Reinforcement Learning: ChatGPT, Games, and More

Slides:
deanwampler.com/talks

Topics

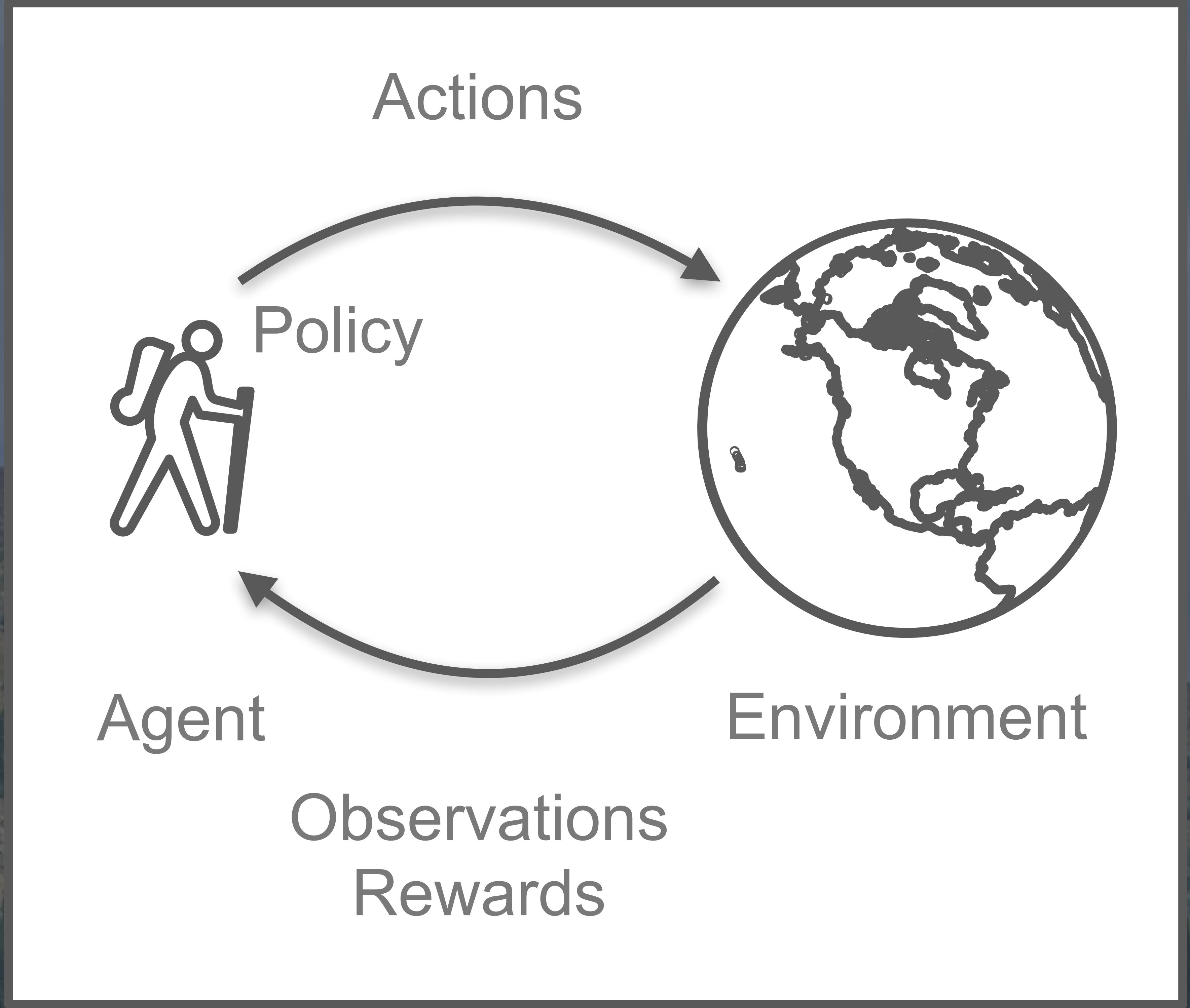
- Why Reinforcement Learning? What Is It? How is it used?
- Ray RLlib, a popular RL system built with Ray.
- More Reinforcement Learning Concepts and Challenges
- Reinforcement Learning and ChatGPT
- Reinforcement Learning for Recommendations
- To Learn More…

Why Reinforcement Learning?

The Agent choses an Action, then Observes any state changes in the Environment and a Reward received, if any.

Through a sequence of these steps, the Agent learns a Policy for picking Actions that maximize the cumulative Reward.

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



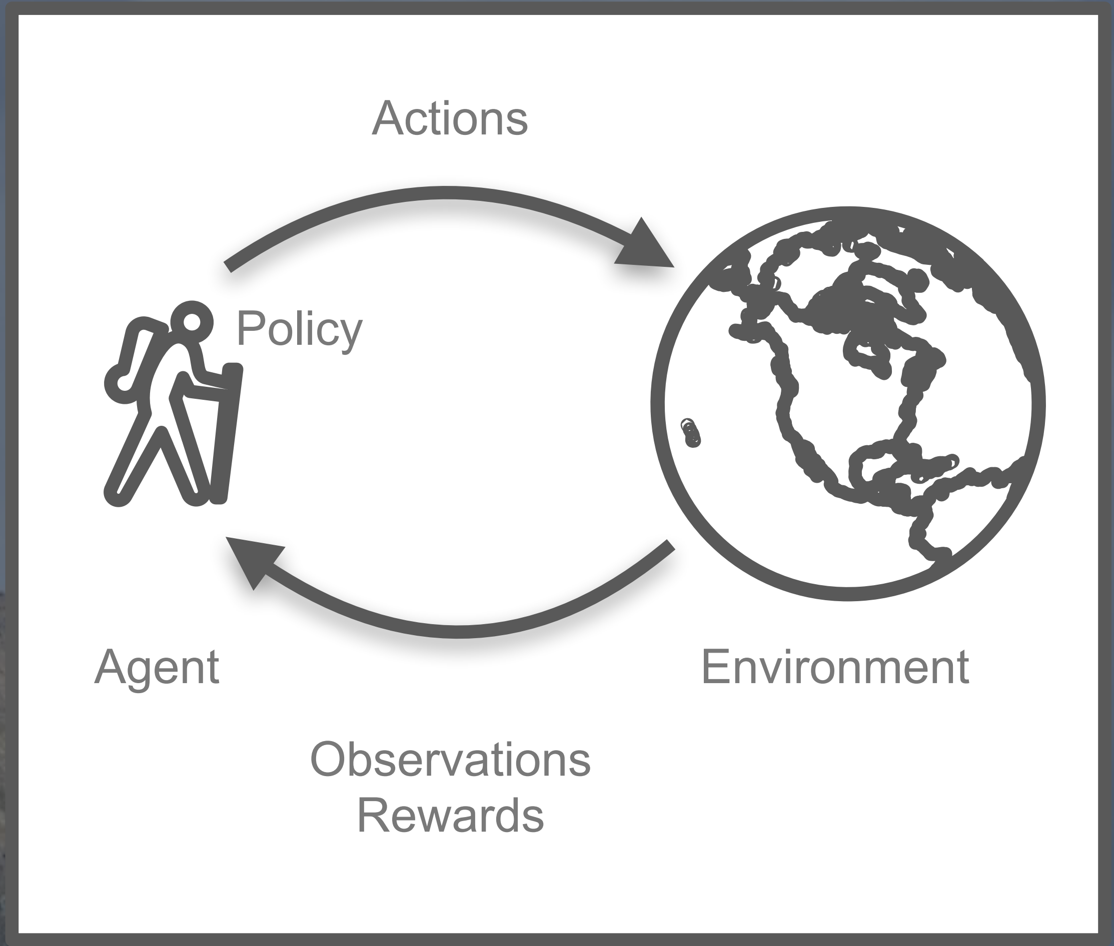
Some systems return a Reward after each Action. Others, only at the Episode end.

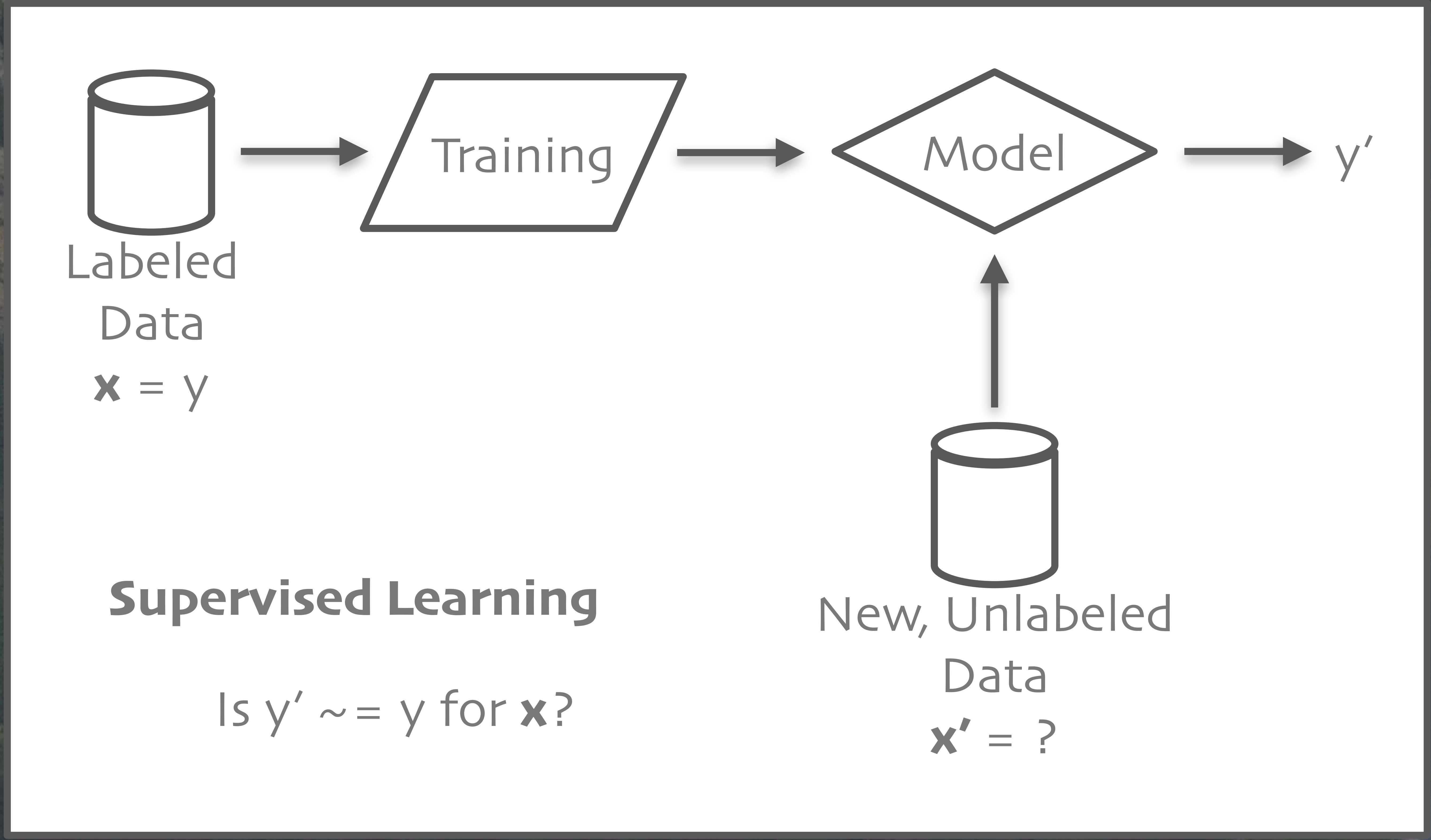
Why Reinforcement Learning?

https://www.youtube.com/watch?v=Lu56xVlZ40M

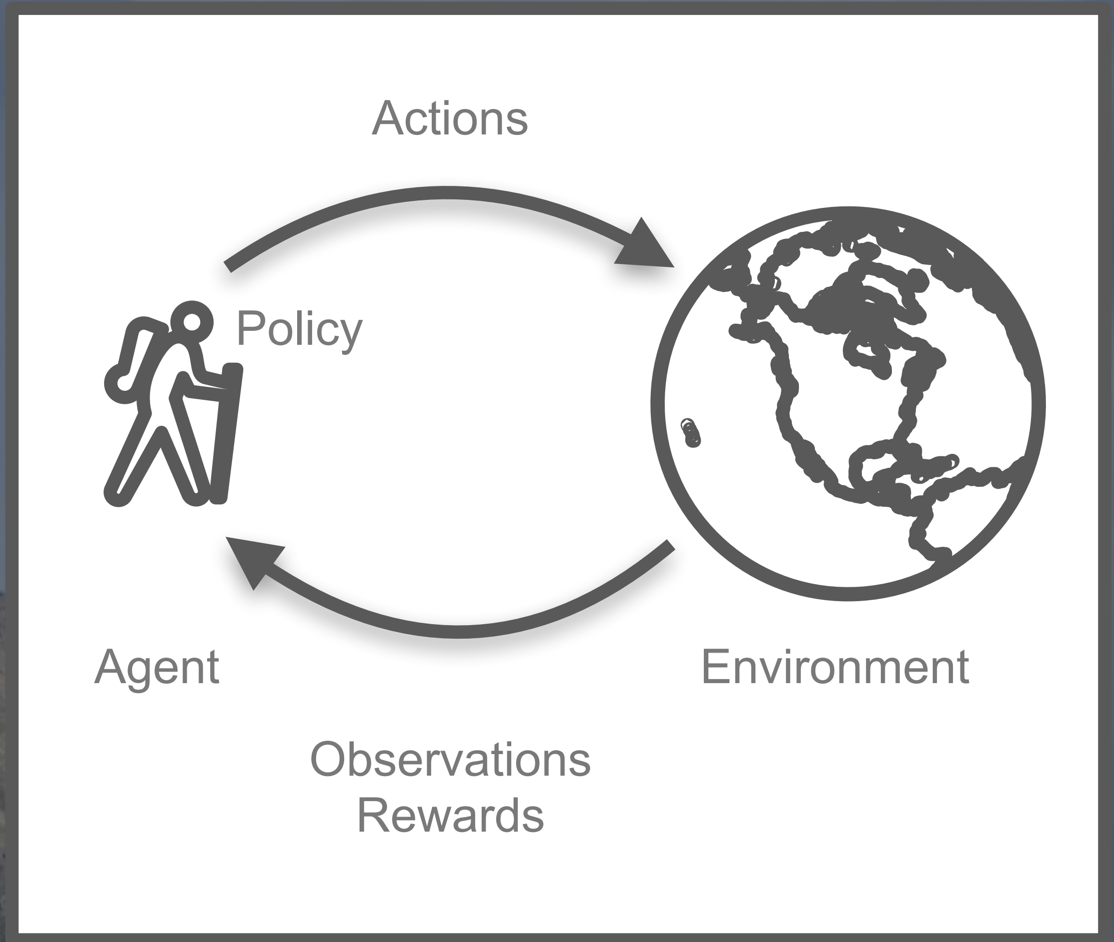

TWO MINUTE
PAPERS
WITH KÁROLY ZSOLNAI-FEHÉR

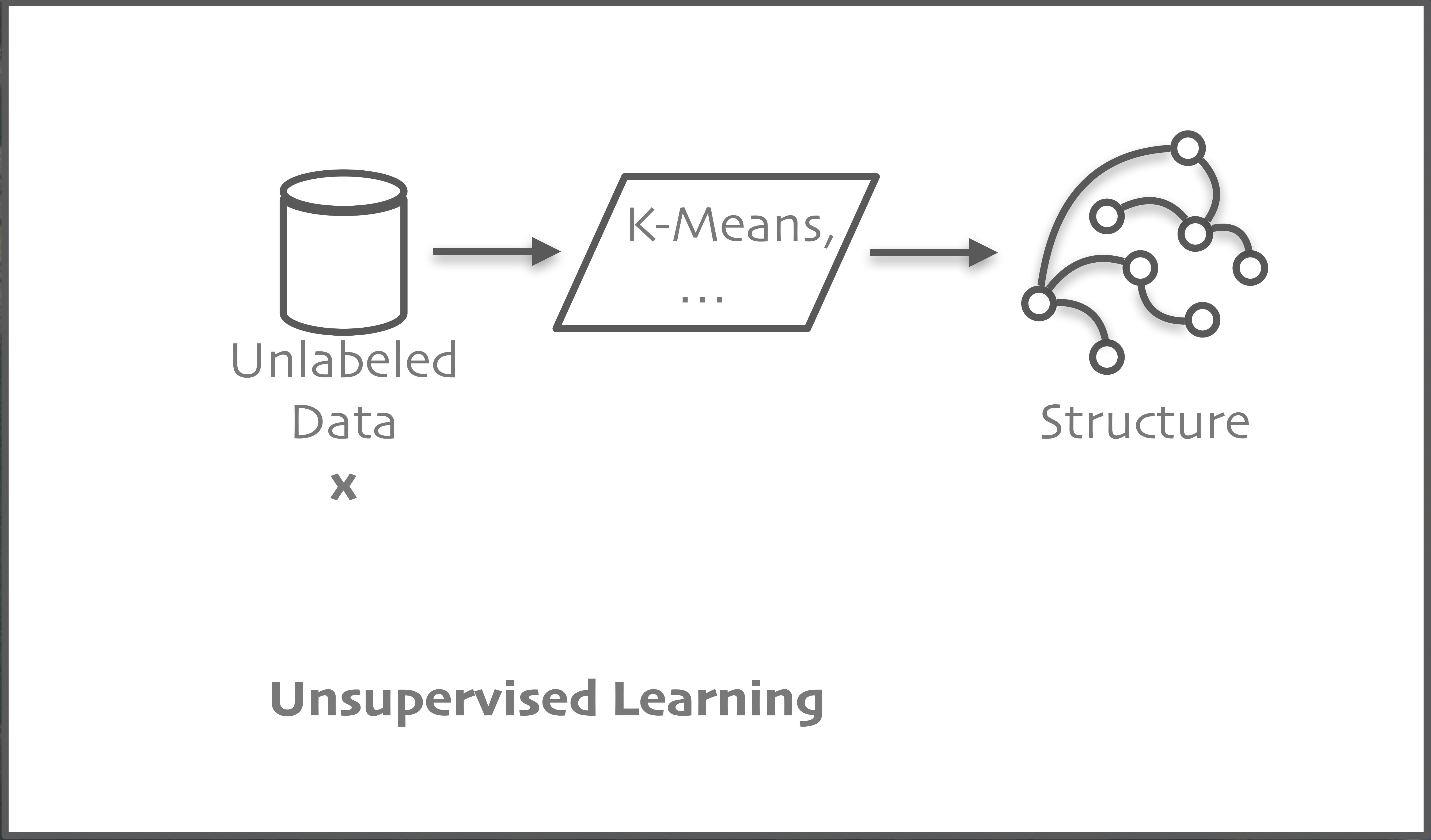
Compared to Supervised Learning



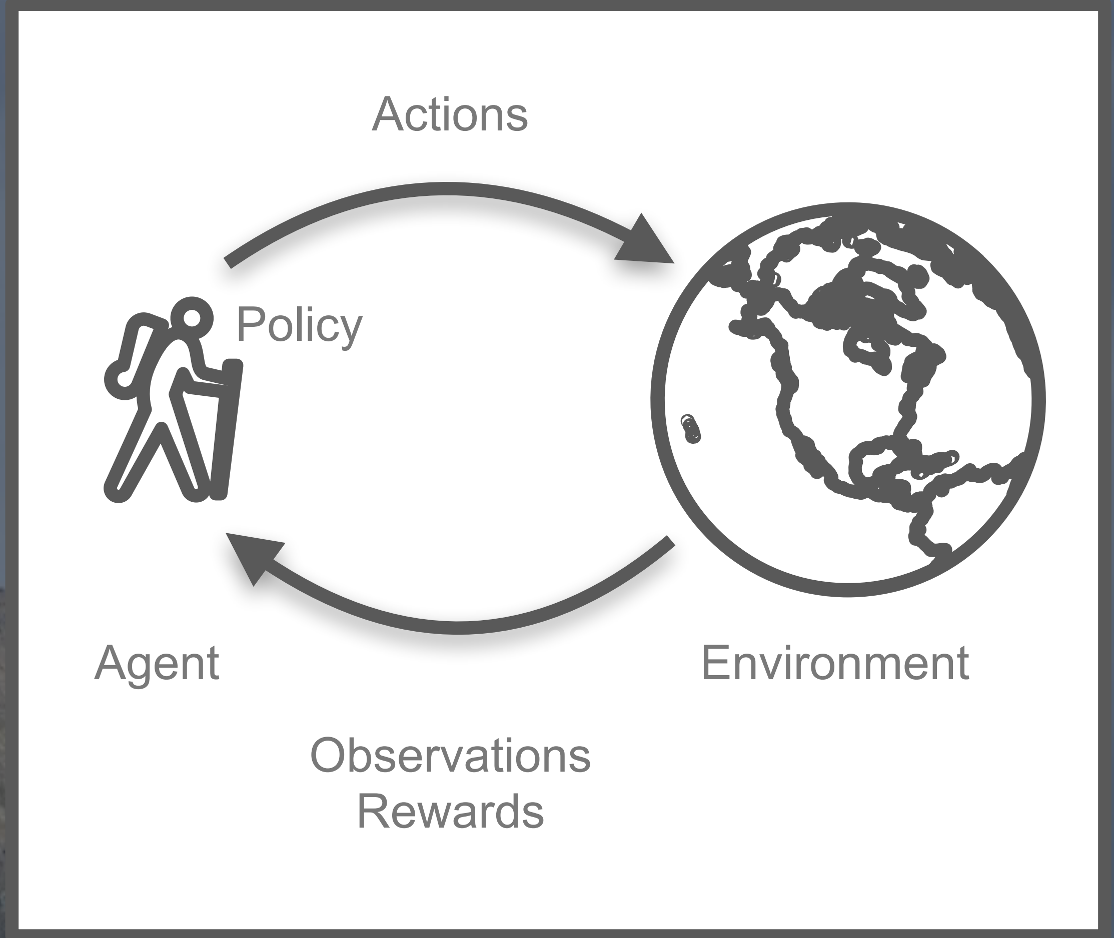


Compared to Unsupervised Learning





RL Applications



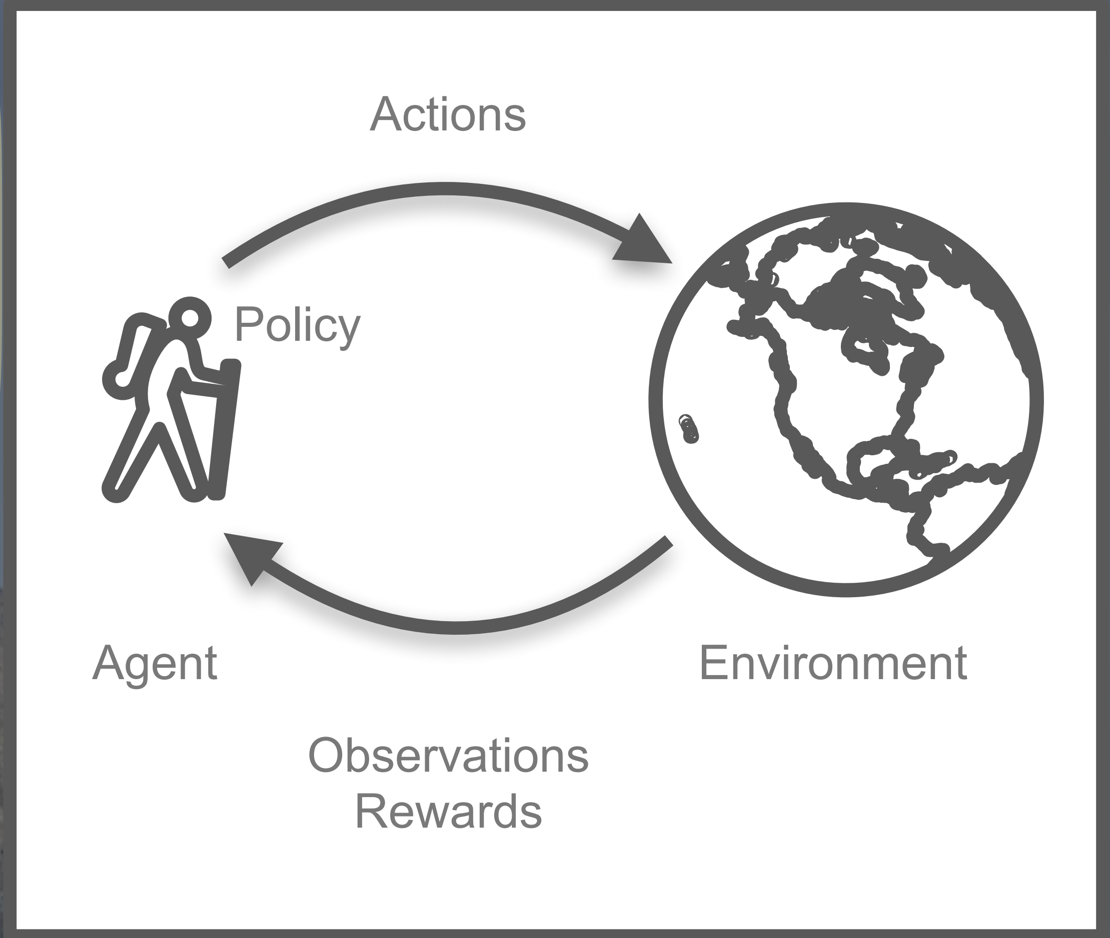
Games	Robotics, Autonomous Vehicles	Industrial Processes	System Optimization	Advertising, Recommendations	Finance

Common Theme:

The ideal applications have sequential, evolving state for the environment and the agent.

RL Applications

AlphaGo, Atari, OpenAI Gym/ Gymnasium, …



Games

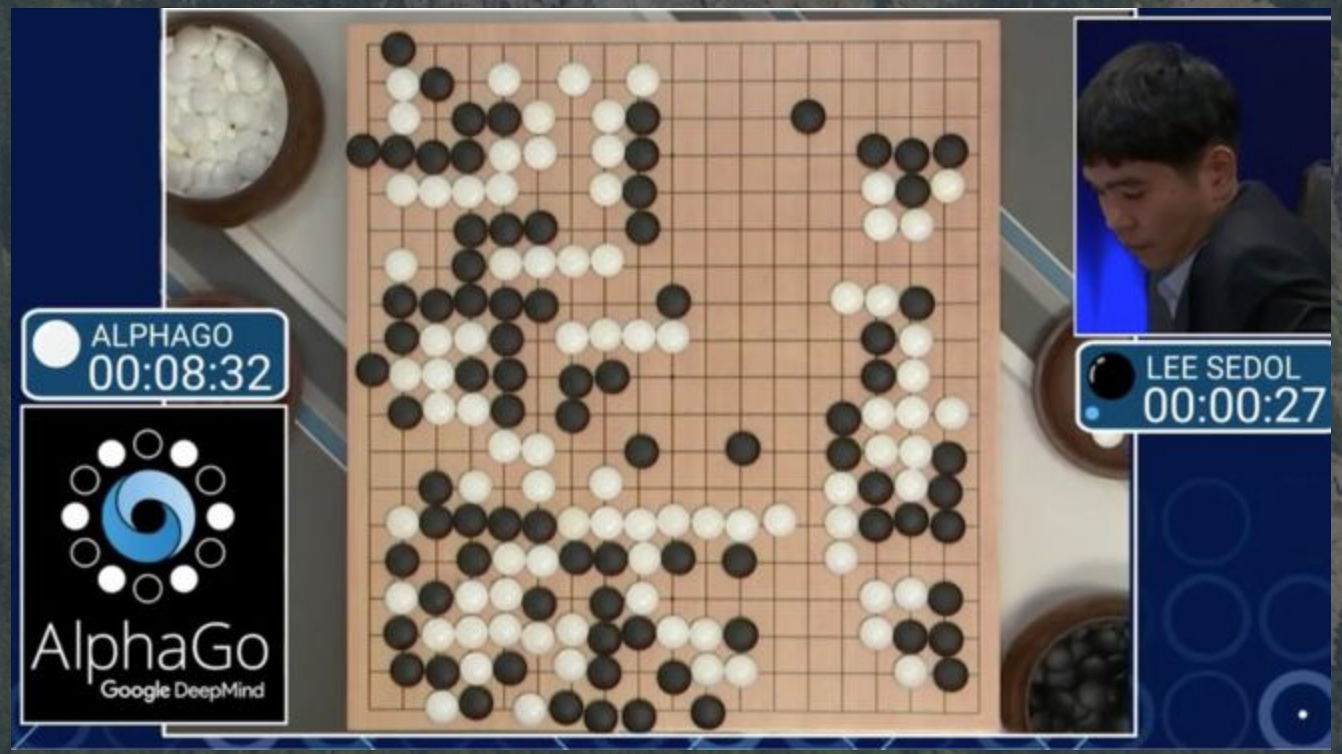
Robotics, Autonomous Vehicles

Industrial Processes

System Optimization

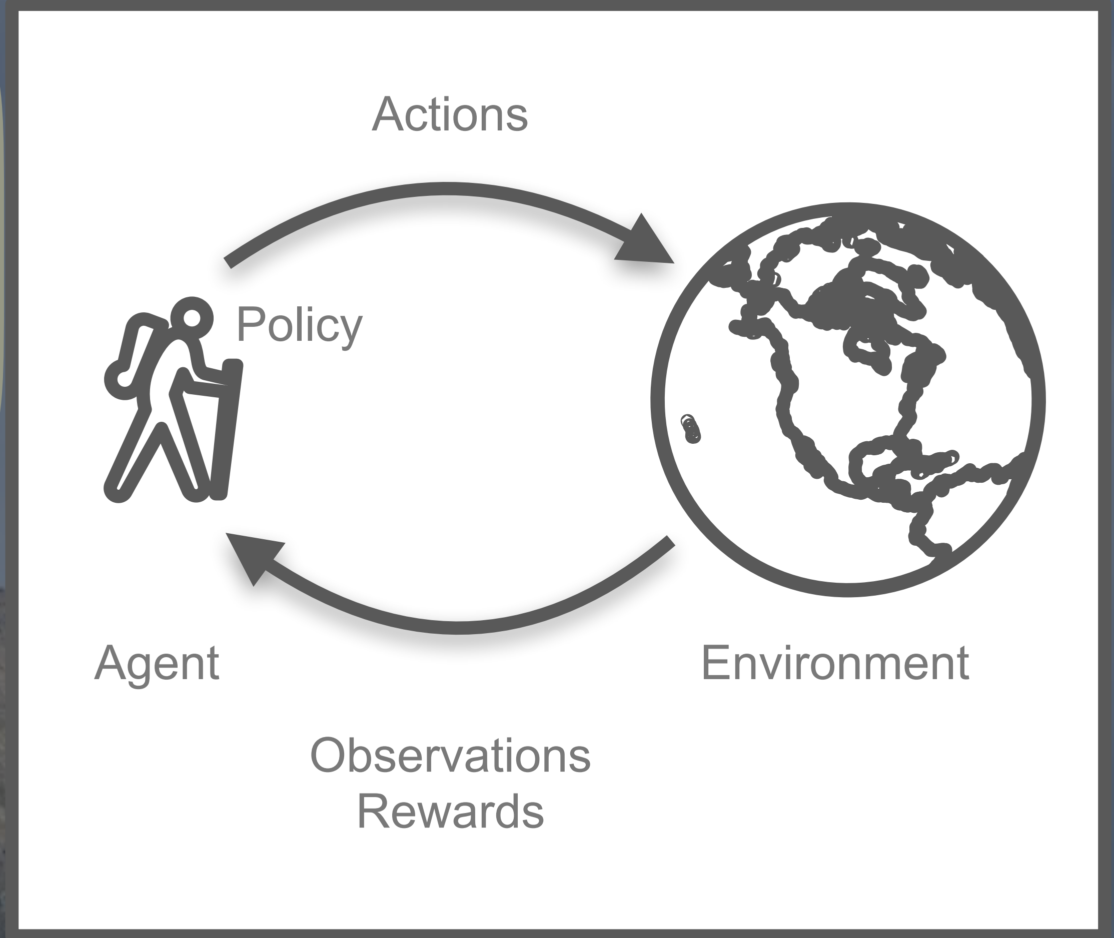
Advertising, Recommendations

Finance

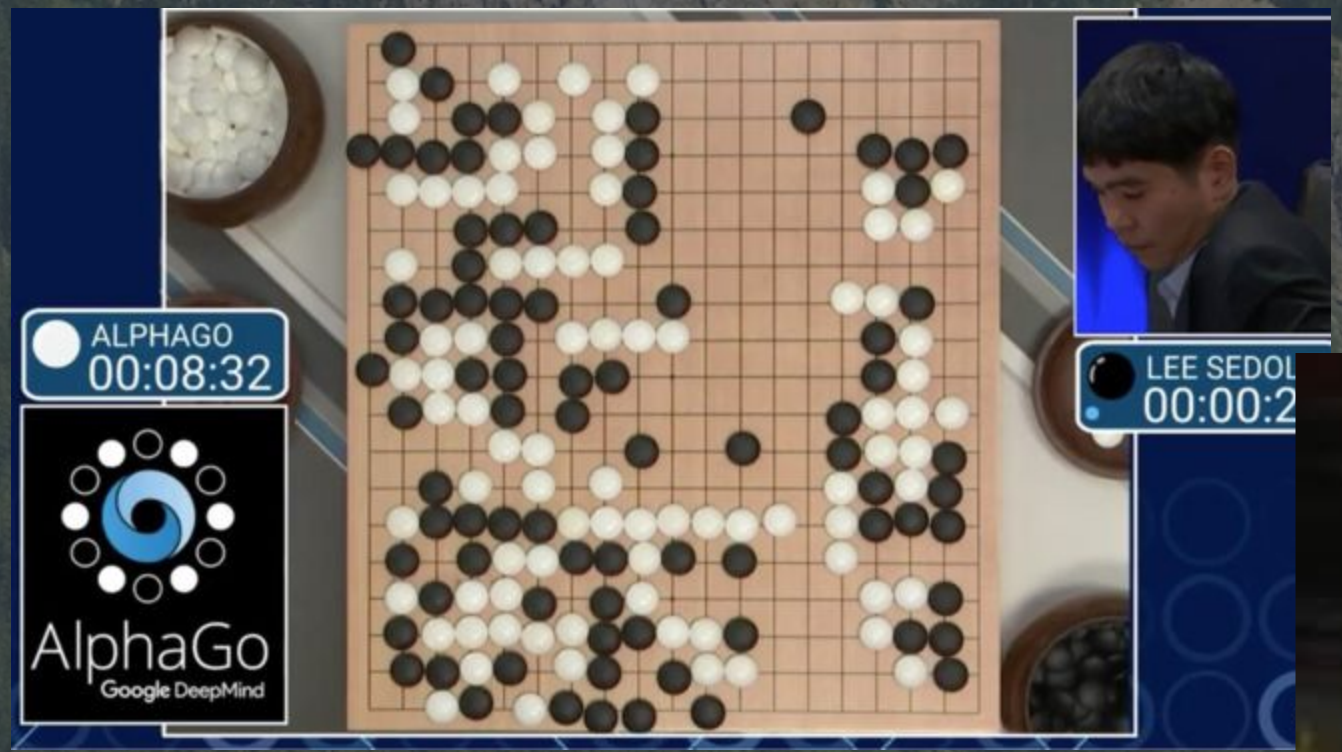


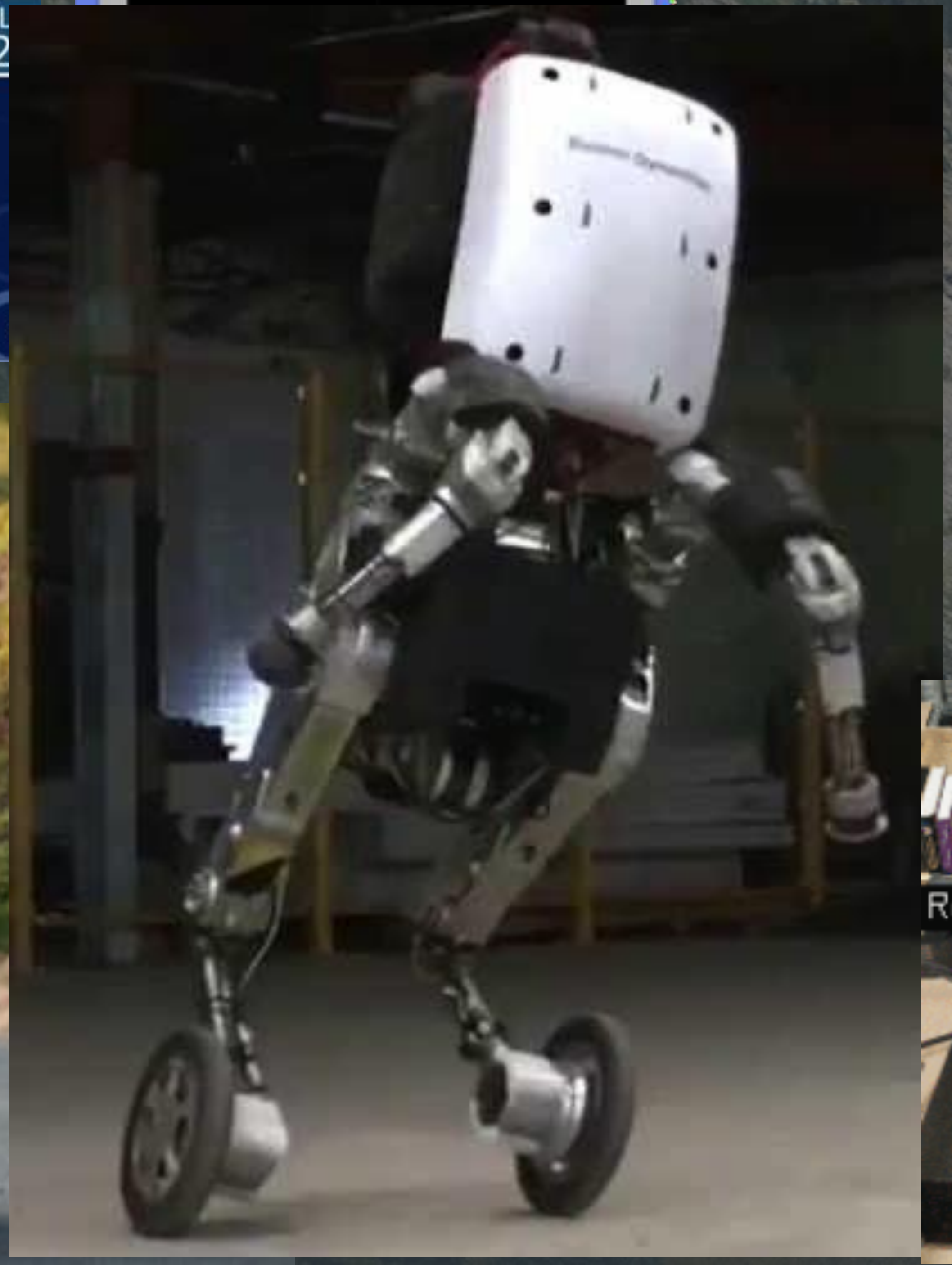
RL Applications

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



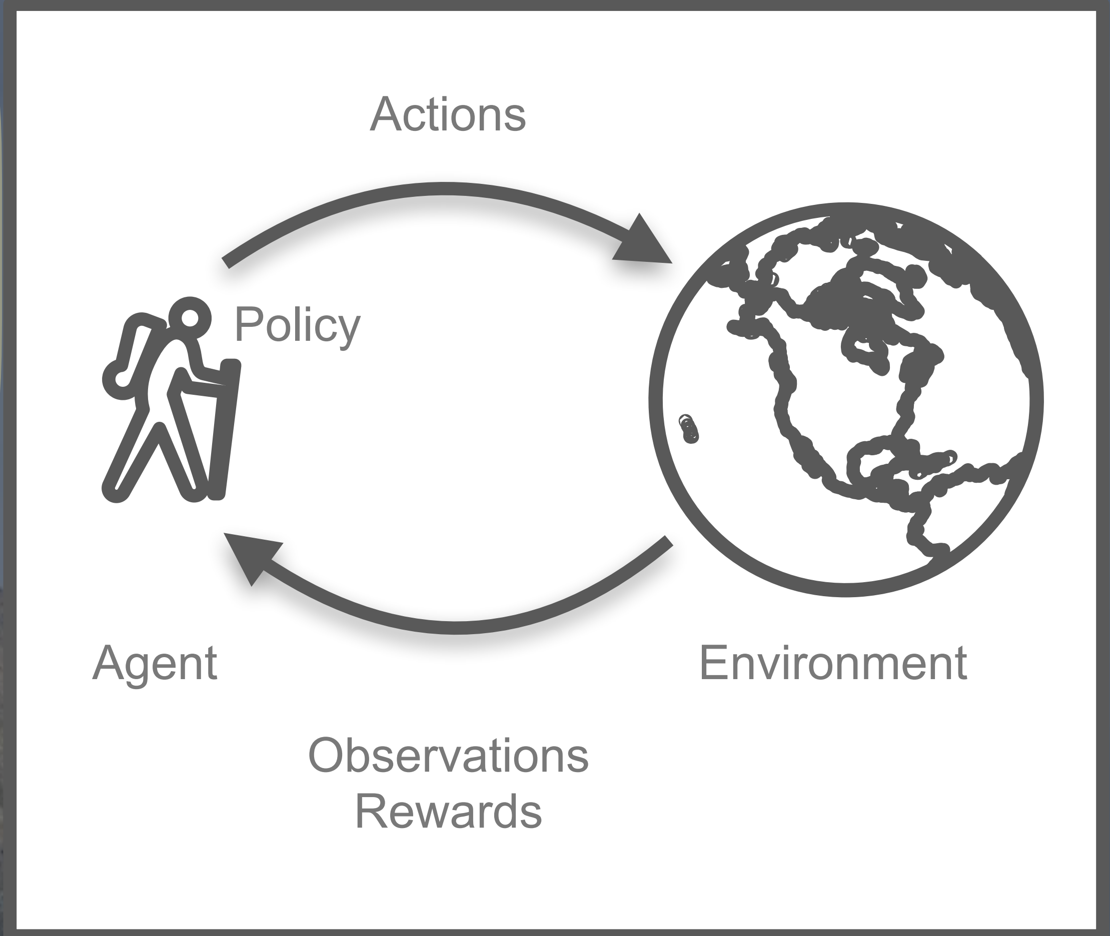
Games	Robotics, Autonomous Vehicles	Industrial Processes	System Optimization	Advertising, Recommendations	Finance





