Scaling ML/AI Applications with Ray

Dean Wampler
October 5, 2020
dean@deanwampler.com
@deanwampler
ray.io

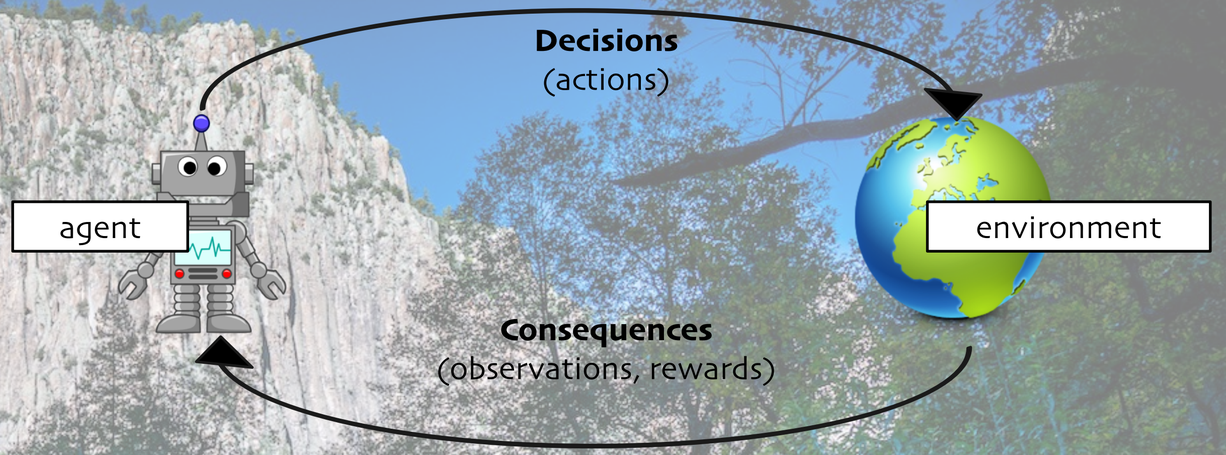
Outline

- Computing Challenges for Reinforcement Learning
- Ray to the Rescue
- Ray RLlib
- Next steps


RAY
Computing Challenges for
Reinforcement Learning
@deanwampler


@deanwampler

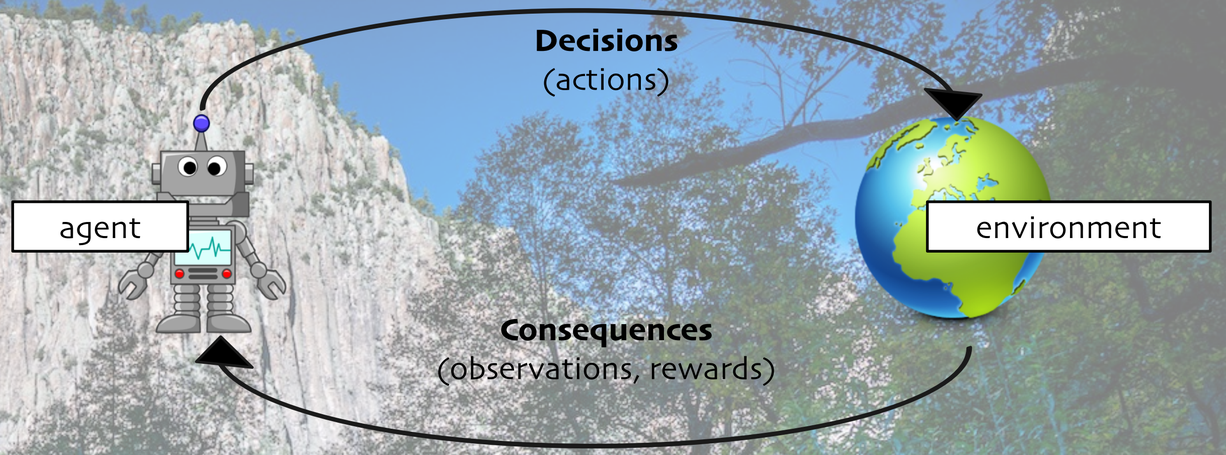
Reinforcement Learning



Games	Robotics, Autonomous Vehicles	Industrial Processes	System Optimization	Advertising, Recommendations	Finance

RL applications

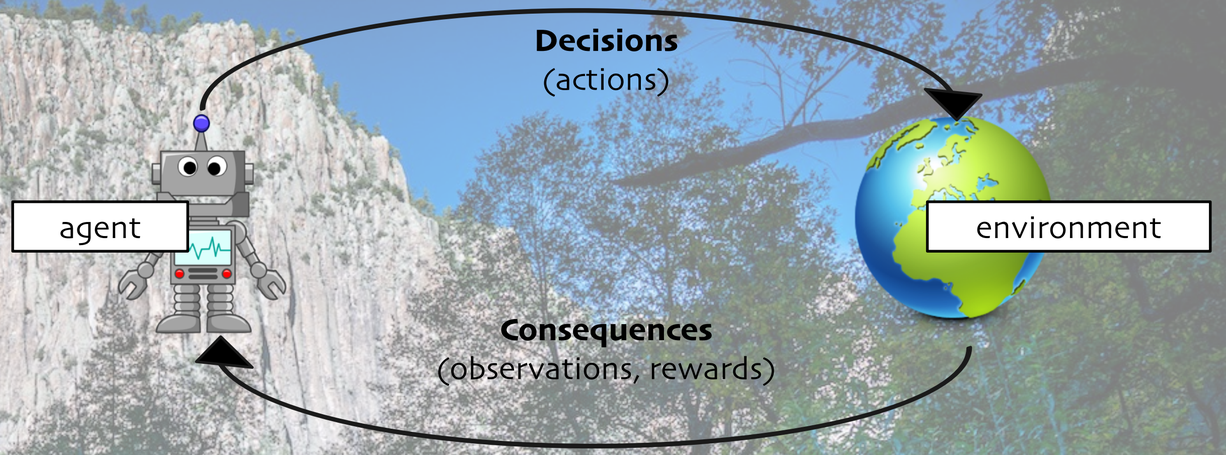
Reinforcement Learning



Games | Robotics, Autonomous Vehicles | Industrial Processes | System Optimization | Advertising, Recommendations | Finance

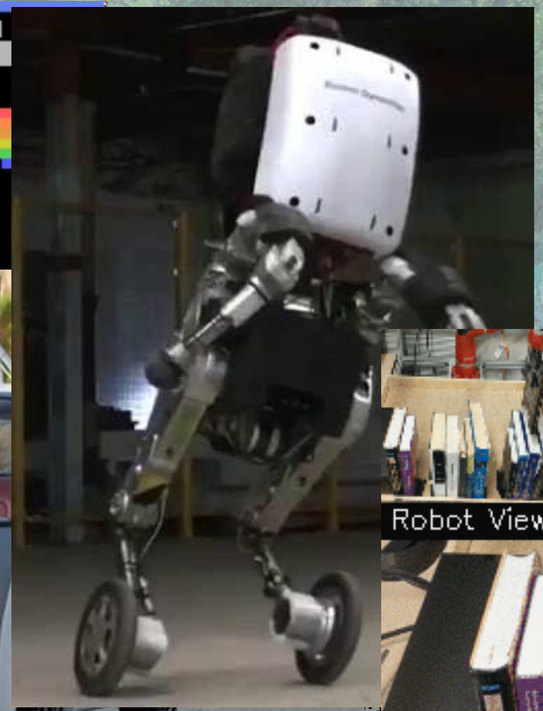
RL applications

Reinforcement Learning

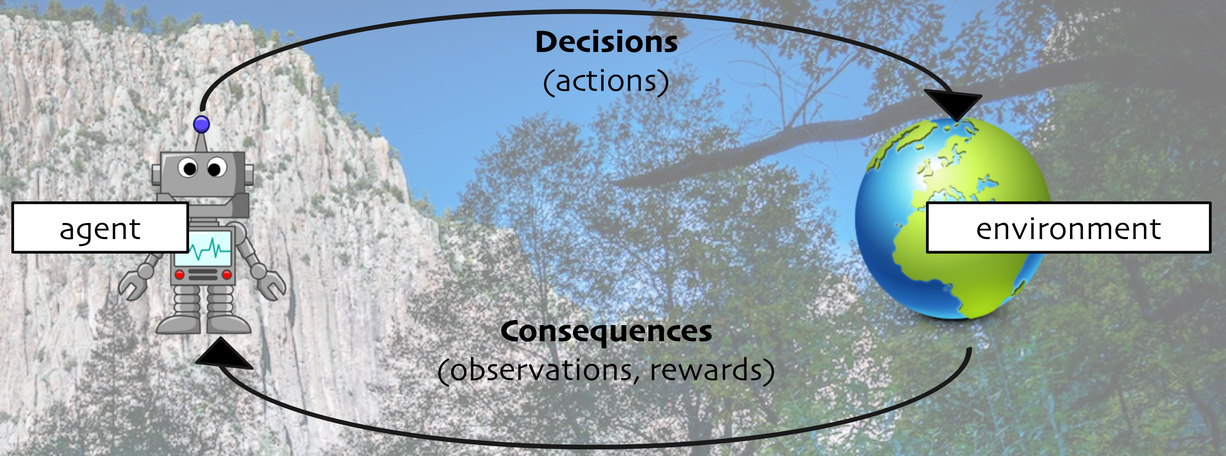


Games | Robotics, Autonomous Vehicles | Industrial Processes | System Optimization | Advertising, Recommendations | Finance

