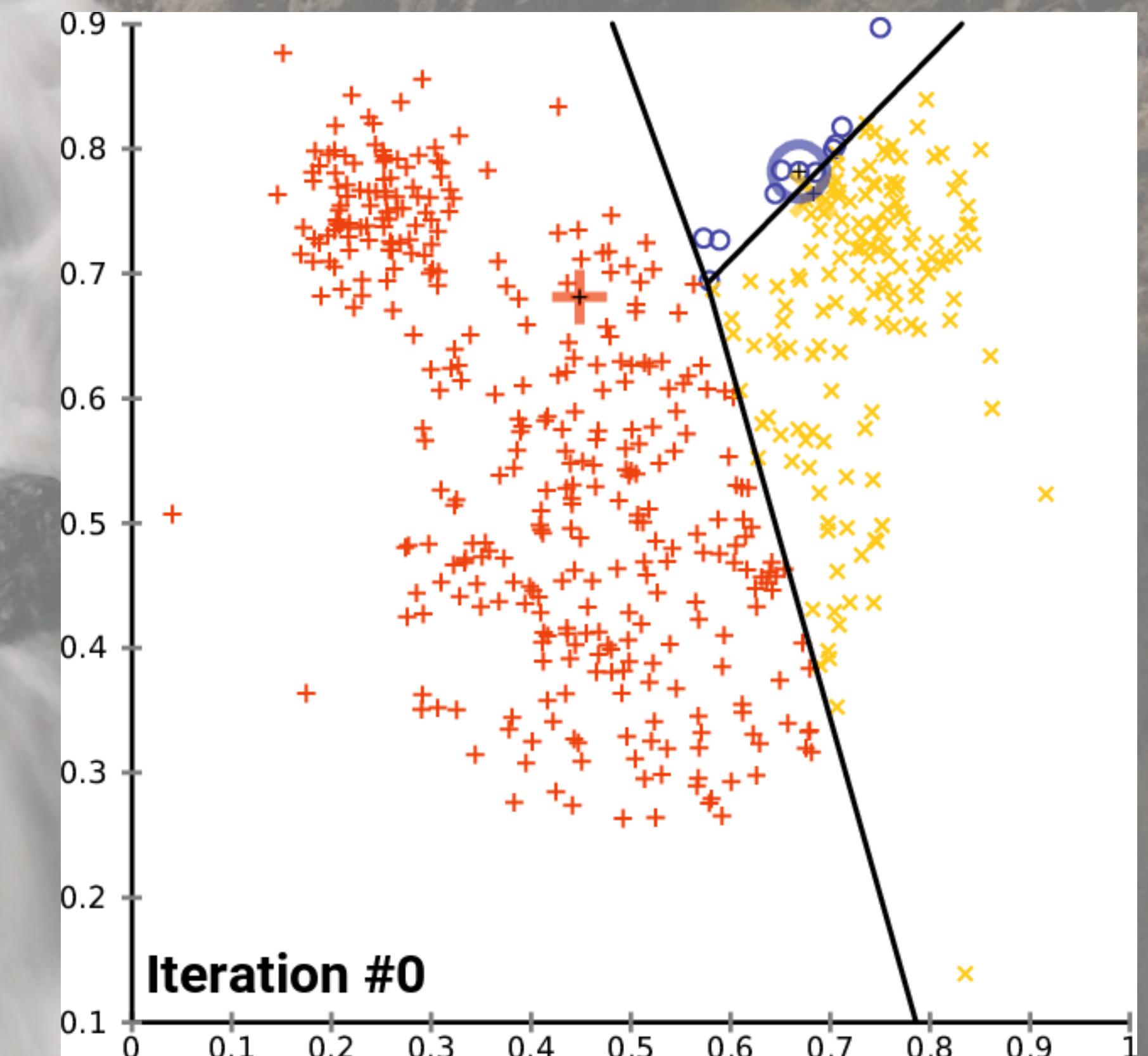
Domain-specific
libraries for each
subsystem



What Is Hyperparameter Tuning?