RL Applications

Assembly lines, warehouse and delivery routing, …



Games

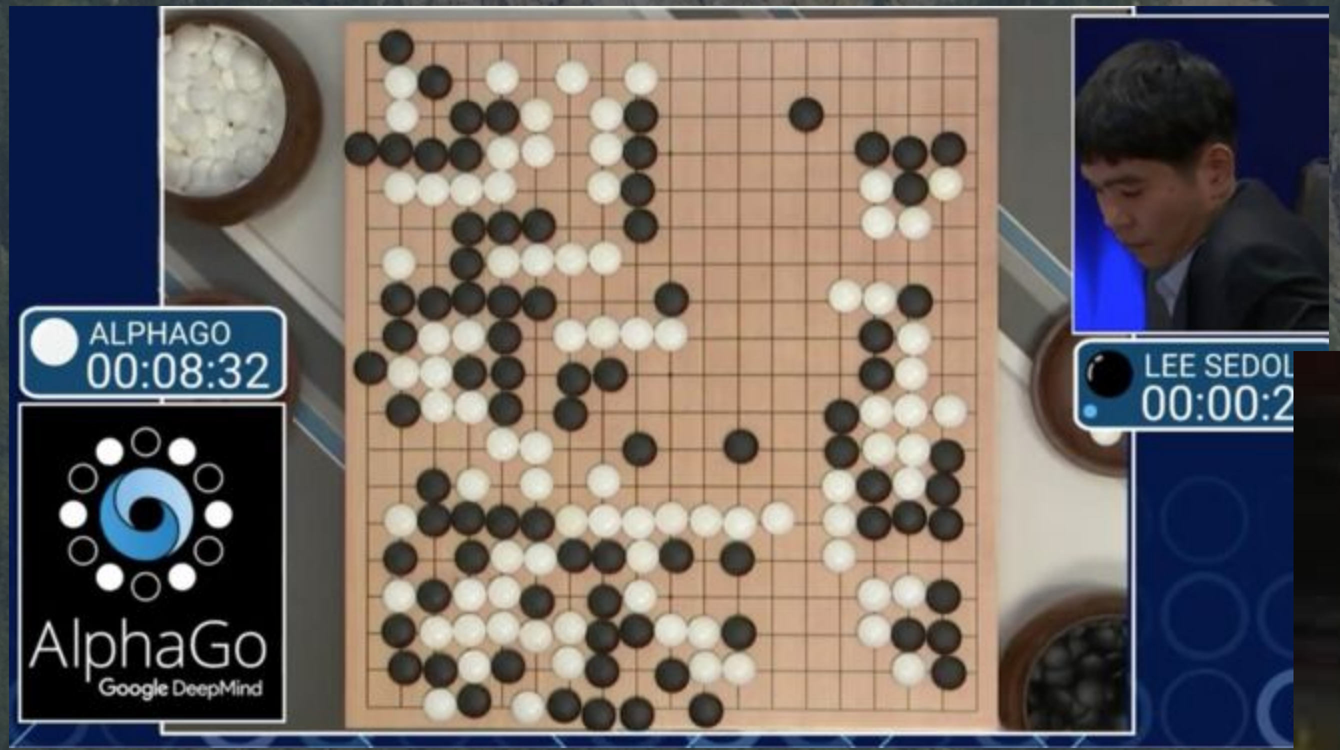
Robotics, Autonomous Vehicles

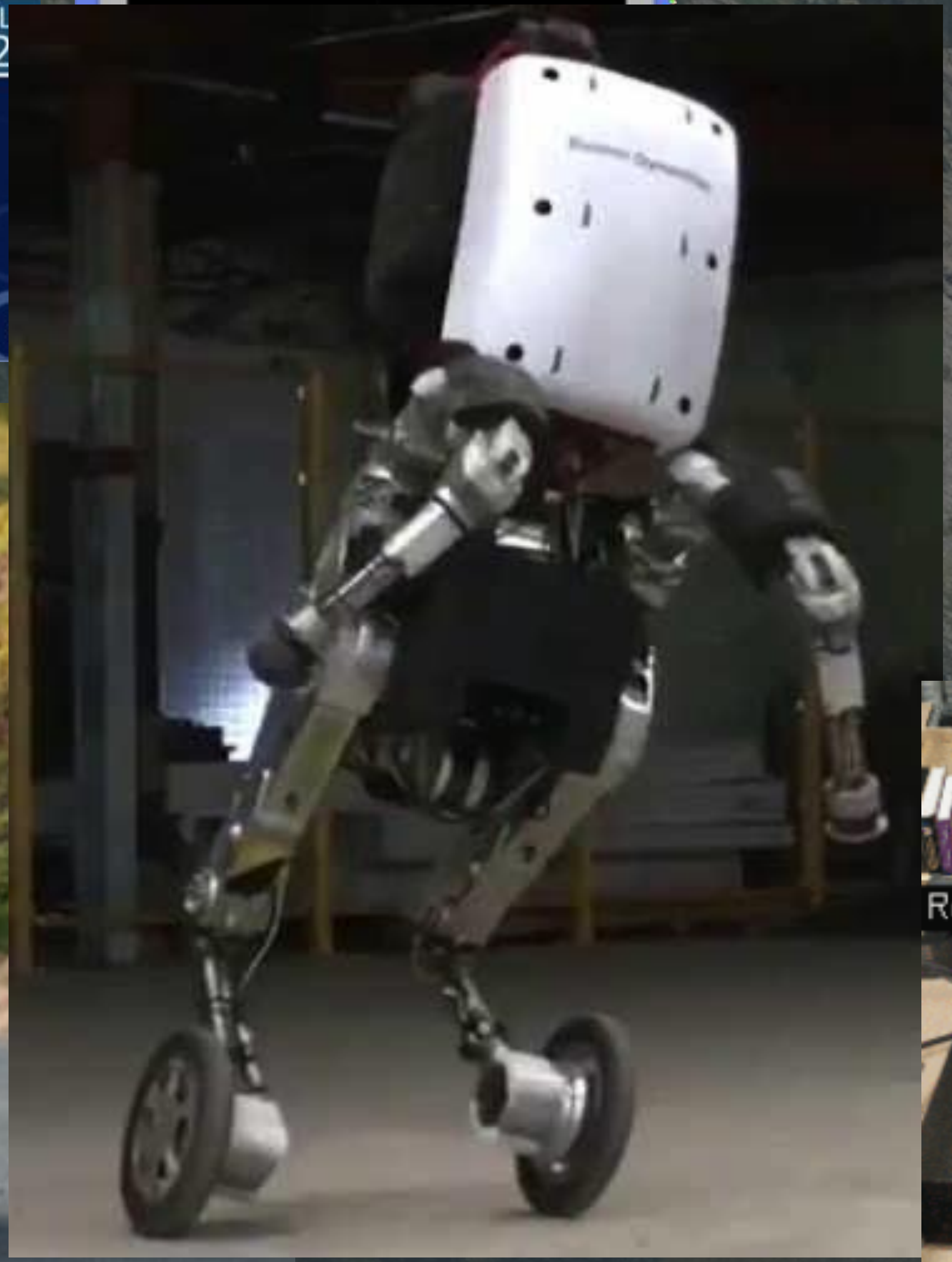
Industrial Processes

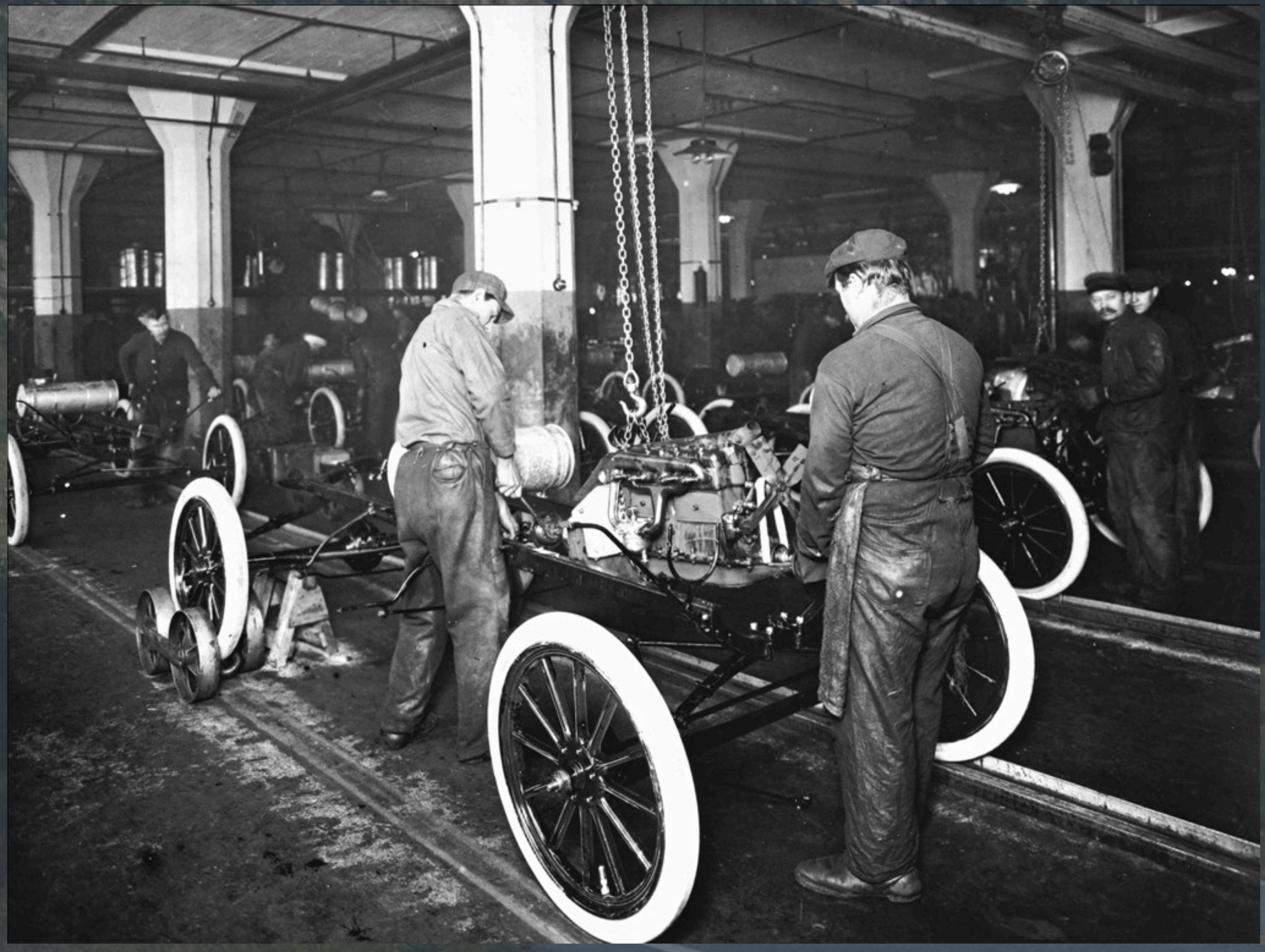
System Optimization

Advertising, Recommendations

Finance

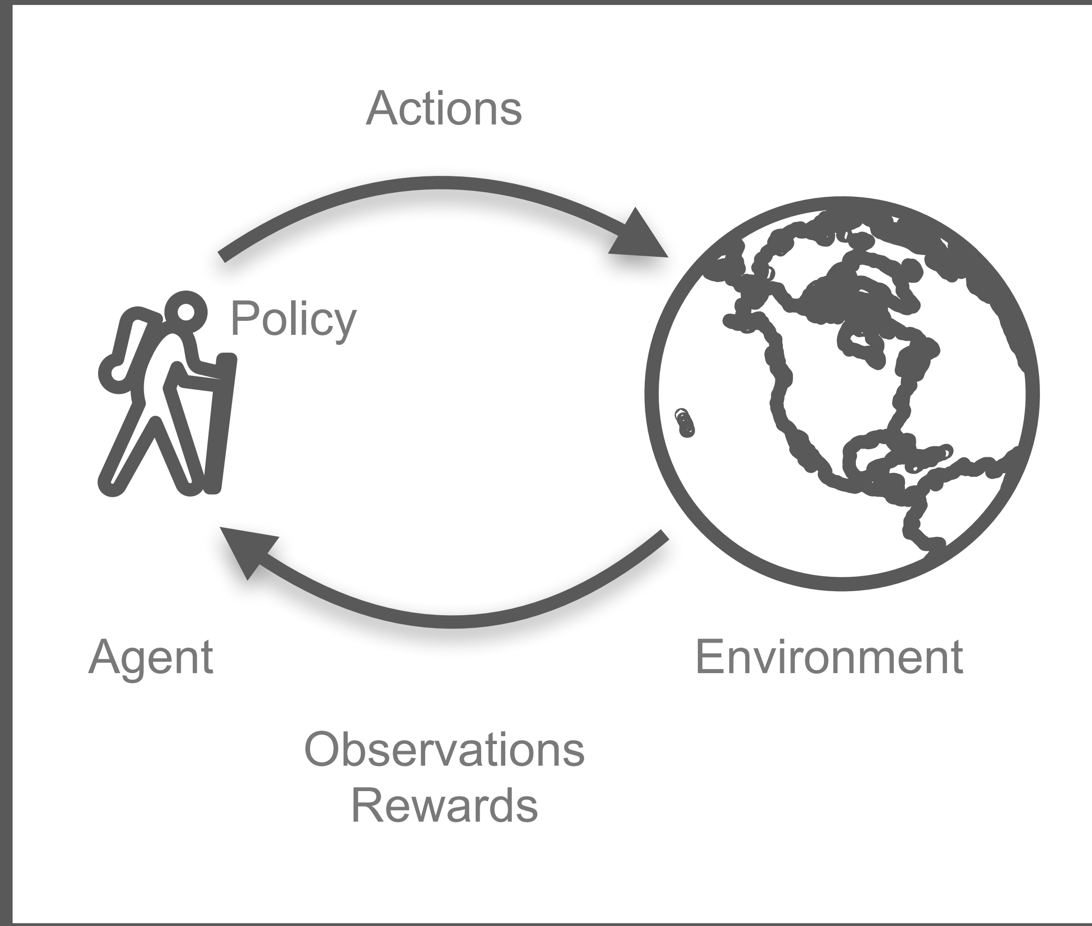






RL Applications

HVAC optimization, networks, business processes, …



Games

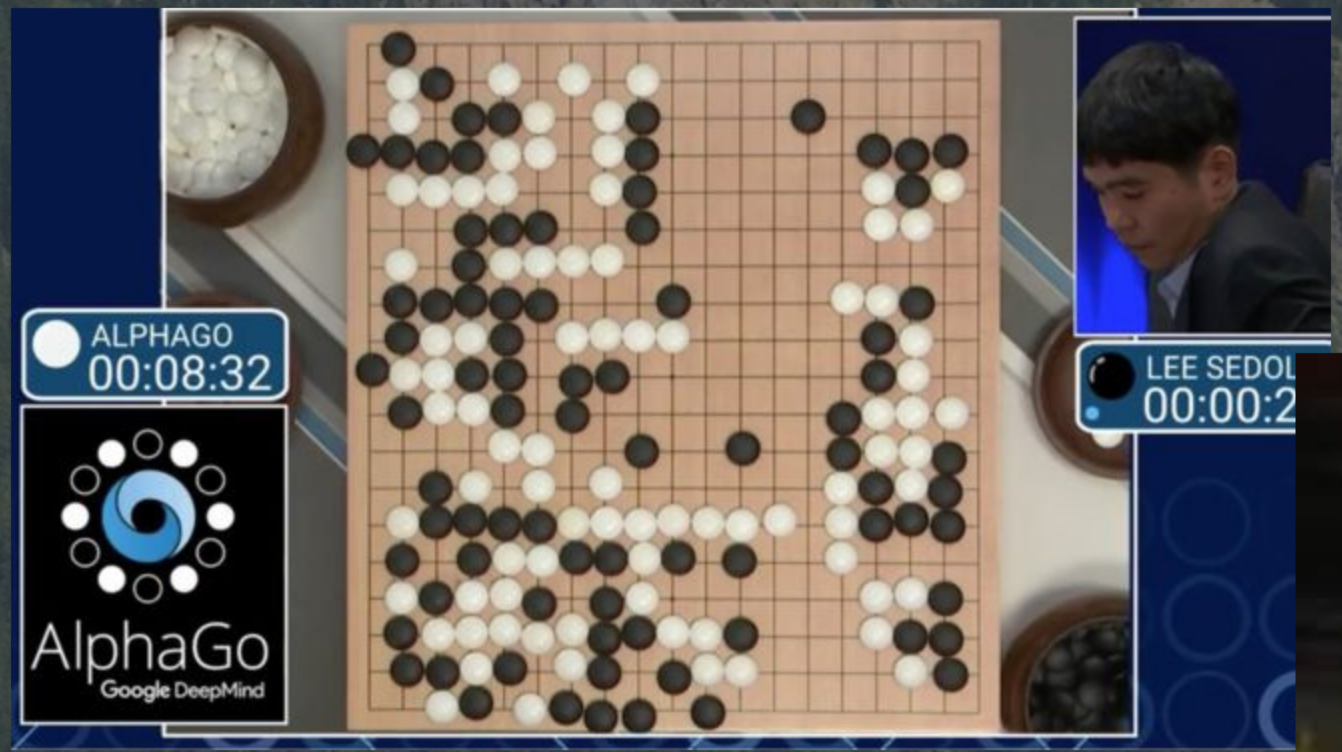
Robotics, Autonomous Vehicles

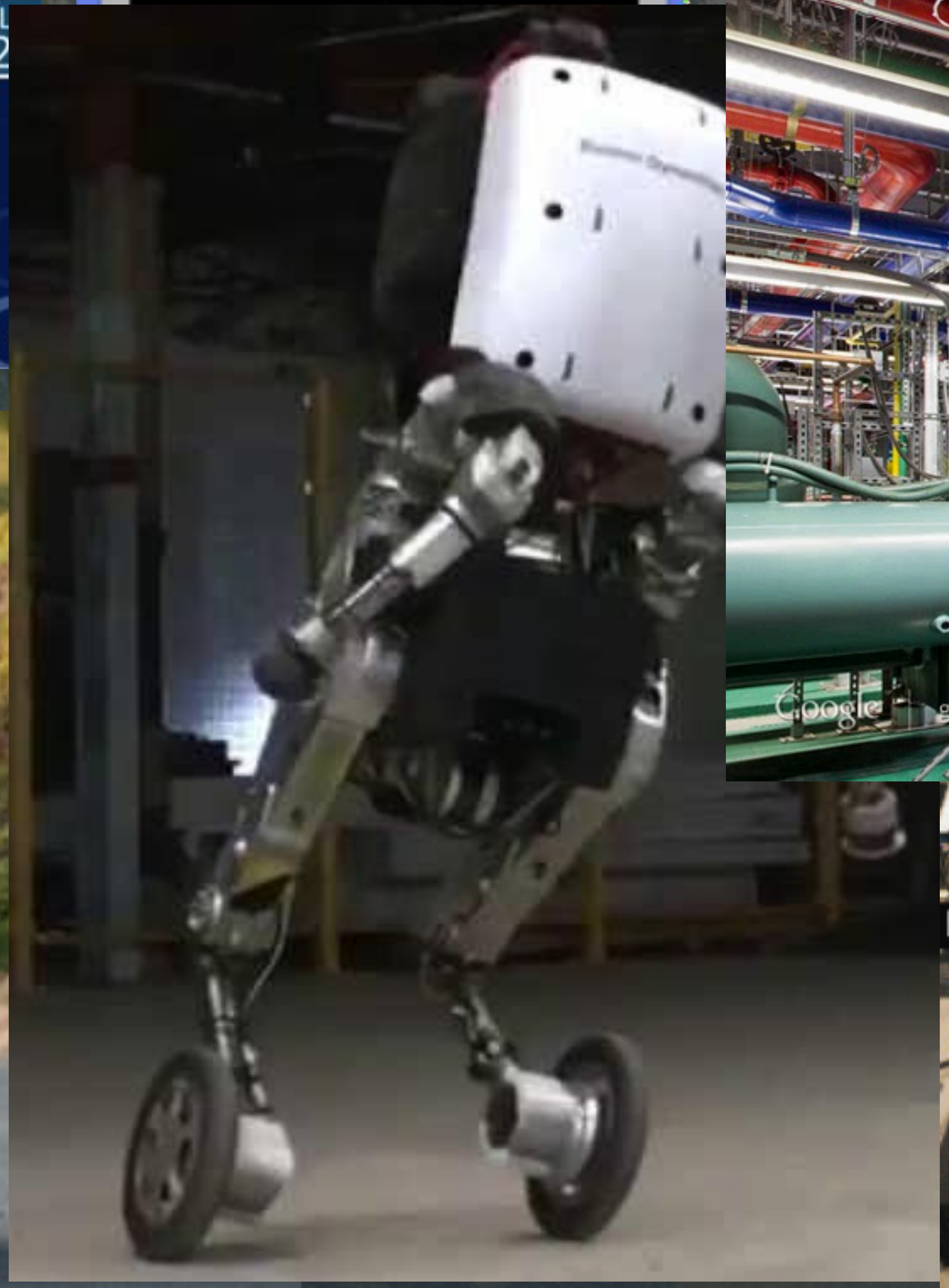
Industrial Processes

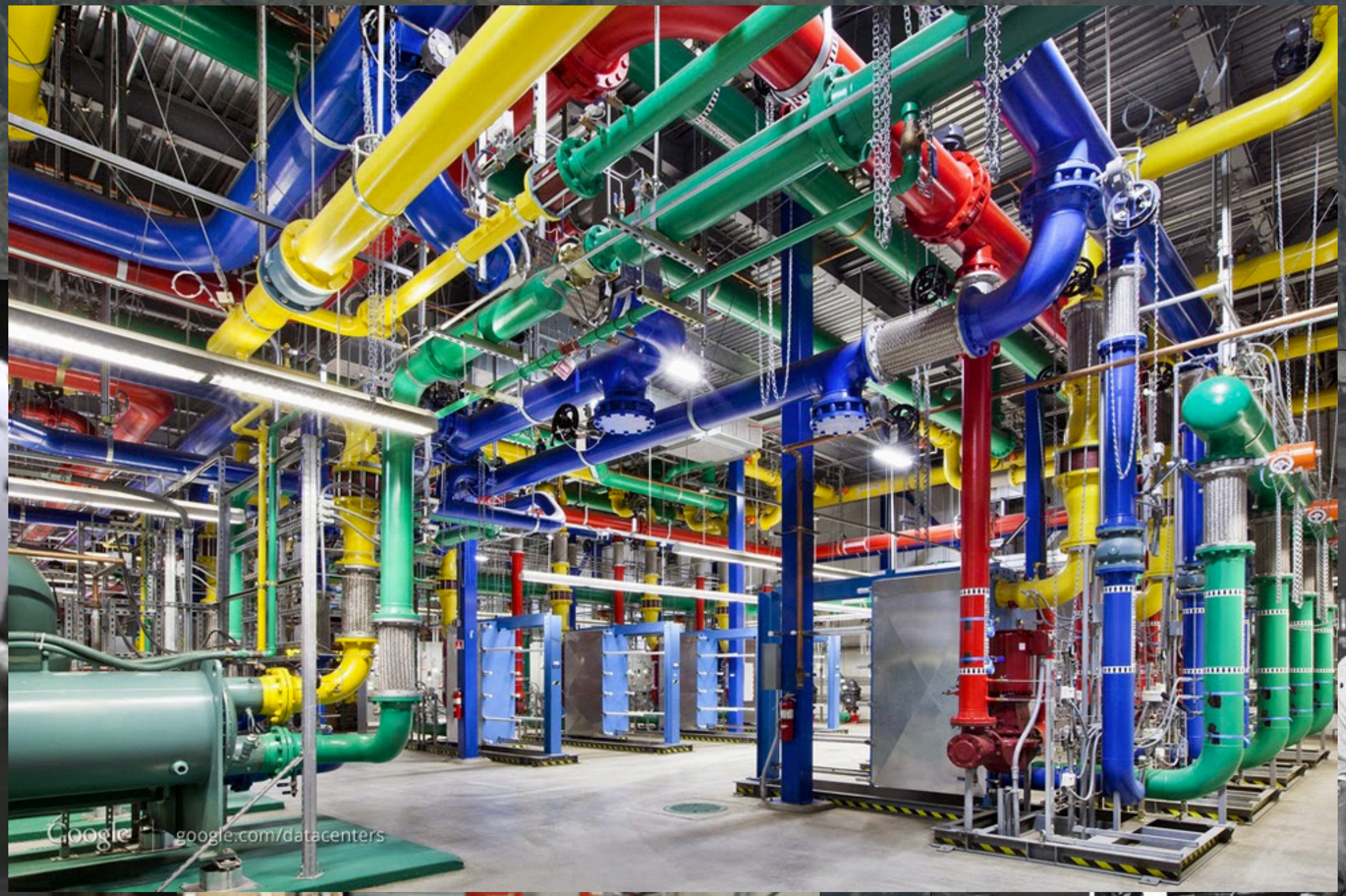
System Optimization

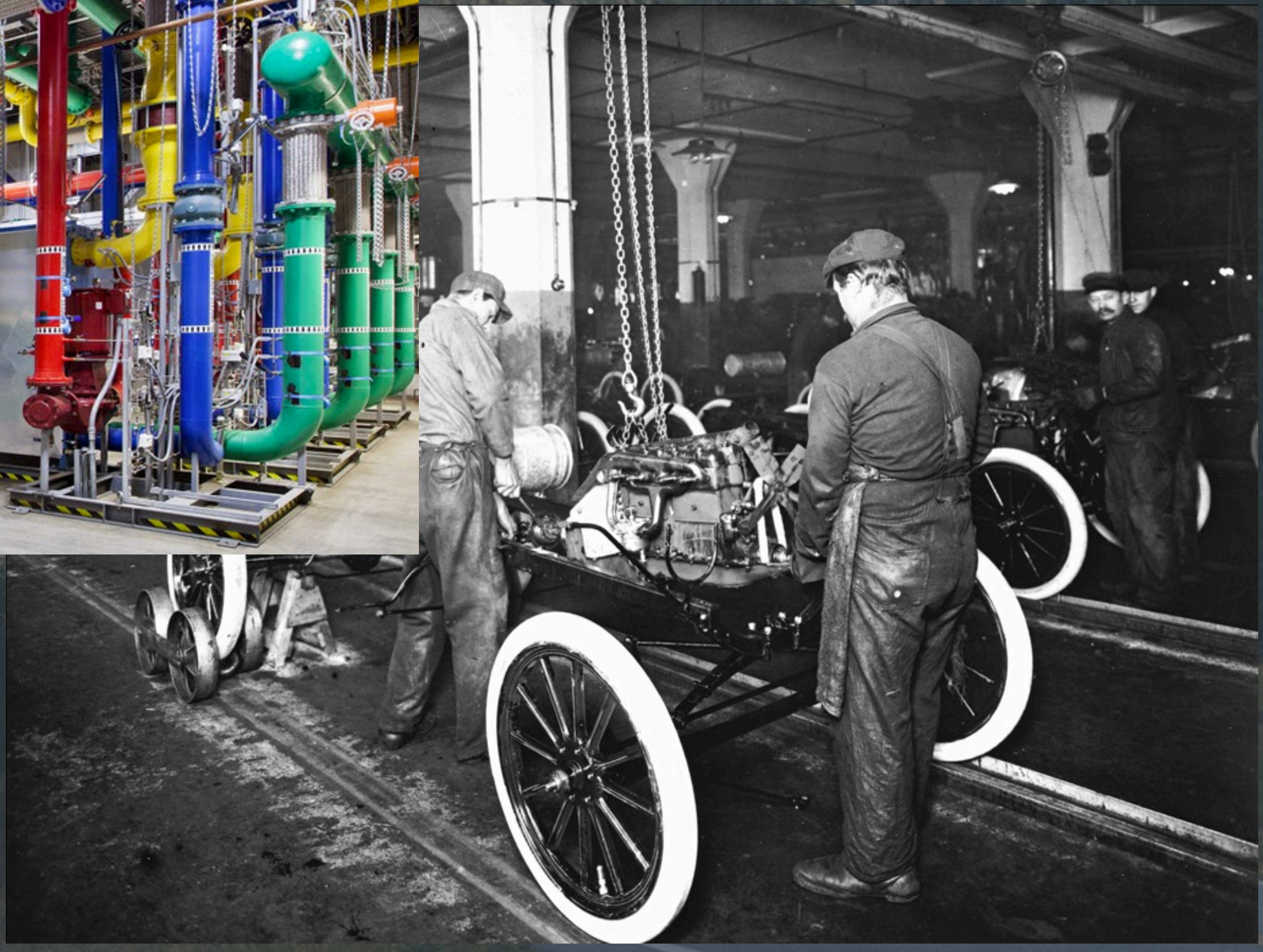
Advertising, Recommendations

Finance



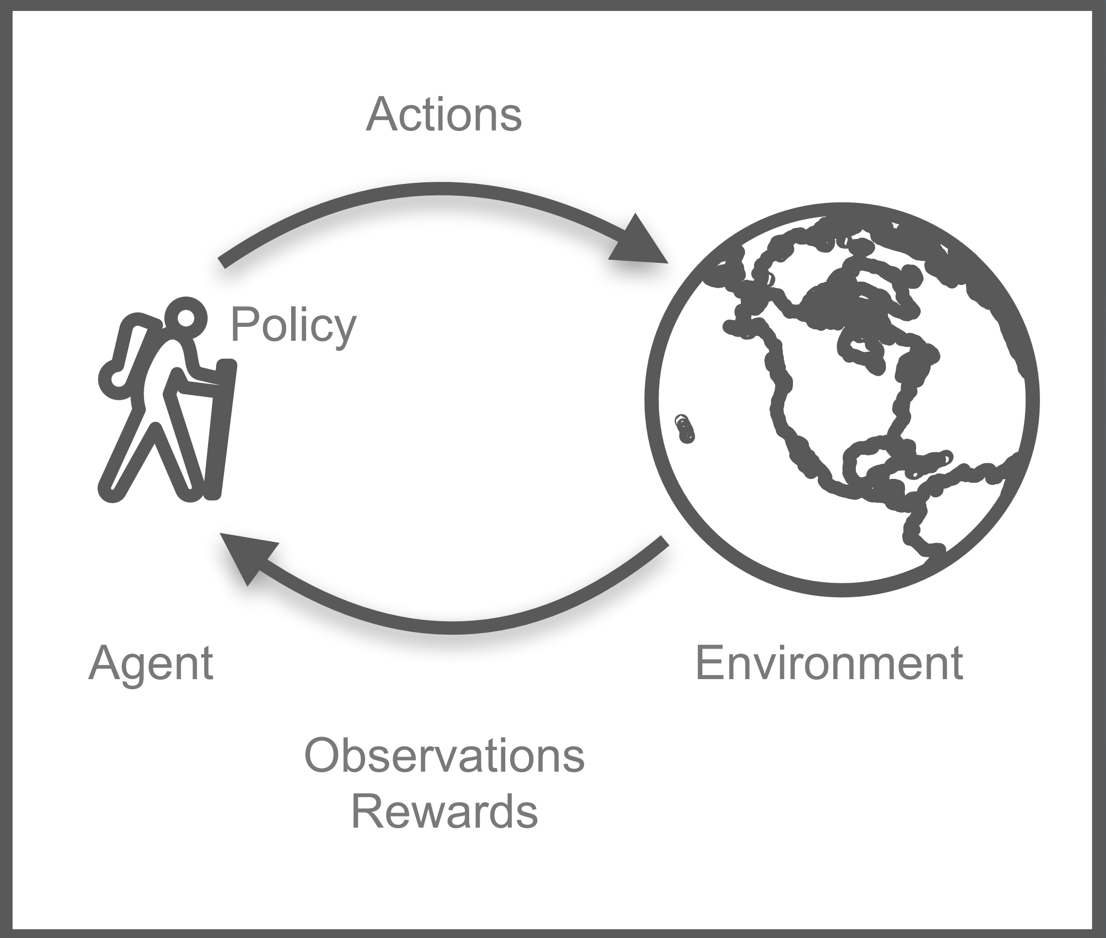




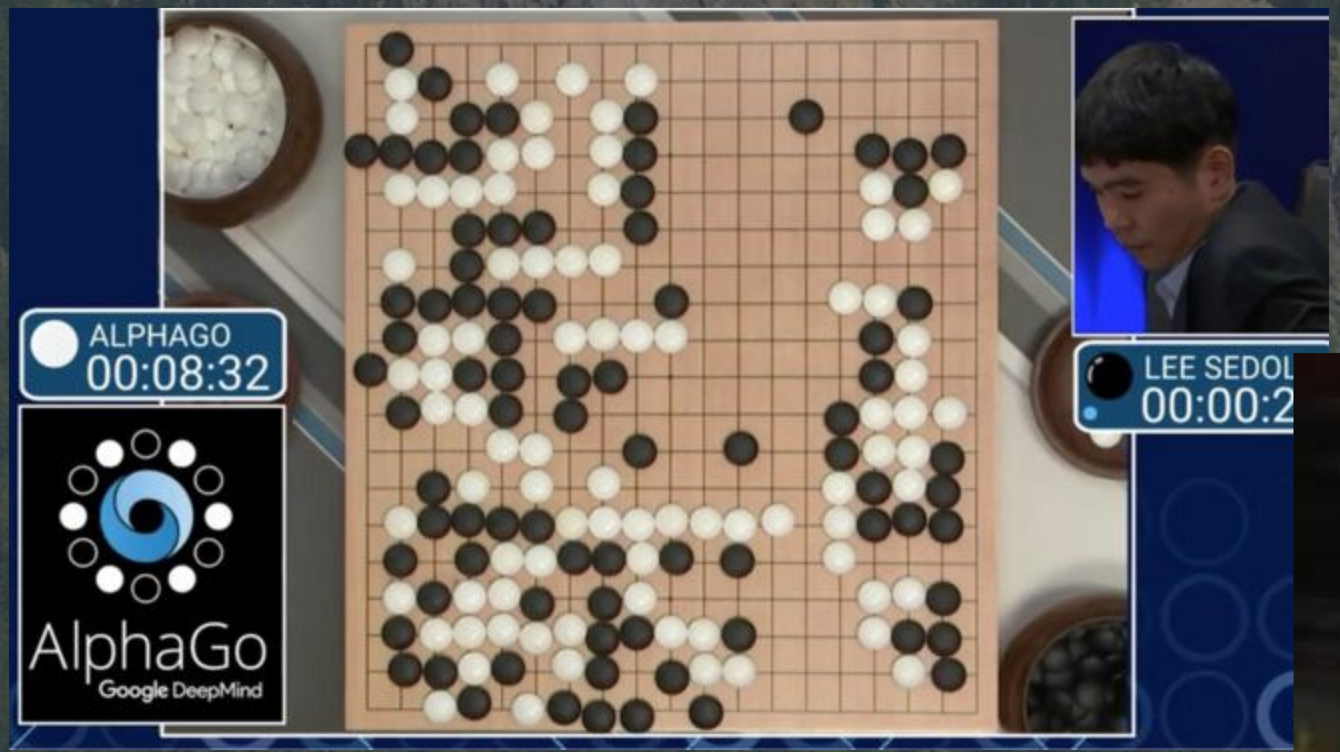


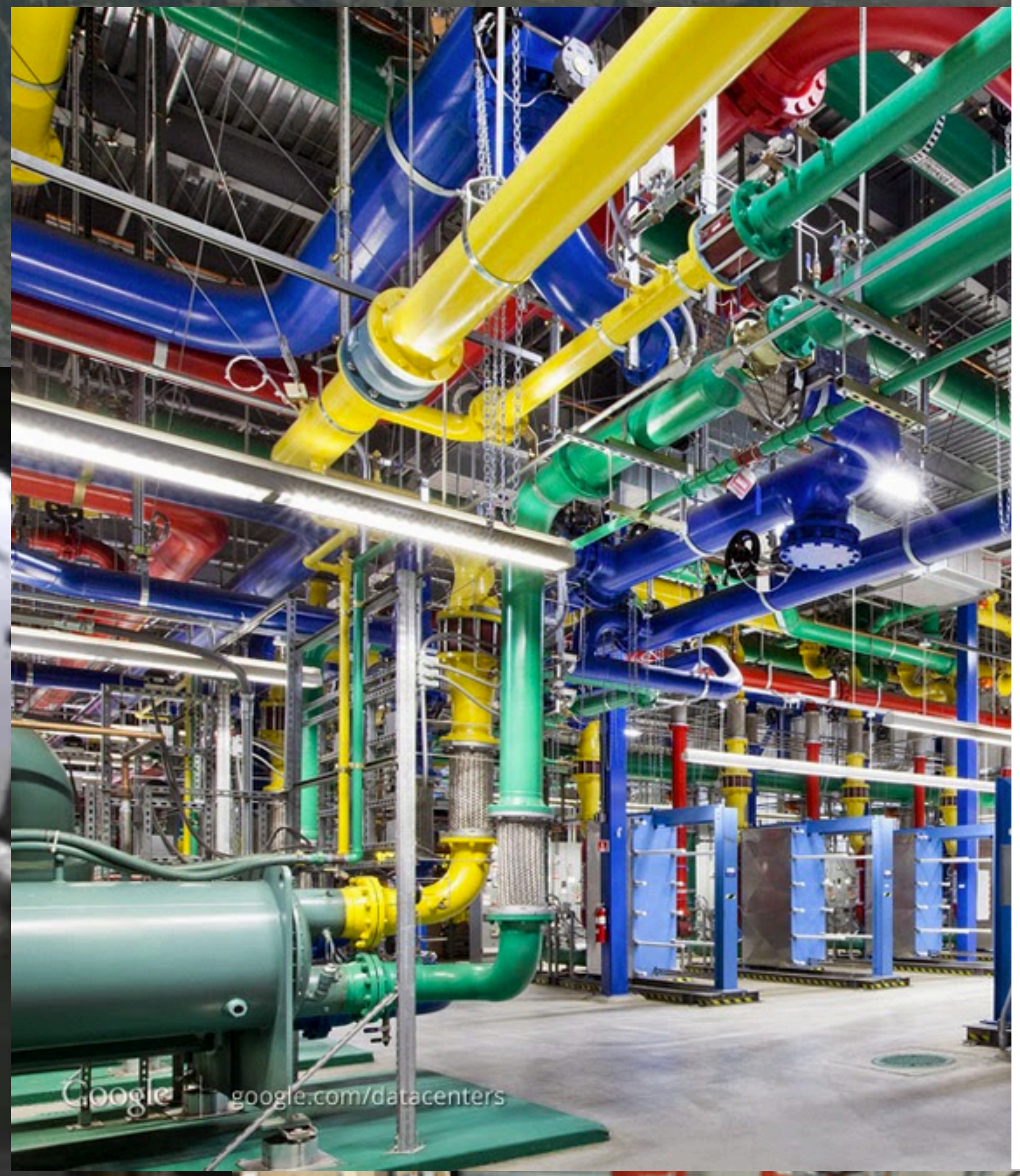
RL Applications

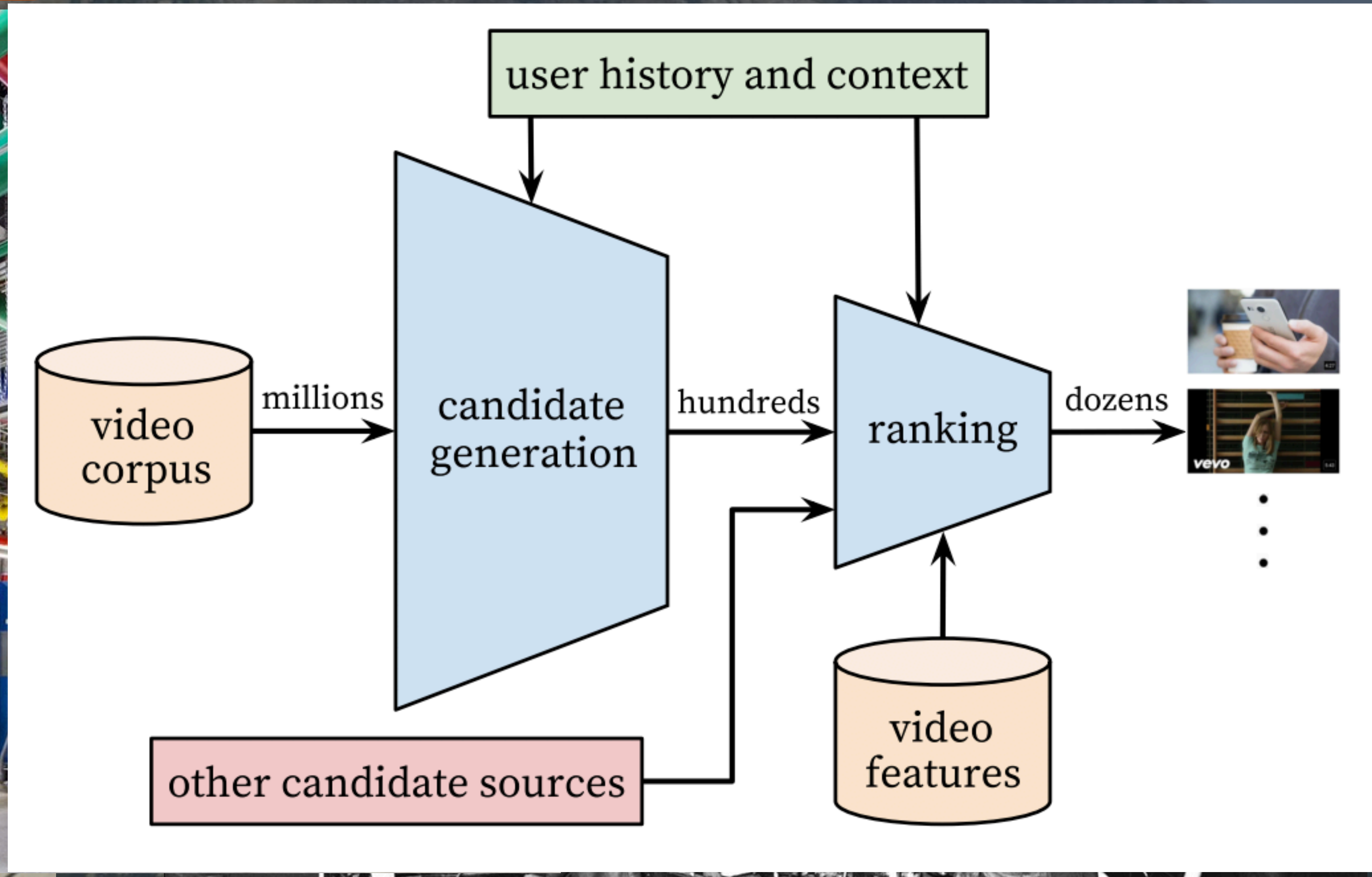
Better recommendations, ad placements, …

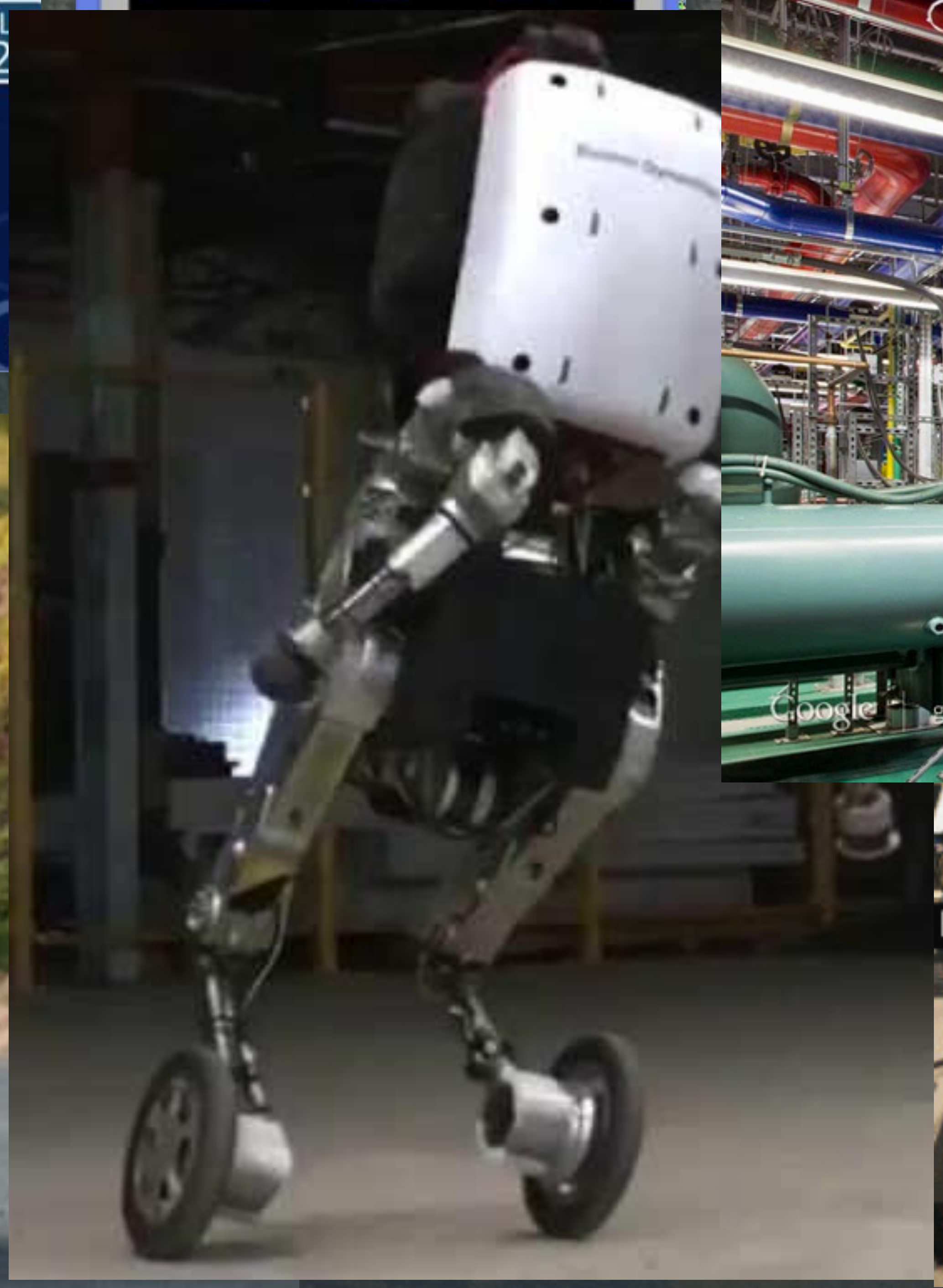


Games | Robotics, Autonomous Vehicles | Industrial Processes | System Optimization | Advertising, Recommendations | Finance



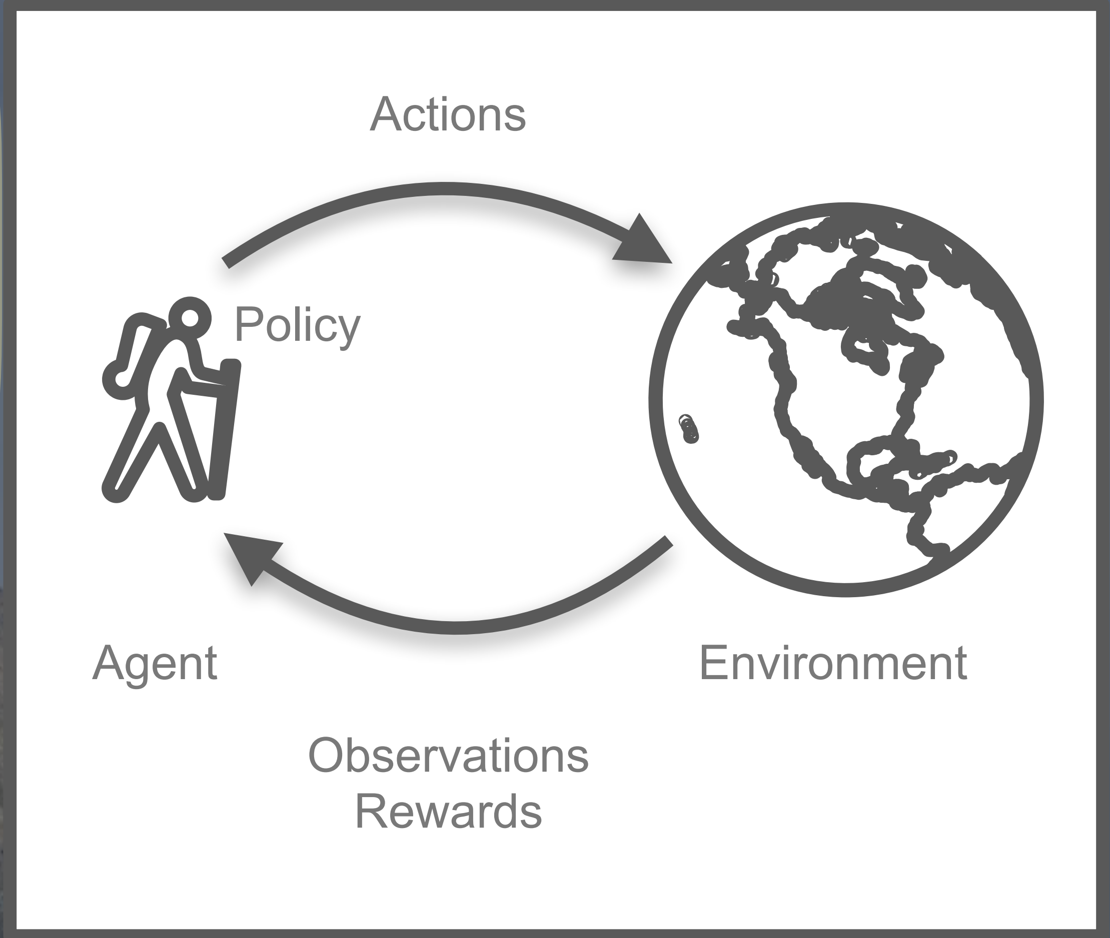






RL Applications

Market trends, timing of trades, …



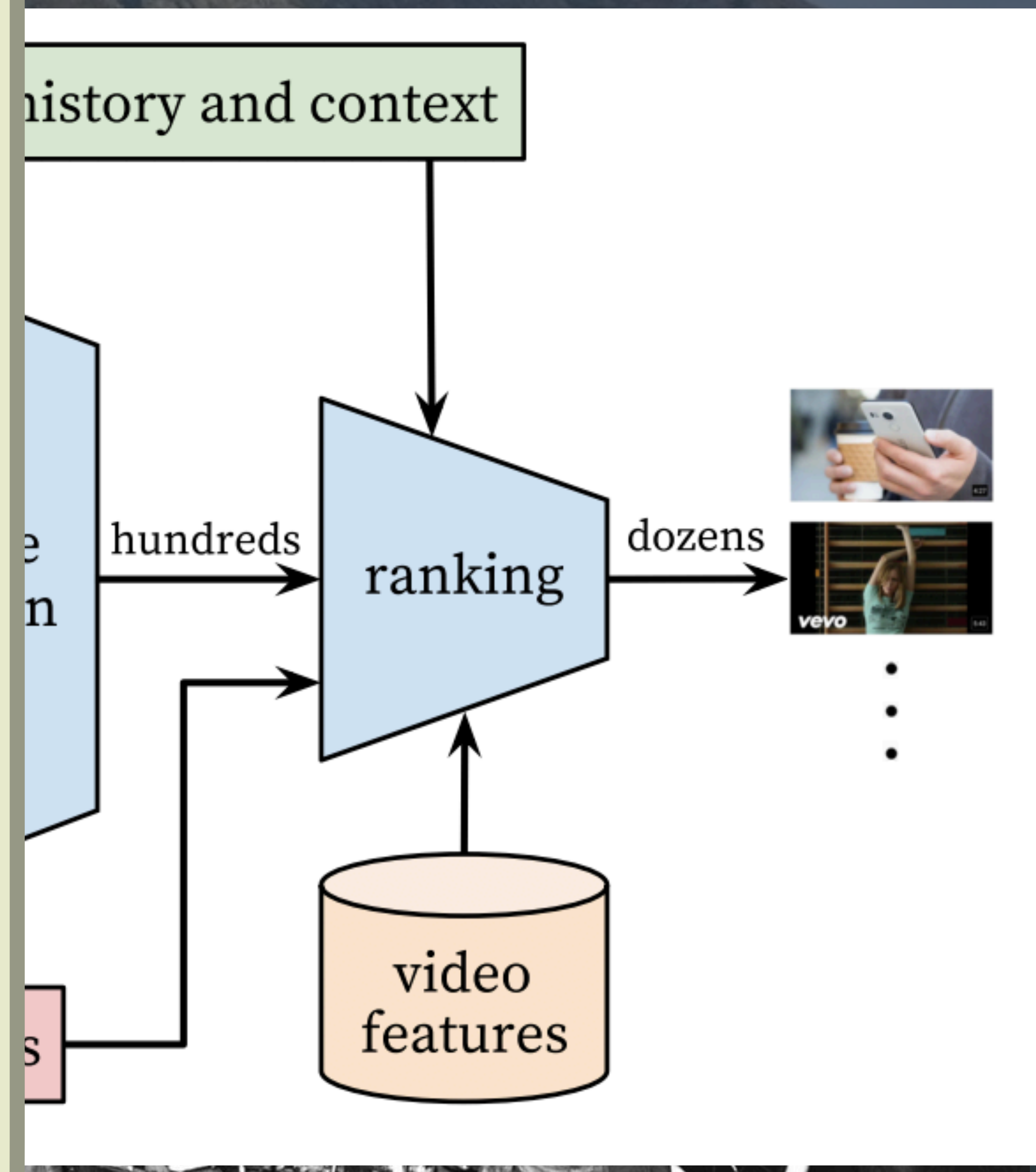
Games	Robotics, Autonomous Vehicles	Industrial Processes	System Optimization	Advertising, Recommendations	Finance