RL applications

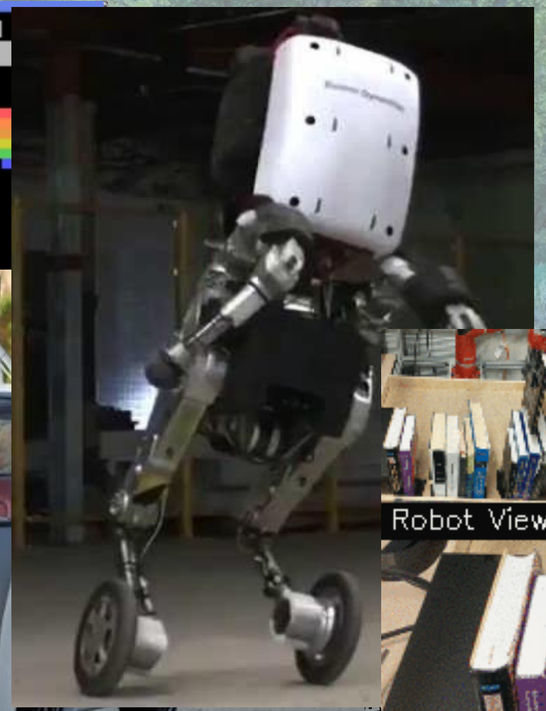


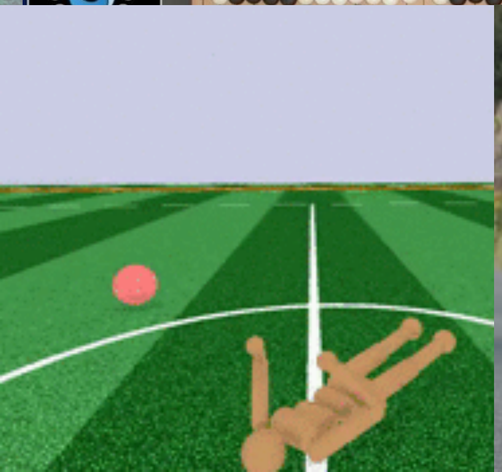
Reinforcement Learning

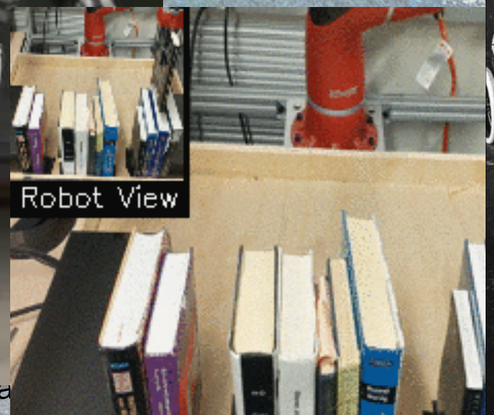


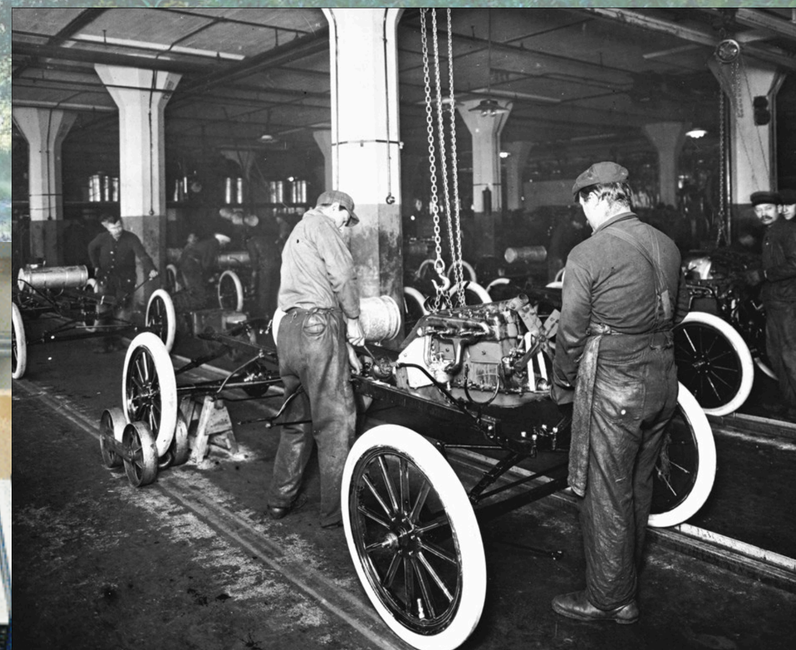
Games | Robotics, Autonomous Vehicles | Industrial Processes | System Optimization | Advertising, Recommendations | Finance

RL applications

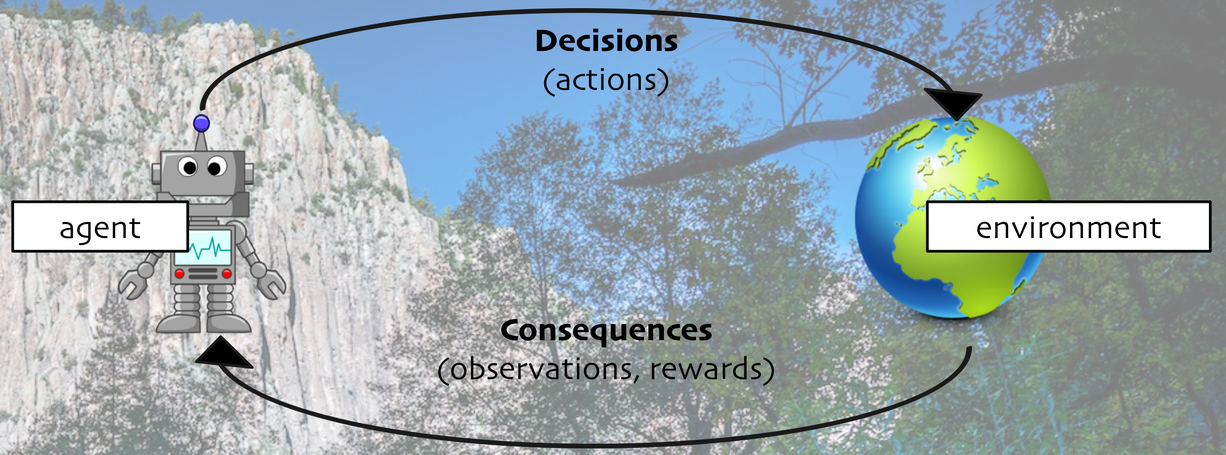






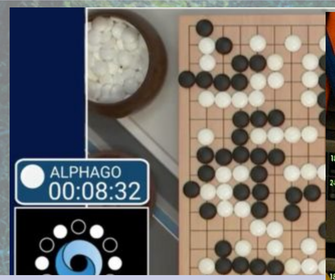


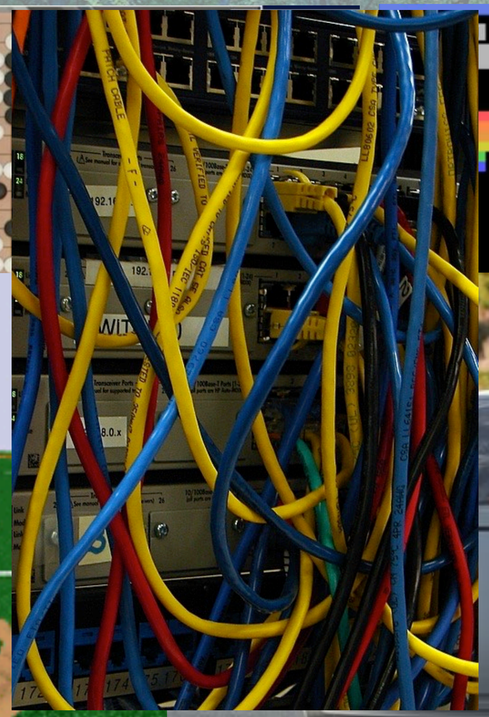
Reinforcement Learning

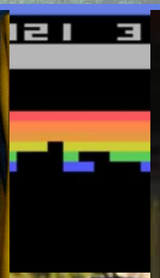


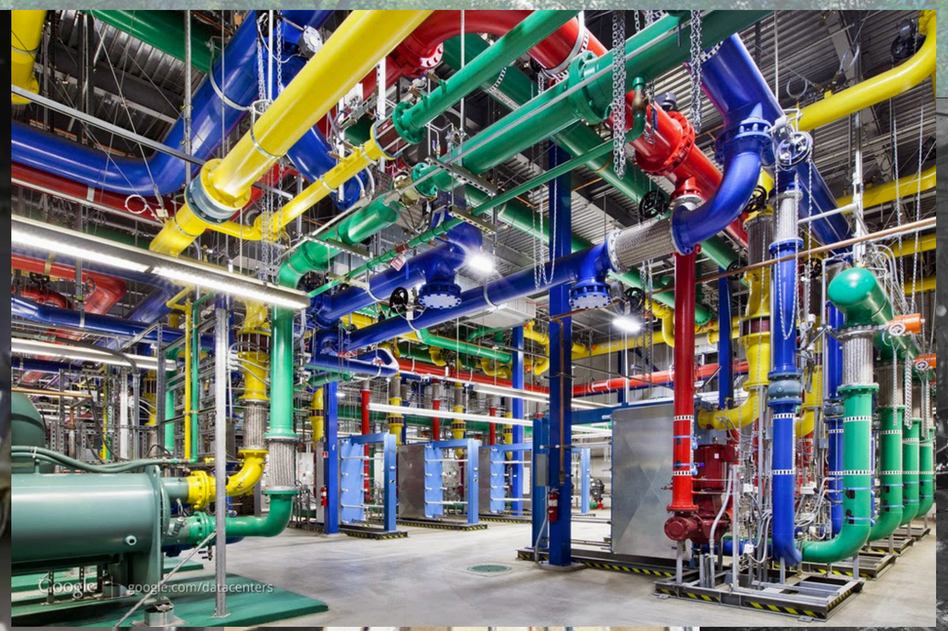
Games | Robotics, Autonomous Vehicles | Industrial Processes | System Optimization | Advertising, Recommendations | Finance