Trivial example:

- What's the best value for "k" in k-means??
- k is a "hyperparameter"
- The resulting clusters are defined by "parameters"



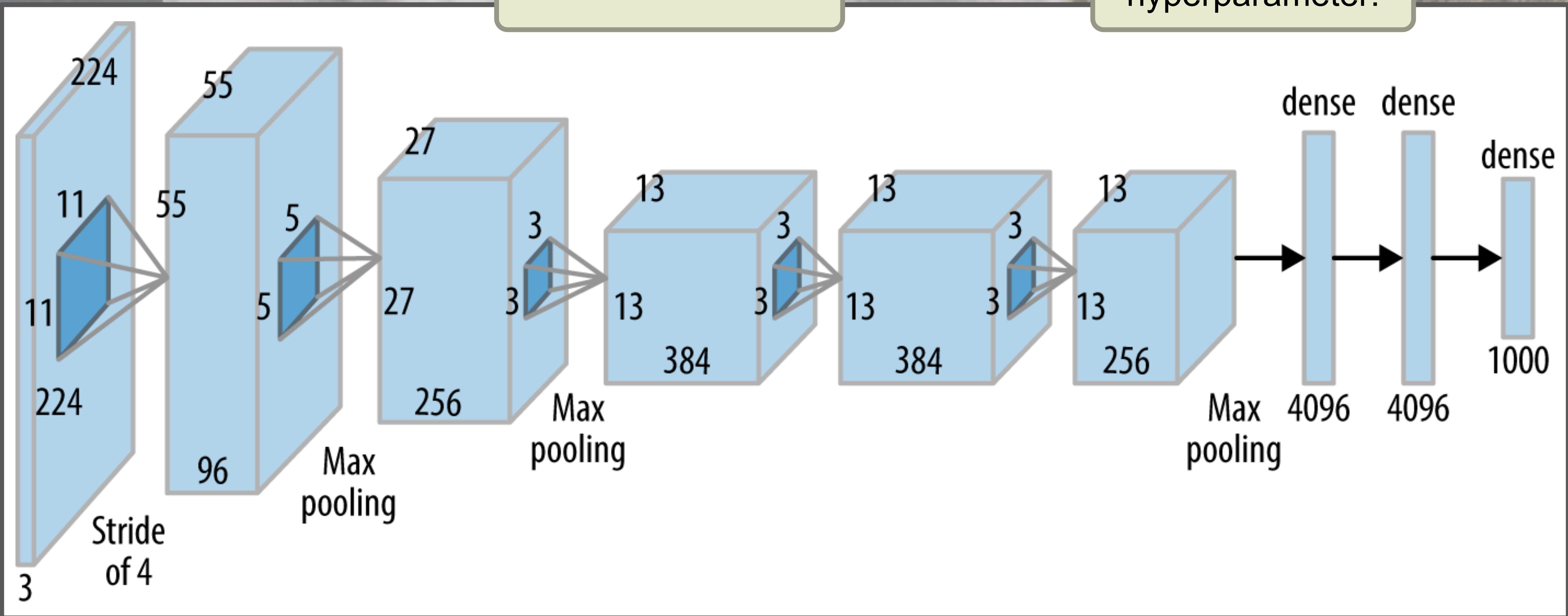
credit: https://commons.wikimedia.org/wiki/File:K-means_convergence.gif



Nontrivial Example - Neural Networks

How many layers?
What kinds of layers?

Every number
shown is a
hyperparameter!