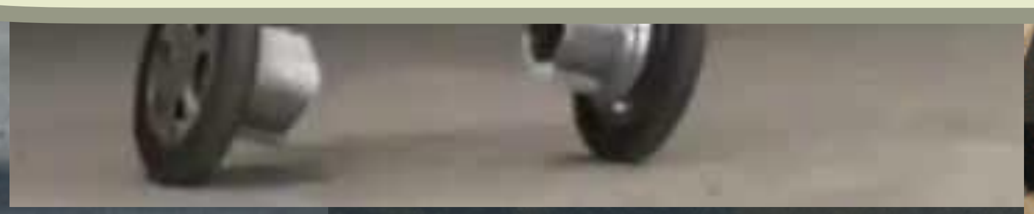
RL Applications

Games

Robotics
Autonomo
Vehicles

Market trends, timing of trades,



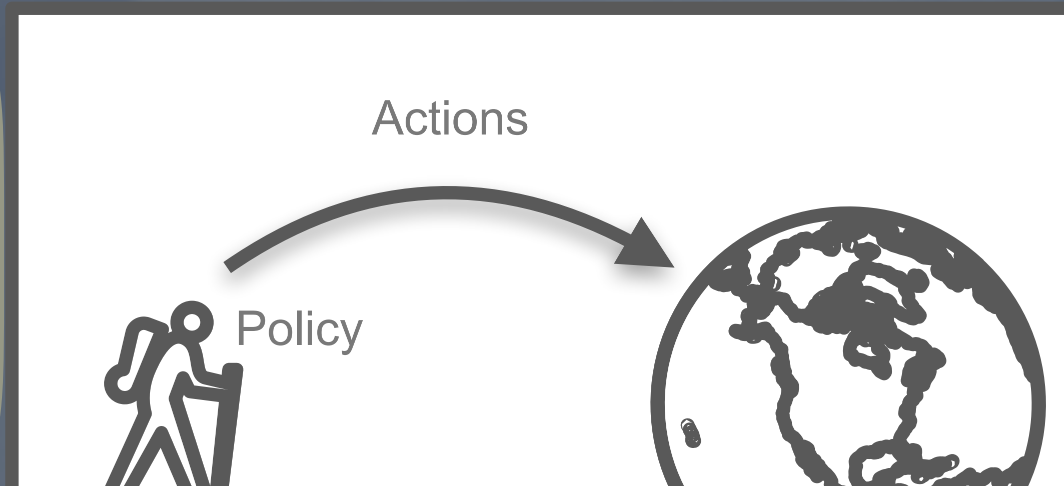


Common Theme:

The ideal applications have sequential, evolving state for the environment and the agent.

RL Applications

ChatGPT!
https://openai.com/blog/chatgpt/



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

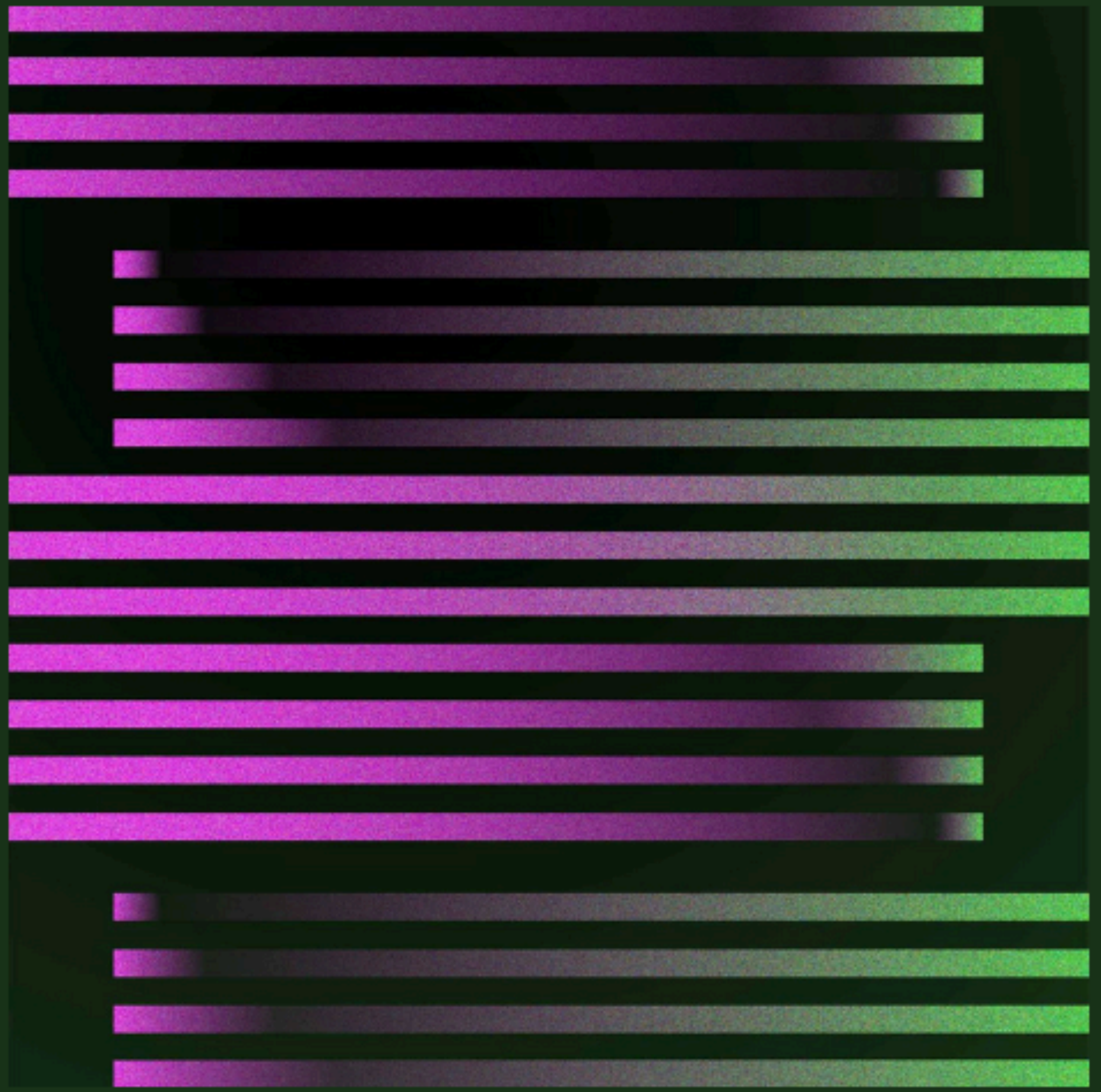
Introducing ChatGPT research release Try ↗ Learn more >

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.



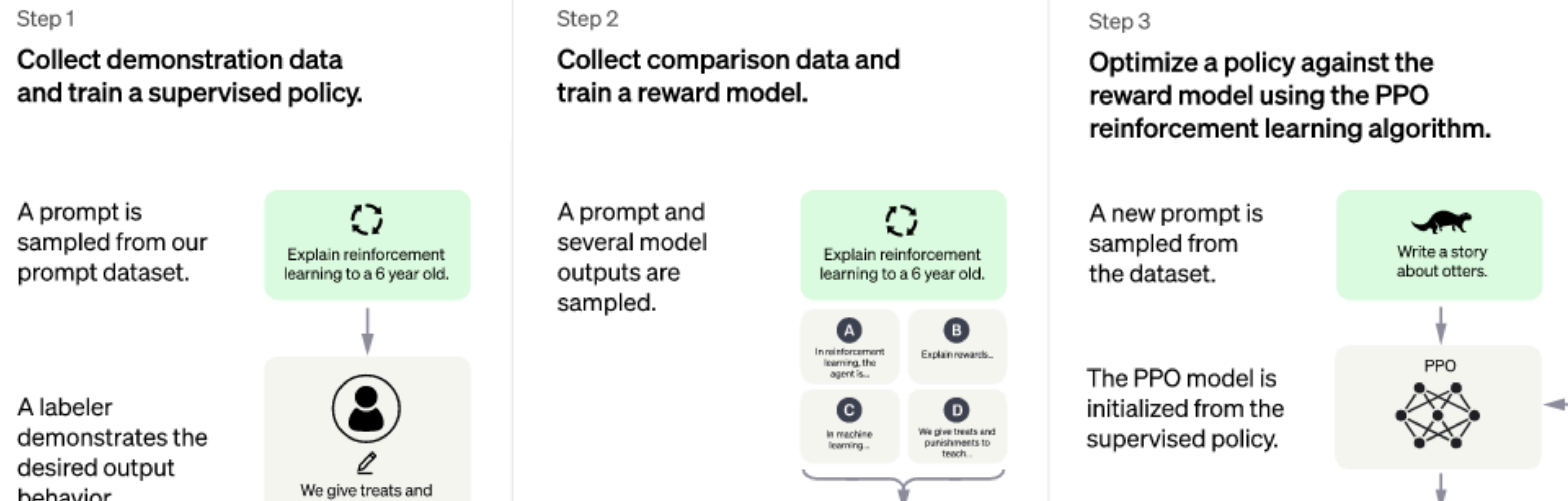
TRY CHATGPT ↗

November 30, 2022
13 minute read

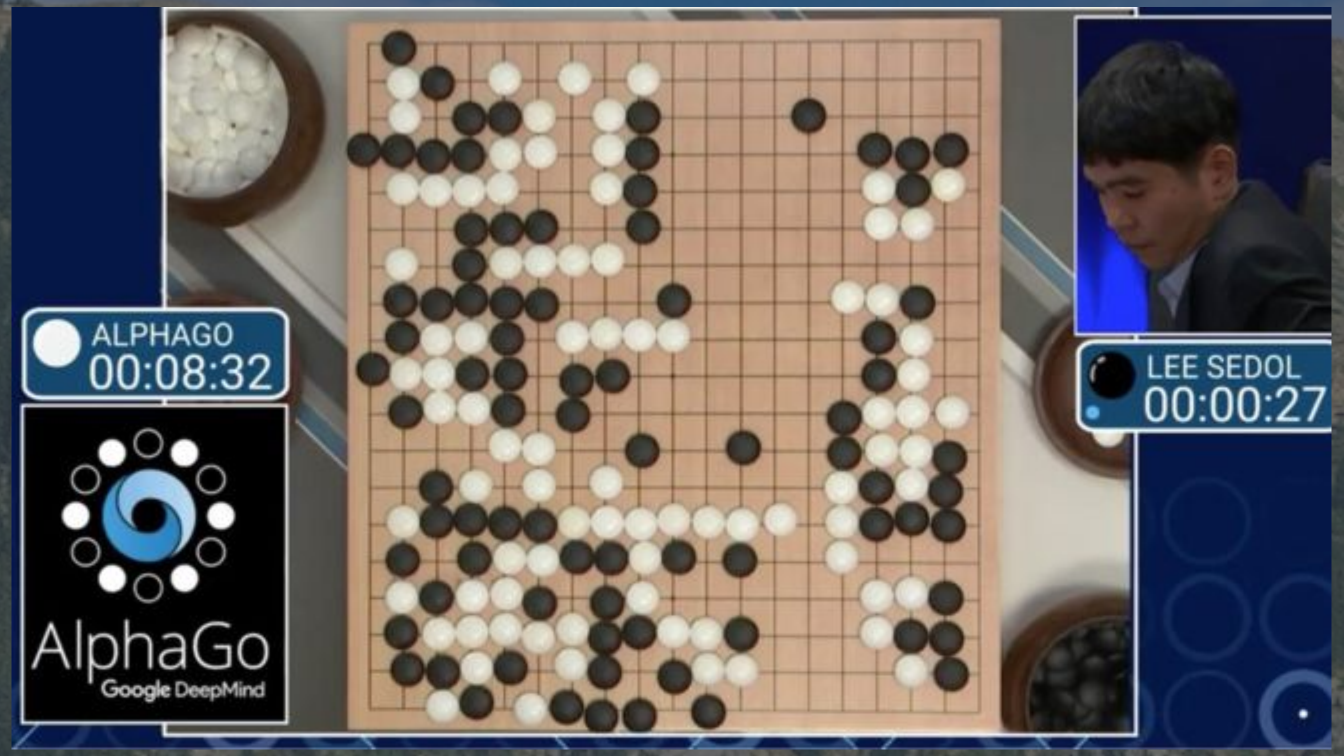
Methods

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.

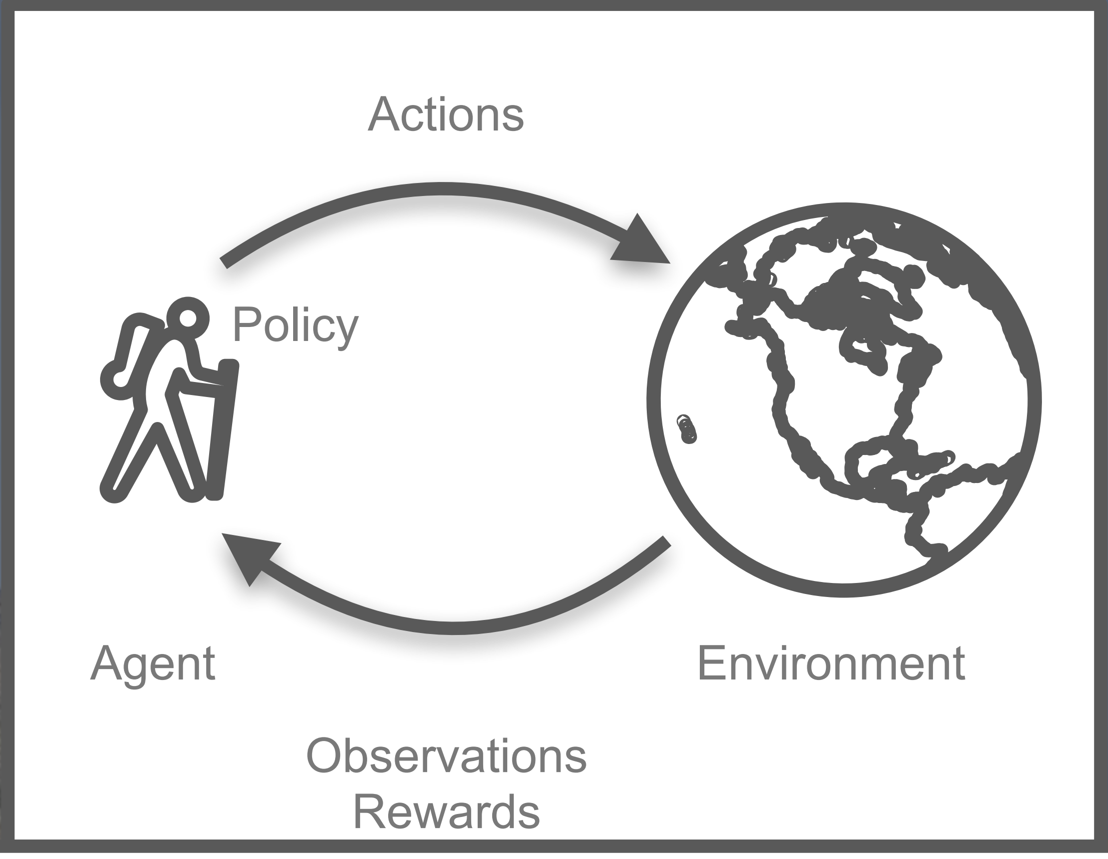
To create a reward model for reinforcement learning, we needed to collect comparison data, which consisted of two or more model responses ranked by quality. To collect this data, we took conversations that AI trainers had with the chatbot. We randomly selected a model-written message, sampled several 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.



AlphaGo example

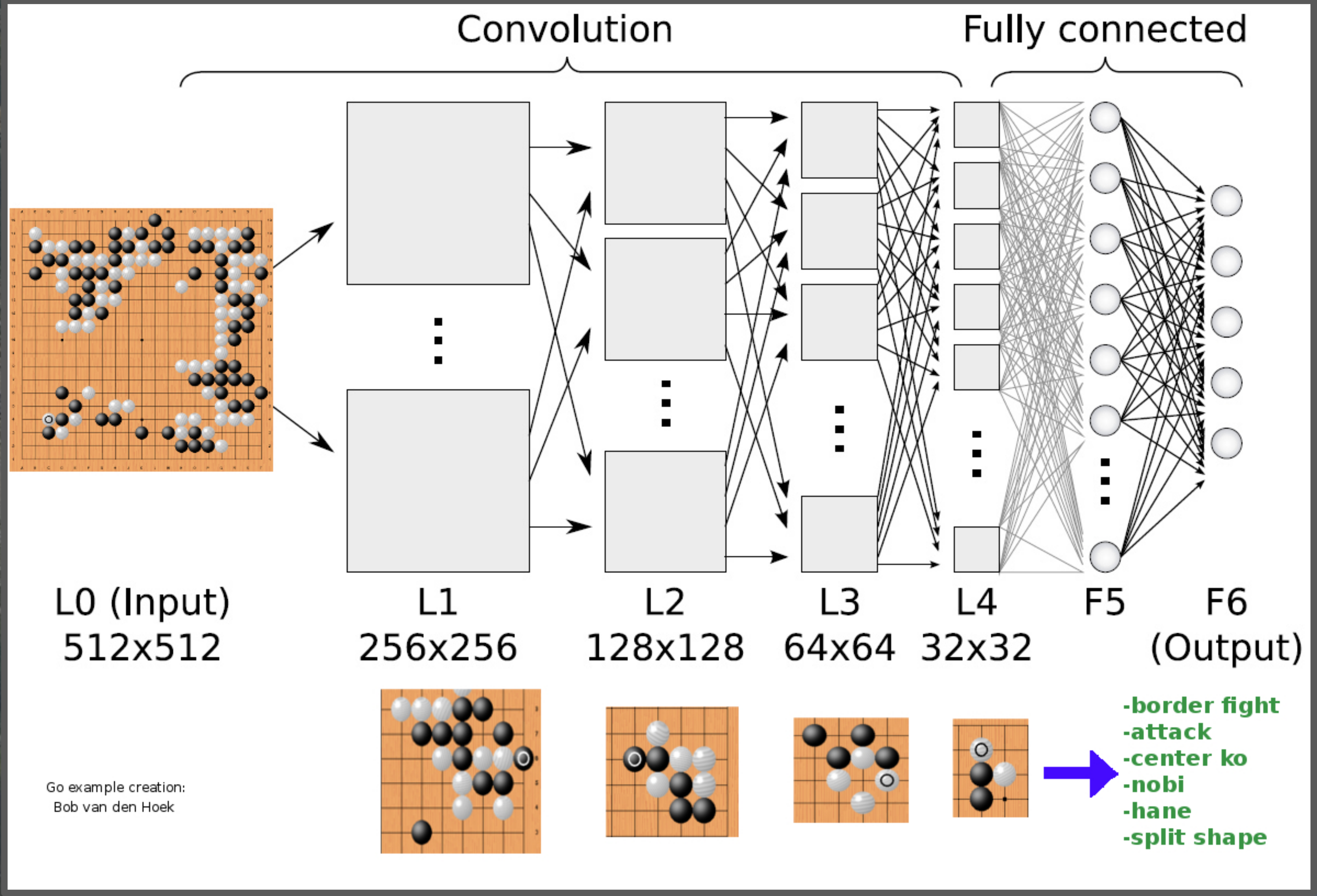


Deep Reinforcement Learning

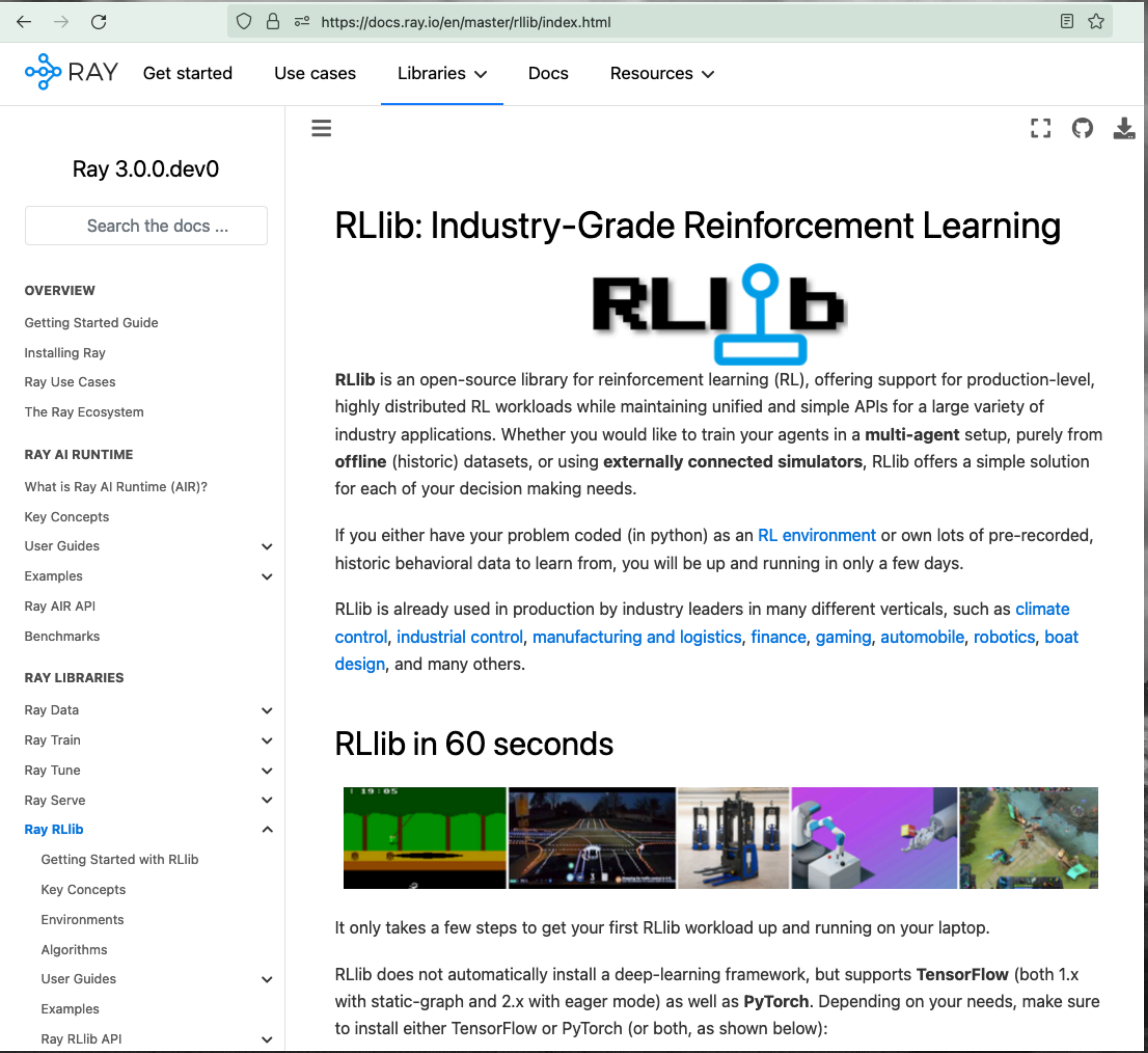


AlphaGo (Silver et al. 2016)

- **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 ...