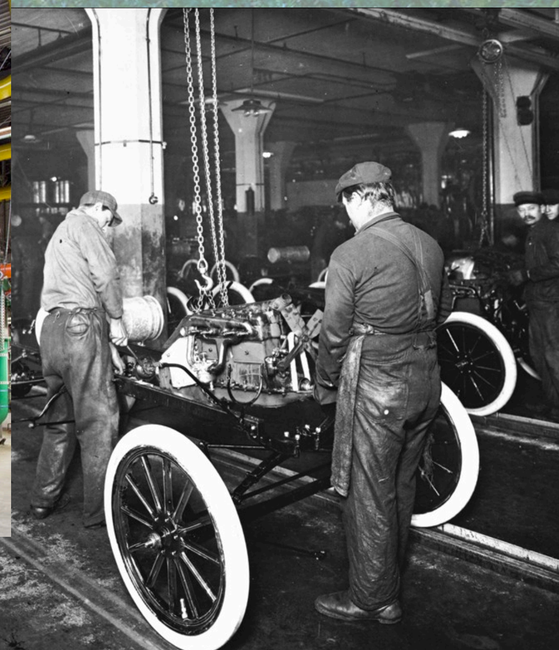
RL applications



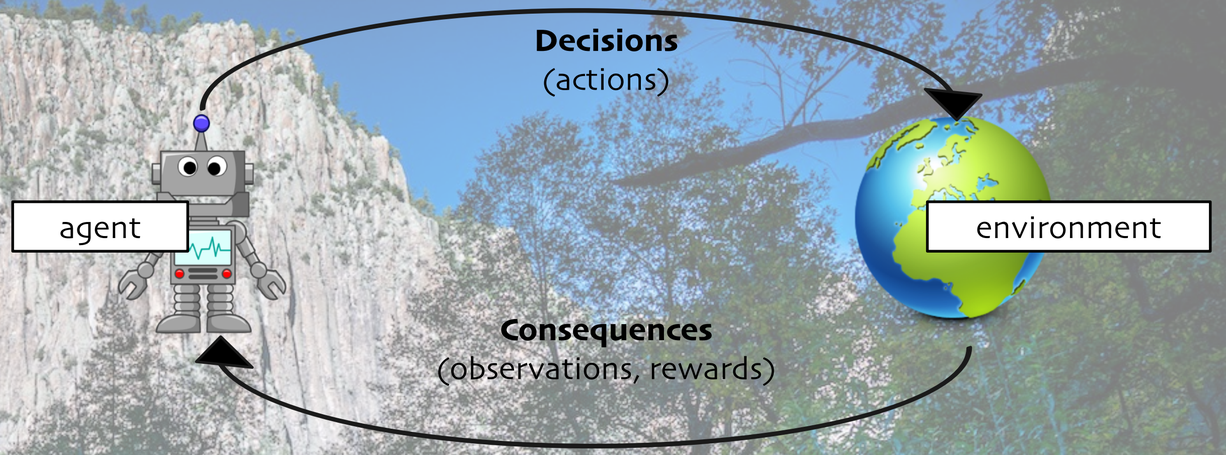






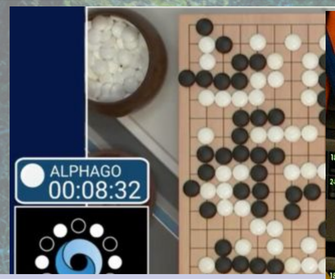


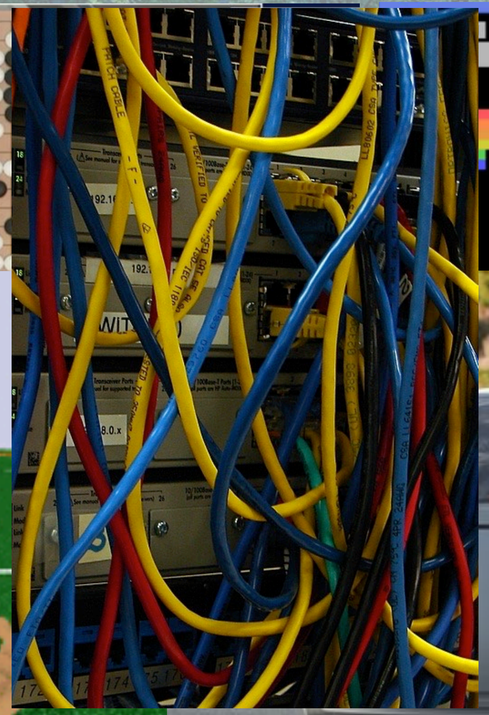
Reinforcement Learning

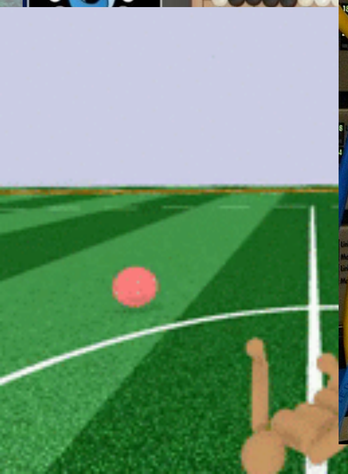


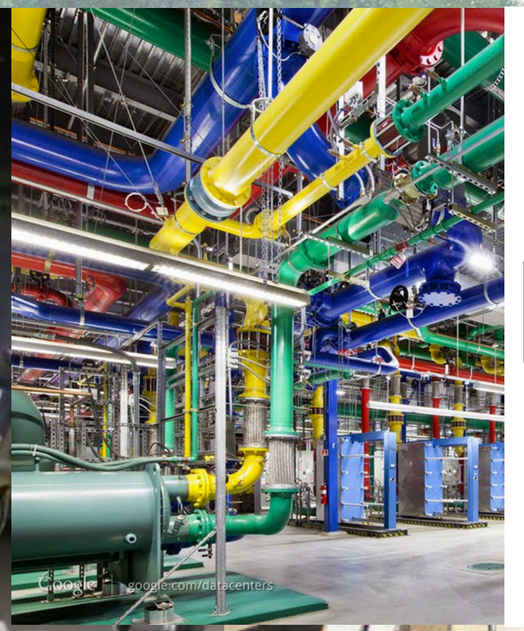
Games | Robotics, Autonomous Vehicles | Industrial Processes | System Optimization | **Advertising, Recommendations** | Finance

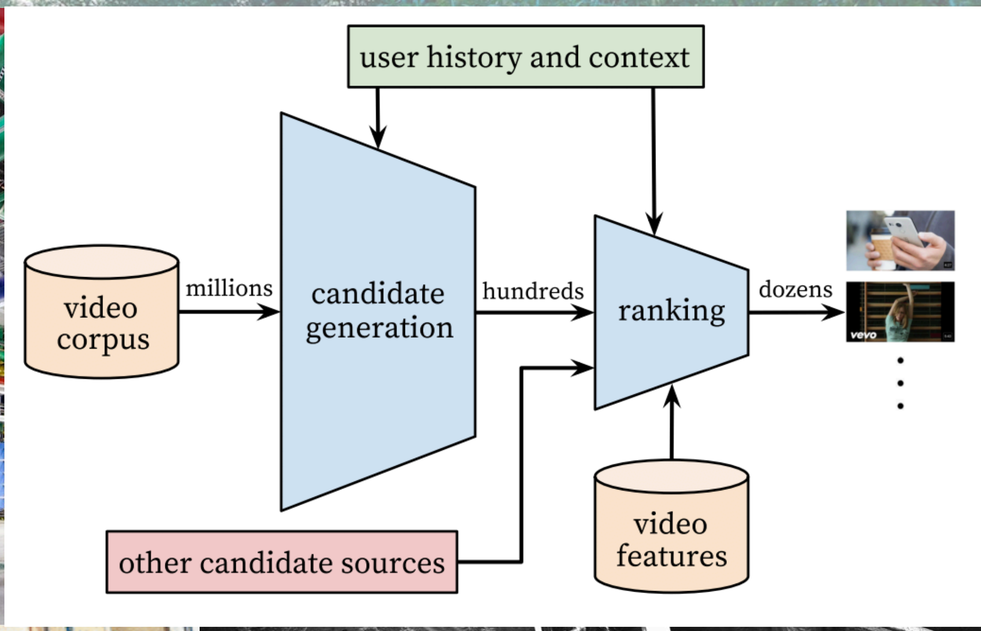
RL applications



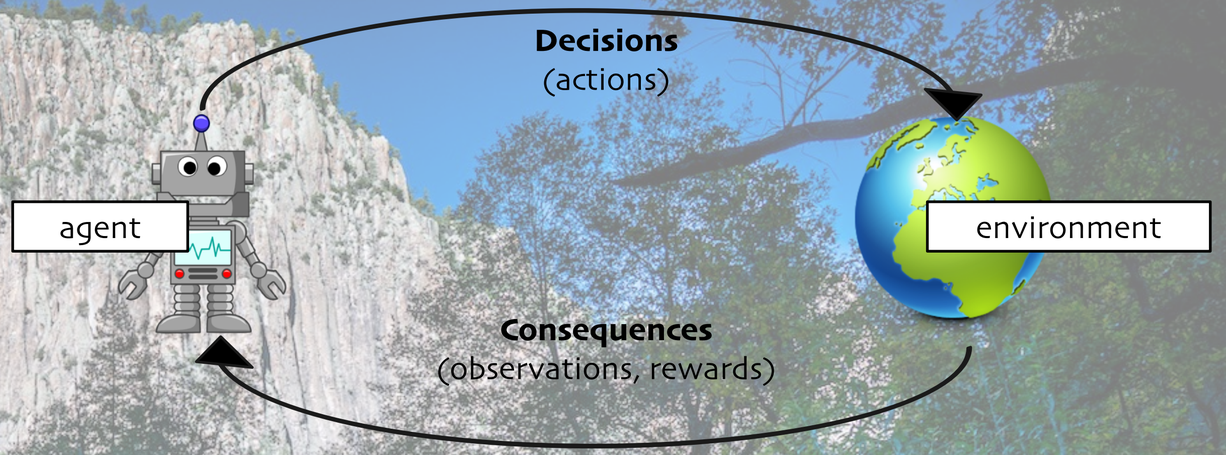






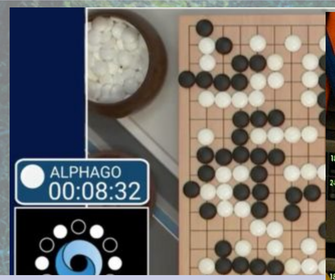


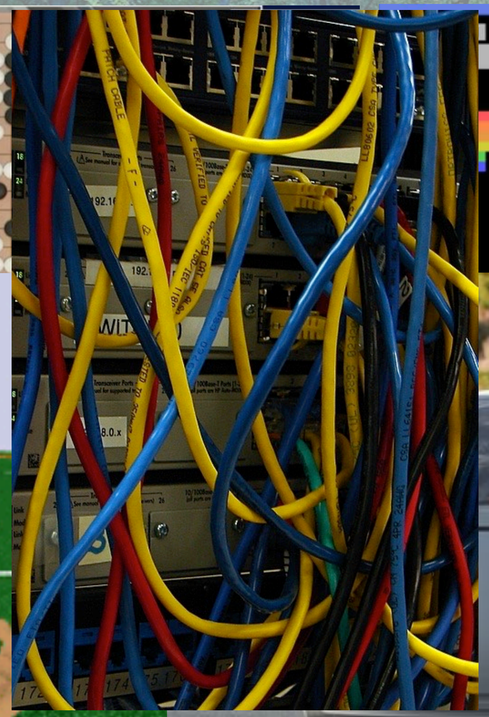
Reinforcement Learning

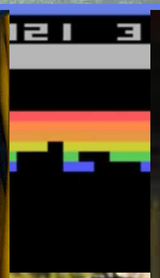


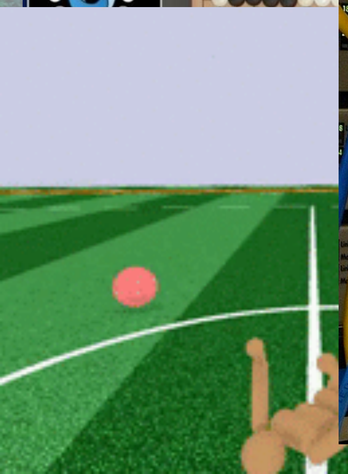
Games | Robotics, Autonomous Vehicles | Industrial Processes | System Optimization | Advertising, Recommendations | Finance