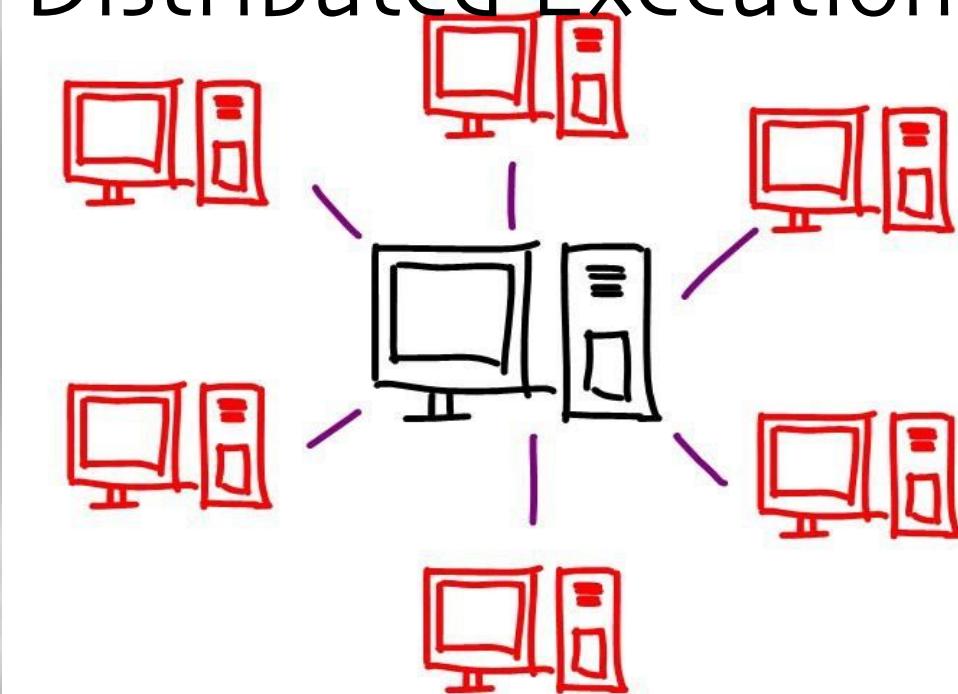


Tune is Built with Deep Learning as a Priority

Resource Aware
Scheduling



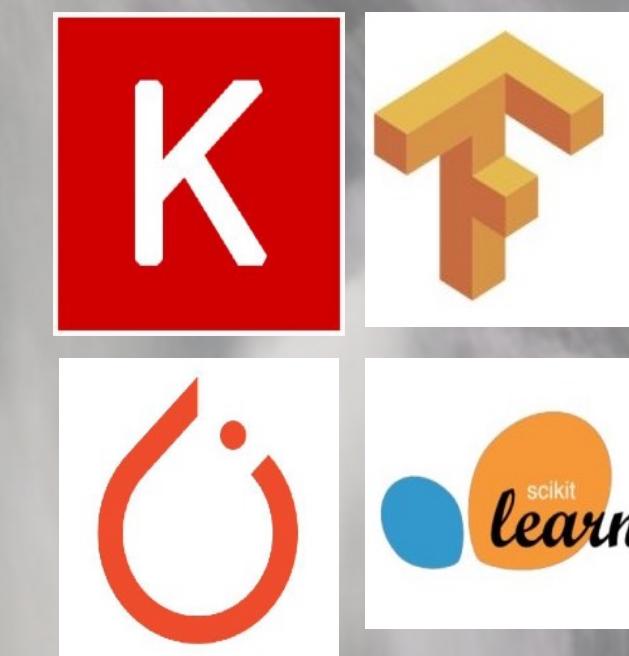
Seamless
Distributed Execution