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.

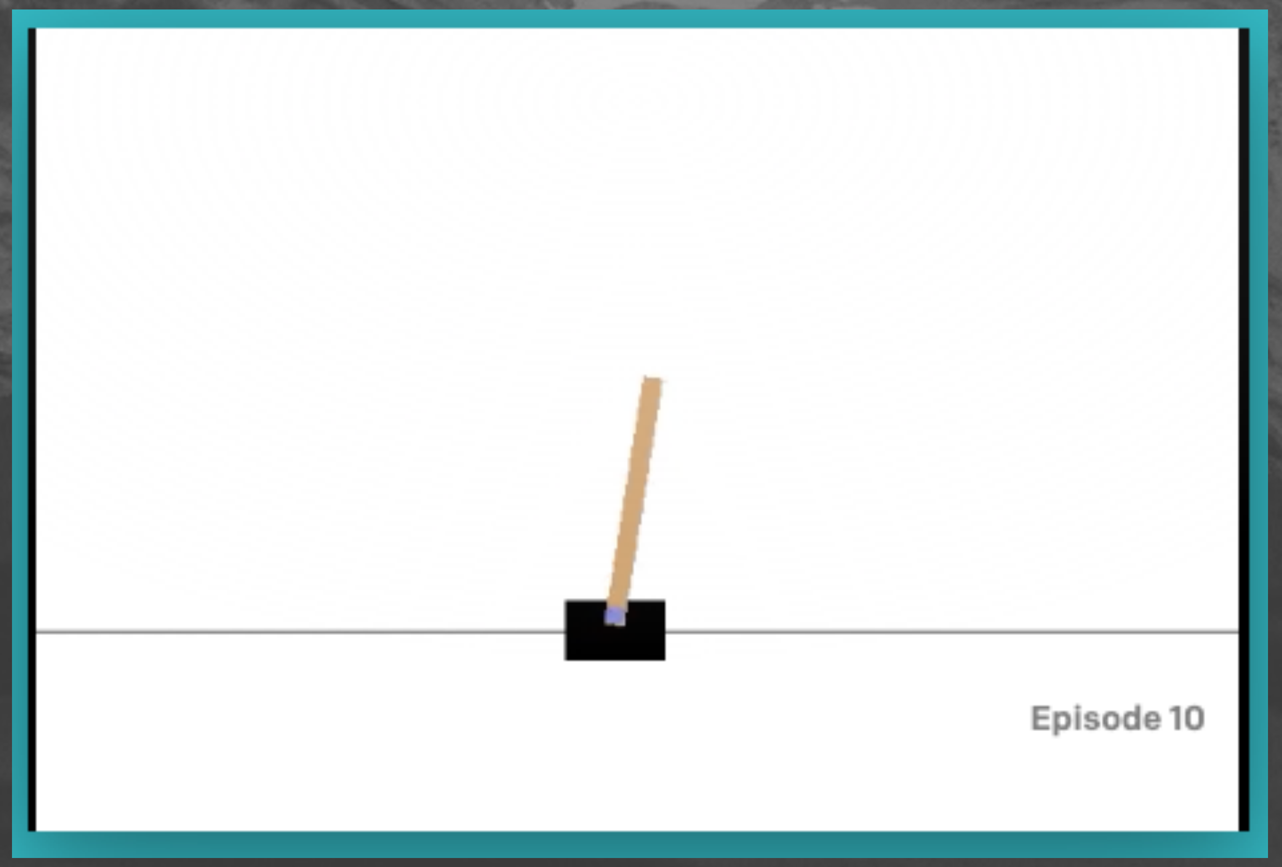
RLlib does not automatically install a deep-learning framework, but supports **TensorFlow** (both 1.x with static-graph and 2.x with eager mode) as well as **PyTorch**. Depending on your needs, make sure to install either TensorFlow or PyTorch (or both, as shown below):

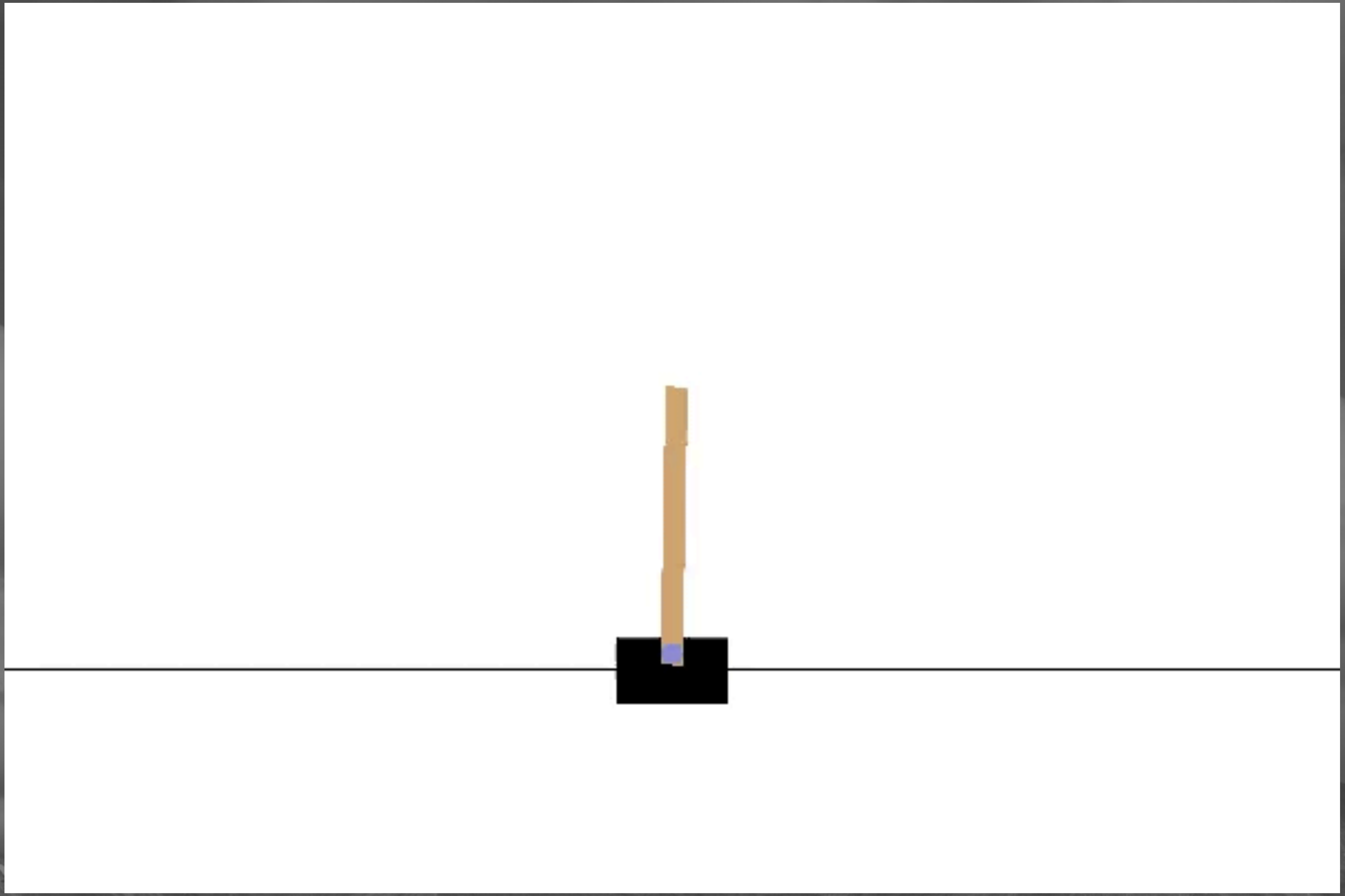
rllib.io

Ray RLlib

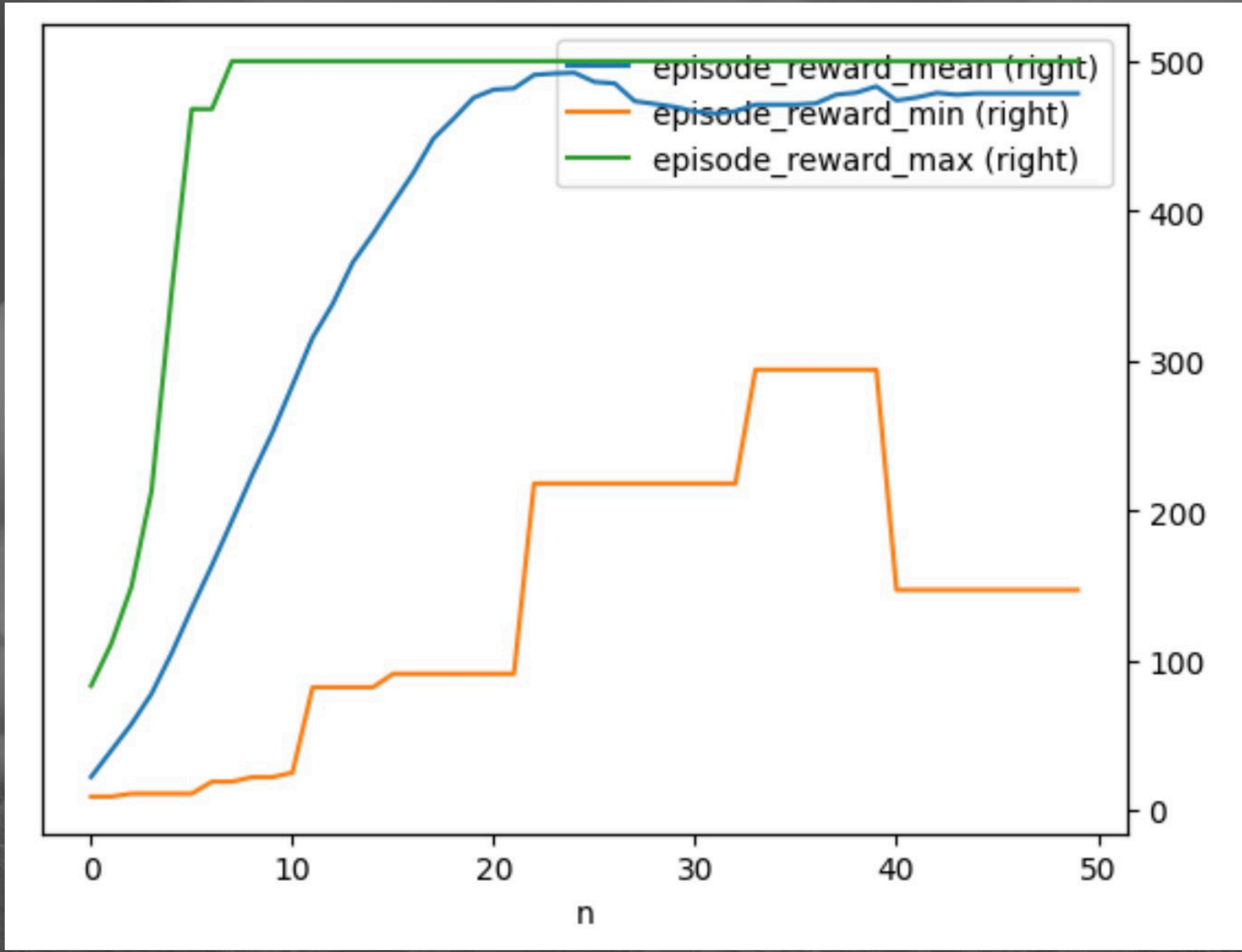

To Try It Out…

```
# Install what we need:
$ pip install "ray[rllib]" tensorflow \
  tensorflow-probability pygame

# Train CartPole using DQN, stop after 100 iterations:
# At end, will print the next command to run:
$ rllib train --algo DQN --env 'CartPole-v1' \
  --stop '{"training_iteration": 200}'

# Run CartPole and see how well it goes:
$ rllib evaluate /path/to/checkpoint --algo DQN
```





Example episode after training.



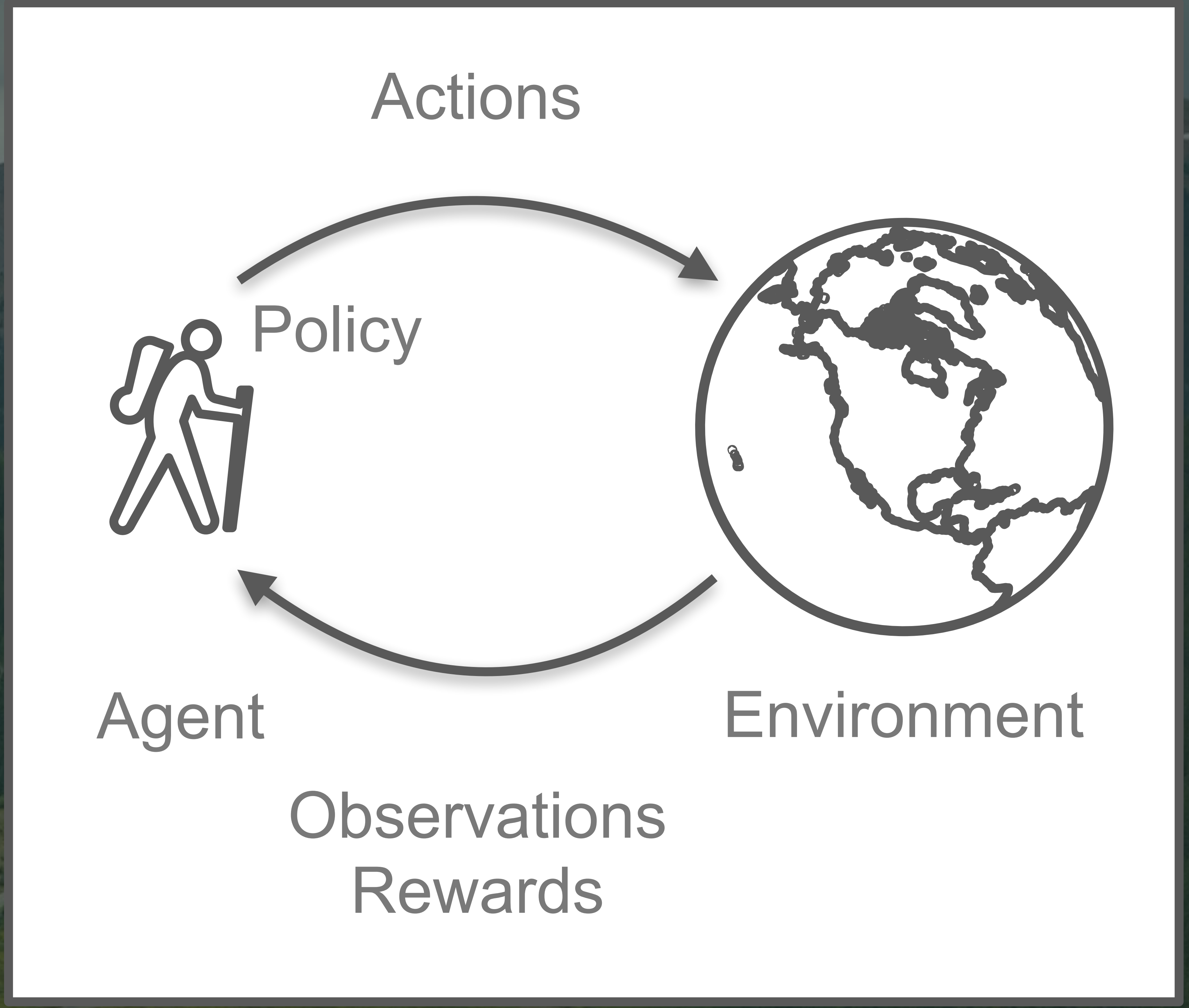
Training n=50 episodes with PPO. Max score is 500. Note that the average actually dips above 20 episodes. Probably overfitting?

RLlib Benefits

- 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, distributed performance… from Ray!

More Reinforcement Learning Concepts and Challenges

Exploitation vs. Exploration

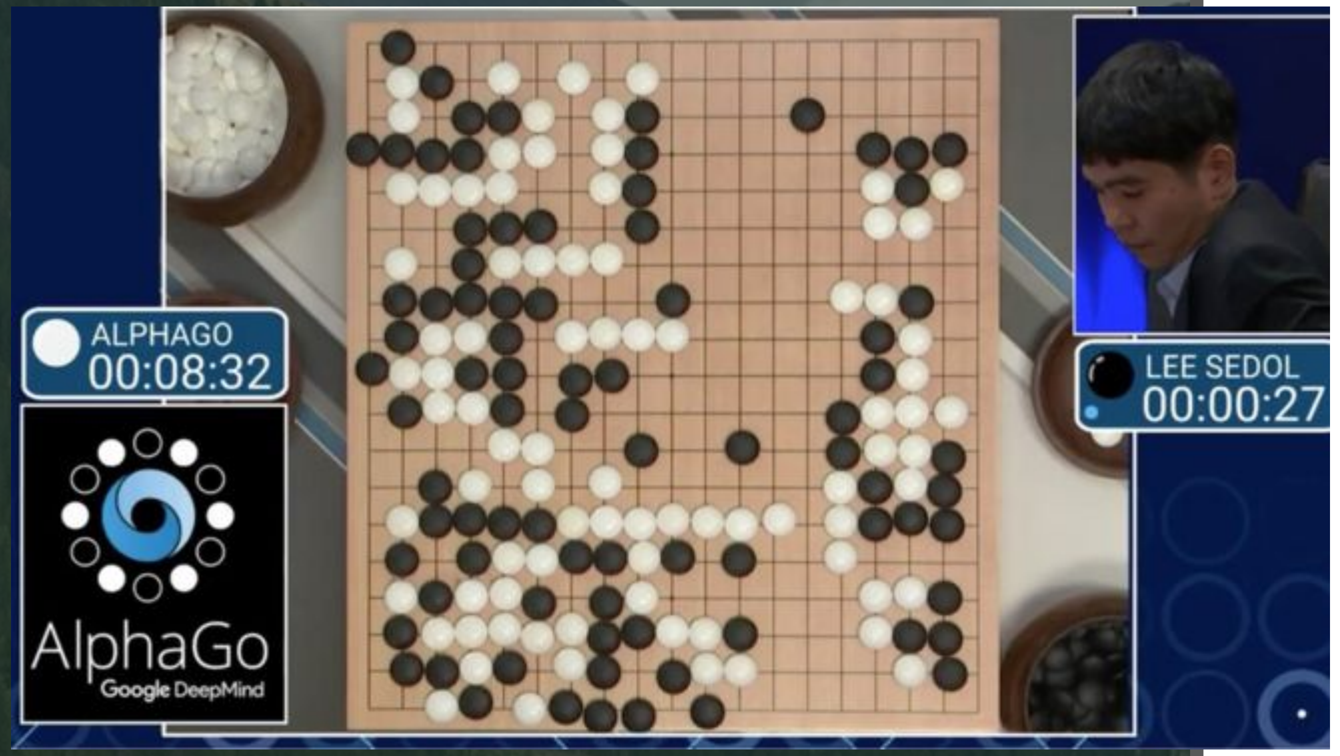


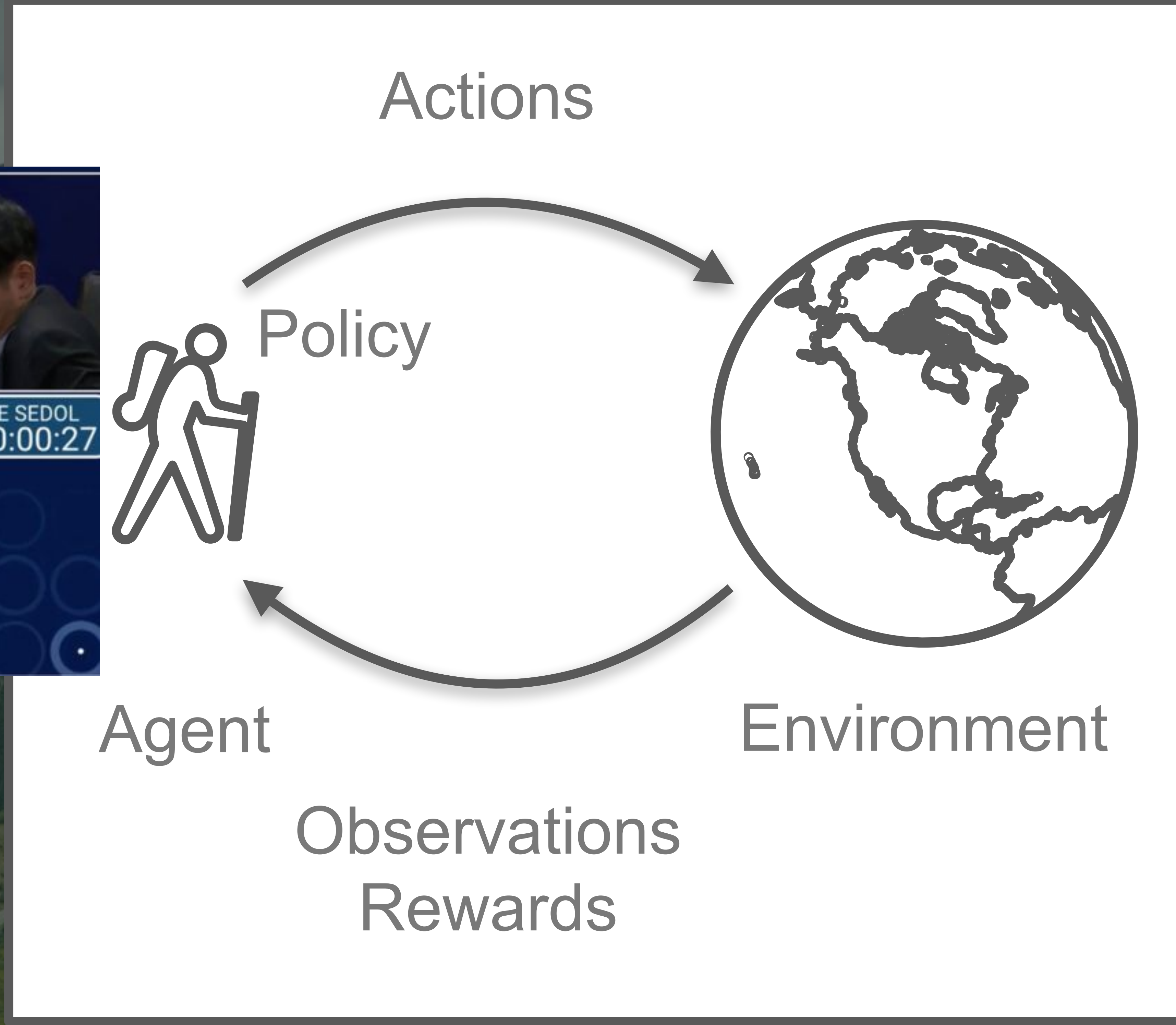
What if the agent finds an action with a good short-term reward? Should it keep exploiting it?

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 an RL system to optimize performance of 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 these environments generate "log" data, what about using this historical data, instead?

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

ChatGPT?

“Reinforcement Learning from Human Feedback” (RLHF)

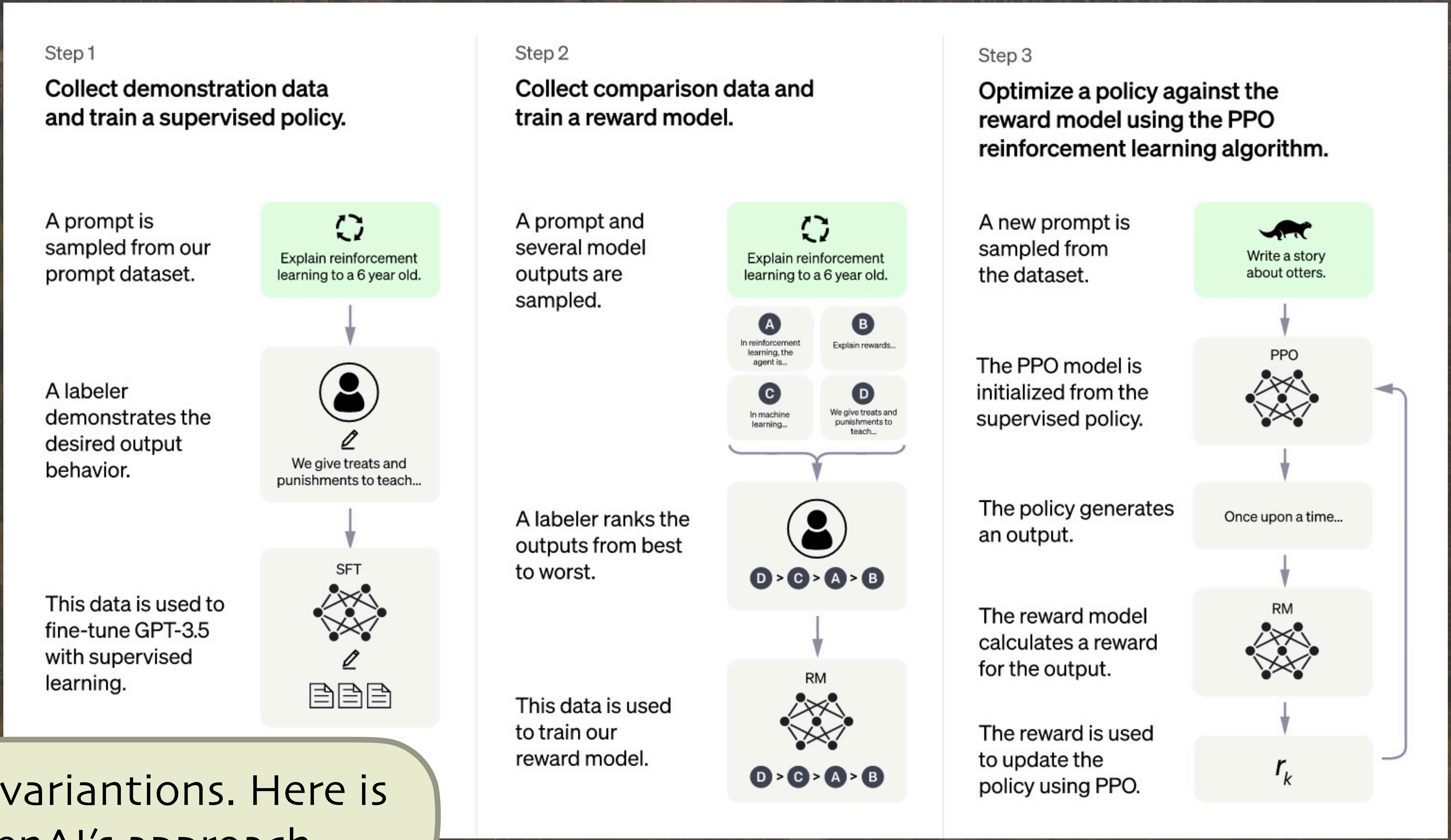
References:

- https://openai.com/blog/chatgpt
- huggingface.co/blog/rlhf

huggingface.co/blog/rlhf

Writing a loss function to capture these attributes seems intractable and most language models are still trained with a simple next token prediction loss (e.g. cross entropy). To compensate for the shortcomings of the loss itself people define metrics that are designed to better capture human preferences such as BLEU or ROUGE. While being better suited than the loss function itself at measuring performance these metrics simply compare generated text to references with simple rules and are thus also limited. Wouldn't it be great if we use human feedback for generated text as a measure of performance or go even one step further and use that feedback as a loss to optimize the model? That's the idea of Reinforcement Learning from Human Feedback (RLHF); use methods from reinforcement learning to directly optimize a language model with human feedback. RLHF has enabled language models to begin to align a model trained on a general corpus of text data to that of complex human values.

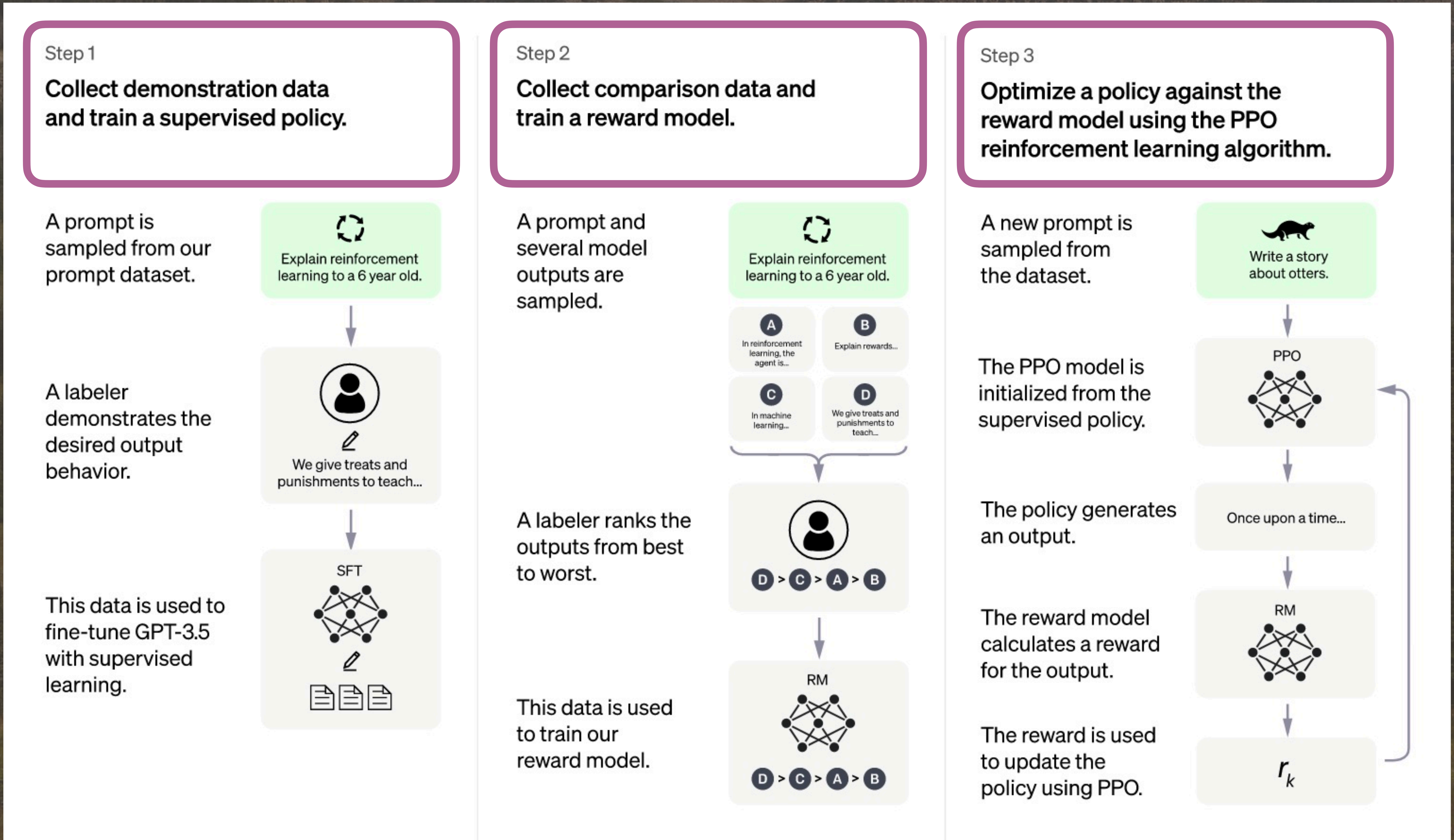
Reinforcement Learning from Human Feedback



Many variantions. Here is OpenAI's approach.

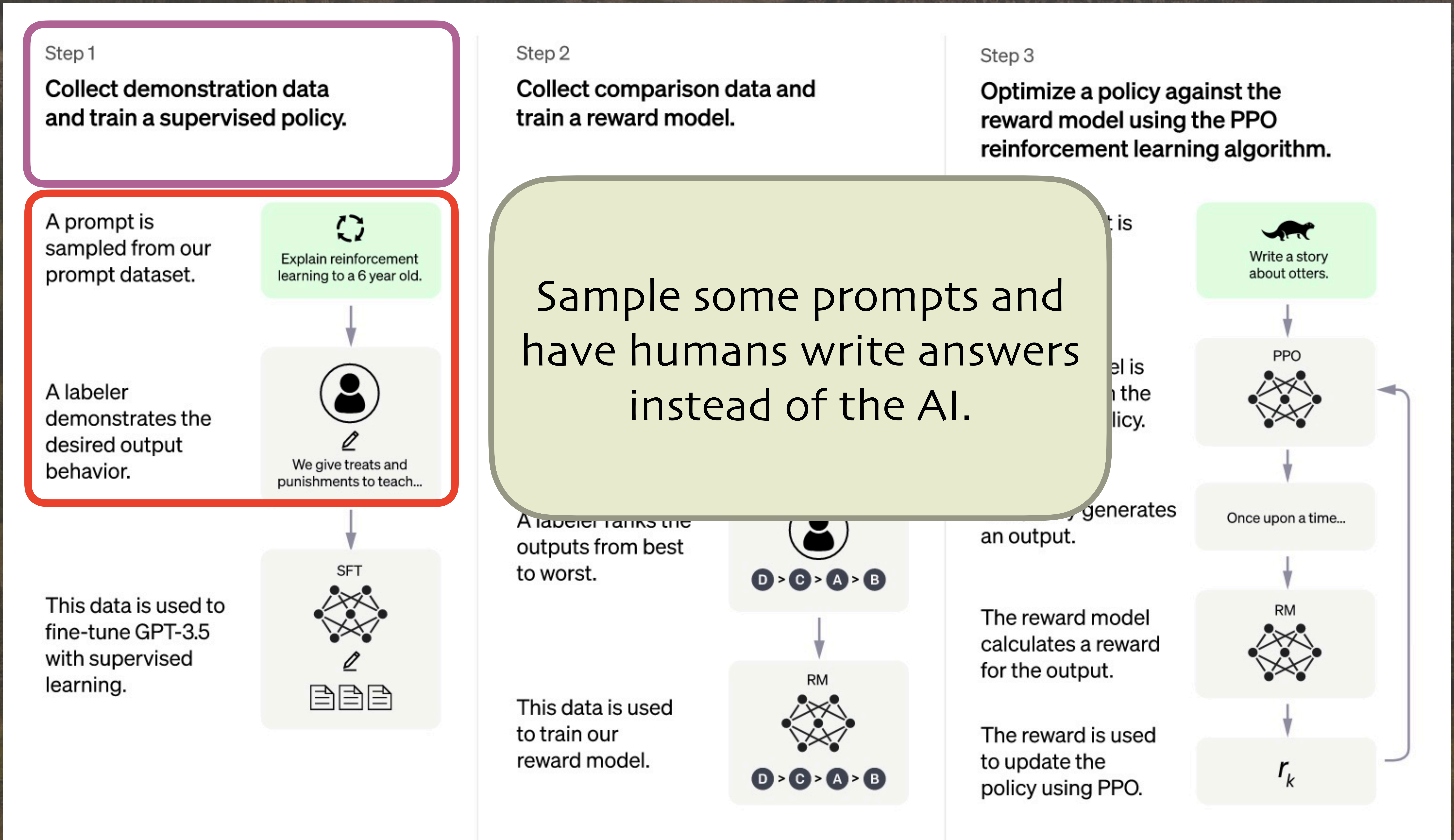
Credit: openai.com/blog/chatgpt

Reinforcement Learning from Human Feedback



Credit: openai.com/blog/chatgpt

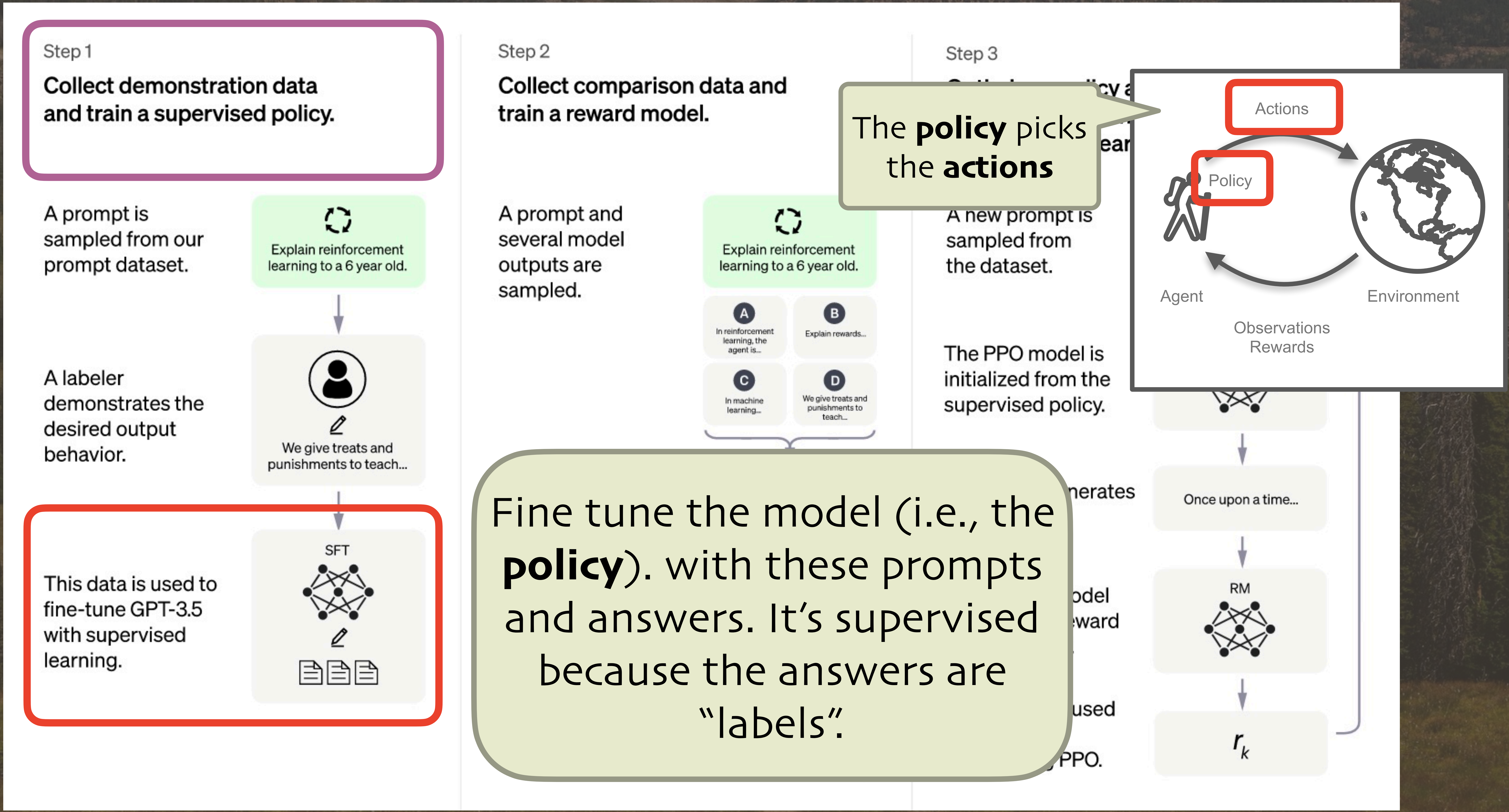
Reinforcement Learning from Human Feedback



Sample some prompts and have humans write answers instead of the AI.