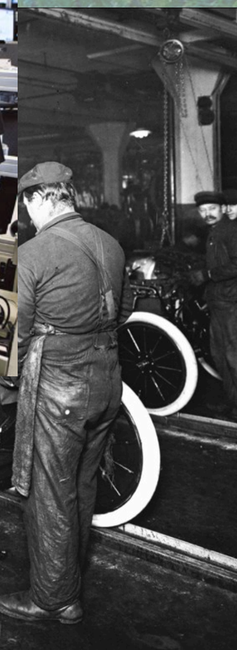
RL applications







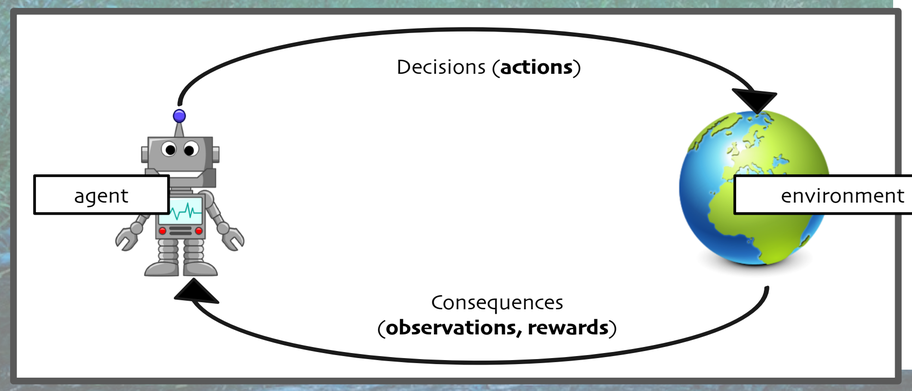


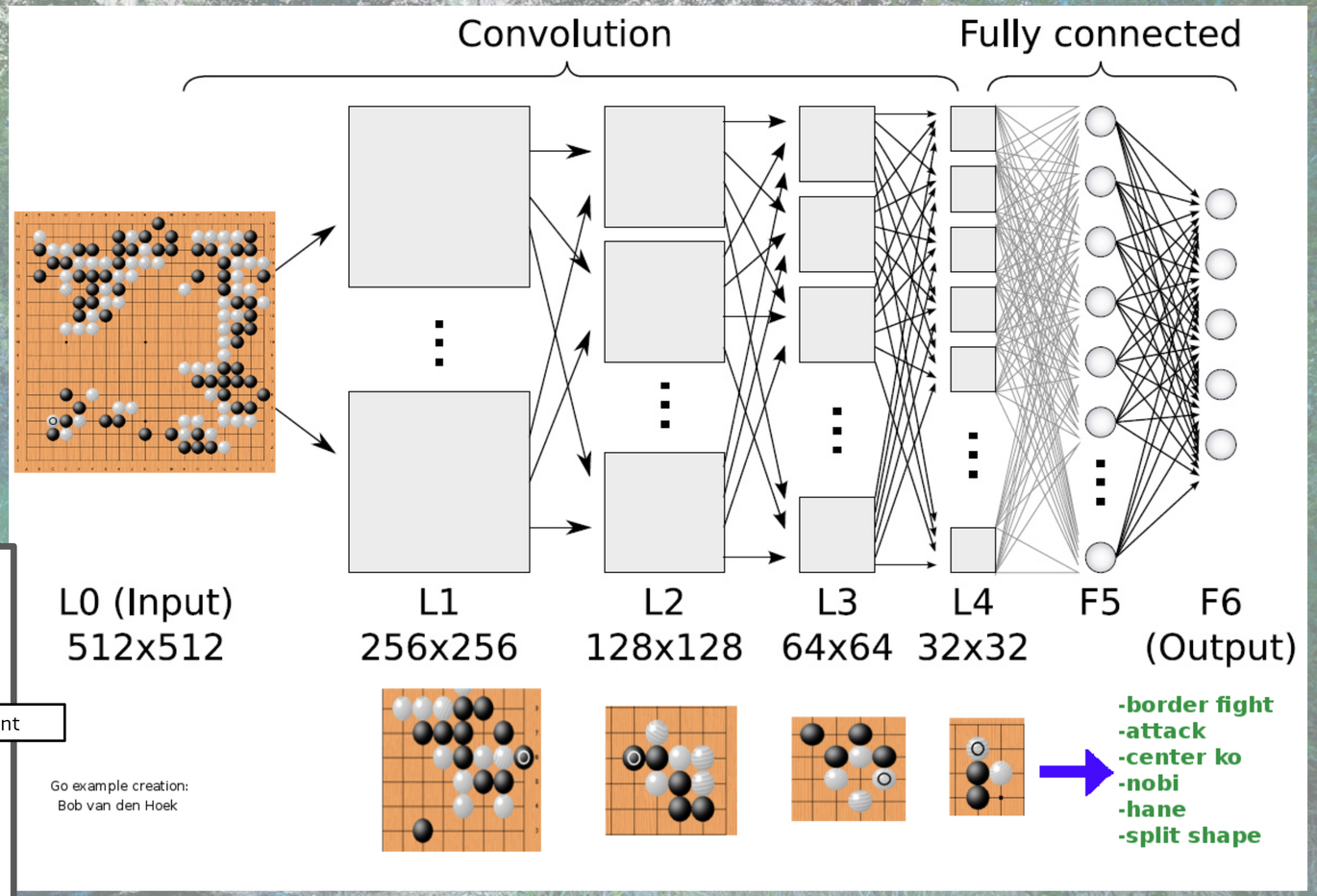


Go as a Reinforcement Learning Problem

AlphaGo (Silver et al. 2016)

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



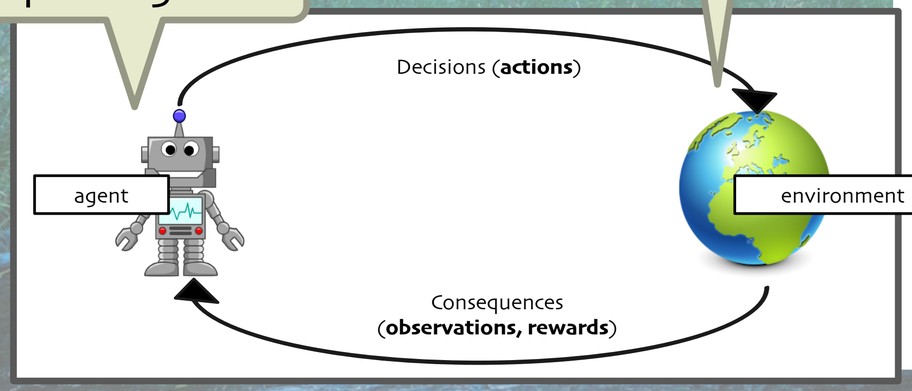


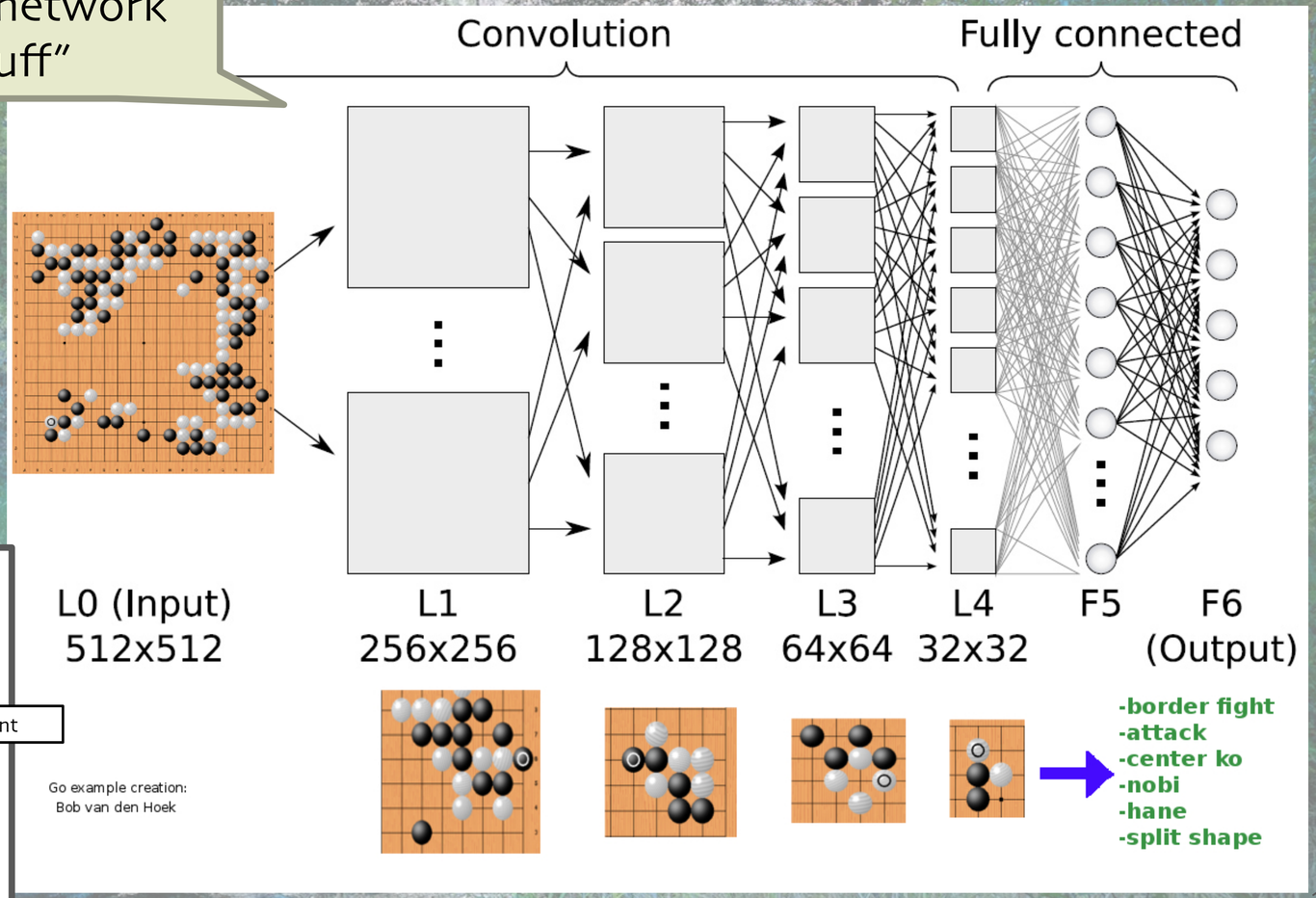
"Worst Case" Diversity of Compute Requirements

Neural network "stuff"

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

Complex agent?





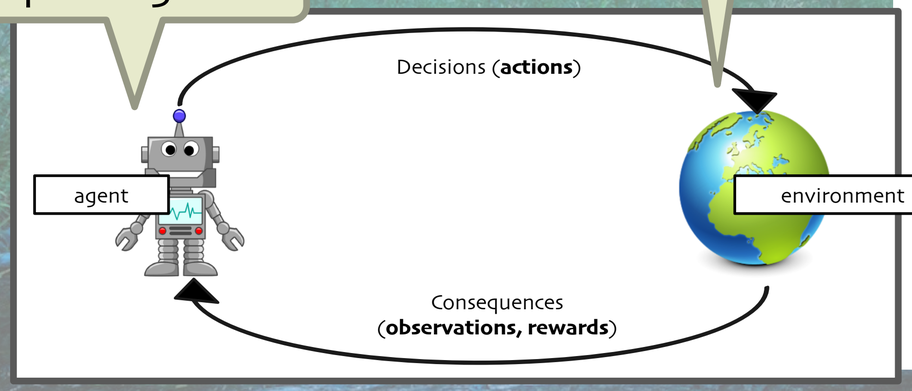
"Worst Case" Diversity of Compute Requirements

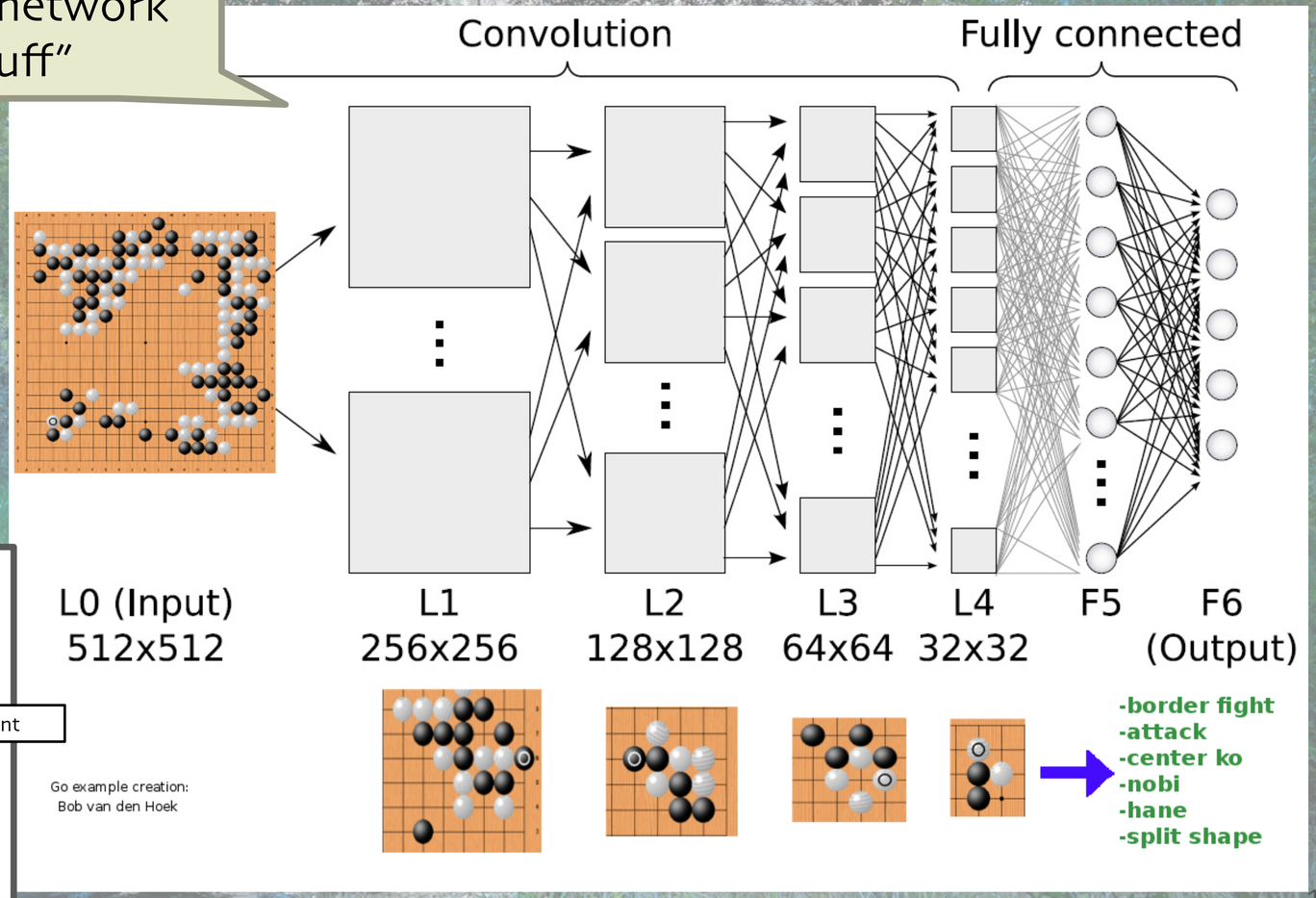
And repeated play, over and over again, to train for achieving the best reward

Neural network "stuff"

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

Complex agent?