Simple API for
new algorithms

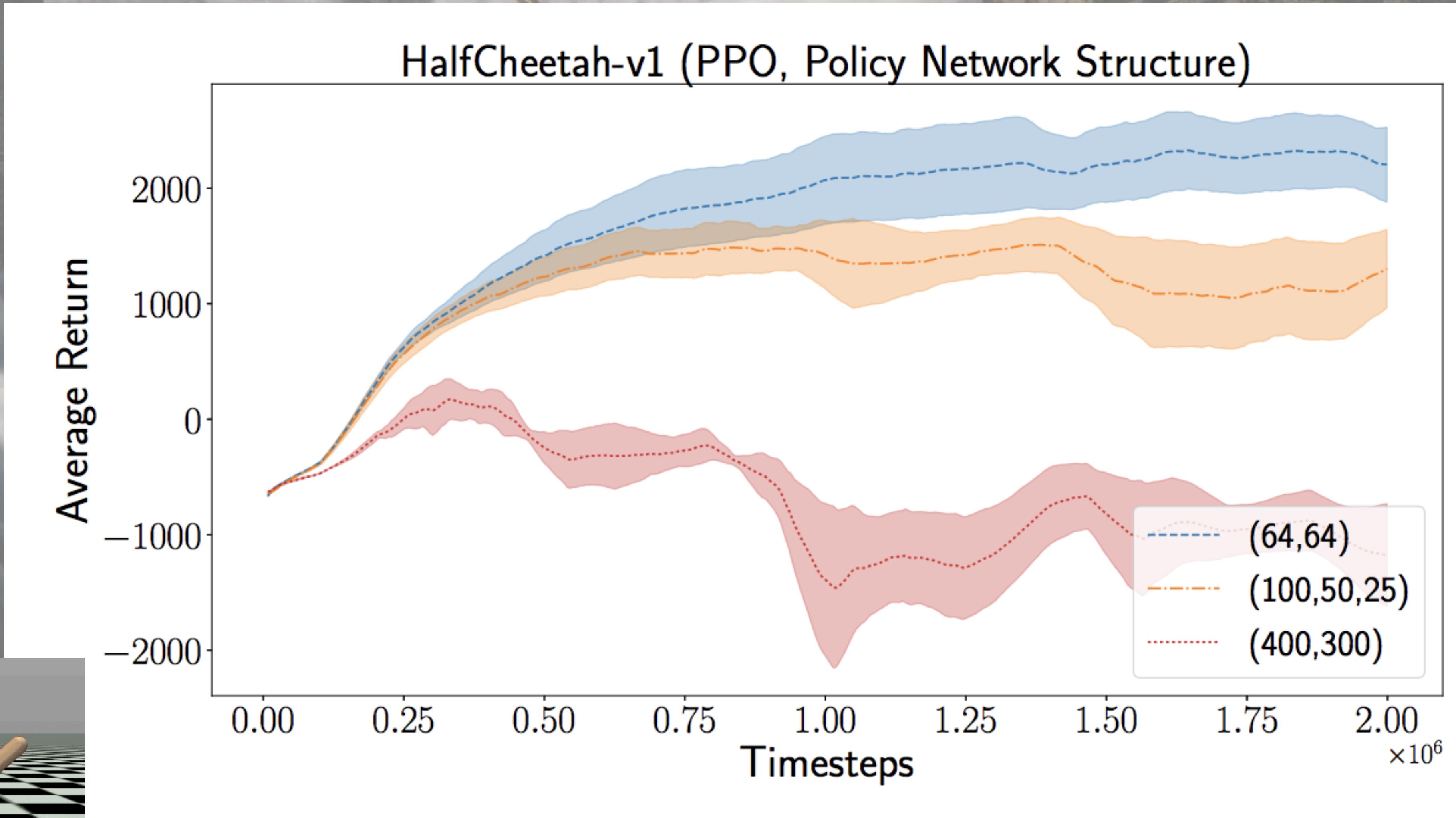
```
class TrialScheduler:  
    def on_result(self, trial, result): ...  
    def choose_trial_to_run(self): ...
```