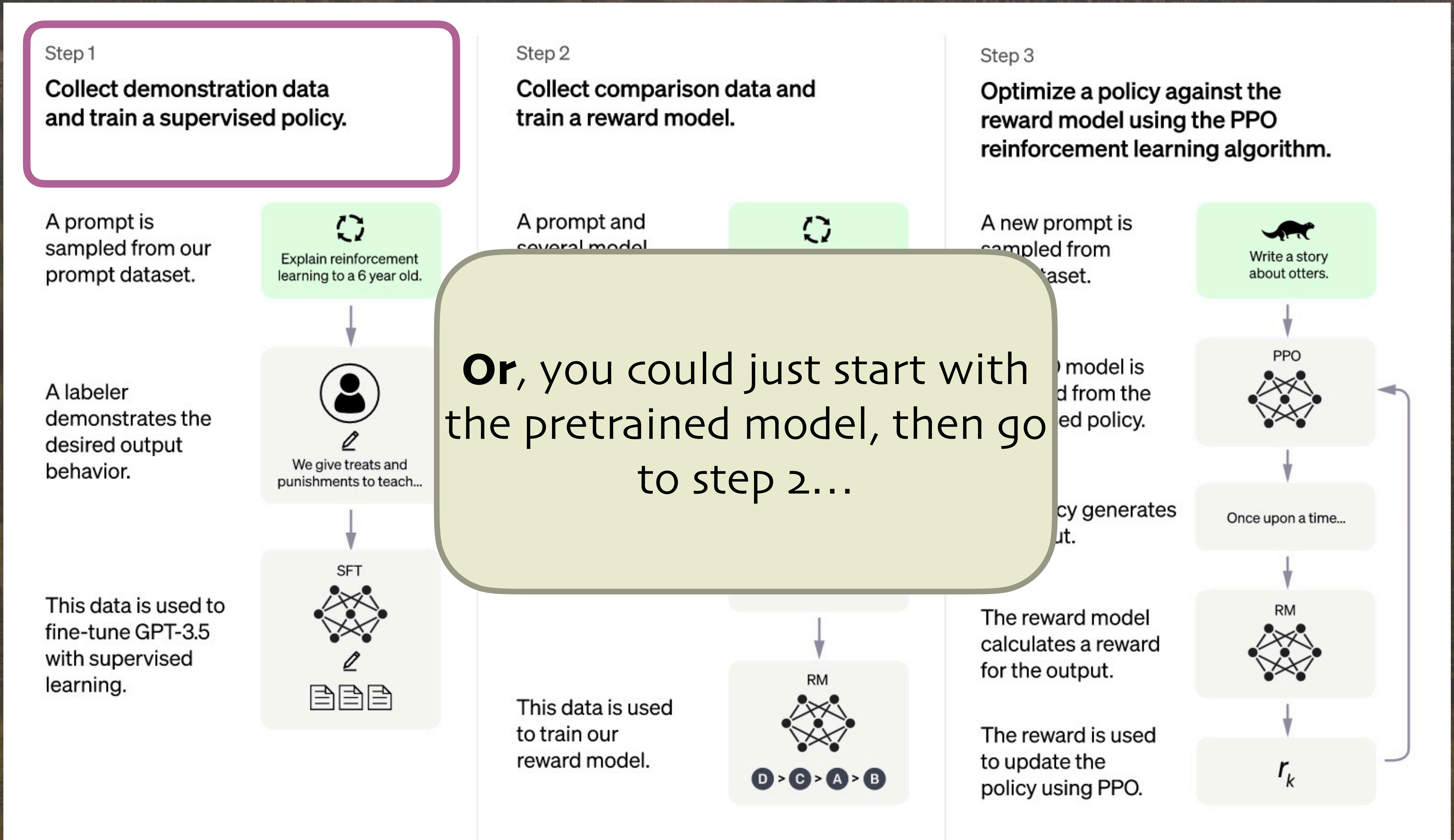
Credit: openai.com/blog/chatgpt

Reinforcement Learning from Human Feedback



Credit: openai.com/blog/chatgpt

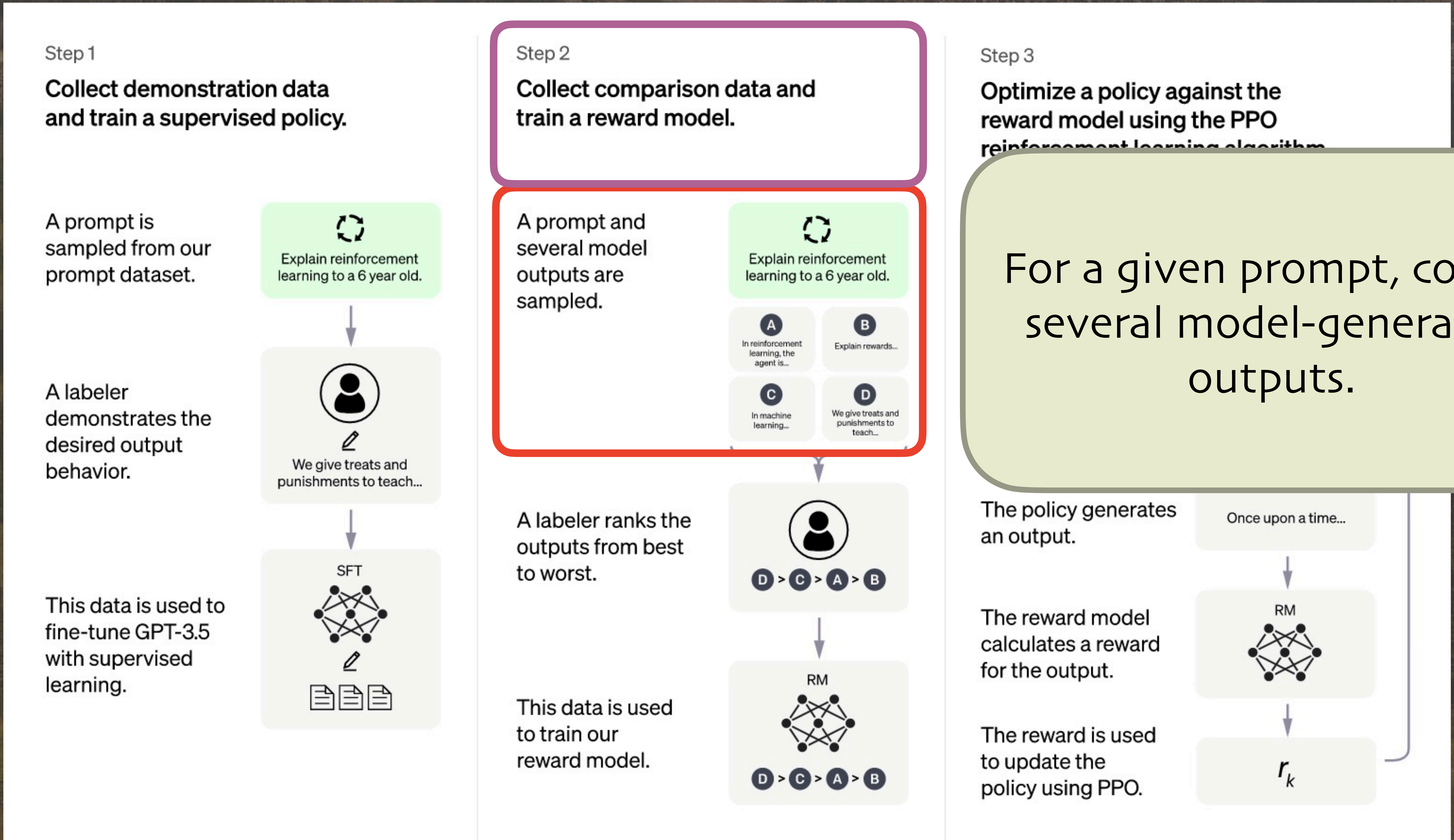
Reinforcement Learning from Human Feedback



Or, you could just start with the pretrained model, then go to step 2…

Credit: openai.com/blog/chatgpt

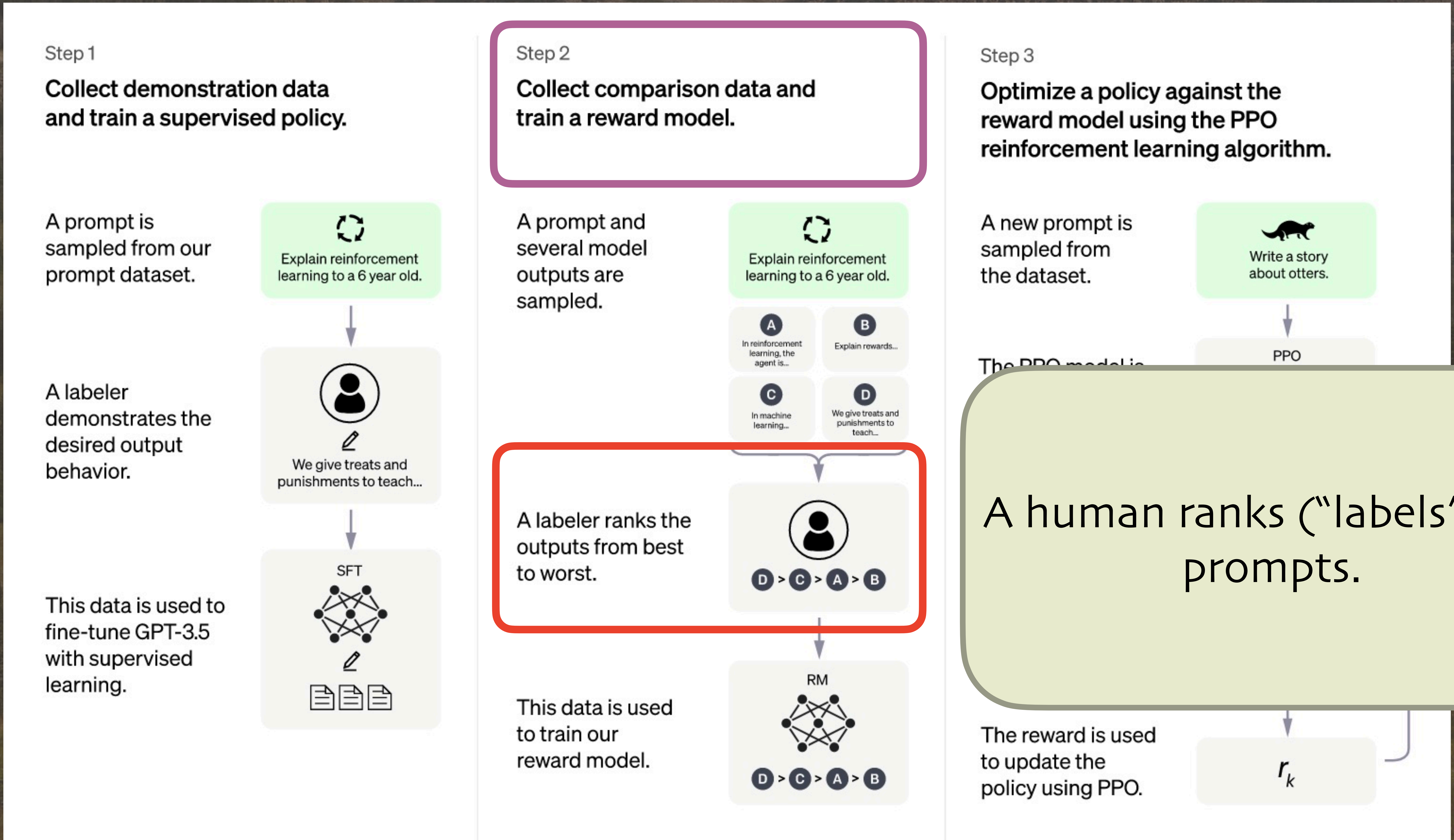
Reinforcement Learning from Human Feedback



For a given prompt, collect several model-generated outputs.

Credit: openai.com/blog/chatgpt

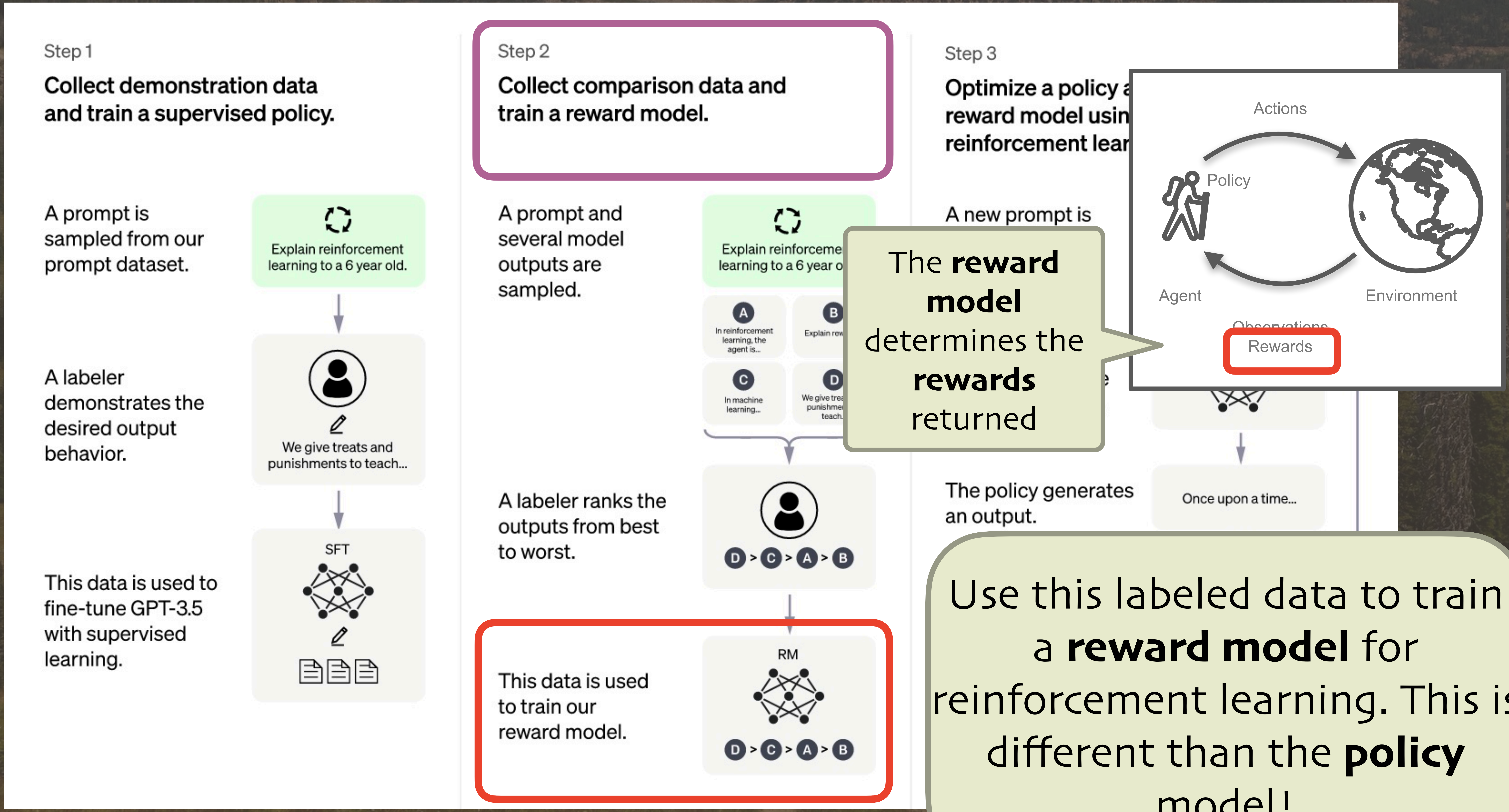
Reinforcement Learning from Human Feedback



A human ranks ("labels") the prompts.

Credit: openai.com/blog/chatgpt

Reinforcement Learning from Human Feedback

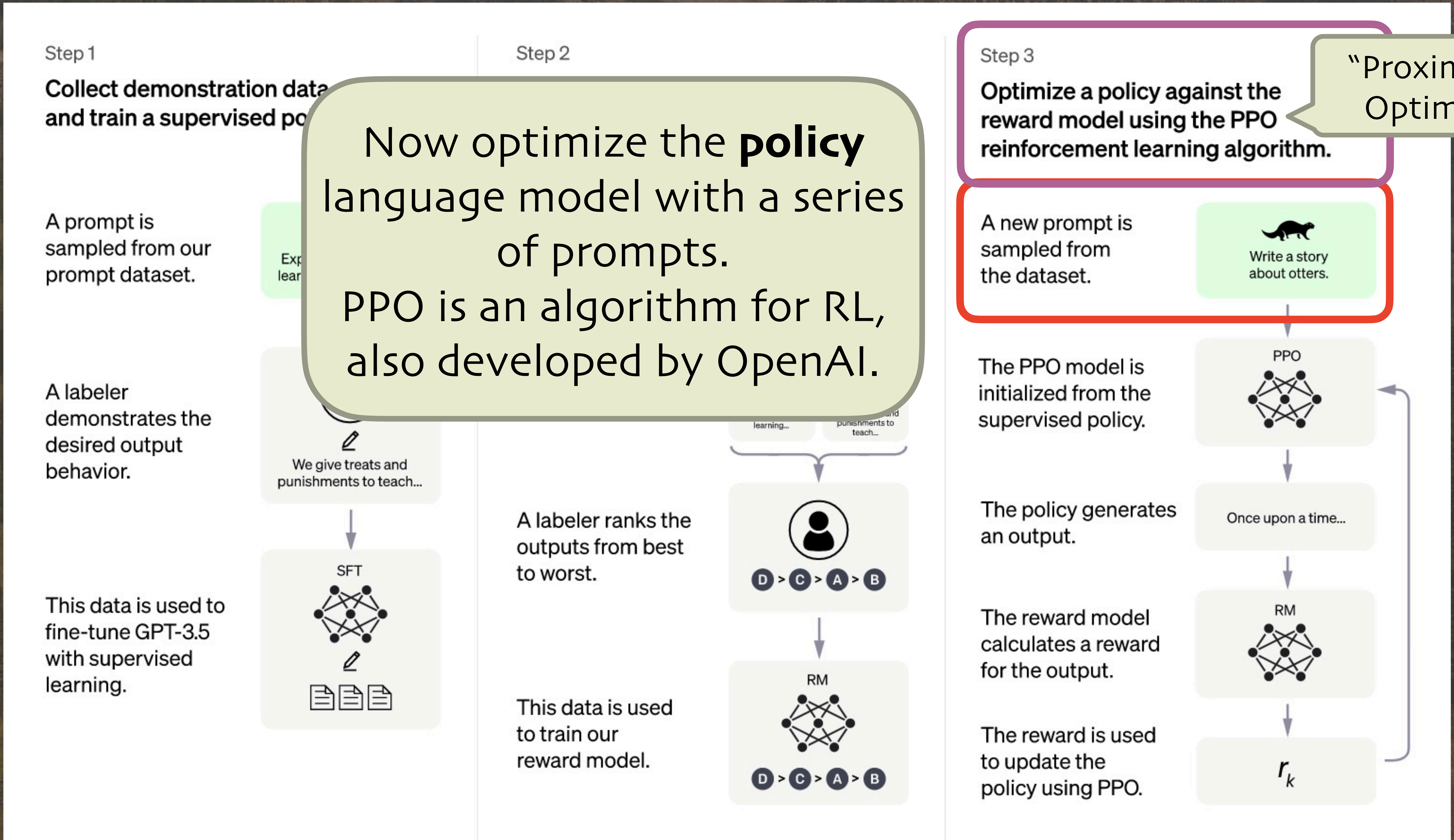


The **reward model** determines the **rewards** returned

Use this labeled data to train a **reward model** for reinforcement learning. This is different than the **policy** model!

Credit: openai.com/blog/chatgpt

Reinforcement Learning from Human Feedback

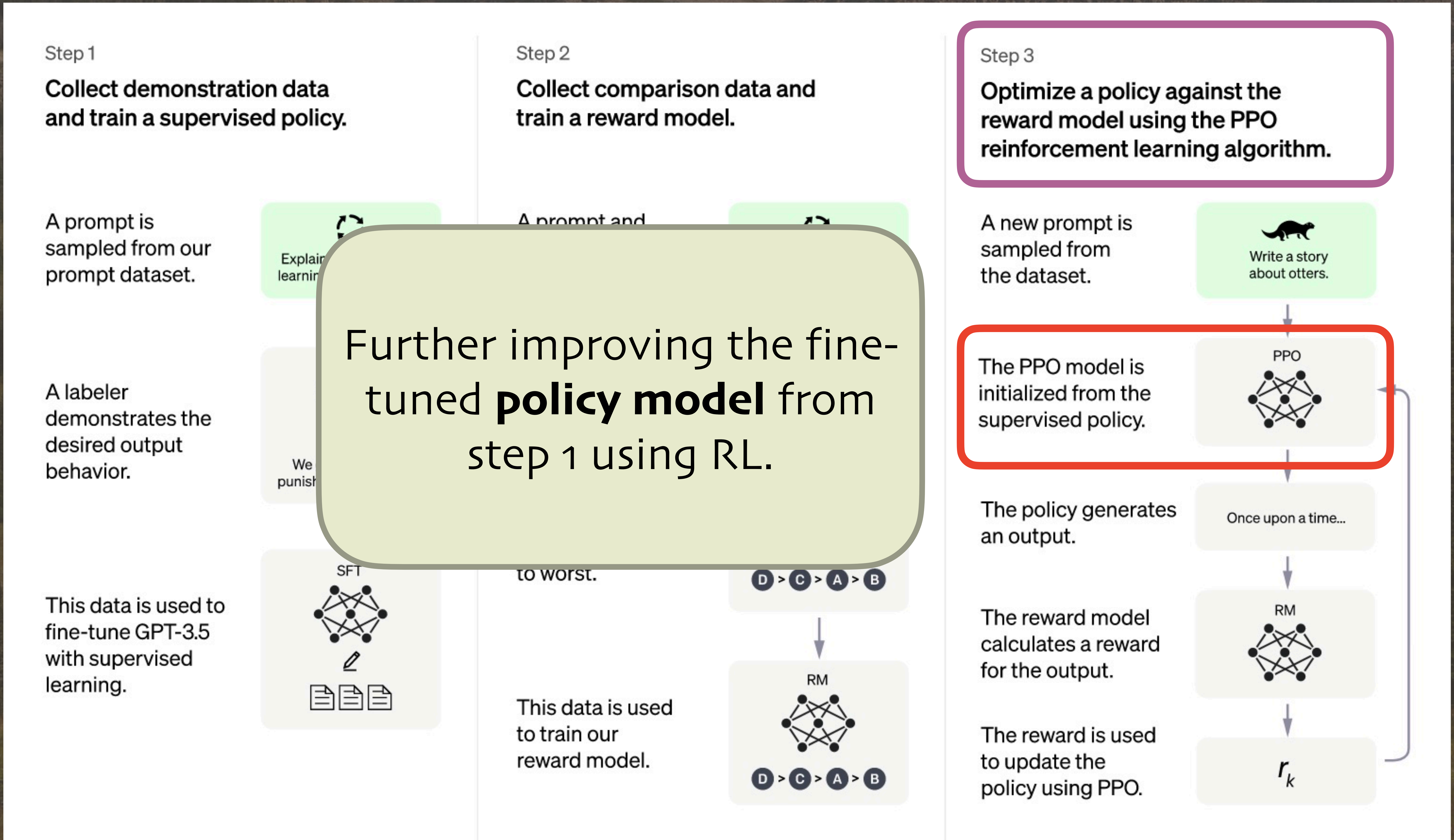


Now optimize the **policy** language model with a series of prompts.
PPO is an algorithm for RL, also developed by OpenAI.

"Proximal Policy Optimization"

Credit: openai.com/blog/chatgpt

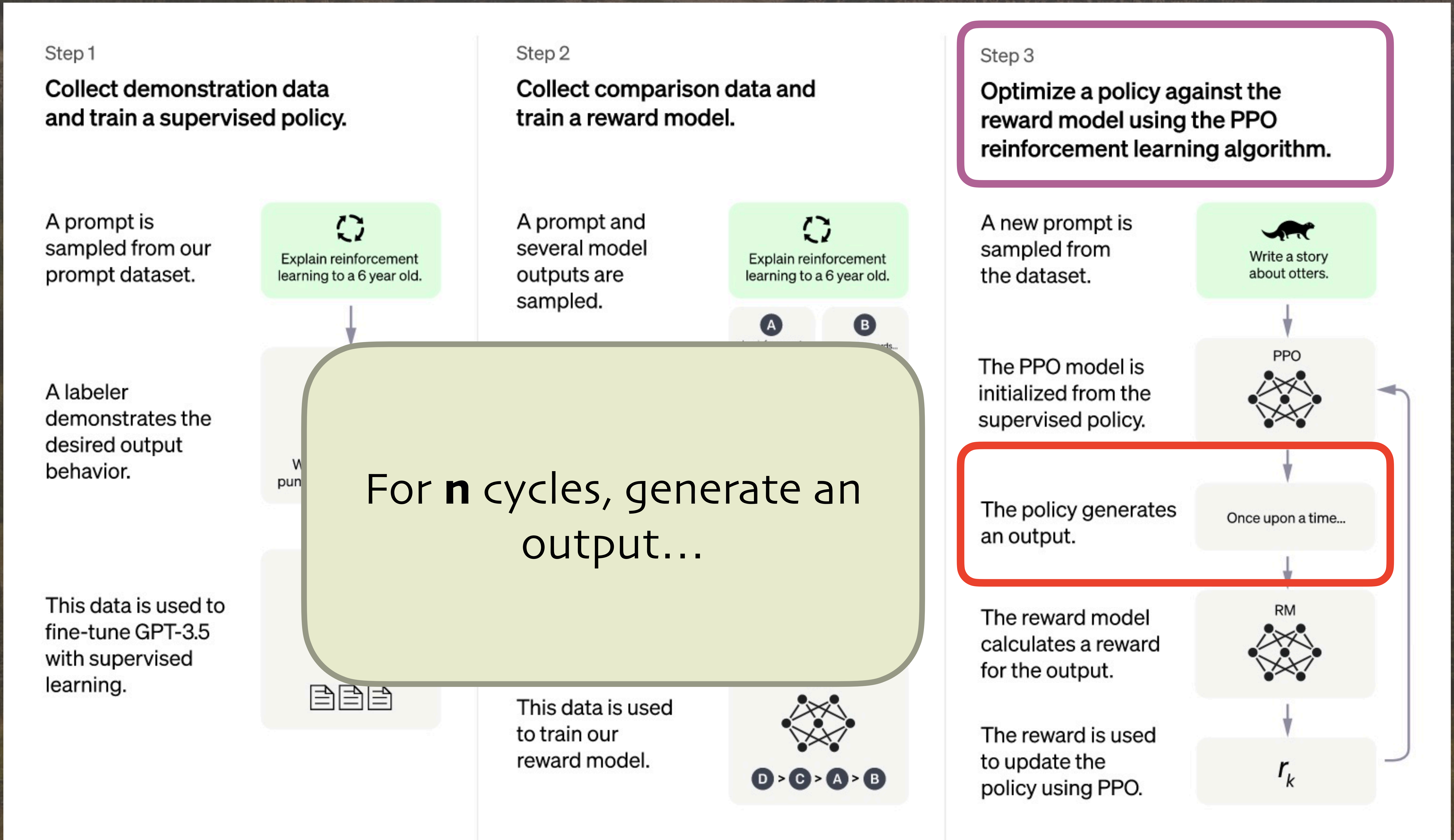
Reinforcement Learning from Human Feedback



Further improving the fine-tuned **policy model** from step 1 using RL.

Credit: openai.com/blog/chatgpt

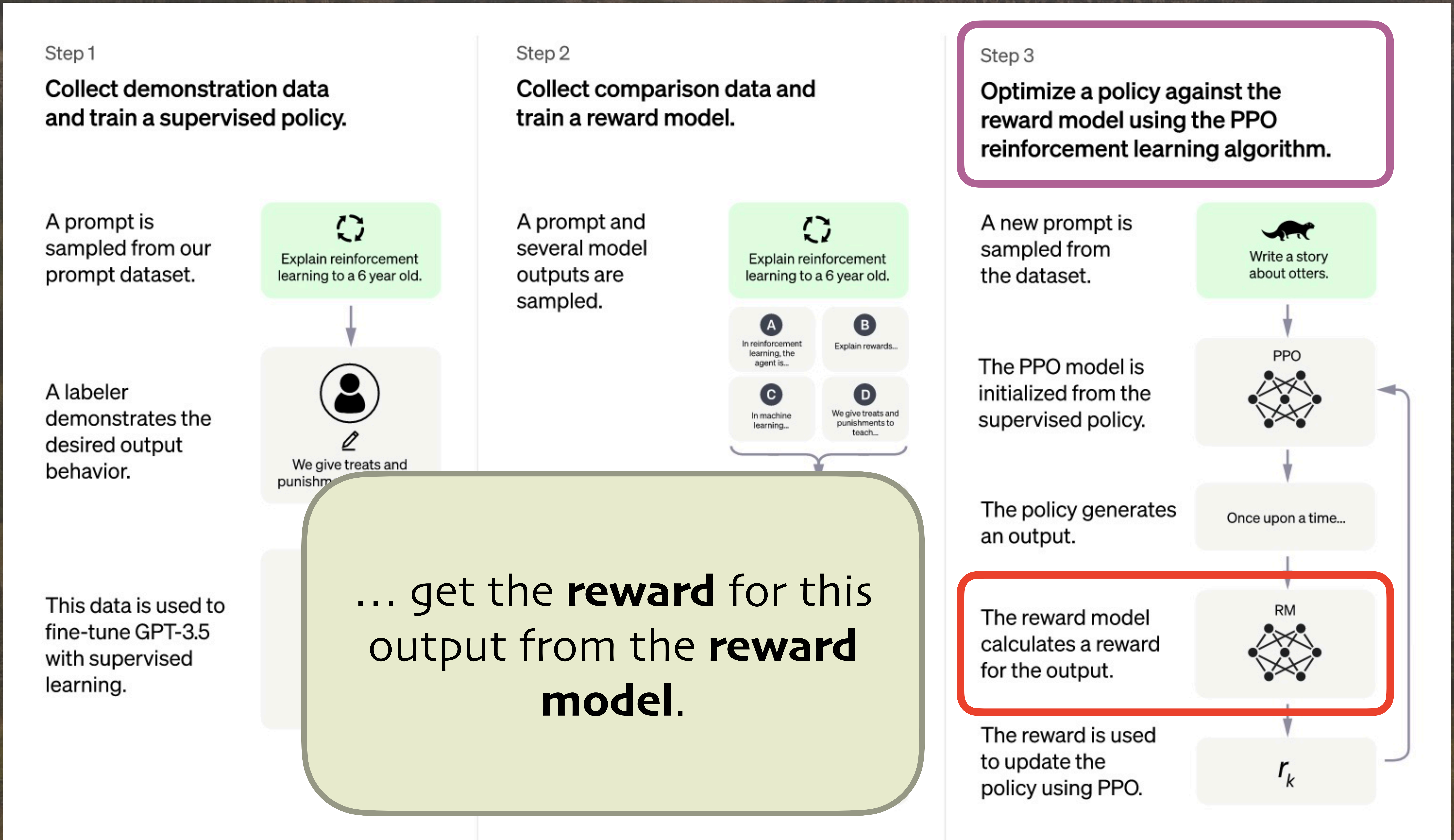
Reinforcement Learning from Human Feedback



For **n** cycles, generate an output…

Credit: openai.com/blog/chatgpt

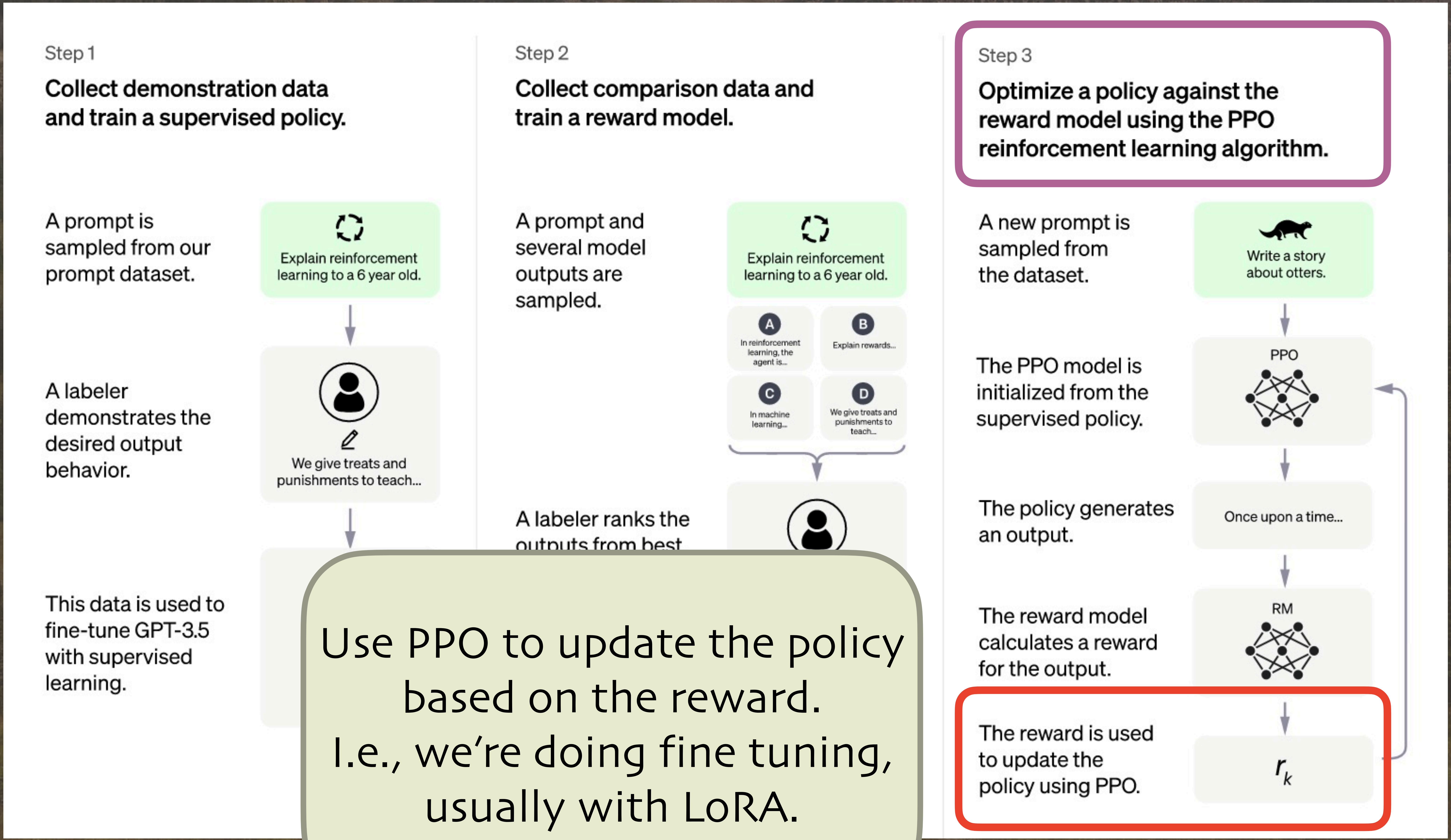
Reinforcement Learning from Human Feedback



… get the **reward** for this output from the **reward model**.

Credit: openai.com/blog/chatgpt

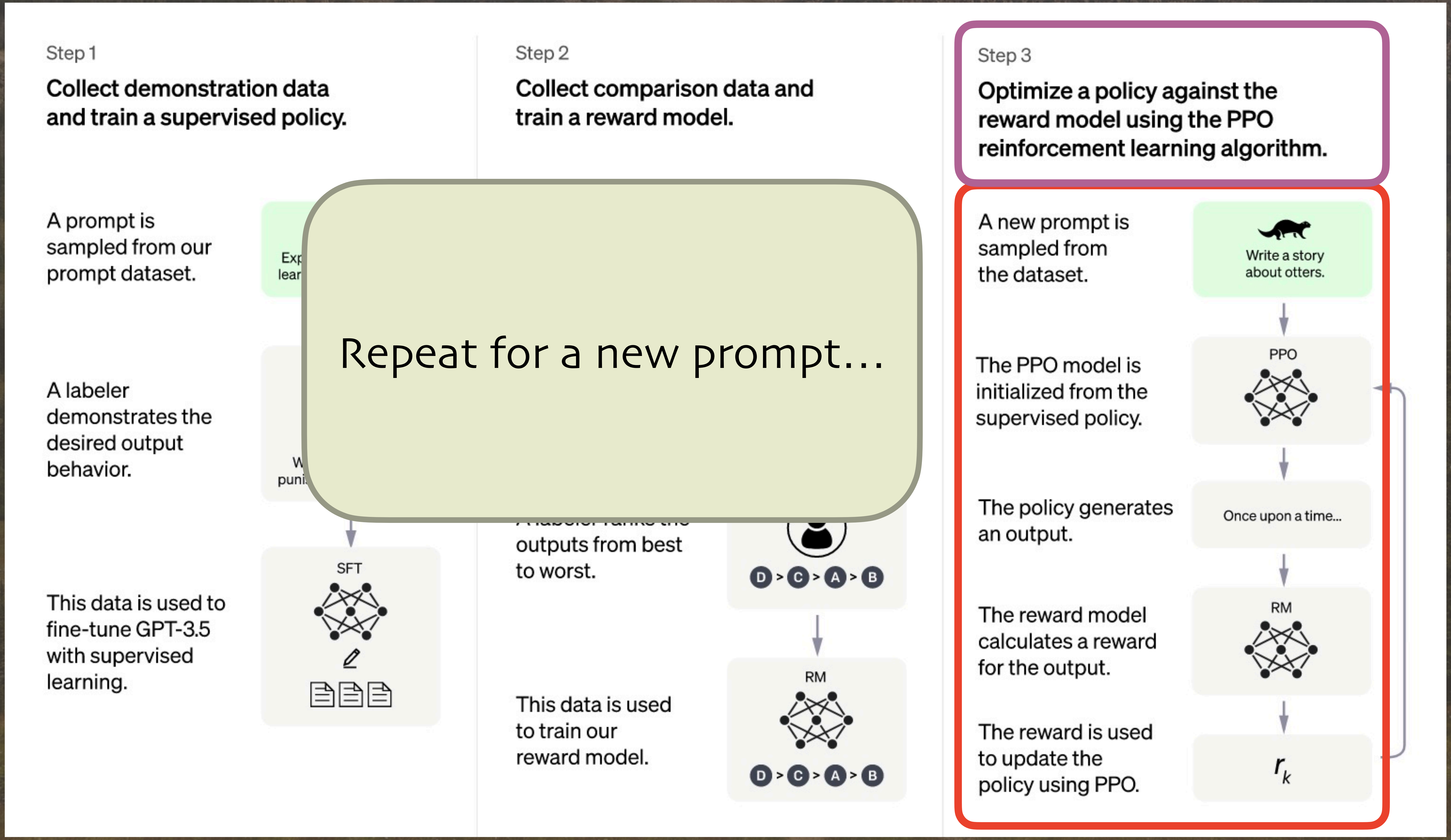
Reinforcement Learning from Human Feedback



Use PPO to update the policy based on the reward.
I.e., we're doing fine tuning, usually with LoRA.