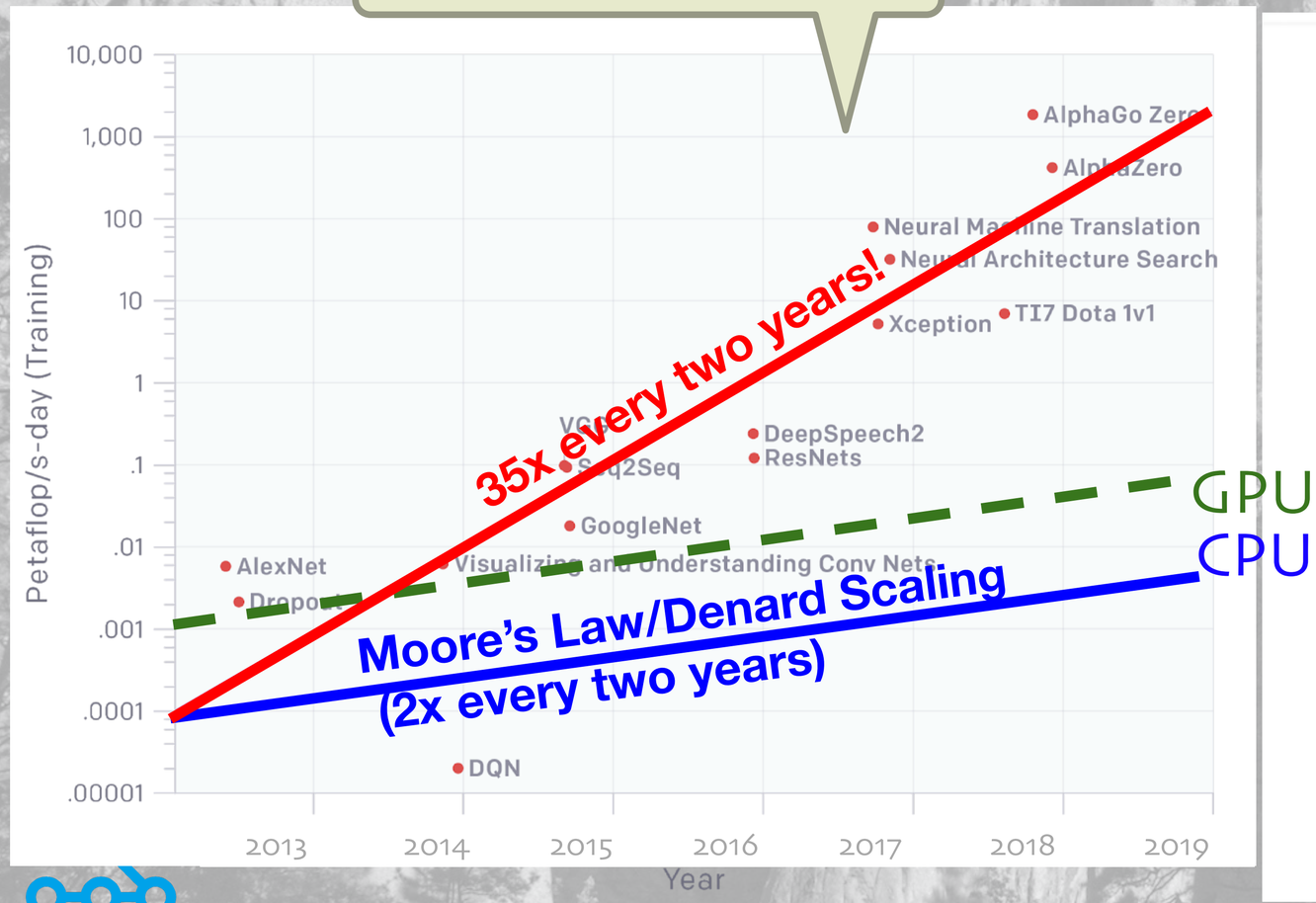




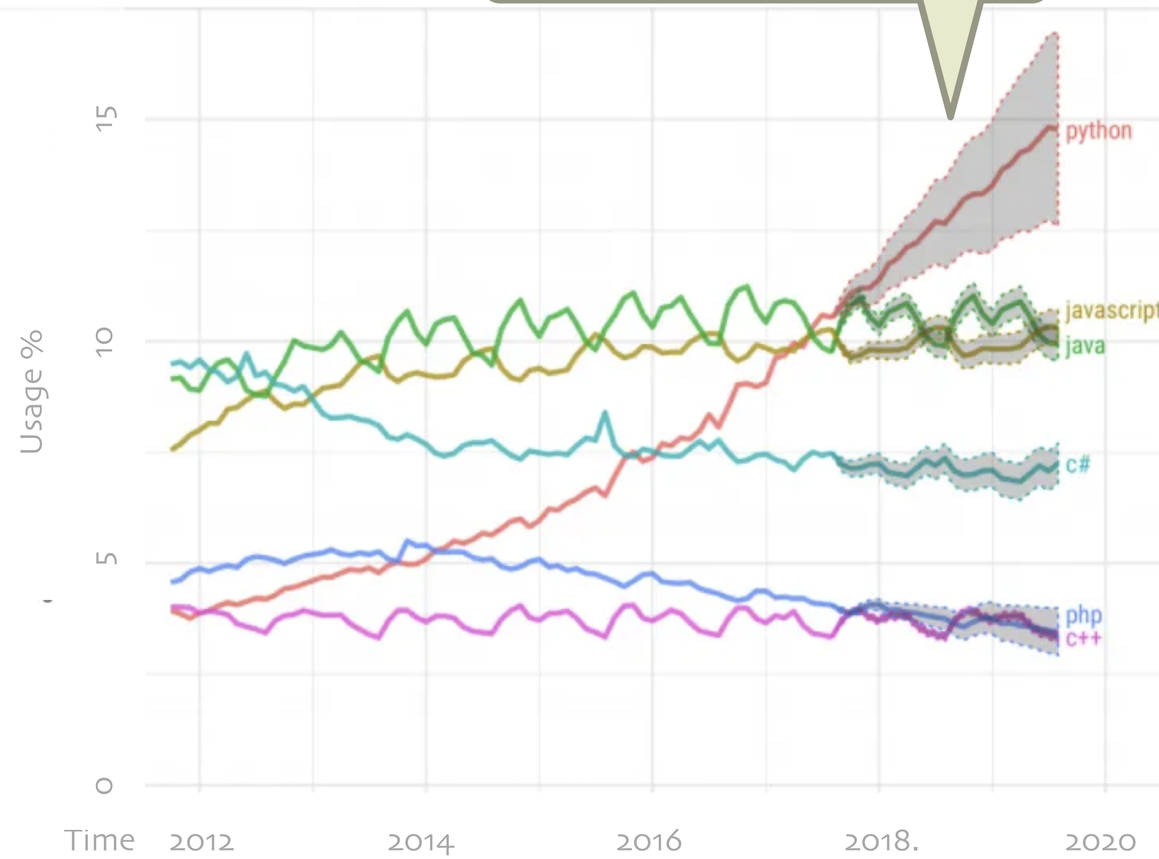

RAY
Ray to the Rescue
@deanwampler

Two Major Trends

Model sizes and therefore compute requirements outstripping Moore's Law



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



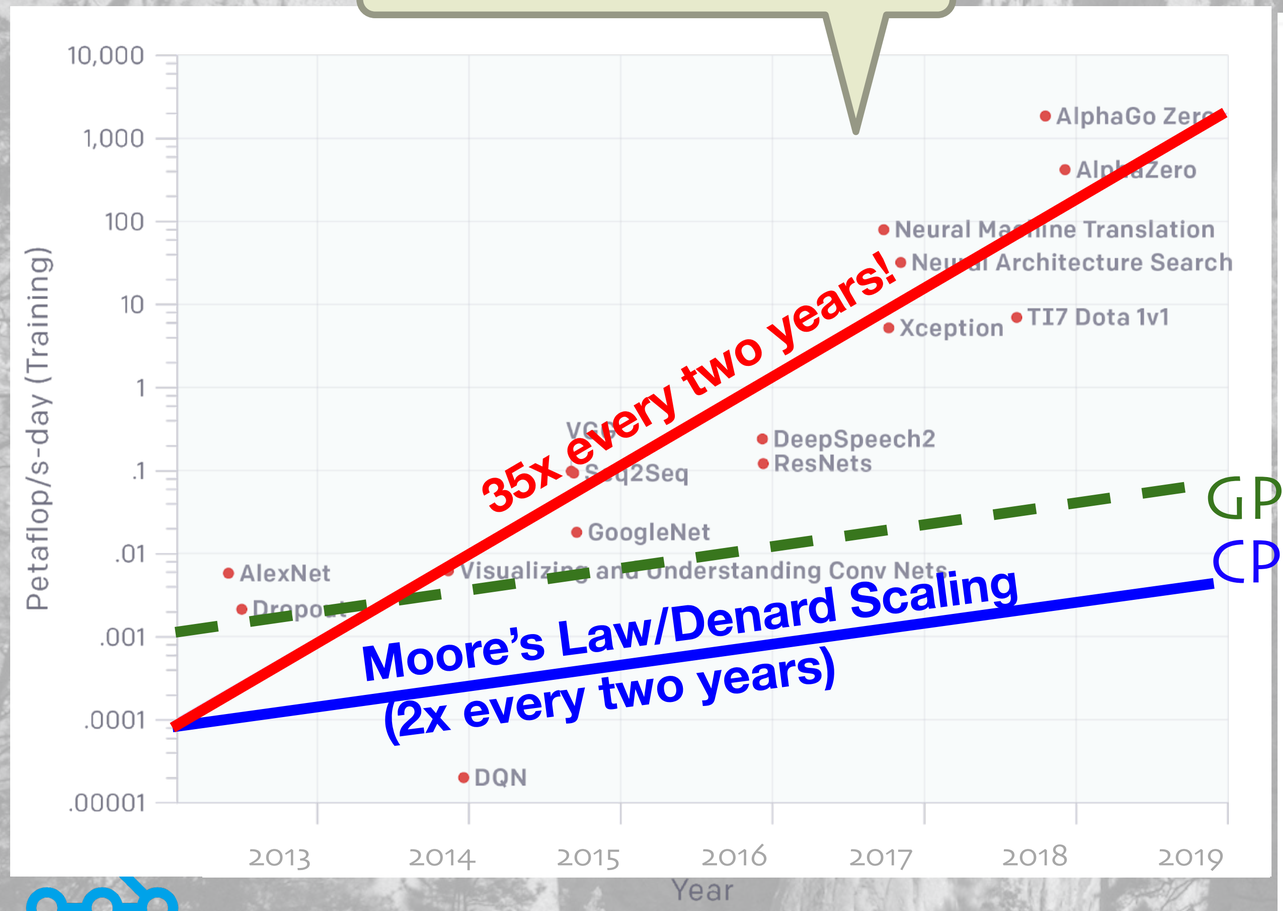
https://openai.com/blog/ai-and-compute/

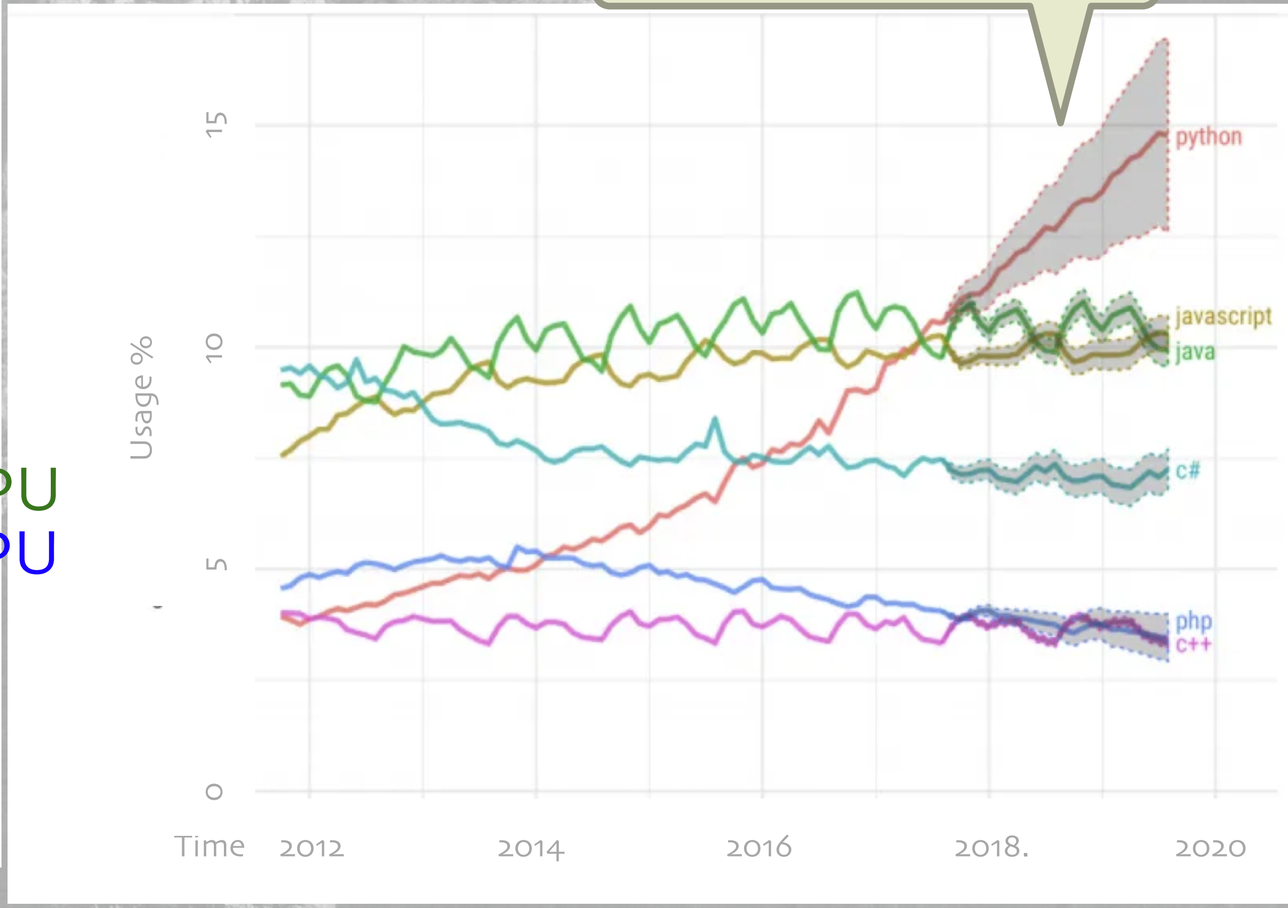
Two Major Trends

Model sizes and therefore compute requirements outstripping Moore's Law

Hence, there is a pressing need for robust, easy to use solutions for distributed Python

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





https://openai.com/blog/ai-and-compute/

The ML Landscape Today

All require distributed implementations to scale

Featurization	Streaming	Hyperparam Tuning	Training	Simulation	Model Serving
Apache Spark	Apache Spark	SIGOPT	HOROVOD	MPI	Clipper
hadoop	kafka	talos AUTONOMY	PyTorch	akka	
			scikit learn	Orleans	

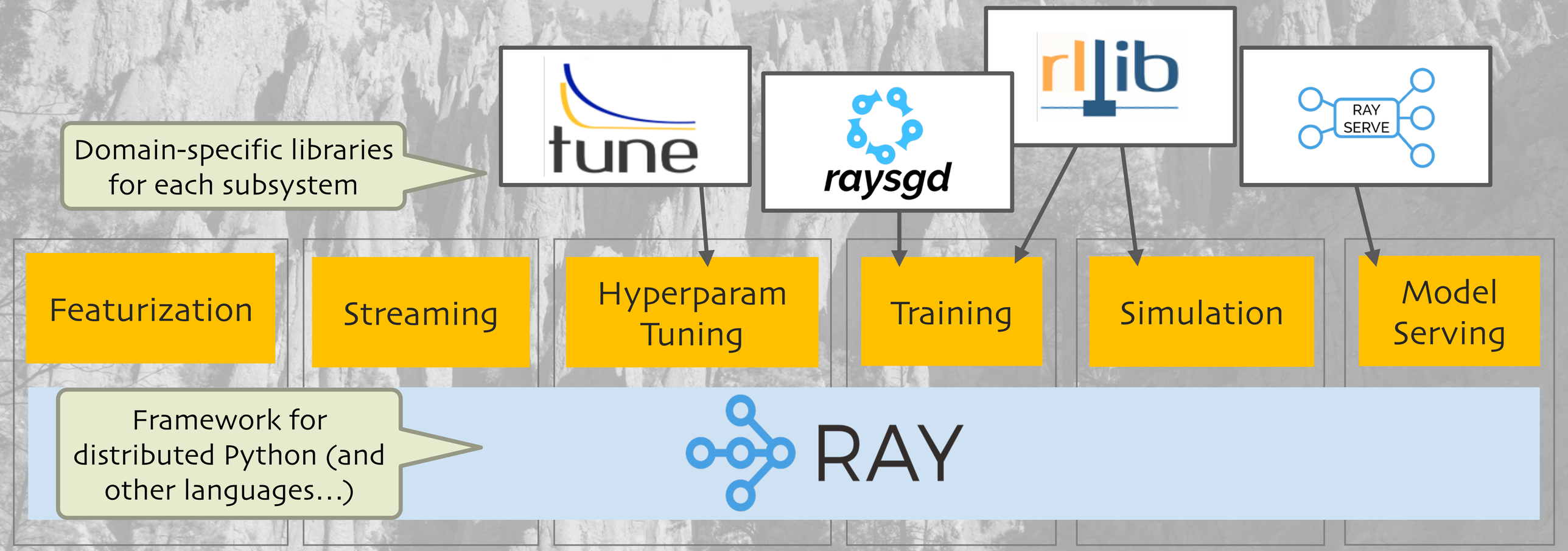
The Ray Vision: Sharing a Common Framework

Featurization | Streaming | Hyperparam Tuning | Training | Simulation | Model Serving