Framework Agnostic



tune.io

Hyperparameters Are Important for Performance

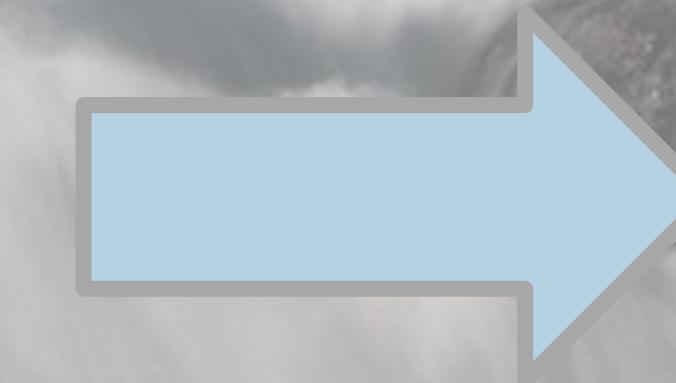


Why We Need a Framework for Tuning Hyperparameters

We want the best model

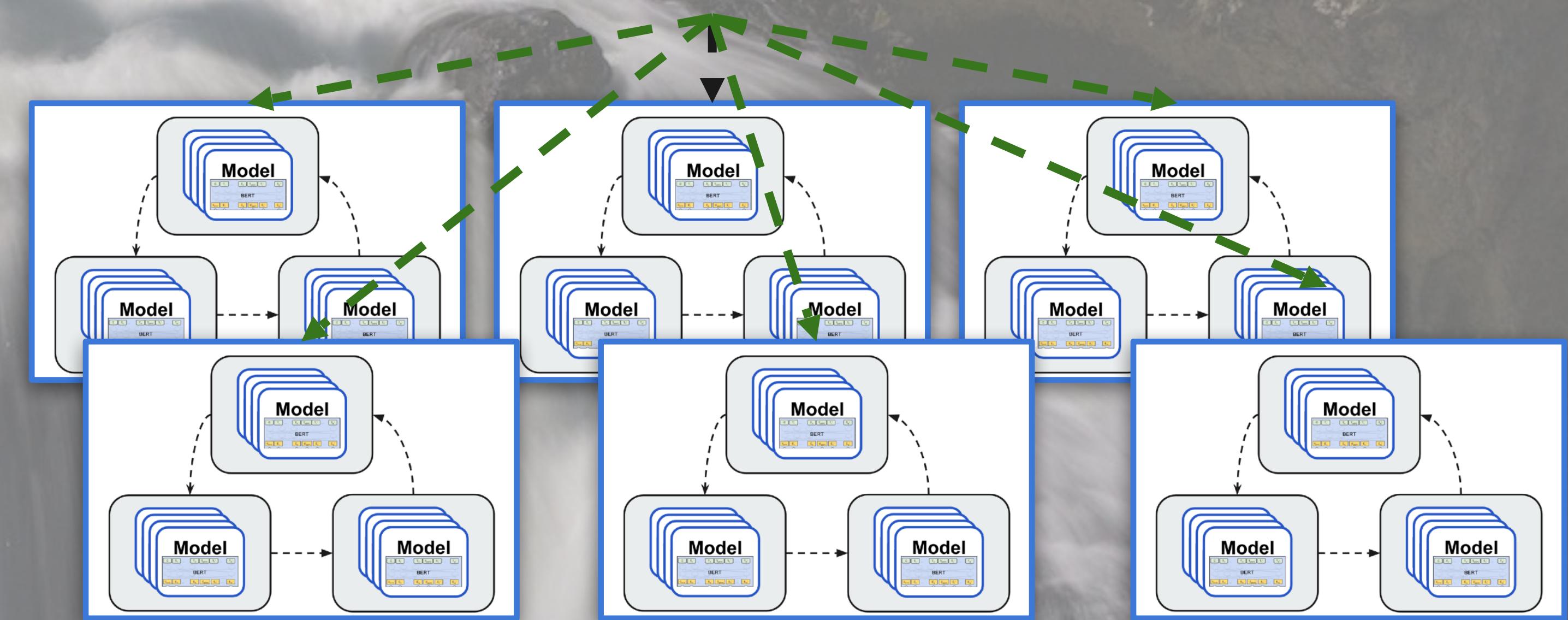
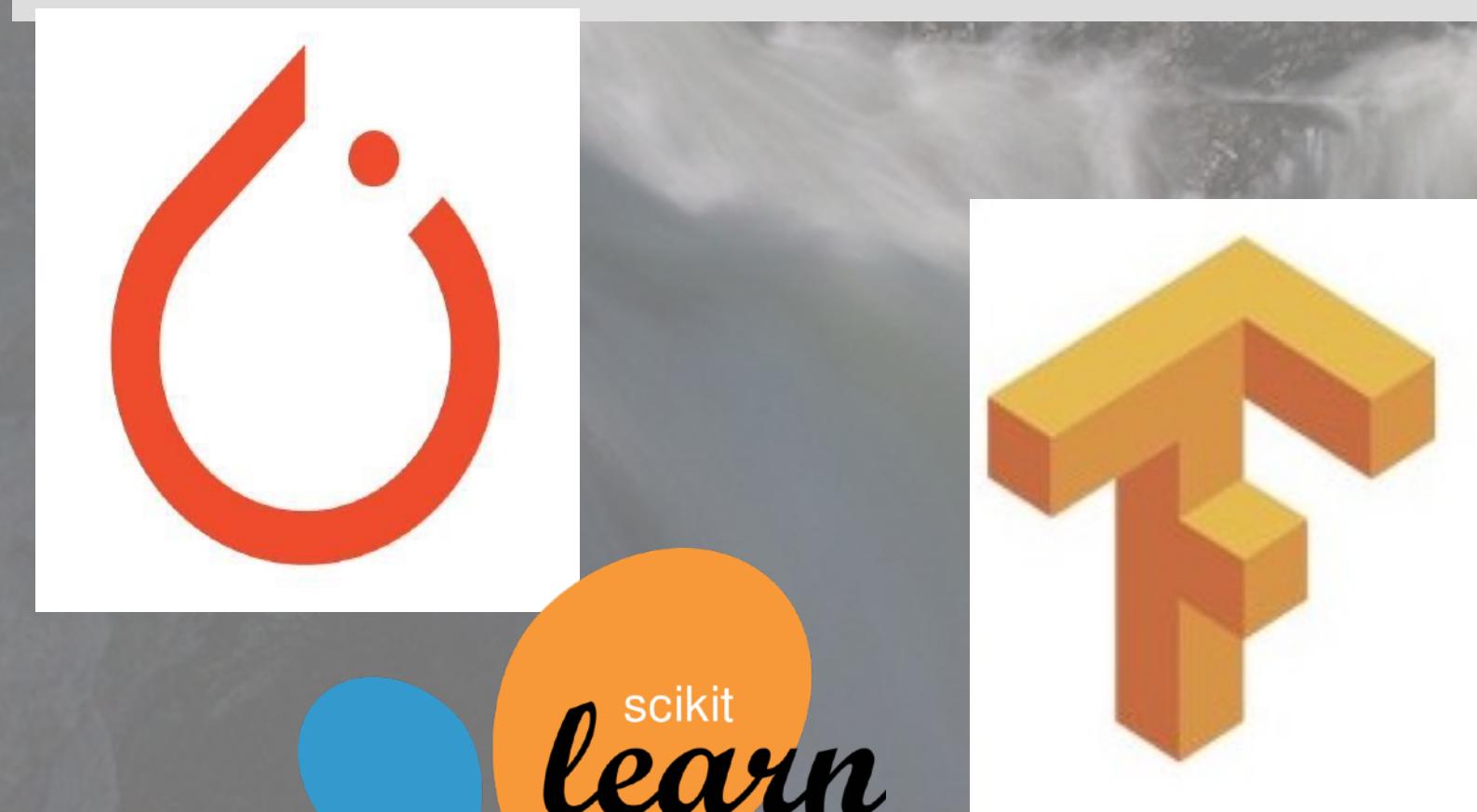
Resources are expensive

Model training is time-consuming



Tuning + Distributed Training

```
tune.run(PytorchTrainable,  
 config={  
     "model_creator": PretrainBERT,  
     "data_creator": create_data_loader,  
     "use_gpu": True,  
     "num_replicas": 8,  
     "lr": tune.uniform(0.001, 0.1)  
 },  
 num_samples=100,  
 search_alg=BayesianOptimization()
```



Native Integration with TensorBoard HParams