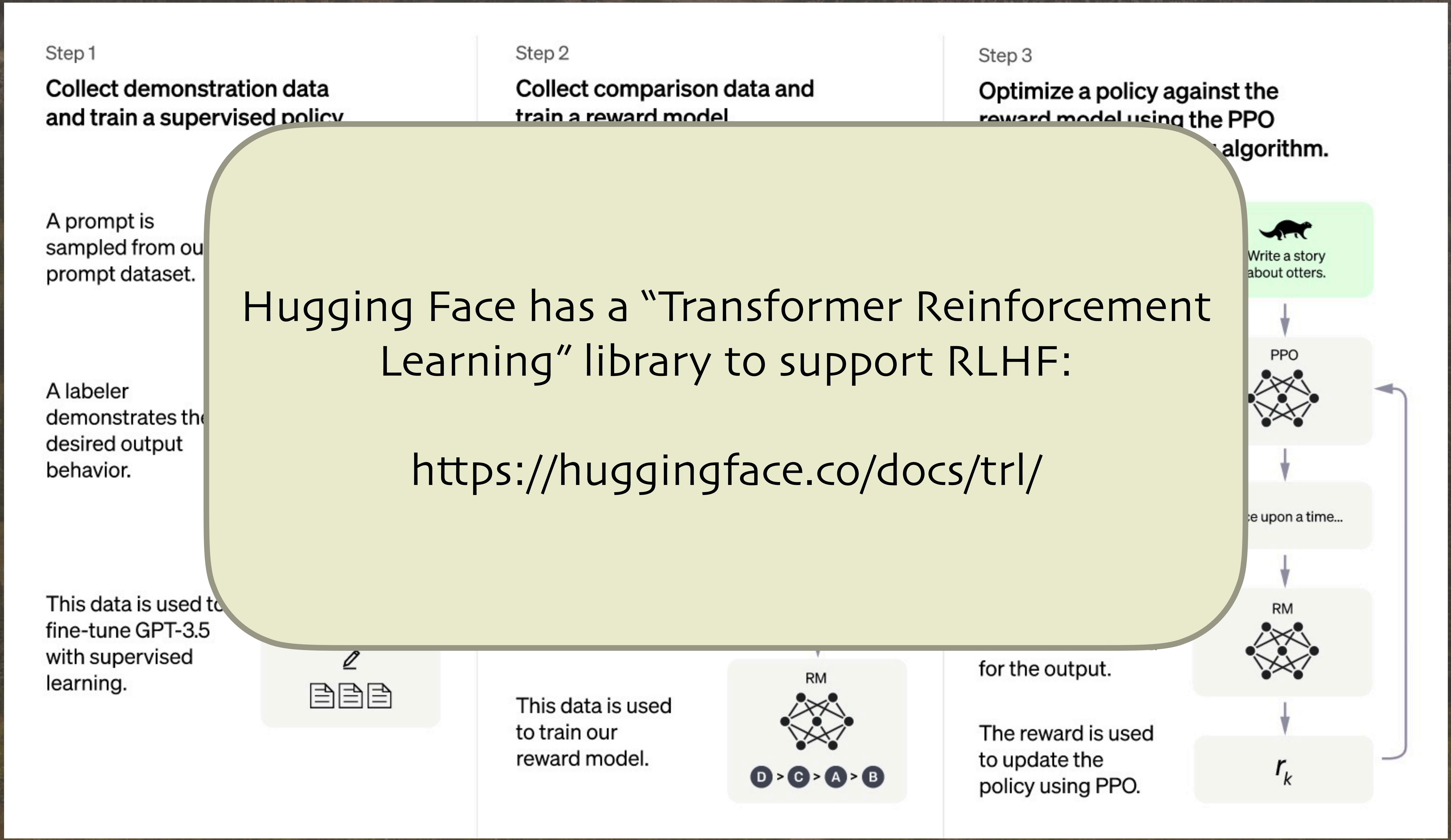
gpt

Reinforcement Learning from Human Feedback



Credit: openai.com/blog/chatgpt

Reinforcement Learning from Human Feedback



Hugging Face has a "Transformer Reinforcement Learning" library to support RLHF:

https://huggingface.co/docs/trl/

Credit: openai.com/blog/chatgpt

Reinforcement Learning from Human Feedback

- However, "human in the loop" techniques are expensive and error prone.
- What if we replace the human part with AI??

arXiv > cs > arXiv:2309.00267

Search...

Help | Adva

Computer Science > Computation and Language

[Submitted on 1 Sep 2023]

RLAIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, Abhinav Rastogi

Reinforcement learning from human feedback (RLHF) is effective at aligning large language models (LLMs) to human preferences, but gathering high quality human preference labels is a key bottleneck. We conduct a head-to-head comparison of RLHF vs. RL from AI Feedback (RLAIF) – a technique where preferences are labeled by an off-the-shelf LLM in lieu of humans, and we find that they result in similar improvements. On the task of summarization, human evaluators prefer generations from both RLAIF and RLHF over a baseline supervised fine-tuned model in ~70% of cases. Furthermore, when asked to rate RLAIF vs. RLHF summaries, humans prefer both at equal rates. These results suggest that RLAIF can yield human-level performance, offering a potential solution to the scalability limitations of RLHF.

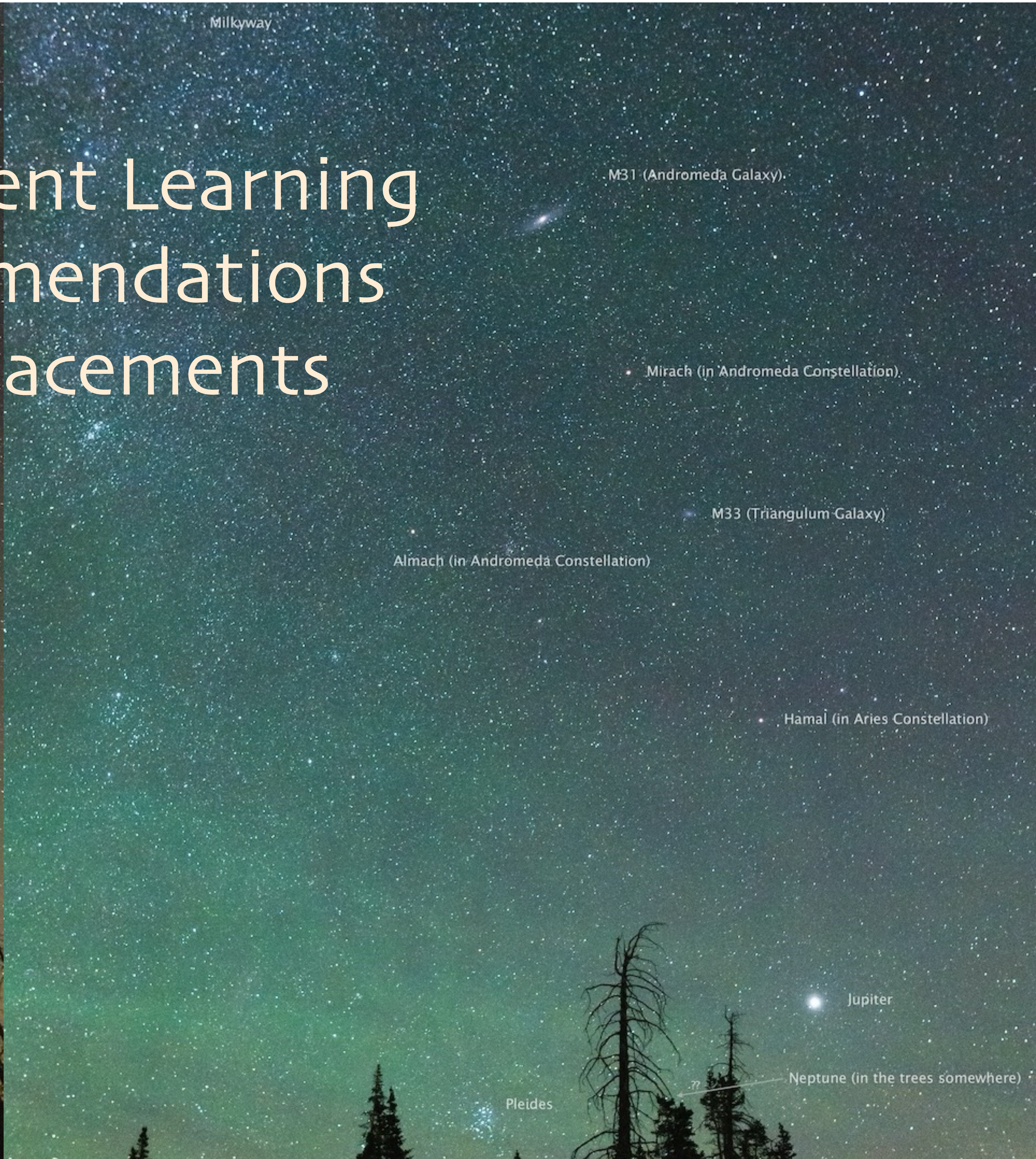
Subjects: **Computation and Language (cs.CL)**; Artificial Intelligence (cs.AI); Machine Learning (cs.LG)

Cite as: arXiv:2309.00267 [cs.CL]

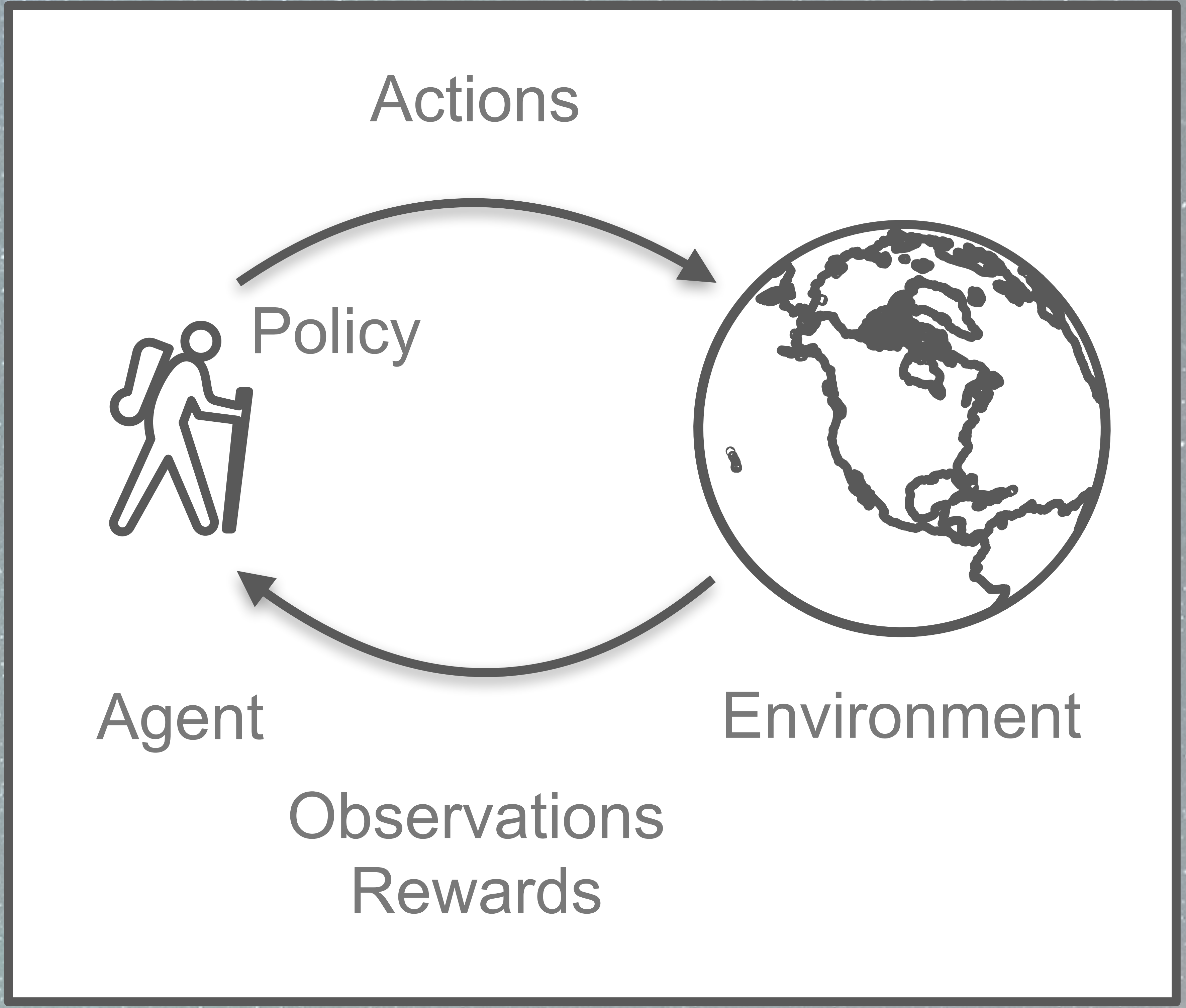
(or arXiv:2309.00267v1 [cs.CL] for this version)

https://arxiv.org/abs/2309.00267

Reinforcement Learning for Recommendations and Ad Placements



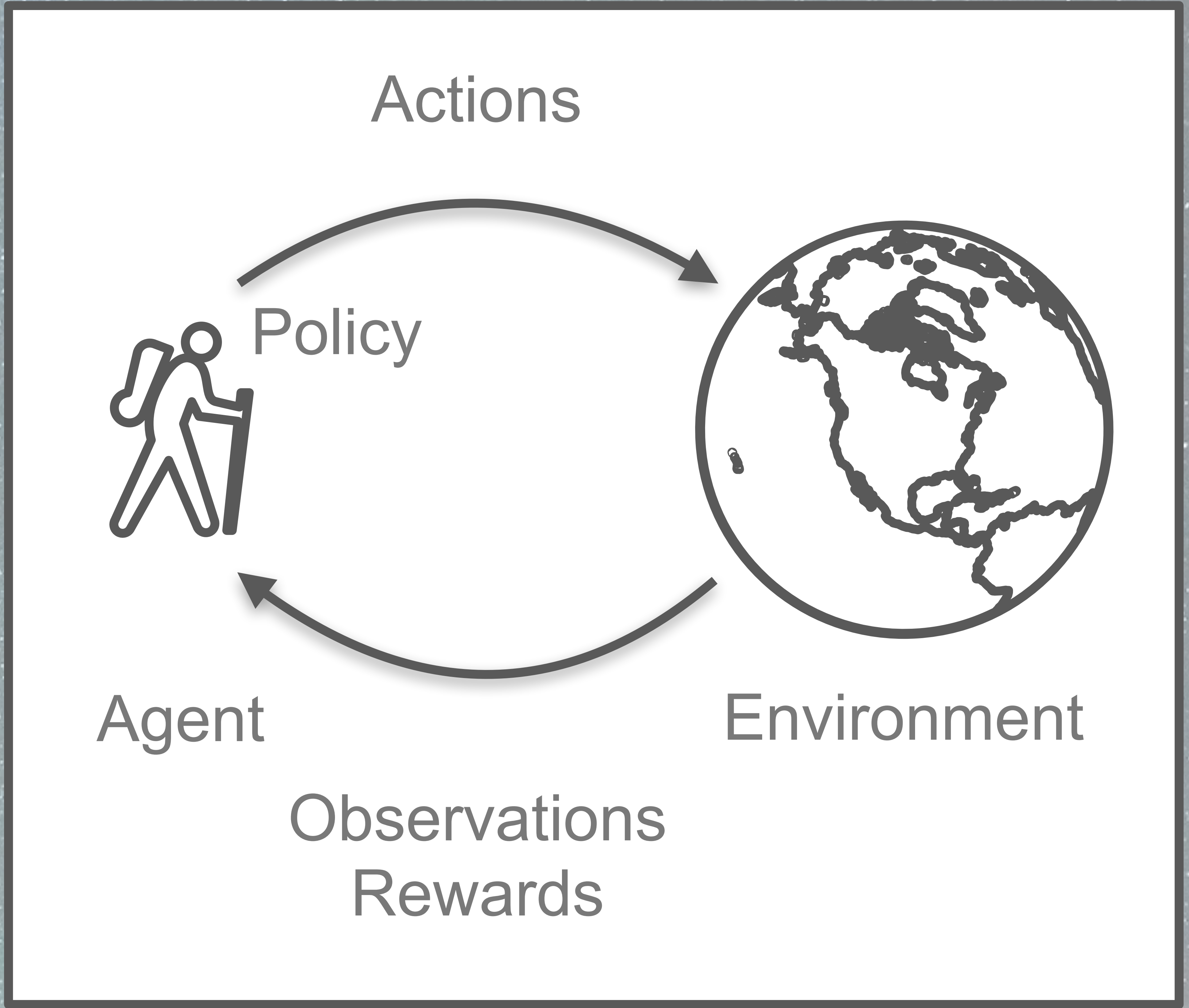
Preferences Change…



- How often has this happened to you?
  - You bought a toilet brush on Amazon…
    - … Do you want to keep seeing ads for toilet brushes?
- You've watched five horror movies in a row.
  - Do you want to watch a sixth horror movie or maybe watch a rom-com for a change?

Challenges

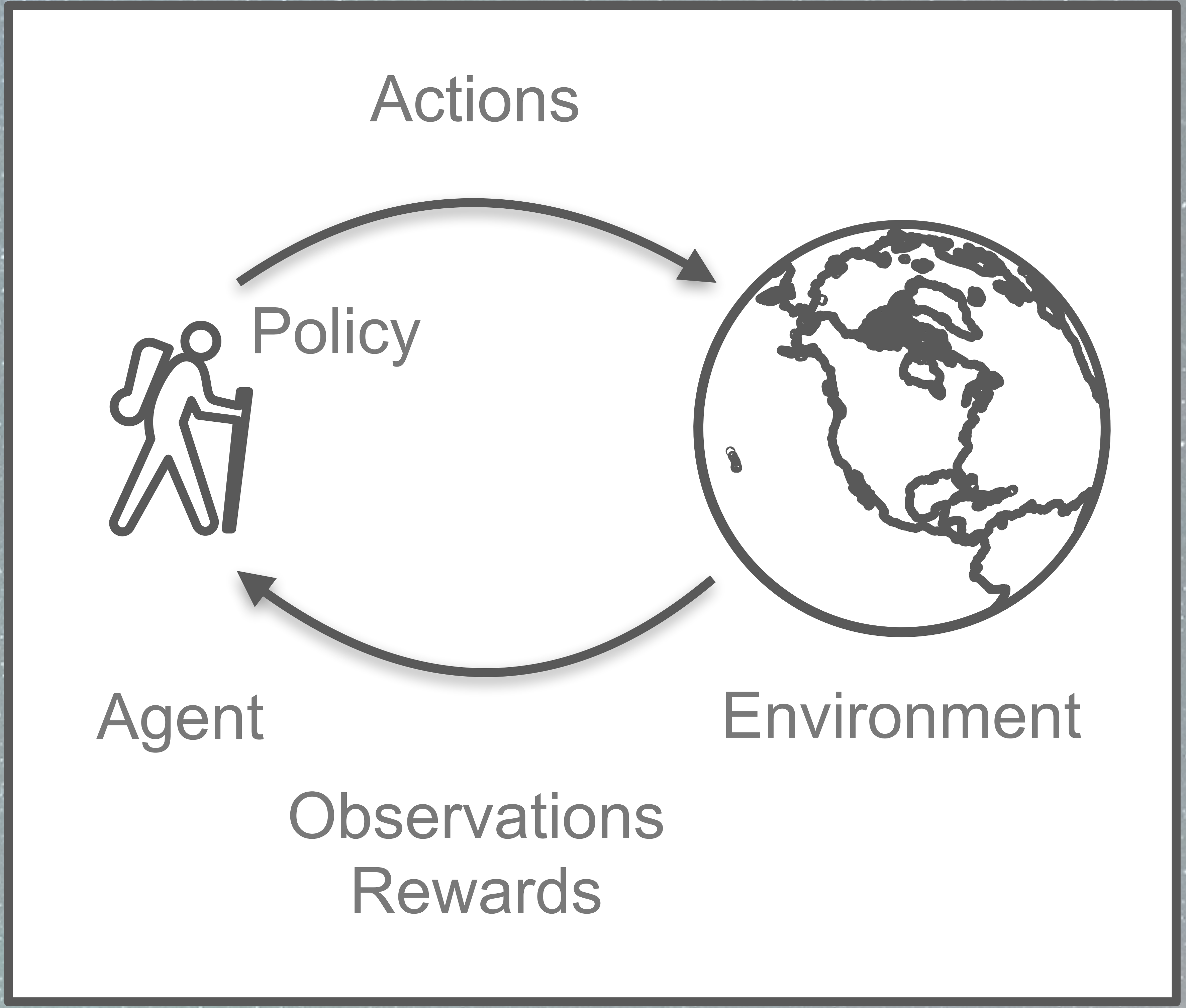
- RL is less able to scale to large state spaces (e.g., all available movies or all catalog items and all users).
- Traditional supervised learning methods are more scalable.



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.

Challenges

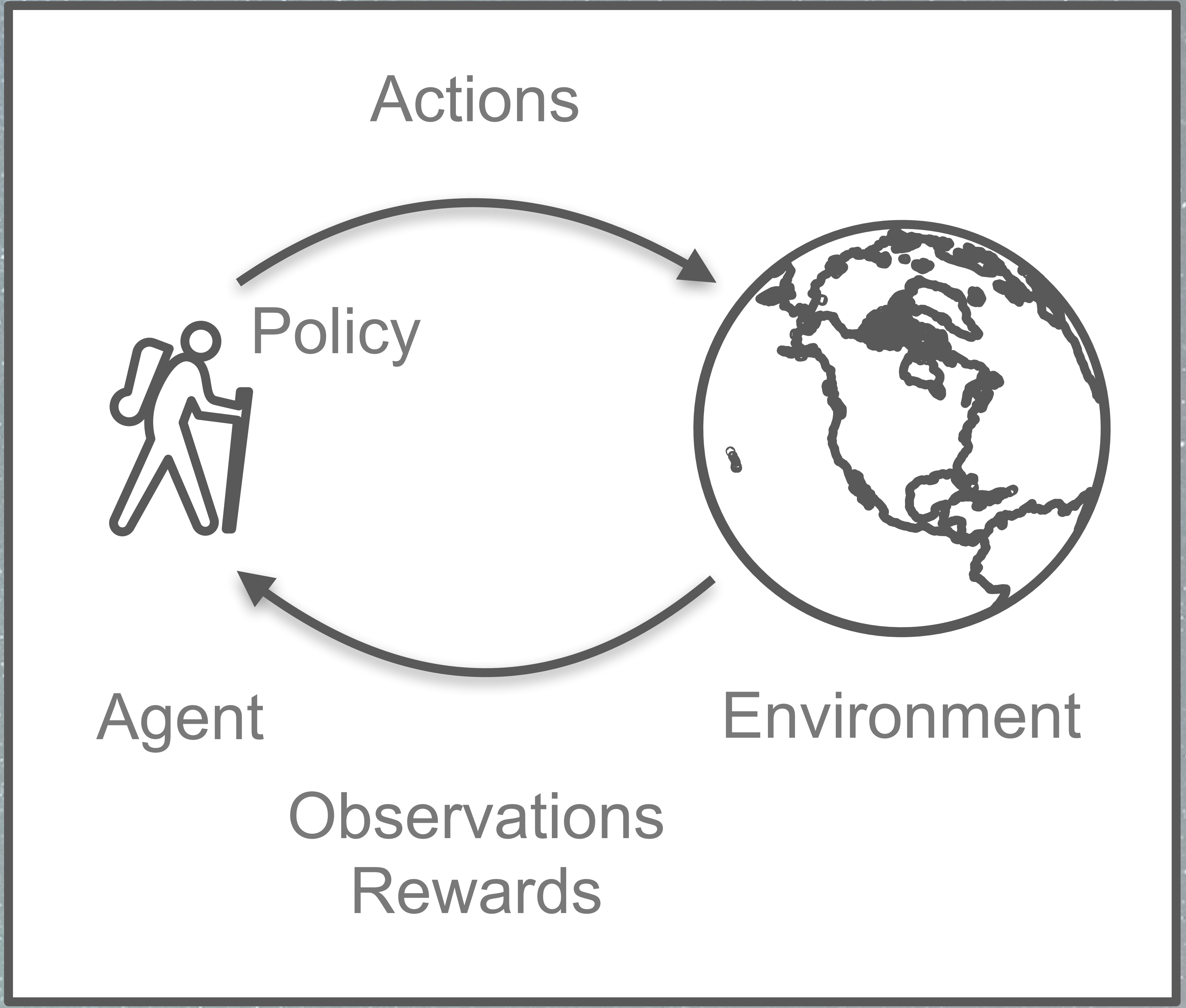
- A simulator is used to model real user behavior. (Training with real users doesn't scale well, etc.)



Or use offline RL with historical data about user behavior!

Challenges

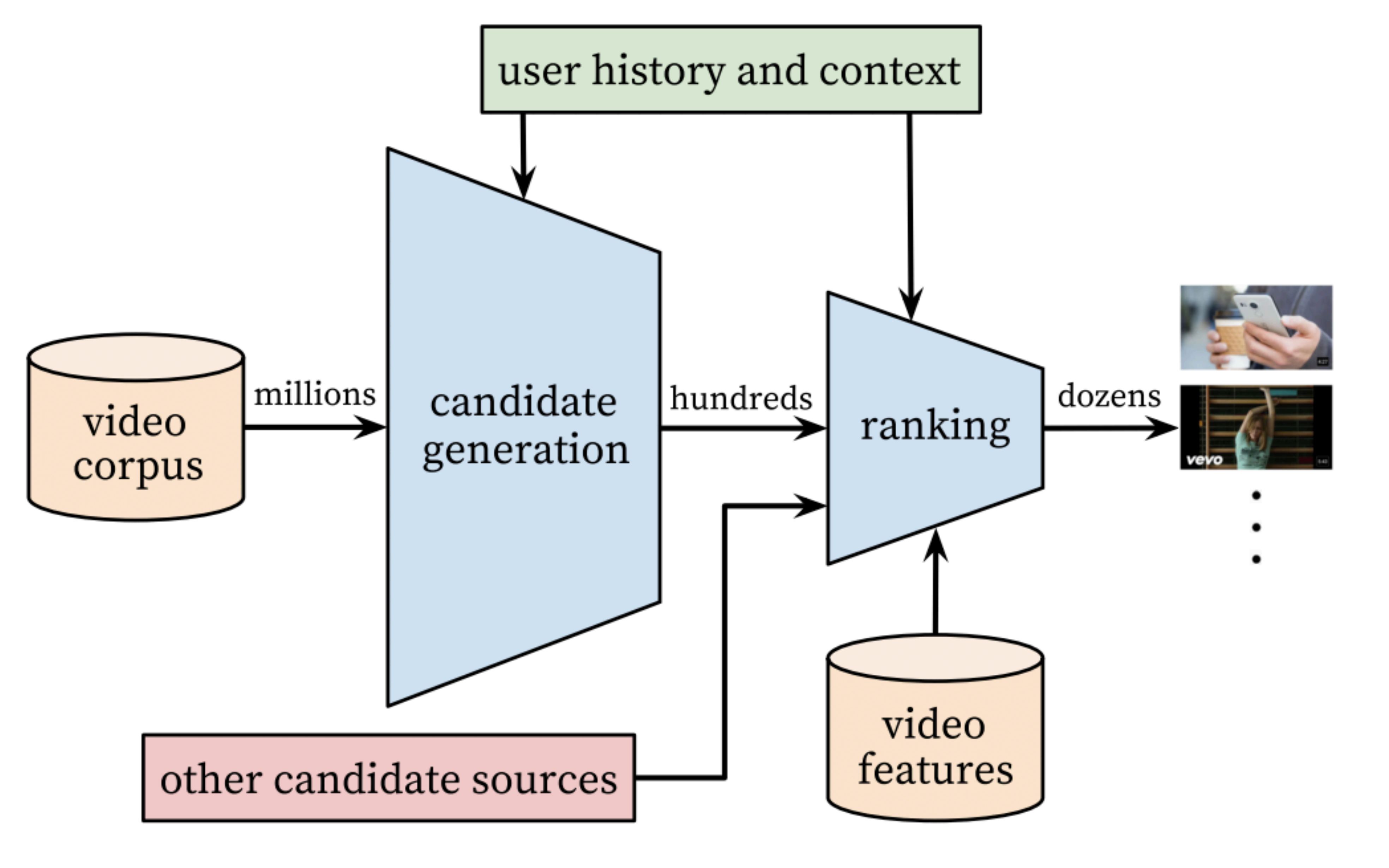
- What is the reward? Some combination of user happiness measures?
    - It could be very specific to the sub-genre of entertainment or product category.

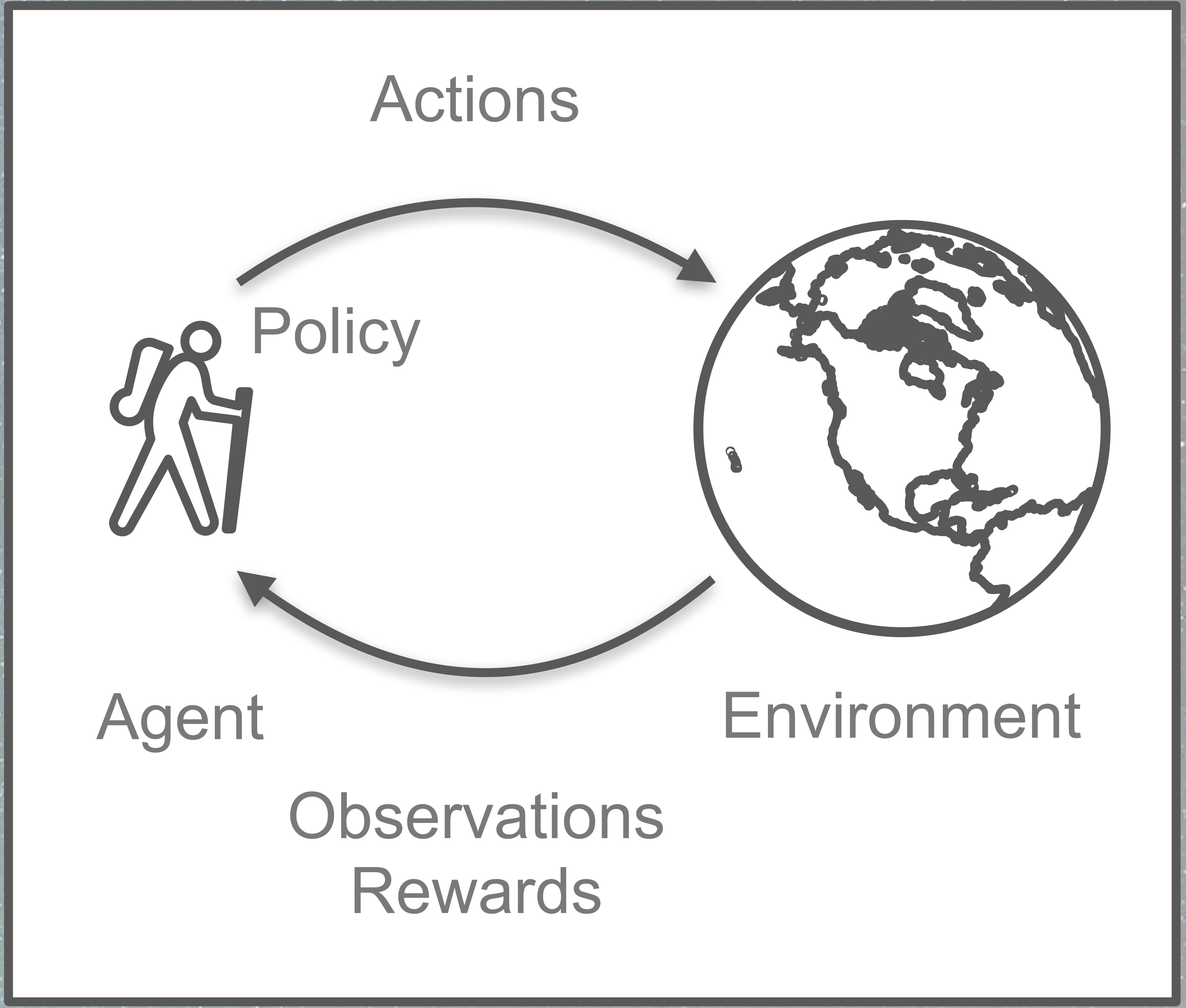


Reward calculation balances mixed preferences & tradeoffs as they evolve in response to use actions.

Challenges

- YouTube! Research:
  - research.google/pubs/pub45530/





See the Anyscale RL tutorial link at the end for a recommendation example

Milkyway

Hamal (in Aries Constellation)

Neptune (in the trees somewhere)

Pleides

To Learn More…

- rllib.io
- Hugging Face Transformer RL (for RLHF)
- Anyscale RL & RLlib course
- More resources in the extra slides!
    - Including more details about Ray
- These slides: deanwampler.com/talks

dean@deanwampler.com
@discuss.systems@deanwampler
IBM Research

Extra Material: References, Ray, …

To Learn More...

- Frameworks
  - Ray RLlib
  - Hugging Face Transformer Reinforcement Learning (TRL) - Biased towards the needs of RLHF.
- 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 coursehttps://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)
  - Illustrated RL from Human Feedback: https://huggingface.co/blog/rlhf

To Learn More…

There is a growing number of RL applications today.

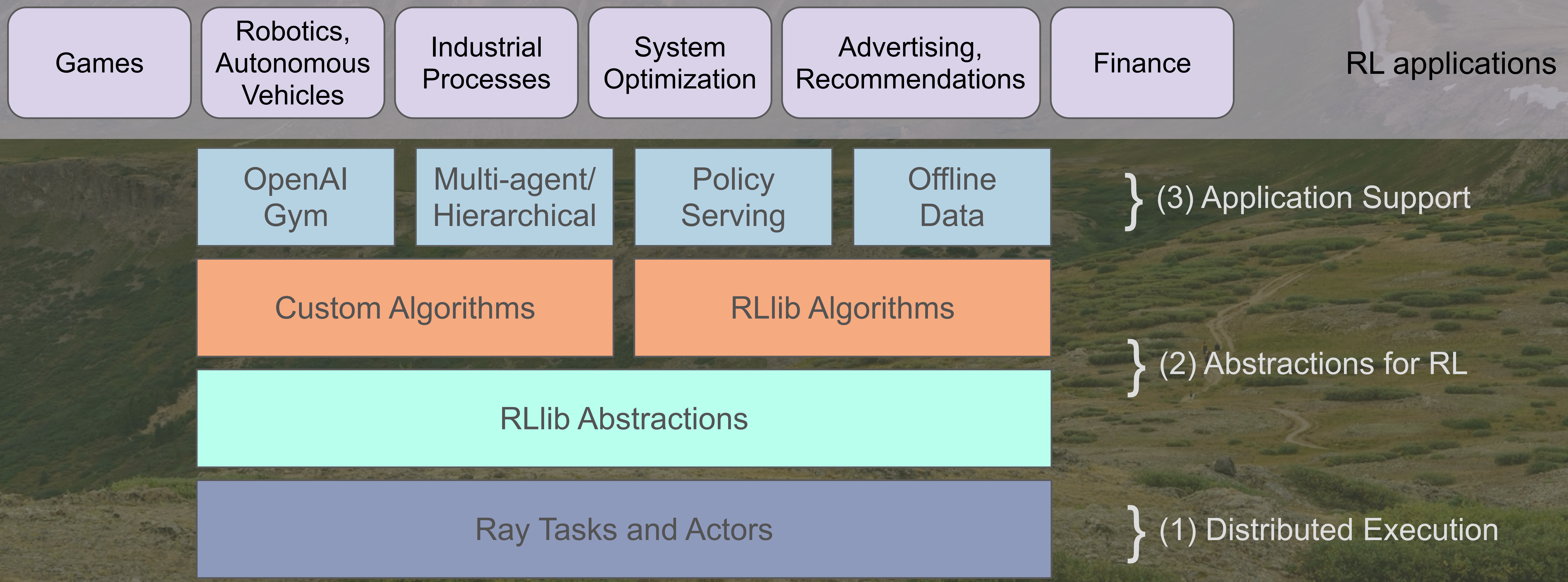
- Games: Google's AlphaGo, released in 2016, which famously beat the 18-time world champion, Lee Sedol, in the game of Go.
- Robotics and autonomous vehicles: simulated locomotion, real-world robotic manipulation, mobility, pick and place.
- Industrial control: We can teach a policy to optimize datacenter cooling and its energy consumption, factory floors (Ford assembly line, credit: https://media.ford.com/content/fordmedia/fna/us/en/features/game-changer--100th-anniversary-of-the-moving-assembly-line.html)
- System optimization: optimize database queries, cooling (Google datacenter photo: https://www.blog.google/inside-google/infrastructure/better-data-centers-through-machine/, patch cabling photo: flickr.com/photos/jerryjohn)
- Advertising, recommendations (Video recommendations on YouTube diagram: https://research.google/pubs/pub45530/)
- Finance (photo: http://tradinghub.co/watch-list-for-mar-26th-2015/)
- ChatGPT: https://openai.com/blog/chatgpt
  - RLHF: openai.com/blog/chatgpt, https://huggingface.co/blog/rlhf, https://huggingface.co/blog/the_n_implementation_details_of_rlhf_with_ppo
  - RLAIF: https://arxiv.org/abs/2309.00267

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

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

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)

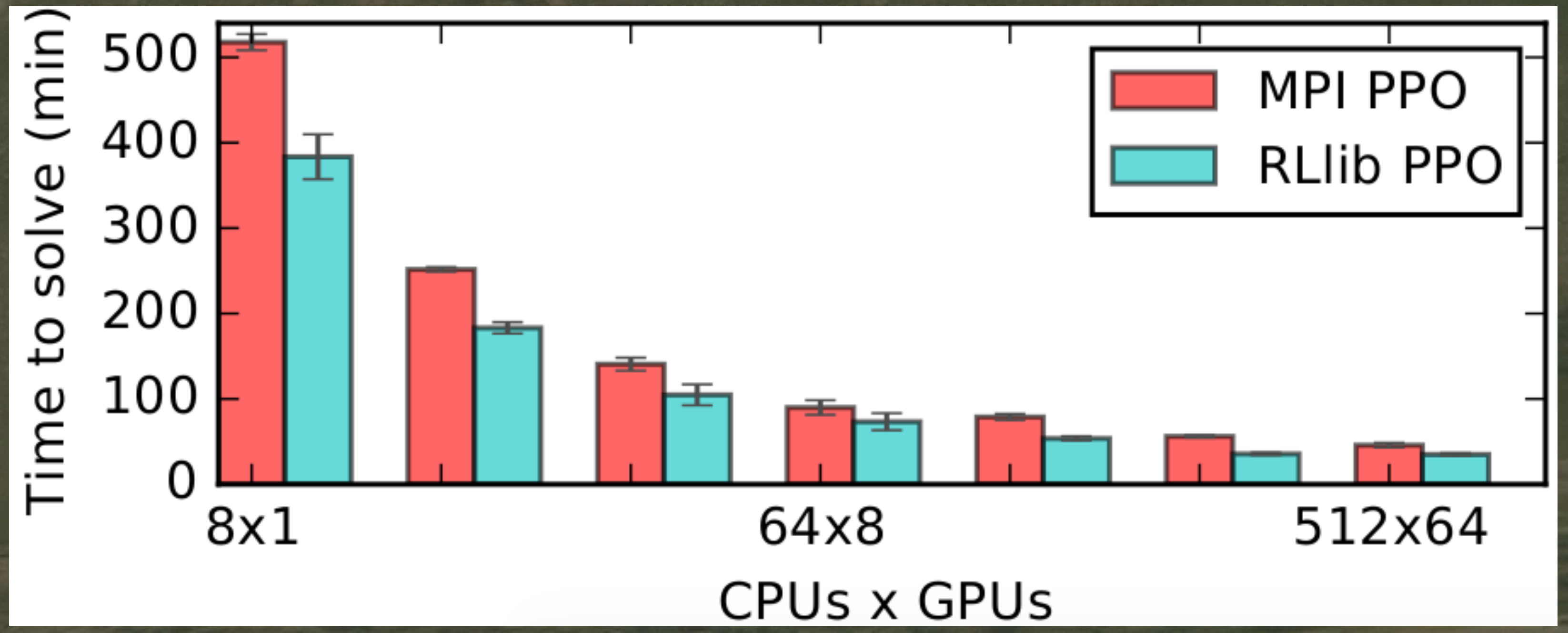
- gradient-free
  - Augmented Random Search (ARS)
  - Evolution Strategies