Framework for distributed Python (and other languages…)

RAY

The Ray Vision: Sharing a Common Framework



Plus a growing list of 3rd-party libraries

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

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

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

For completeness, add these first:

```
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(…)
```

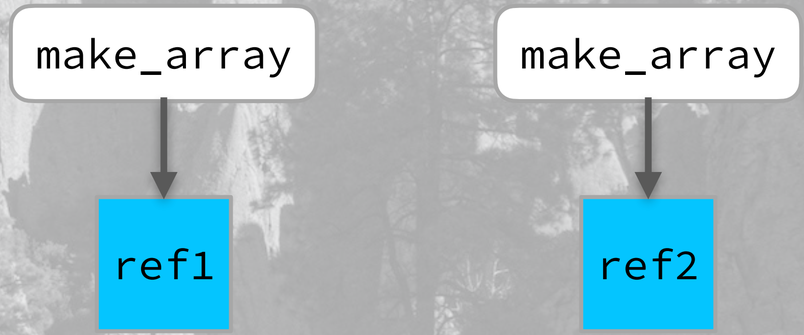
API - Designed to Be Intuitive and Concise

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(…)
```



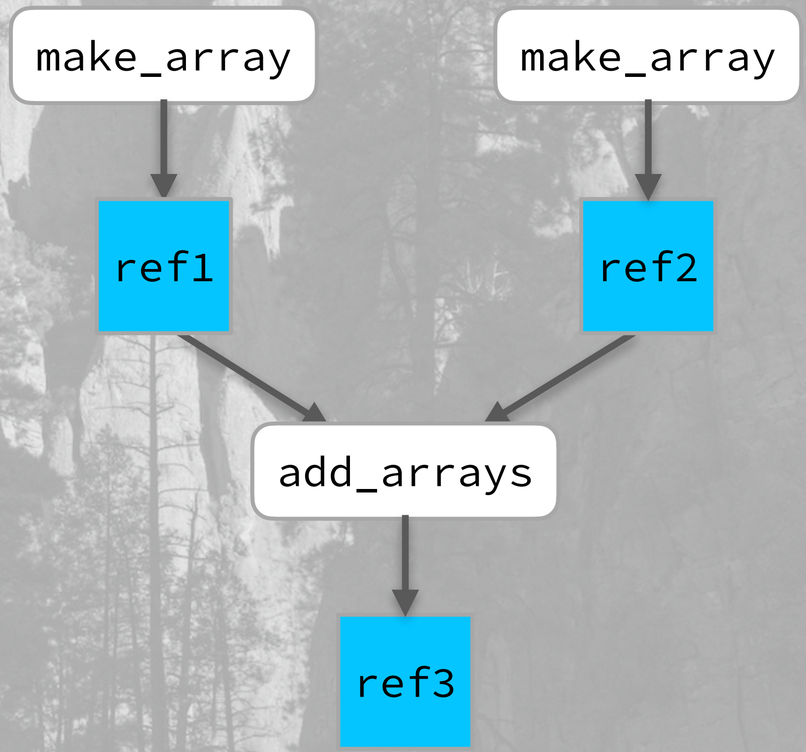
API - Designed to Be Intuitive and Concise

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

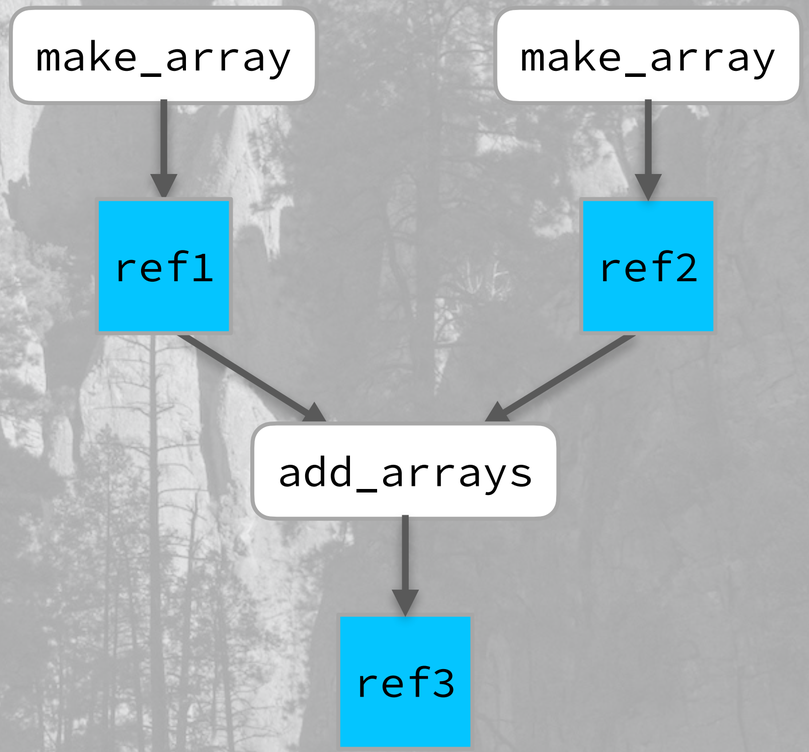
```
@ray.remote
def make_array(…):
    a = … # Construct a NumPy array
    return a

@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 extracting the arrays from the object refs

Ray handles sequencing of async dependencies



API - Designed to Be Intuitive and Concise

What about distributed state?