- Multi-agent specific
  - QMIX Monotonic Value Factorisation (QMIX, VDN, IQN)

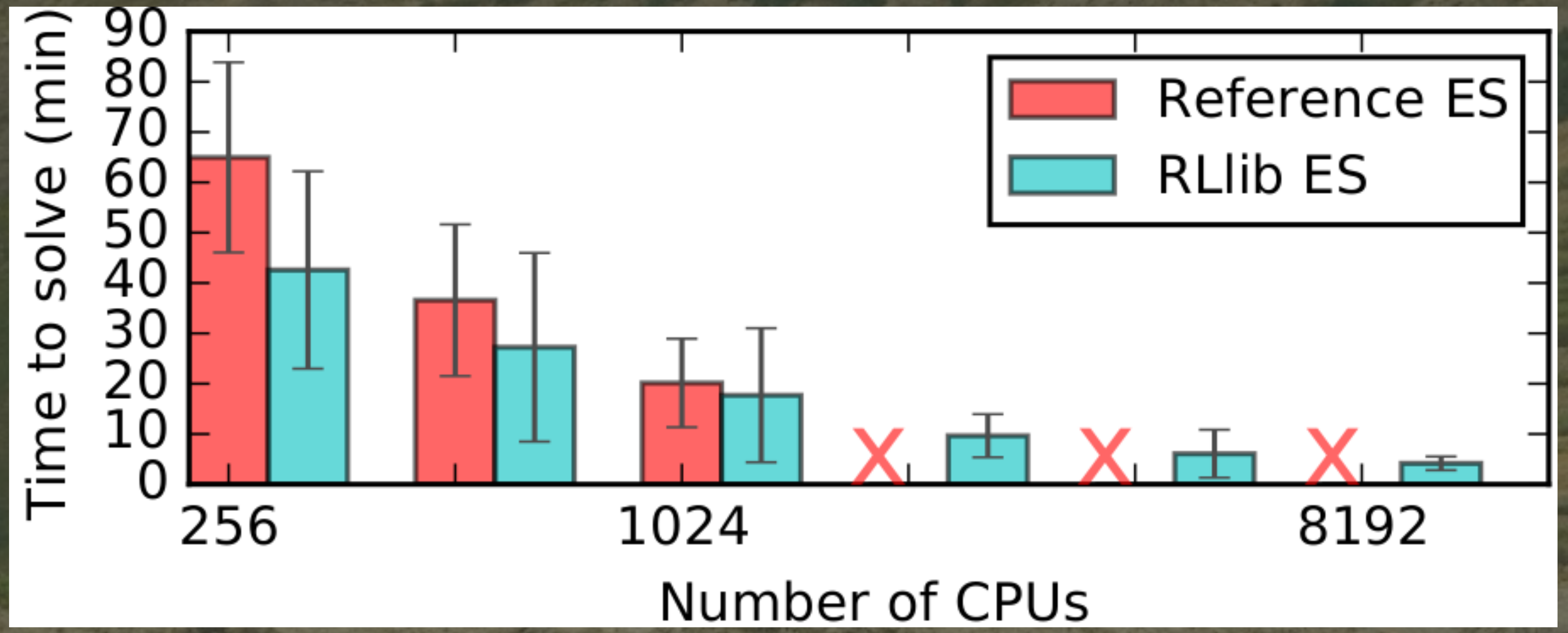
- Offline
  - Advantage Re-Weighted Imitation Learning (MARWIL)

Excellent Performance vs. "Hand-tuned" Implementations

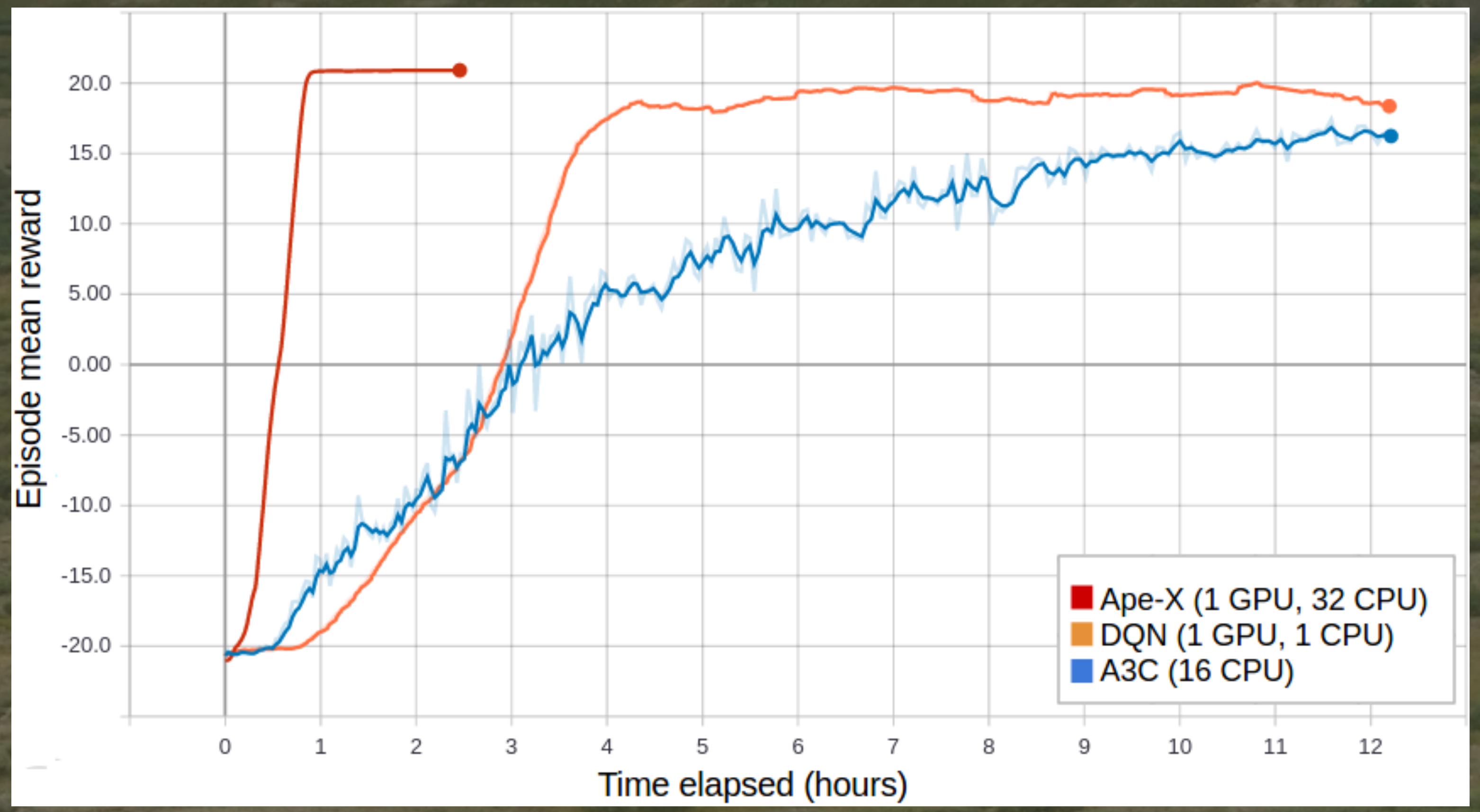
Distributed PPO



Evolution Strategies



Ape-X Distributed DQN, DDPG



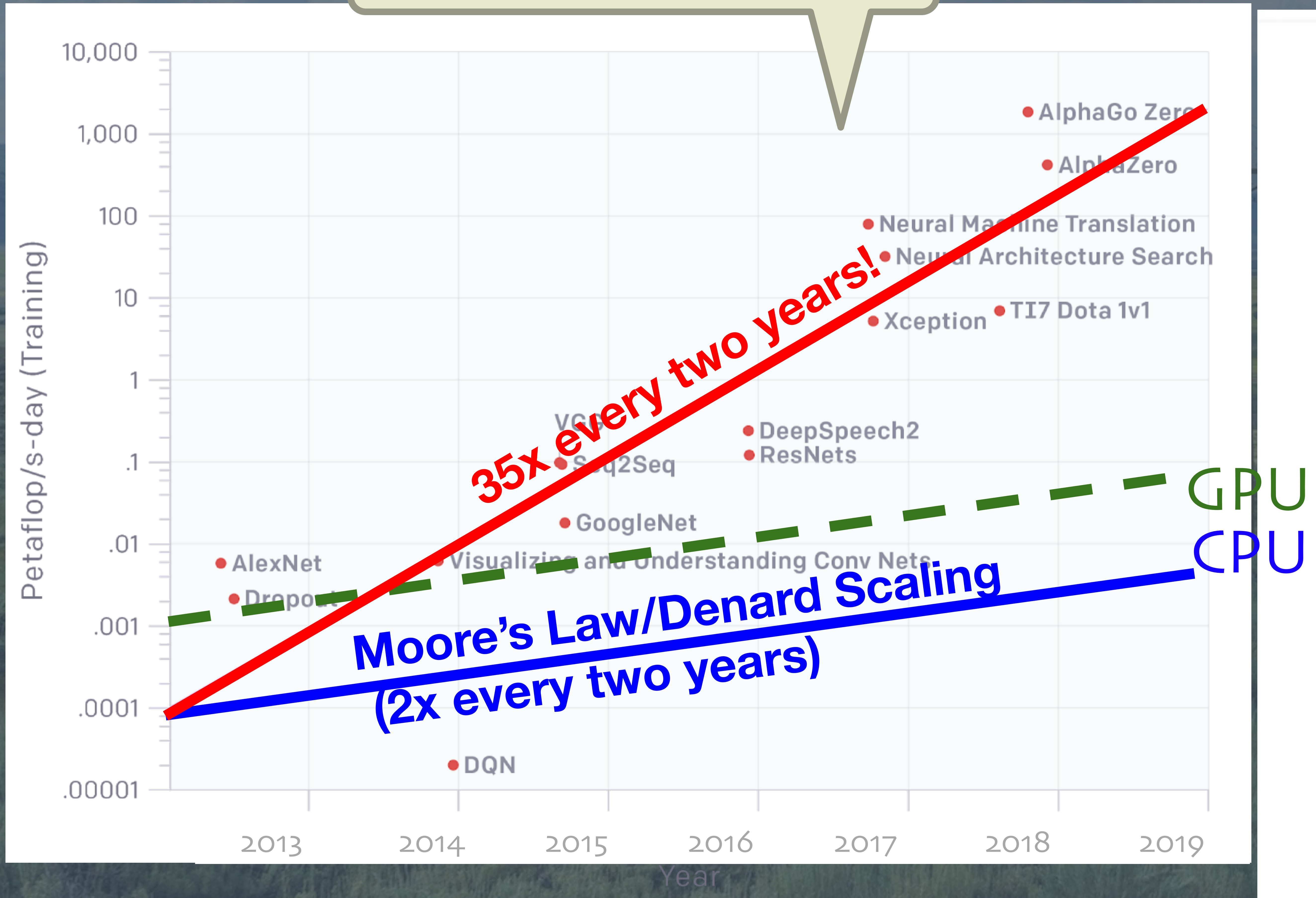

Why Ray??

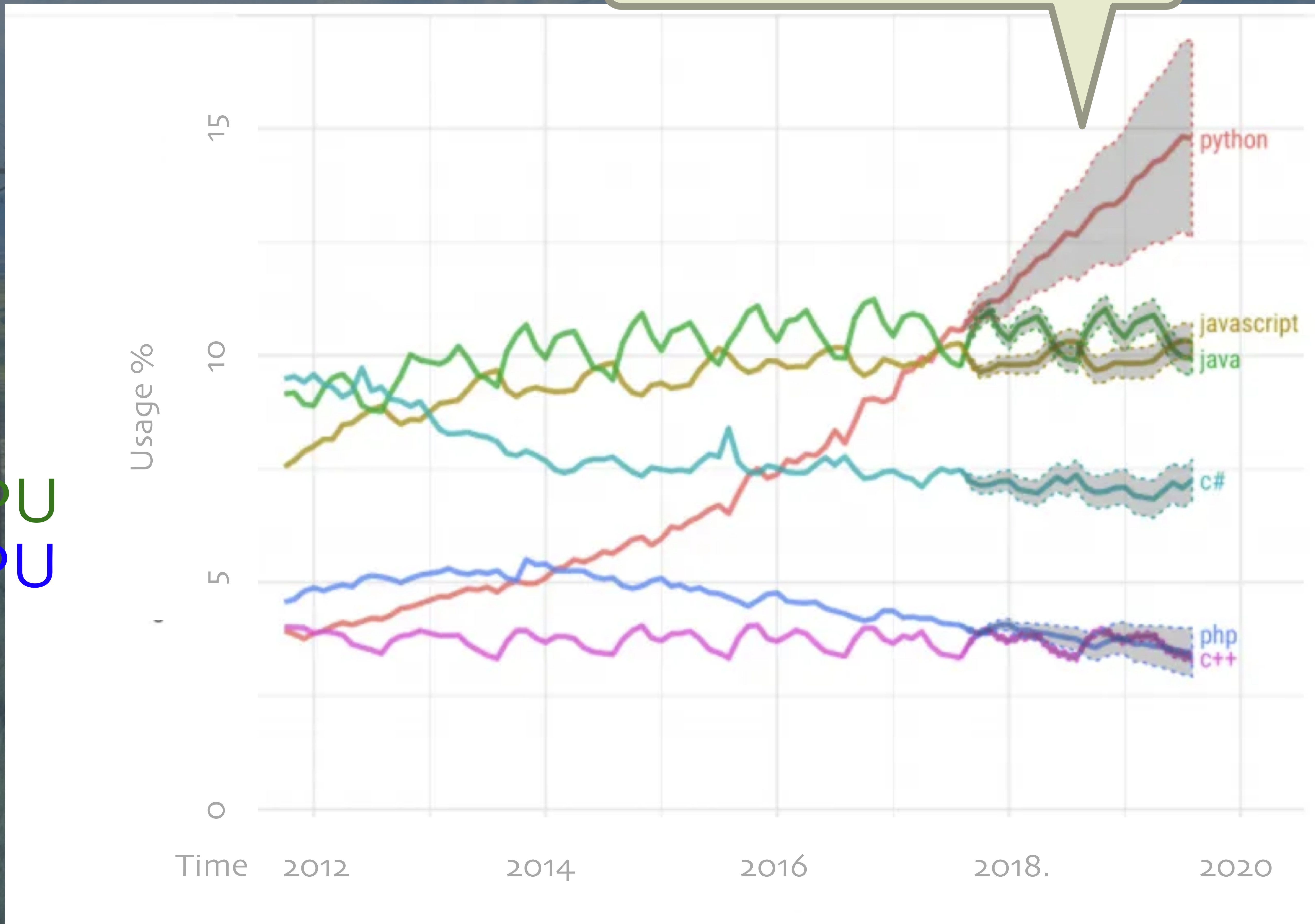
Two Trends

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





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

The Overall Data & ML Landscape Today

All require distributed implementations to scale

ETL	Streaming	HPO Tuning	Training	Simulation	Model Serving
Apache Spark	Apache Spark, Flink, kafka	SIGOPT, talos AUTONOMY	Horovod, PyTorch, scikit learn	MPI, akka, Orleans	Clipper

The Ray Vision: a Common Framework

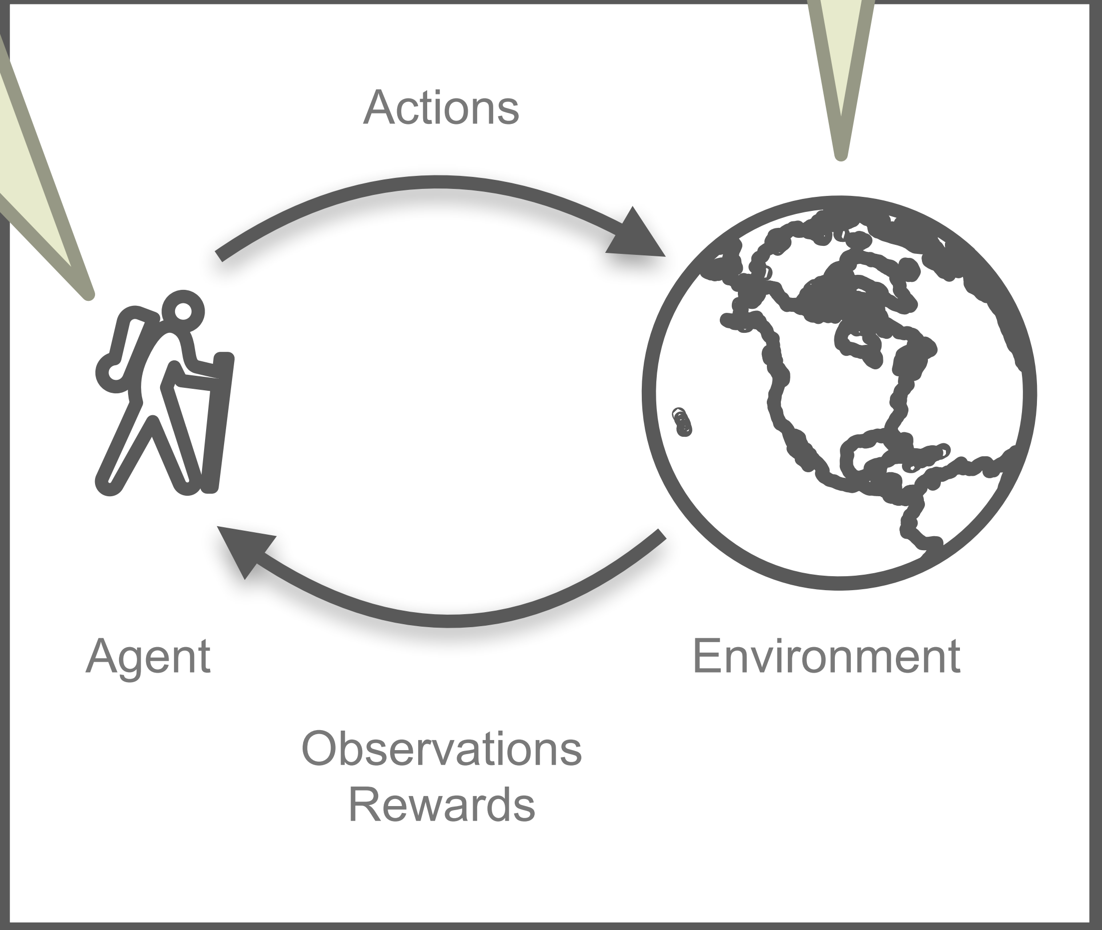
Diverse Compute Requirements Motivated the Creation of Ray!

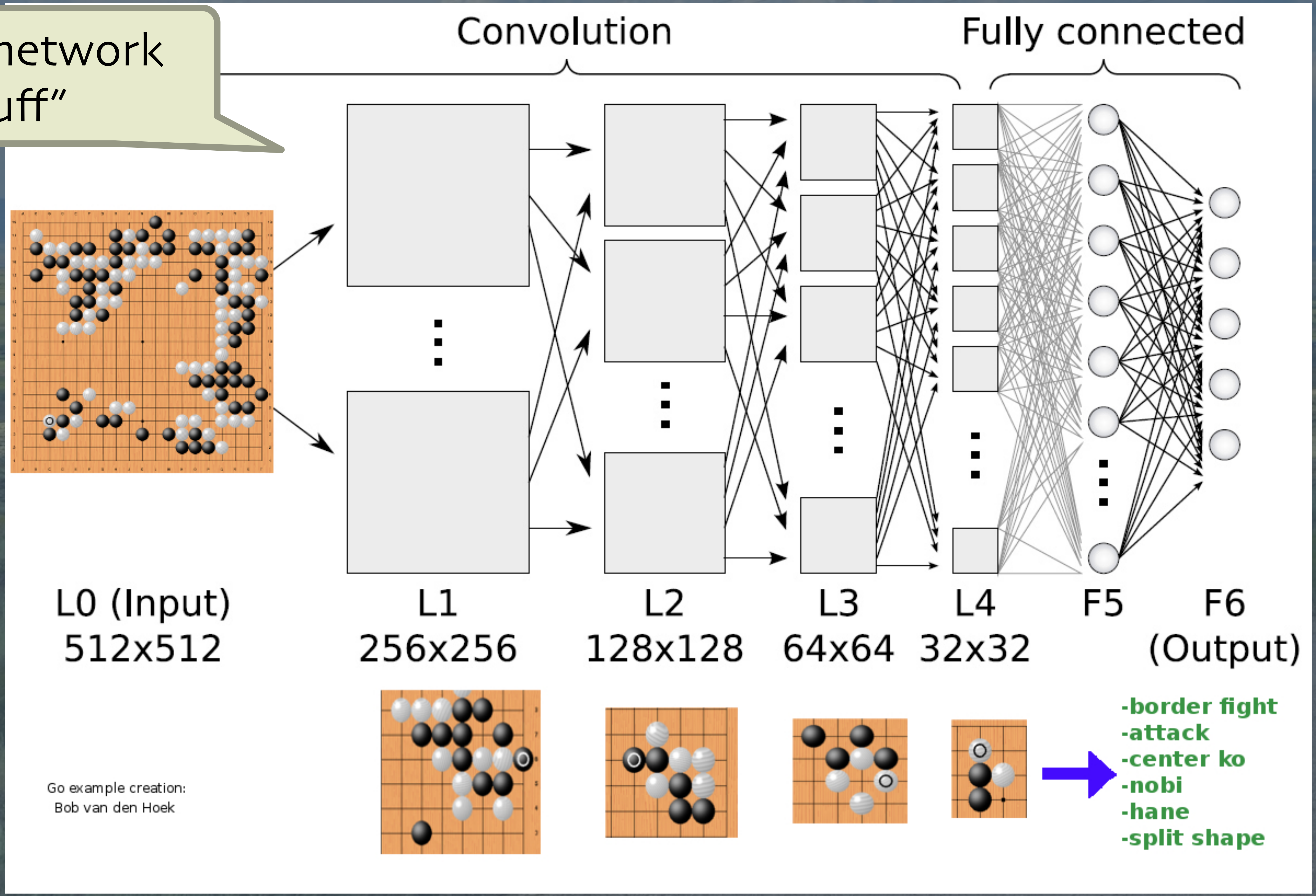
And repeated play, over and over again, to train for achieving the best reward

Neural network "stuff"

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

Complex agent?





Quick Intro to the Ray API

https://docs.ray.io/en/latest/

An Intuitive and Concise API

```
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…

An Intuitive and Concise API

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)
```

Now these functions are remote “tasks"

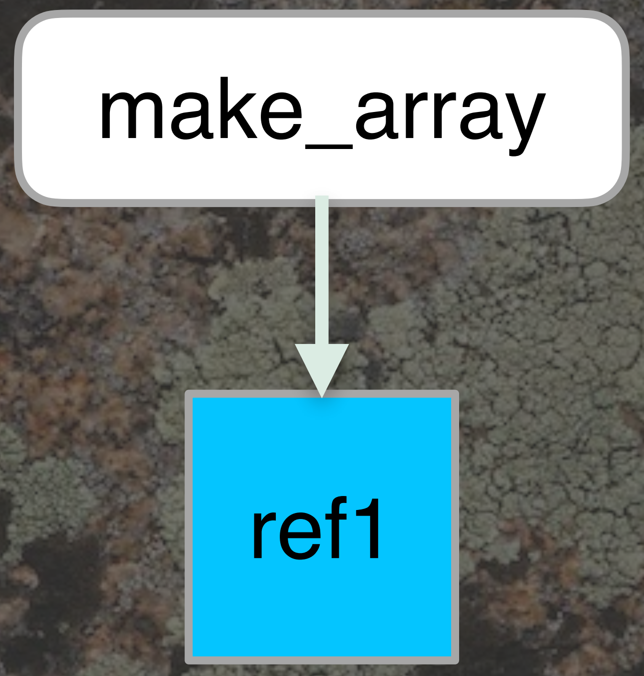
For completeness, add these first:

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

An Intuitive and Concise API

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(...)
```



An Intuitive and Concise API

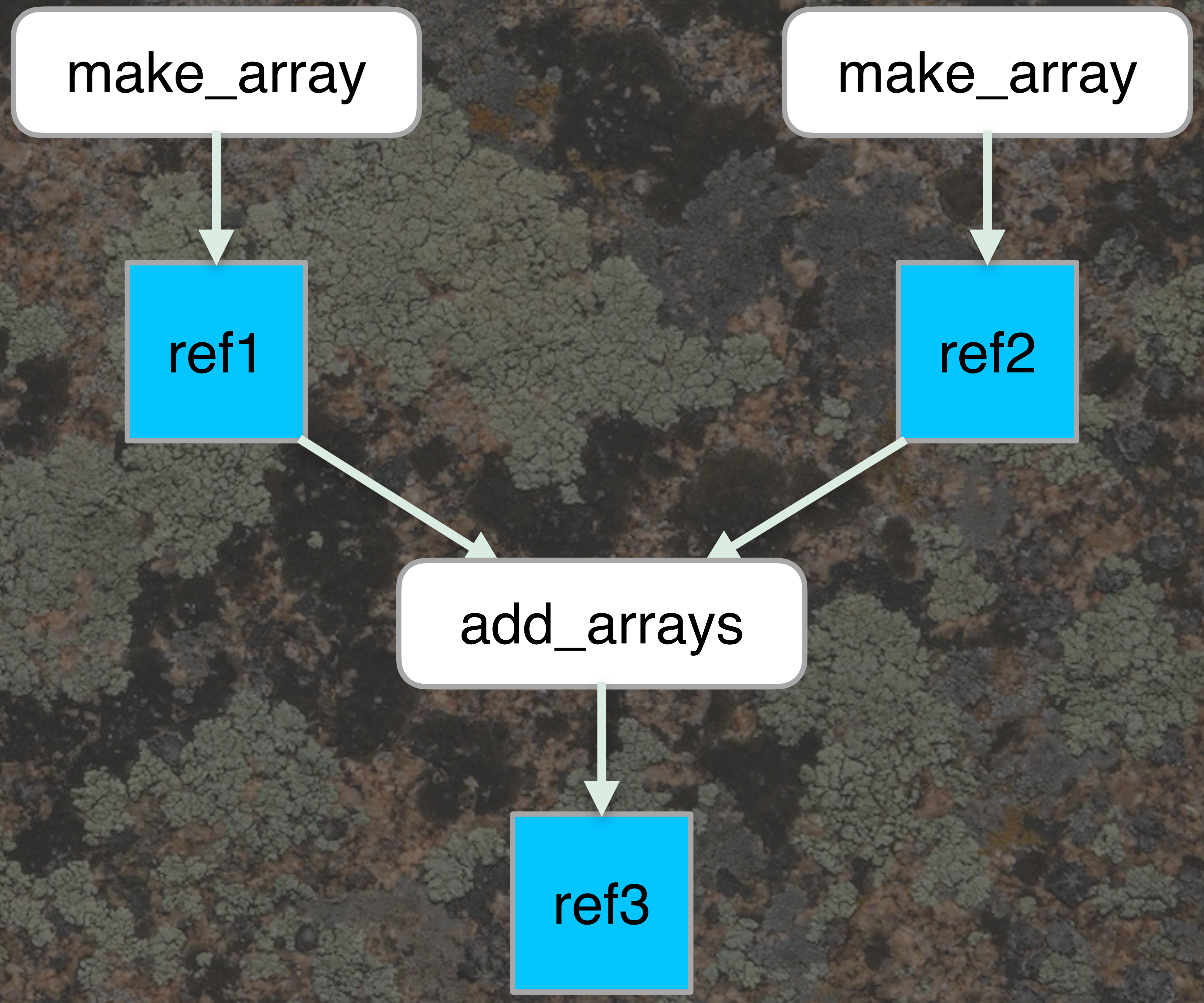
Functions -> Tasks

```
@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(...)
```

An Intuitive and Concise API

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)
```

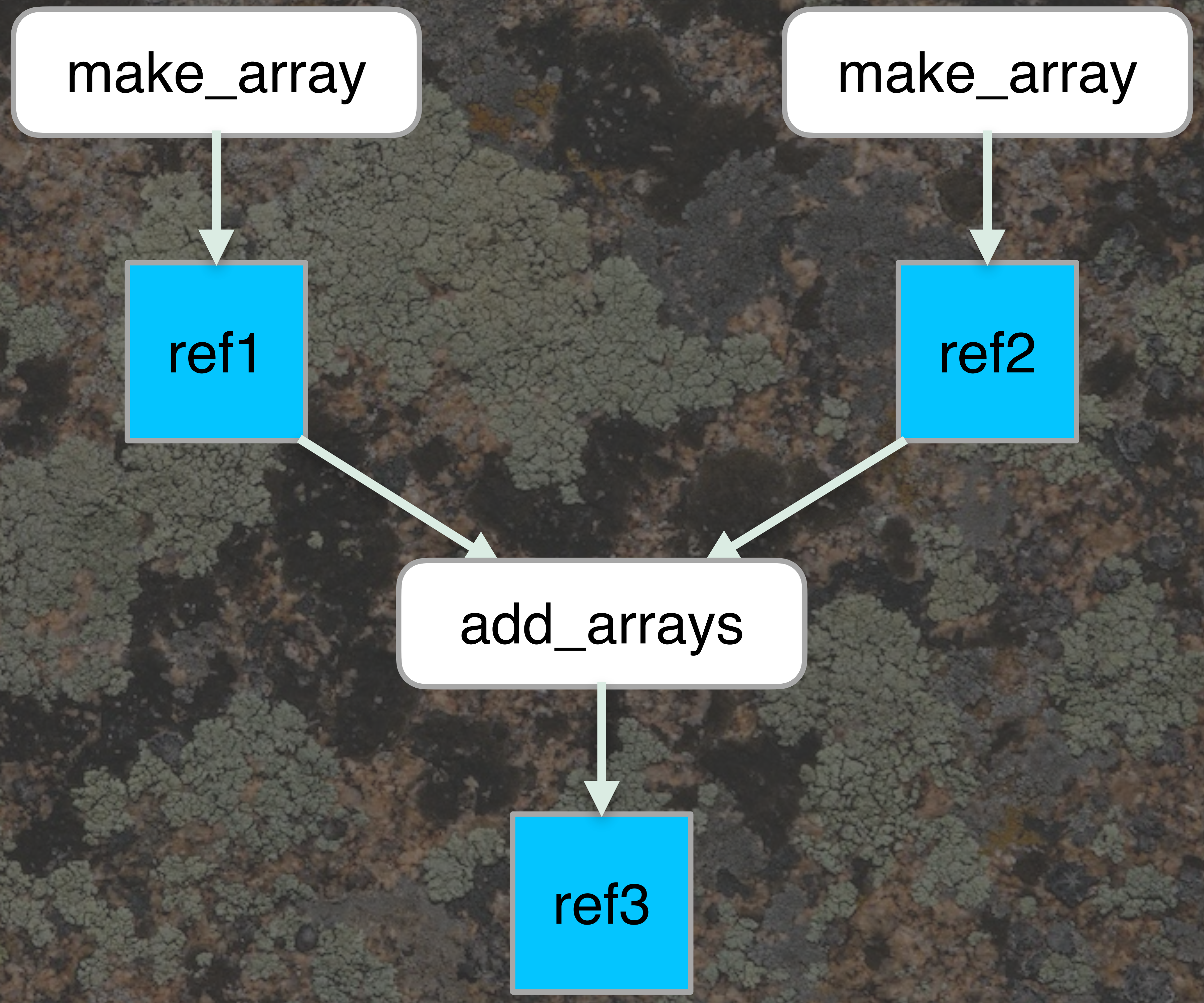


An Intuitive and Concise API

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)
```

Retrieve results



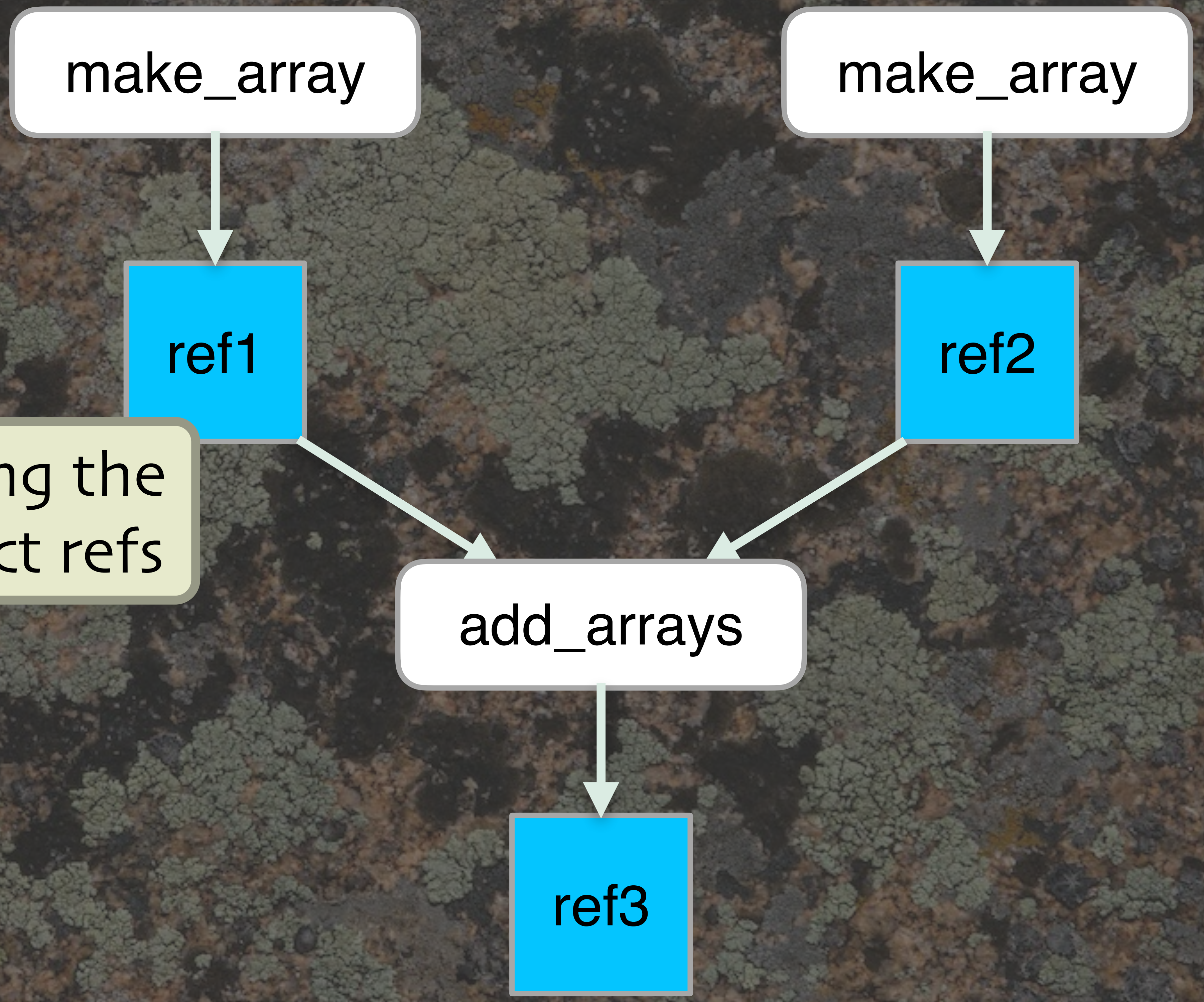
An Intuitive and Concise API

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



An Intuitive and Concise API

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)
```

An Intuitive and Concise API

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)
```

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

The Python classes you love...

An Intuitive and Concise API

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)
```

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.

An Intuitive and Concise API

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)
```

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]
```

An Intuitive and Concise API

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)
```

Classes -> Actors

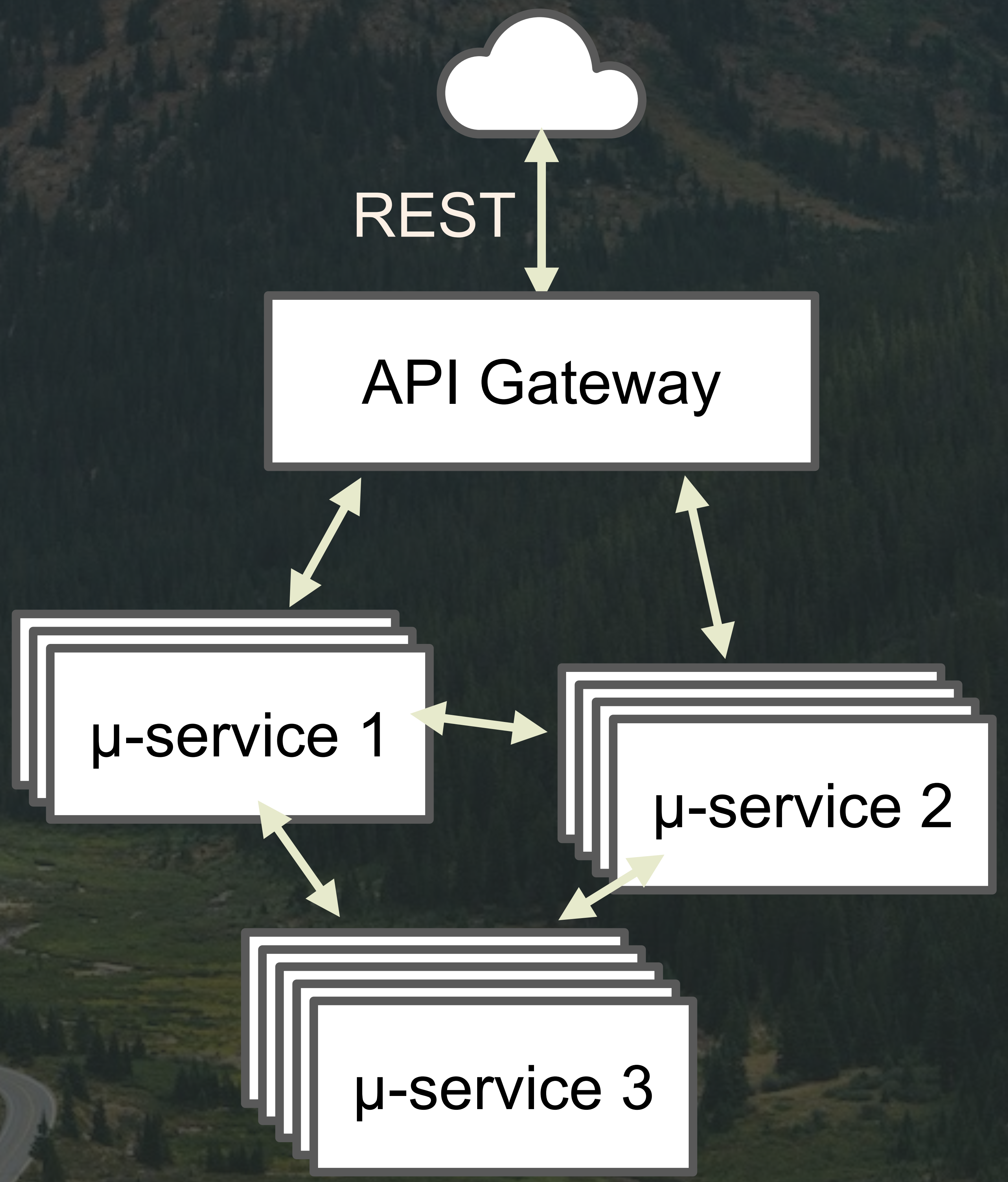
```
@ray.remote(num_gpus=1)
class Counter(object):
    def __init__(self):
        self.value = 0
    def increment(self):
        self.value += 1
        return self.value
    def get_count(self):
        return self.value

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

Configure with optional key-value args.