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)
```

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
```

… now a remote "actor"

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)
```

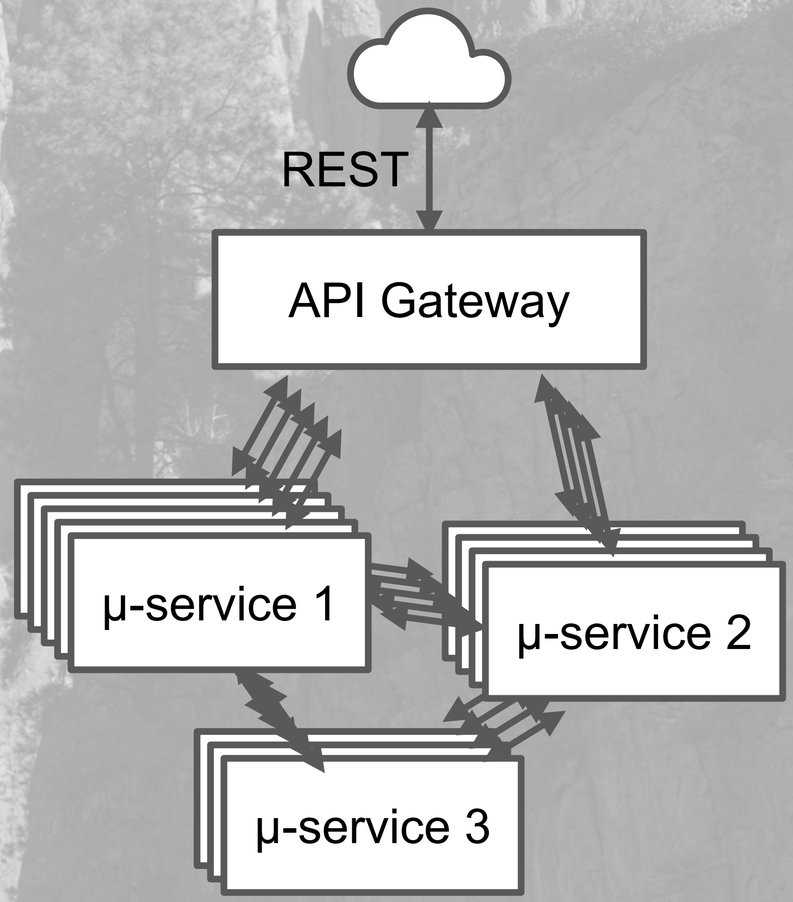
Classes -> Actors

```
@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]
```

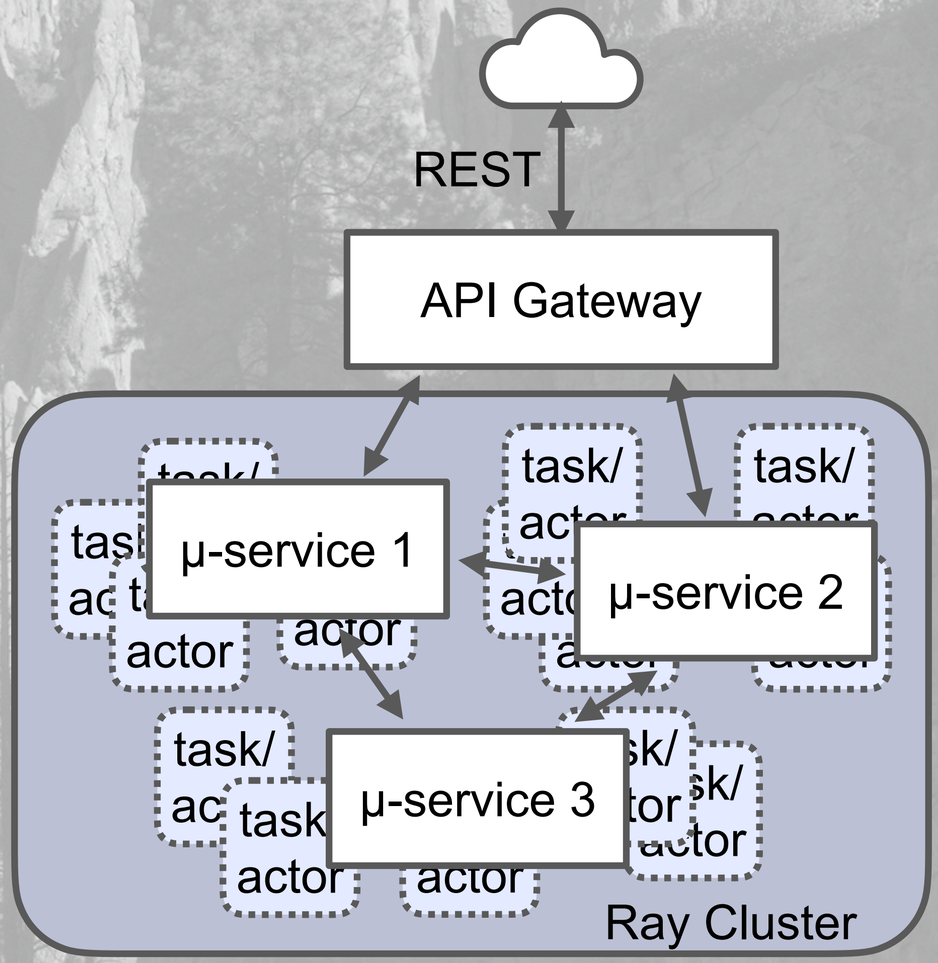
Another View - Distributed ML/AI as Microservices?

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



Microservices - Simplified

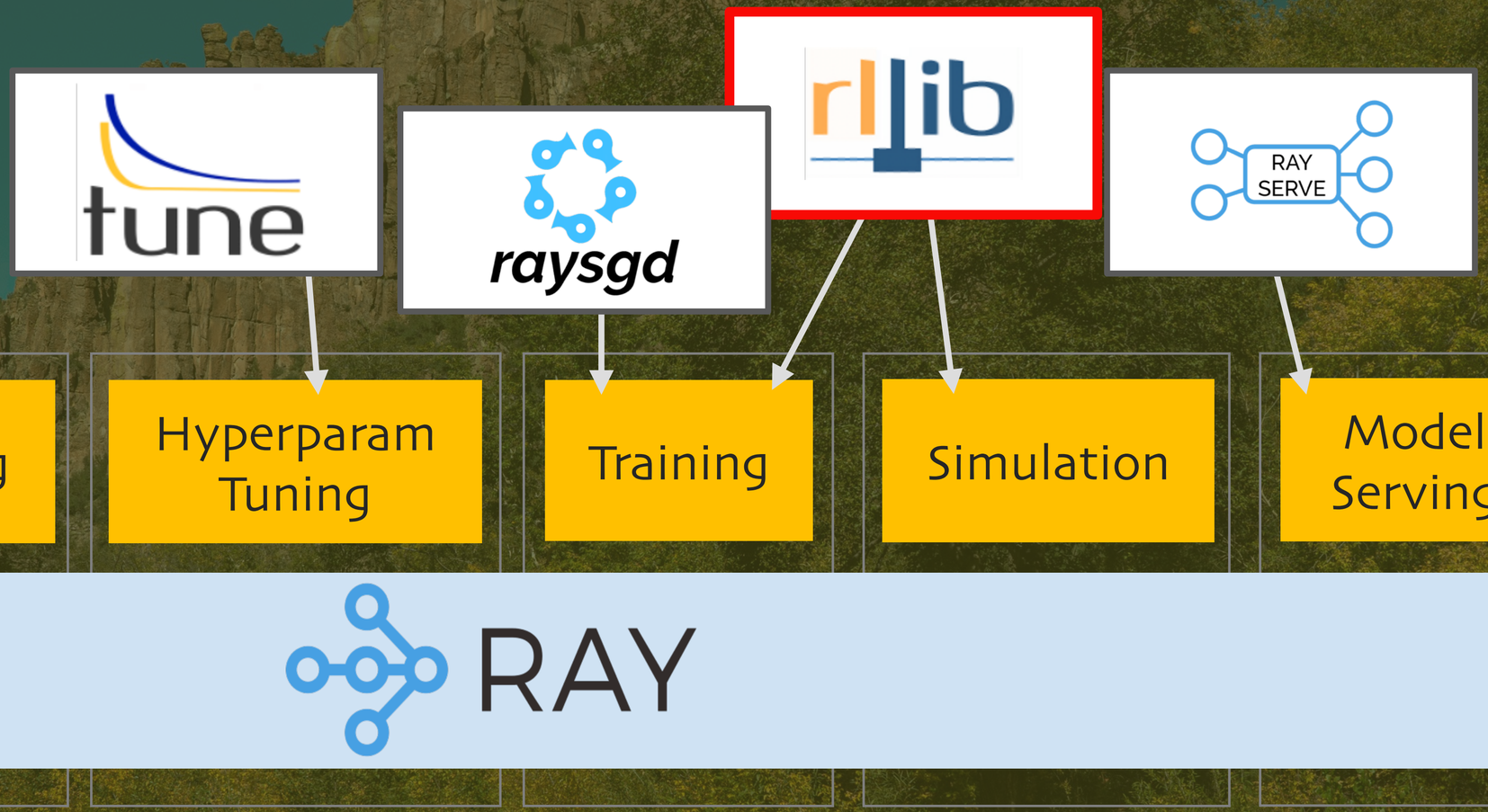
- With Ray, you have one "logical" instance to manage and Ray does the cluster-wide scaling for you.




RAY
Ray RLlib

@deanwampler

Reinforcement Learning - Ray RLlib



rllib.io

RLlib: A Scalable, Unified Library for RL

Games | Robotics, Autonomous Vehicles | Industrial Processes | System Optimization | Advertising, Recommendations | Finance

RL applications

OpenAI Gym | Multi-agent/ Hierarchical | Policy Serving | Offline Data

} (1) Application Support

Custom Algorithms | RLlib Algorithms

} (2) Abstractions for RL

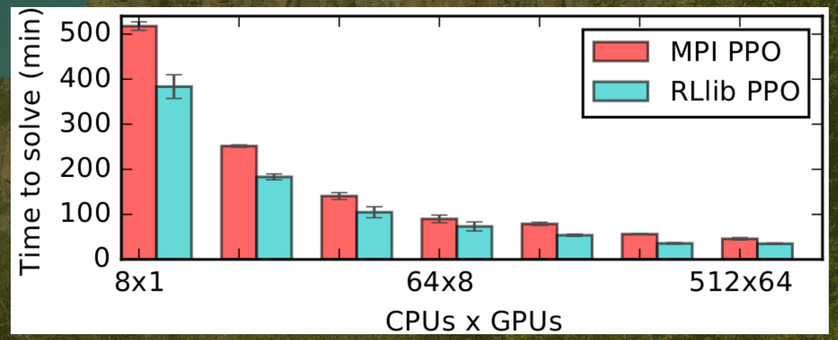
RLlib Abstractions

Ray Tasks and Actors

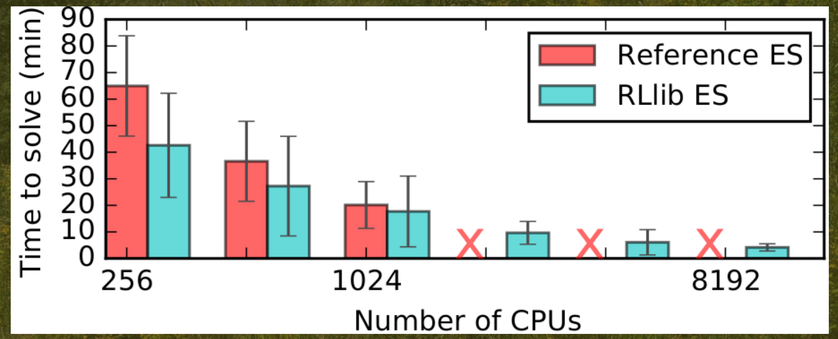
} (3) Distributed Execution

RLlib Provides a Unified Framework for Scalable RL that Doesn't Compromise on Performance

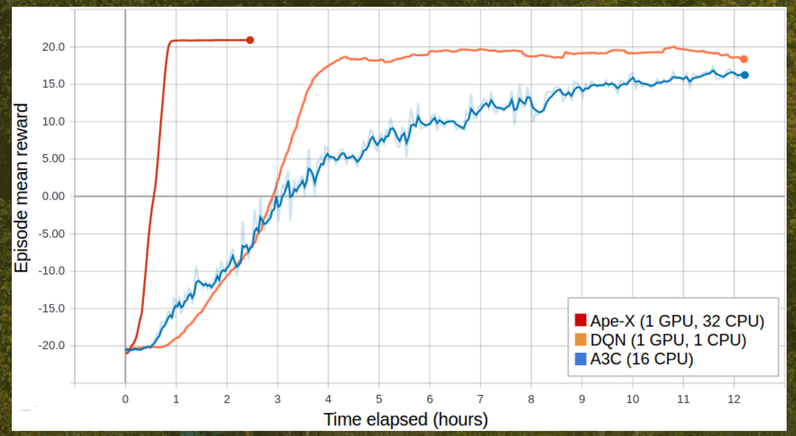
Distributed PPO



Evolution Strategies



Ape-X Distributed DQN, DDPG



A Broad Range of Popular Algorithms

- 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)

Amazon SageMaker RL

End-to-end examples for classic RL and real-world RL applications

Robotics | Industrial Control | HVAC | Autonomous Vehicles | Operations | Finance | Games | NLP

RL Environments to model real-world problems

- **AWS Simulation Environments**: Amazon Sumerian | AWS RoboMaker
- **Open Source Environments**: EnergyPlus | RoboSchool | PyBullet | ...
- **Custom Environments**: Bring Your Own
- **Commercial simulators**: MATLAB & Simulink

Open AI Gym

RL Toolkits that provide RL agent algorithm implementations

- **RL-Coach**: DQN | PPO | HER | Rainbow | ...
- **RL-Ray RLLib**: APEX | ES | IMPALA | A3C | ...
- **Open AI Baselines**: TRPO | GAIL | ...
- ...

SageMaker Deep Learning Frameworks

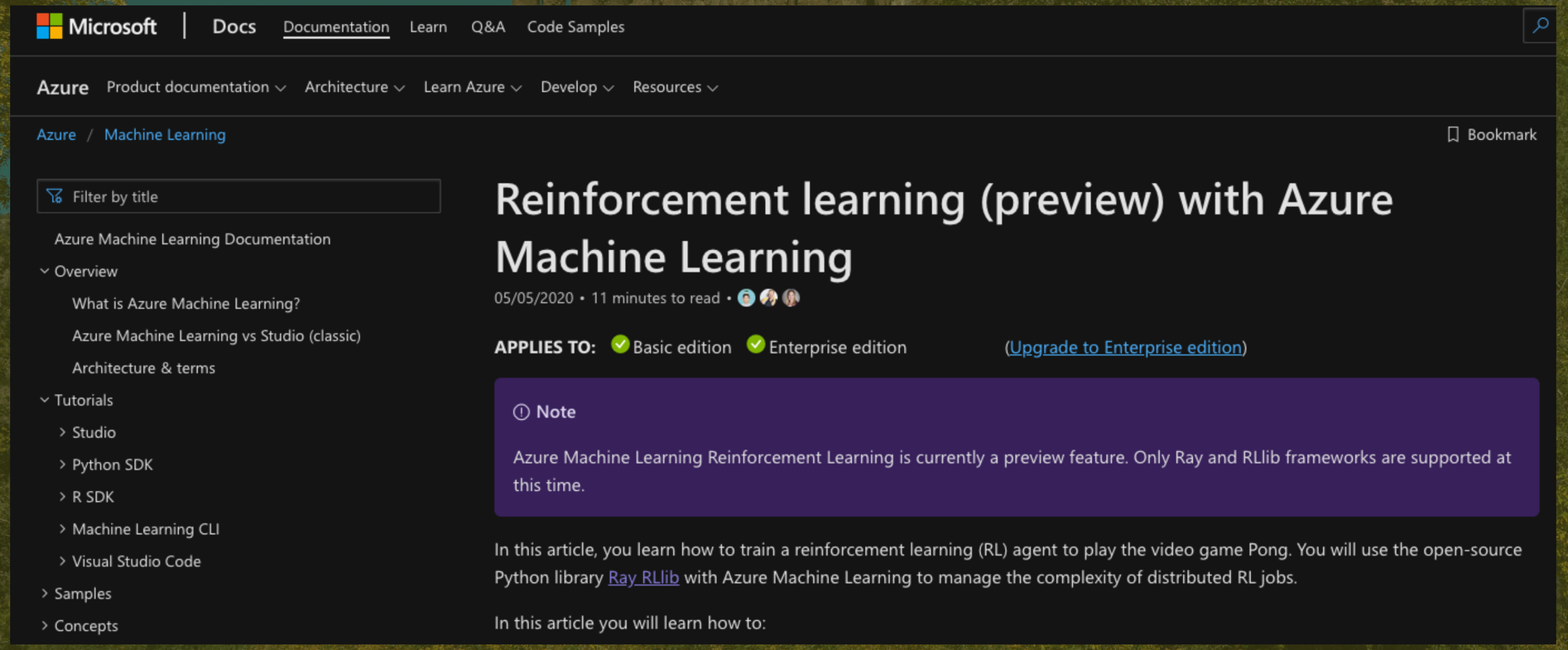
TensorFlow | MxNet | PyTorch | Chainer

Training Options

Single Machine / Distributed | Local / Remote simulation | CPU / GPU Hardware

SageMaker supported | Customer BYO

Now in Azure


Microsoft | Docs Documentation Learn Q&A Code Samples
Azure Product documentation Architecture Learn Azure Develop Resources
Azure / Machine Learning
Bookmark
Filter by title
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:

@deanwampler

Hyperparameter Tuning - Ray Tune

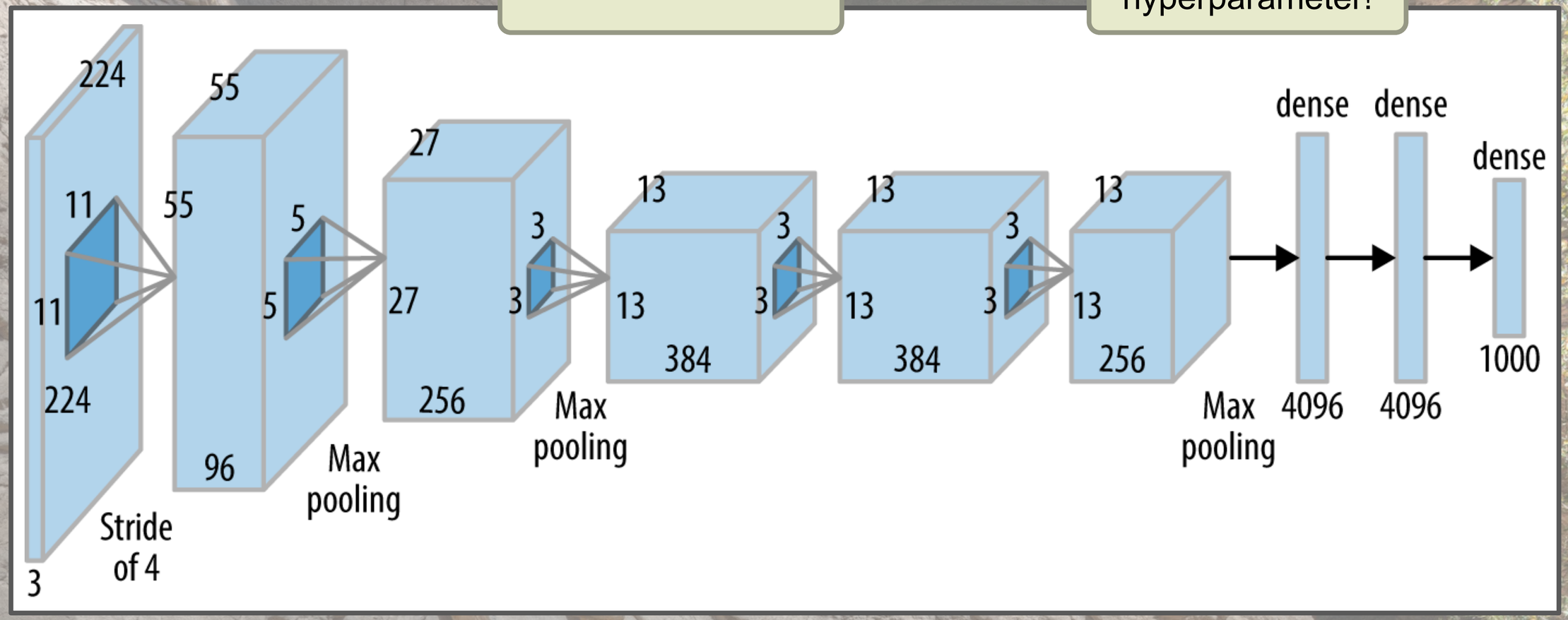
Featurization | Streaming | Hyperparam Tuning | Training | Simulation | Model Serving

RAY

Nontrivial Example - Neural Networks

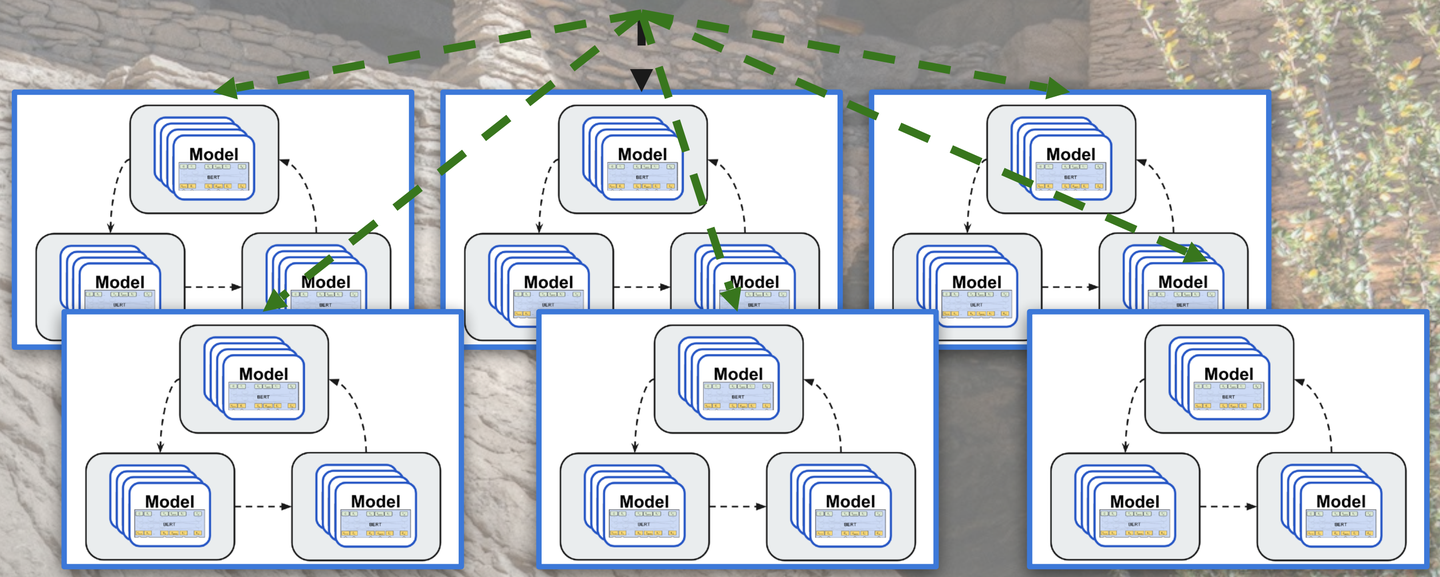
How many layers?
What kinds of layers?

Every number shown is a hyperparameter!



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()
)
```




RAY
Next Steps:
Adopting Ray
@deanwampler

@deanwampler

Ray Community and Resources

- ray.io
  - Ray Slack: ray-distributed.slack.com
- Tutorials: anyscale.com/academy


Thank You
ray.io
dean@deanwampler.com
@deanwampler
polyglotprogramming.com/talks
raysummit.org
anyscale.com/events
RAY SUMMIT
Presented by Anyscale
September 30–October 1
RAY