News Flash!
Ray Completely Rethinks
Microservices!

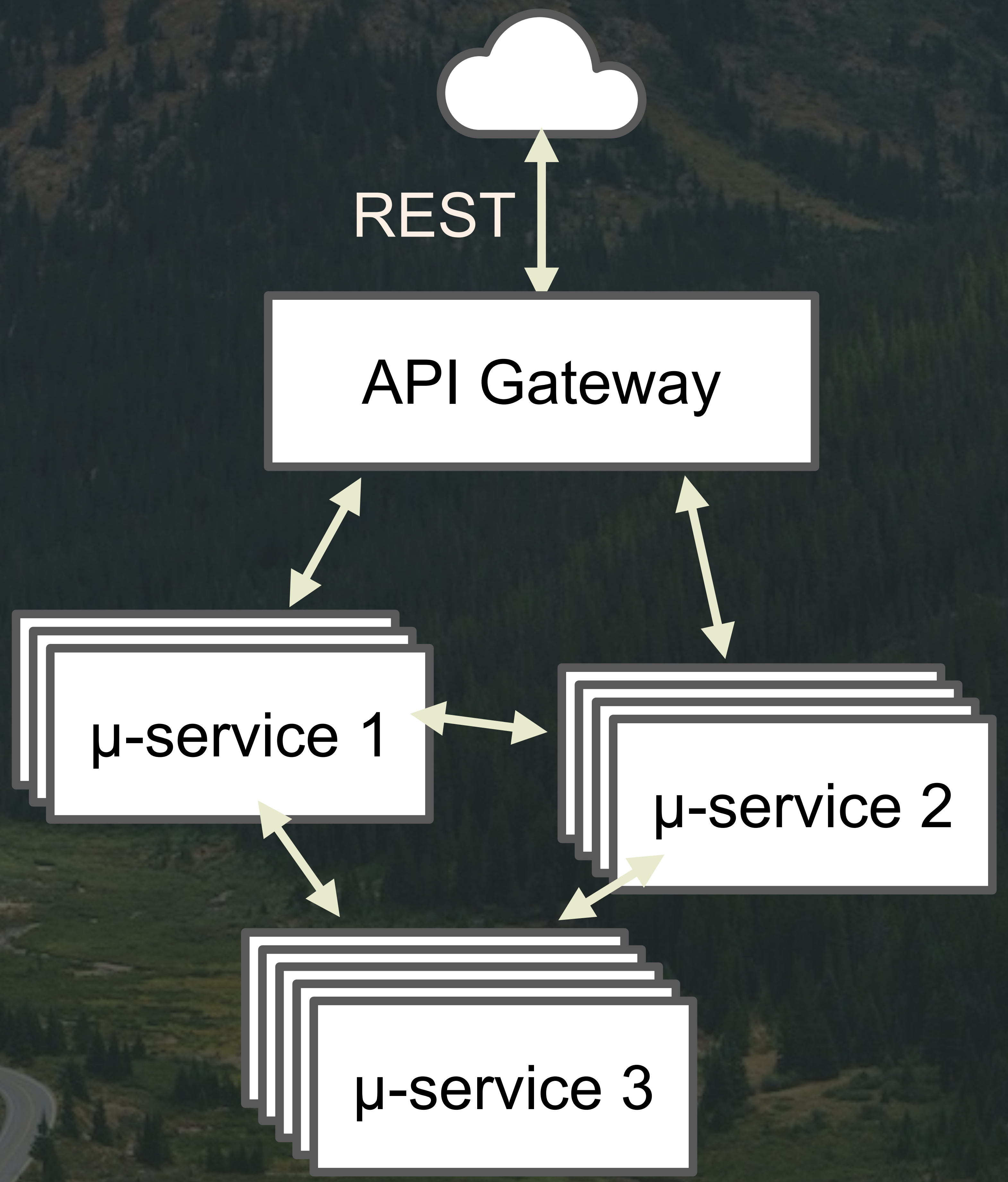
What Are Microservices?

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



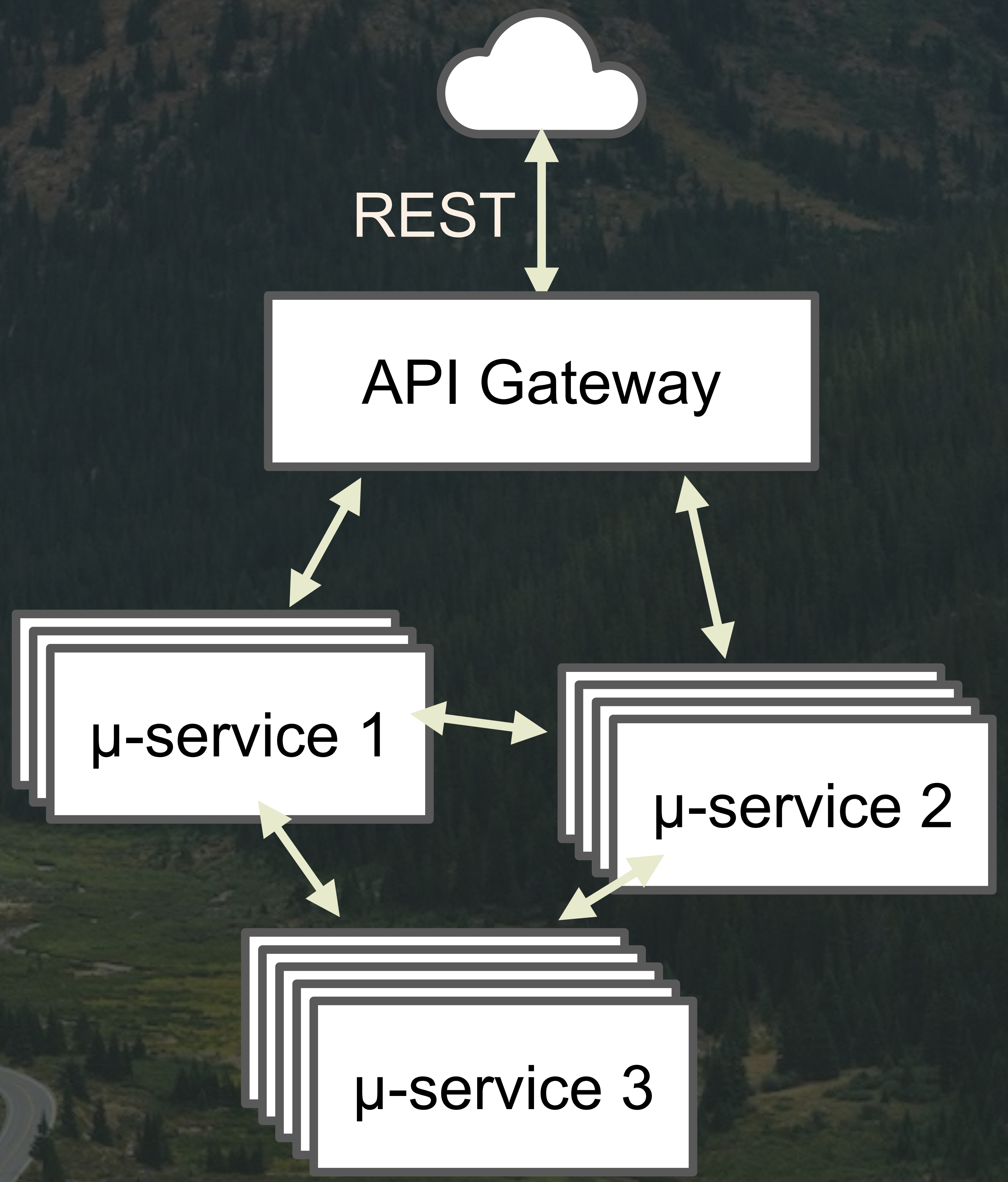
Conway’s Law: Embraced

- “Any organization that designs a system will produce a design whose structure is a copy of the organization's communication structure”
- Rather than fight this, let each team own and manage the services for its part of the domain!



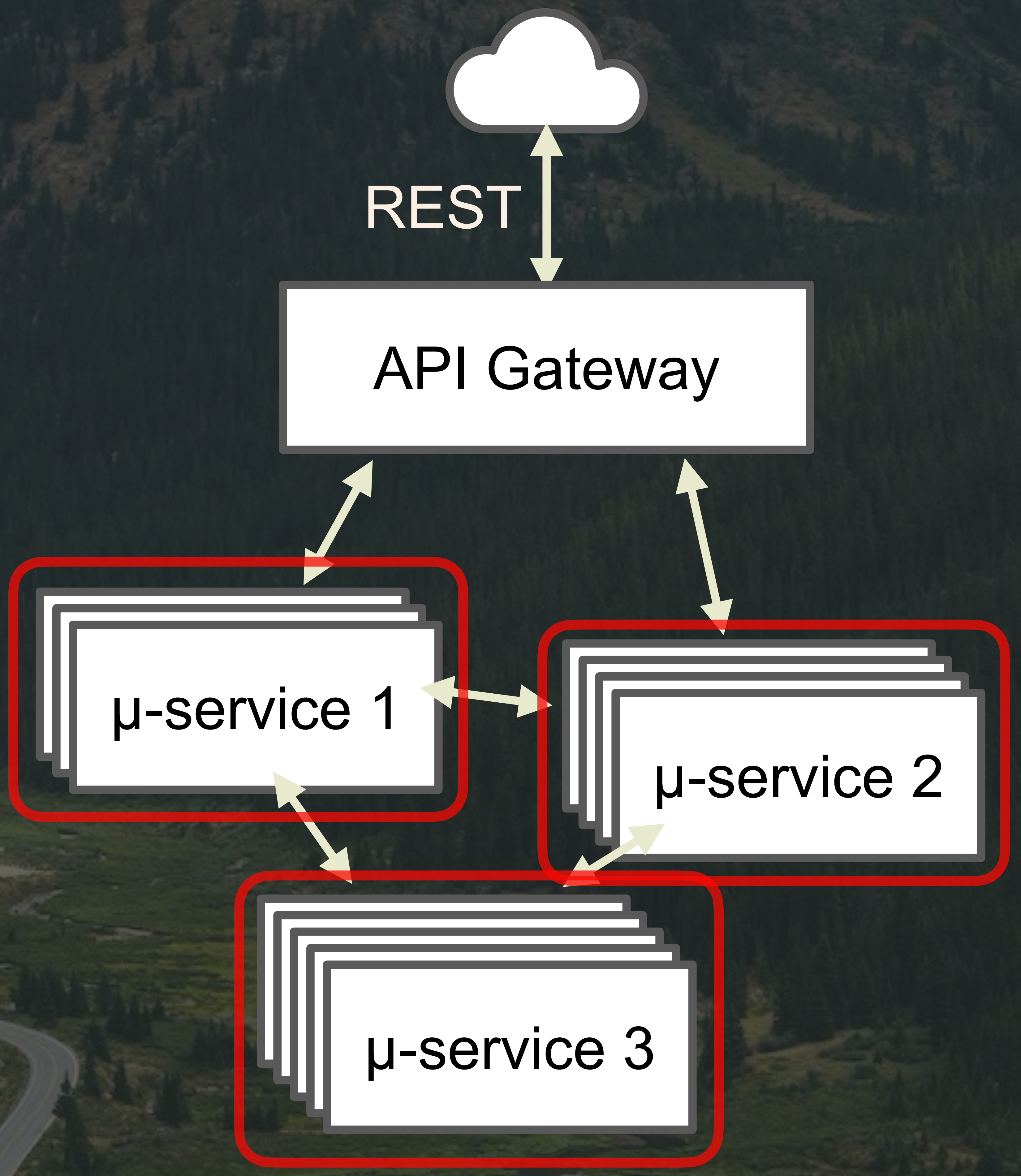
Separate Responsibilities

- Each microservice does "one thing", a single responsibility with minimally-sufficient coupling to the other microservices
- ...



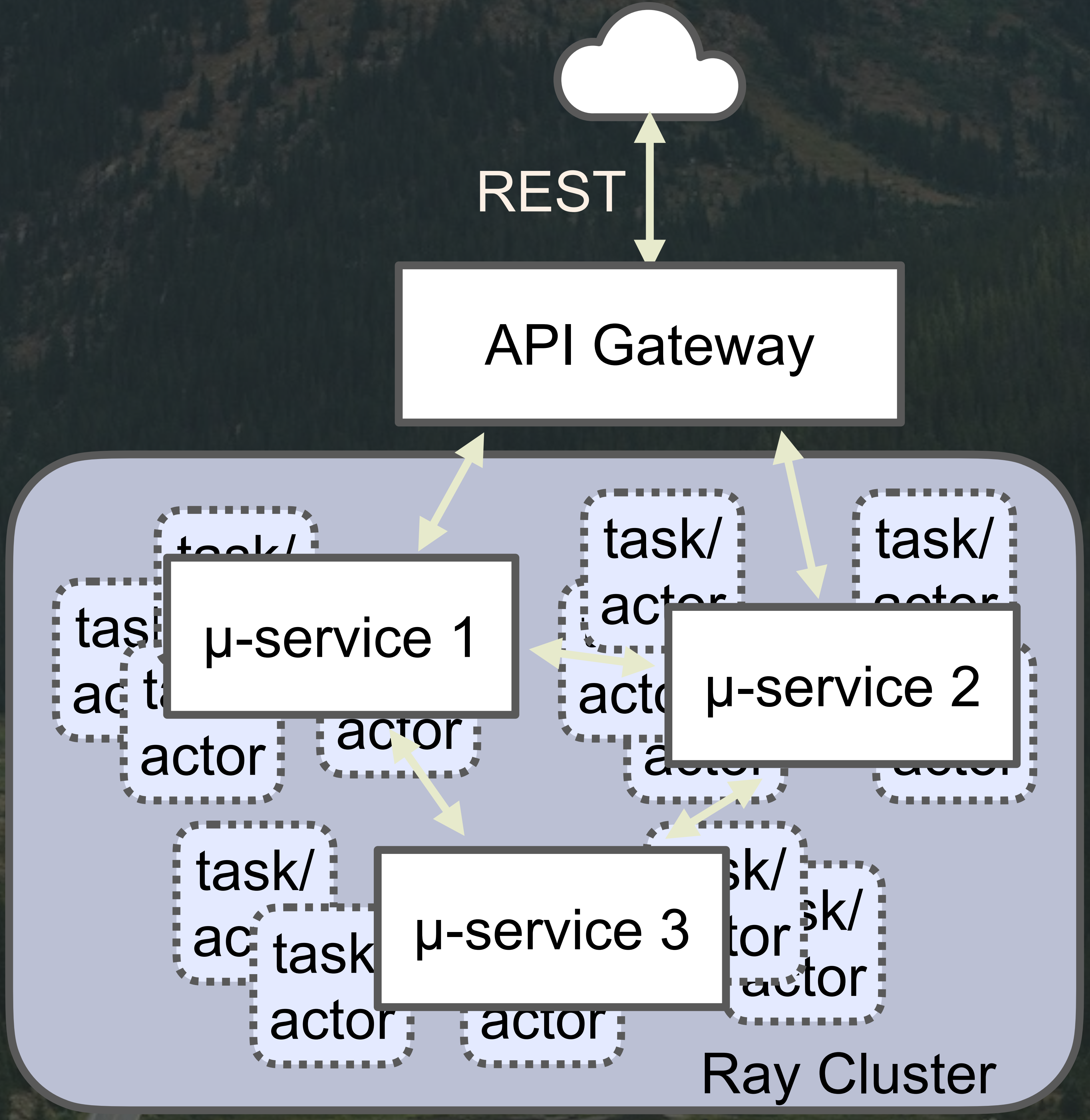
Separate Responsibilities

- ...
- 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**



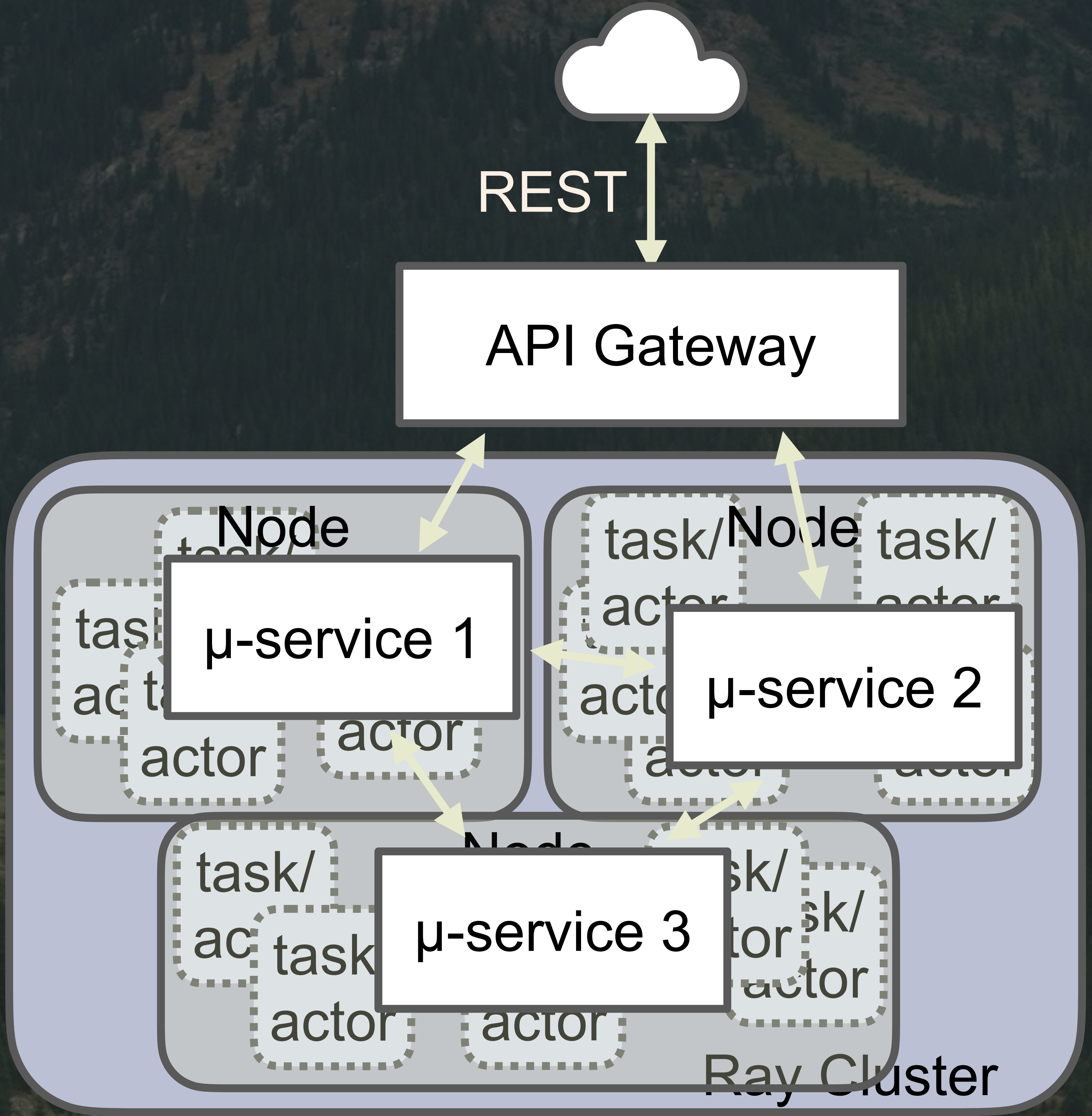
Management Drastically Simplified!

- With Ray, you can have one "logical" instance to manage and Ray does the cluster-wide scaling for you.
  - No need for explicit container scaling.



What About Kubernetes?

- Ray scaling is very fine grained.
- It operates within the "nodes" of coarse-grained managers
  - Containers, pods, VMs, or physical machines
- I.e. a Ray cluster within a K8s cluster




Hyper Parameter Tuning with Ray Tune
http://tune.io

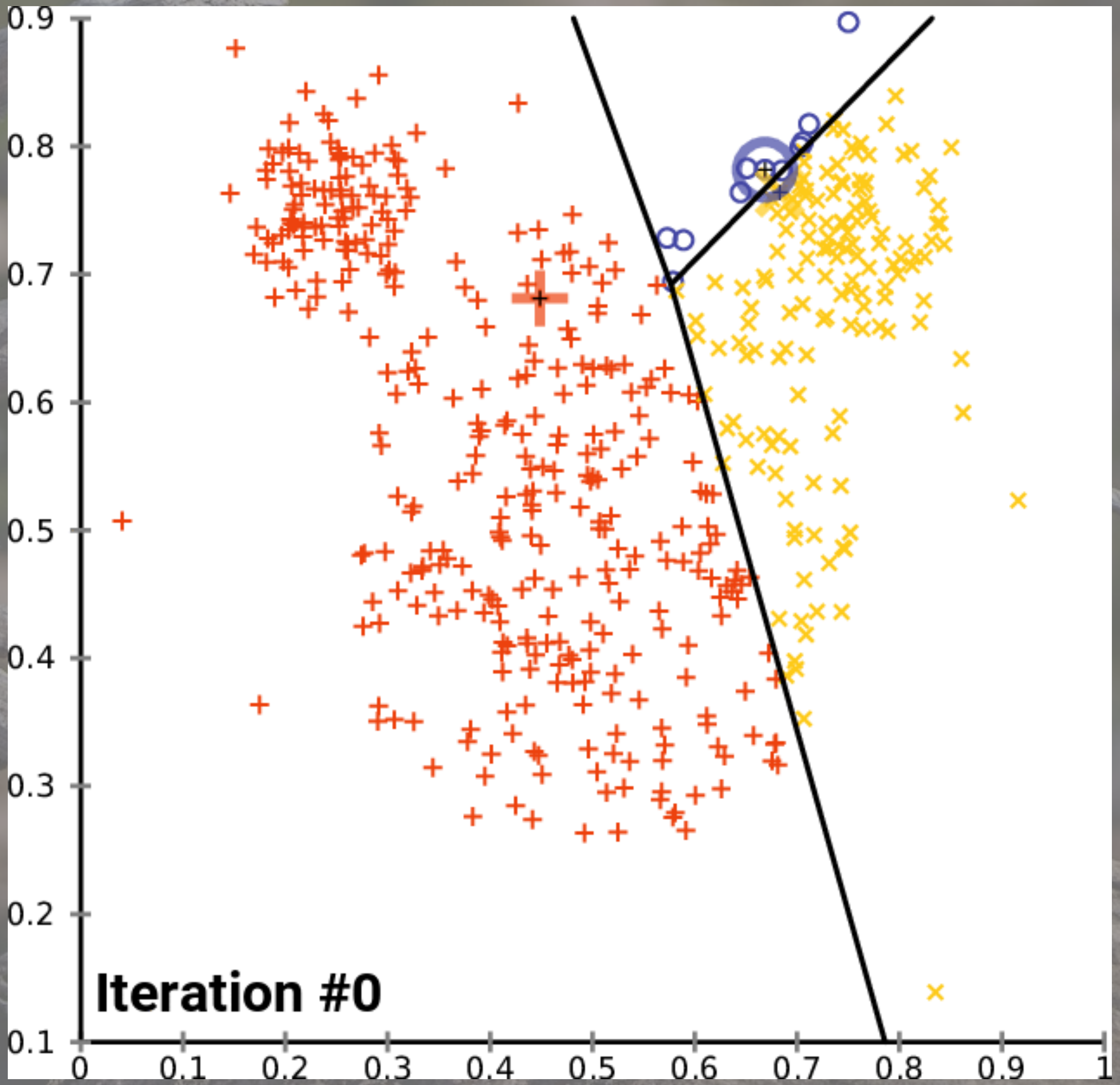
Hyper Parameter Tuning with Ray Tune

HPO: Hyper Parameter Optimization

What Is Hyper Parameter Tuning?

Trivial example:

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

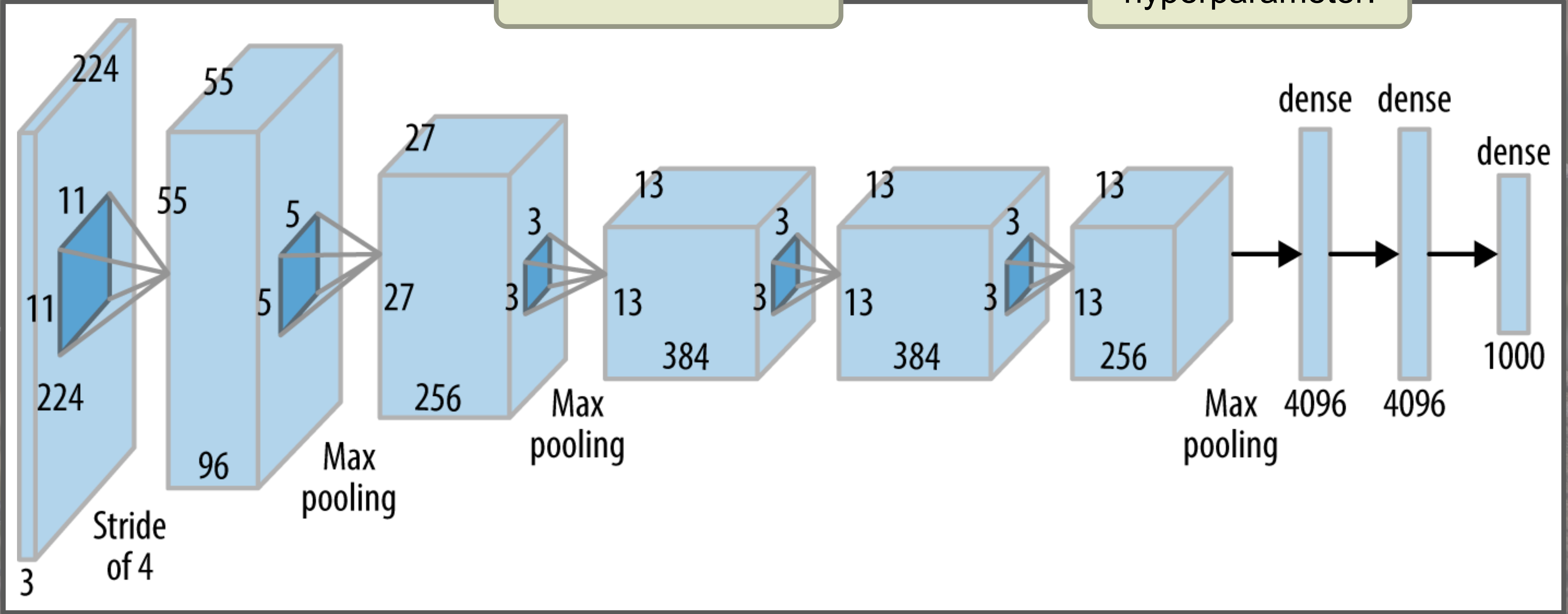


credit: https://commons.wikimedia.org/wiki/File:K-means_convergence.gif

Non-trivial Example: Neural Nets

How many layers?
What kinds of layers?

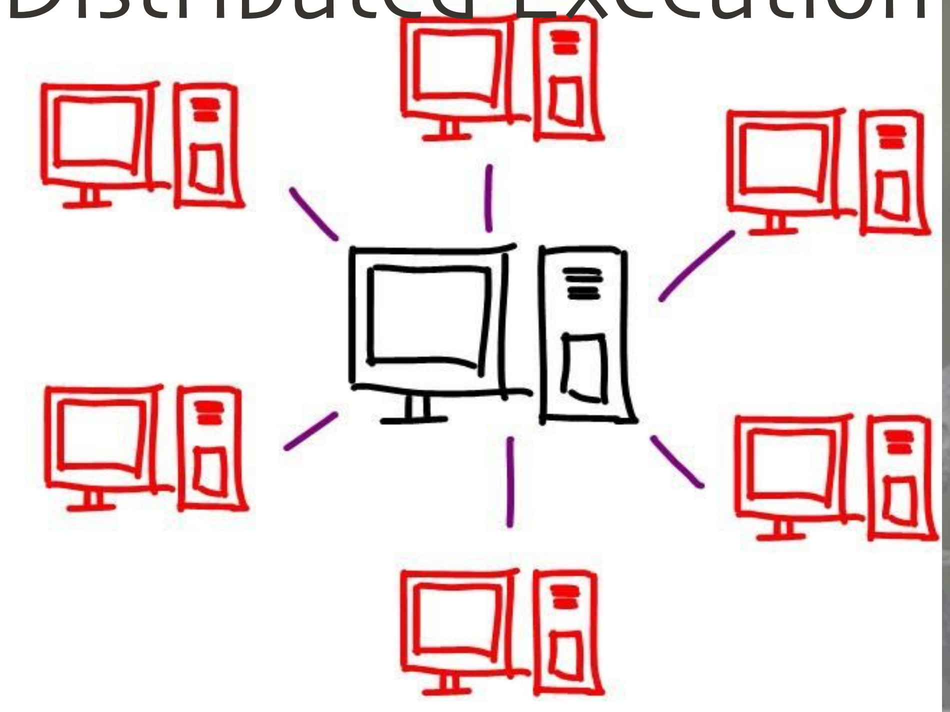
Every number shown is a hyperparameter!



Tune Is Optimized for Deep Learning

Resource Aware Scheduling

Seamless Distributed Execution

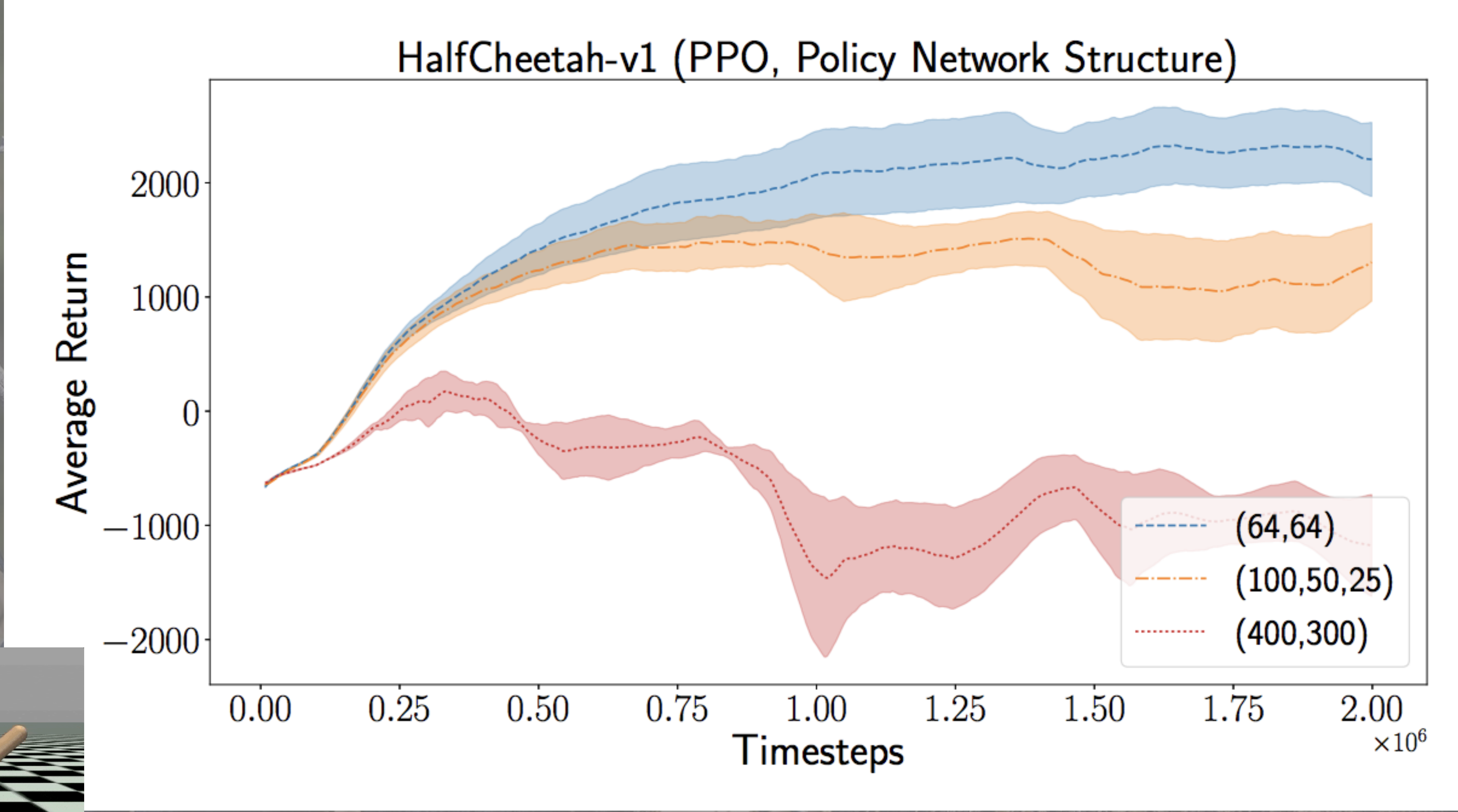


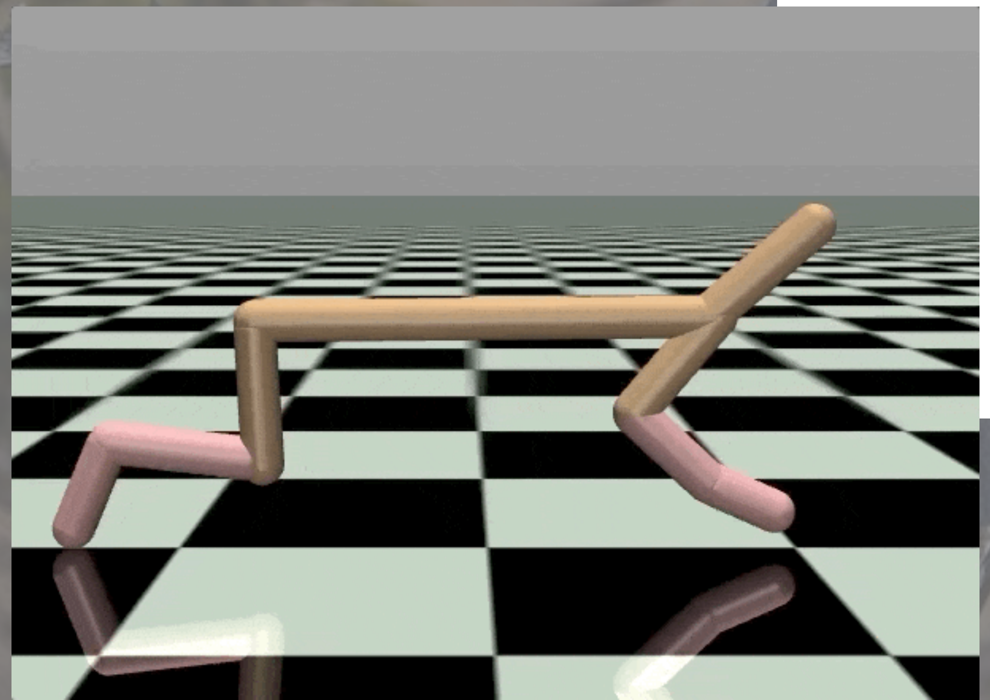
Simple API for new algorithms

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

Framework Agnostic

Hyper Parameters Optimized for Performance





Why We Need a Framework for Tuning Hyper Parameters

We want the best model

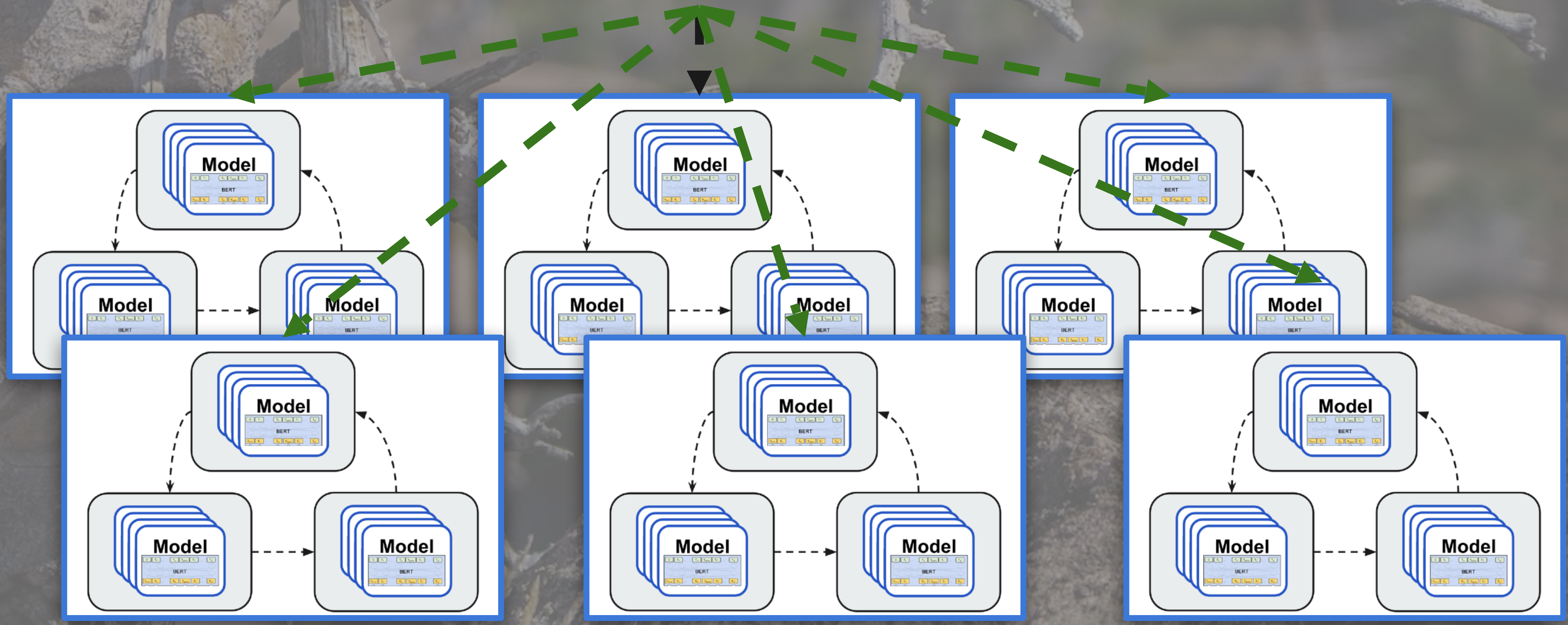
Resources are expensive

Model training is time-consuming

tune

Tuning with 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