Capgemini

Technische Universität München

End-to-End Learning AI project

Team:
Haris Jabbar, Christian Holland, Adrian Sieler, Egor Labintcev

Supervision by Capgemini:
Matthias Wissel

TUM Data Innovation Lab

Team

Adrian Sieler

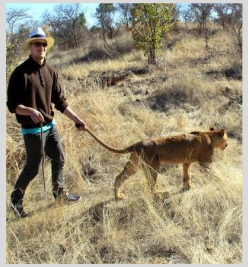
- Master Mathematics in Data Science
- Expert in Big Data related Buzzwords
- Loves scrambled egg

Haris Jabbar

- Master CSE Student
- Roboticist
- Wantrepreneur
- #LikeHumansDo

Christian Holland

- Master Mathematics
- AI enthusiast
- Love to travel
- Tea connoisseur



Egor Labintcev

- first-year Master student in CSE
- ML/DS
- CNN part, general coding
- loves bretzels

Agenda

Introduction
- Theory
- Applications

Algorithms
- Approaches
- Architecture

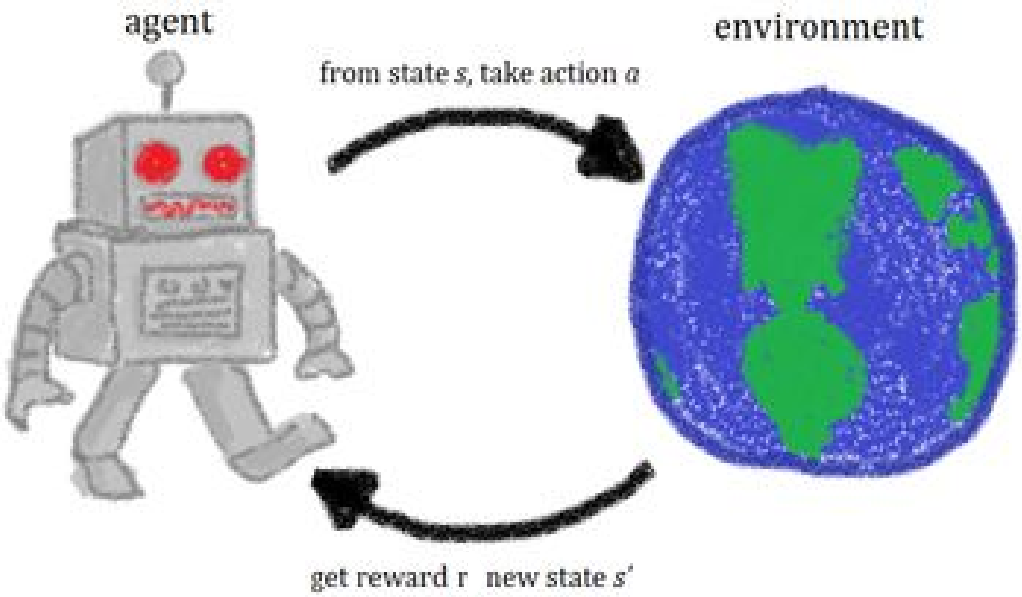
Scaling
- Distributed Tensorflow
- ROS

Tests
- Scalability
- Hyperparameters

Technische Universität München

Introduction

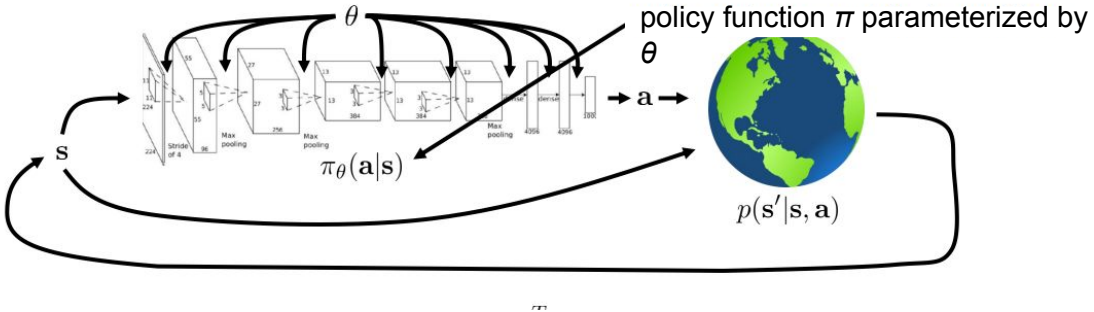
What is reinforcement learning?



Definitions

- state **s**
- action **a**
- reward **r**
- new state **s'**
- transition function **p** from **s** to **s'** after agent performs action **a**

The goal of reinforcement learning: find a policy



Definitions

- state **s**
- action **a**
- reward **r**
- new state **s'**
- transition function **p** from **s** to **s'** after agent performs action **a**
- **policy function π**

- **Markov decision process (MDP)**

Some important functions to consider

Policy function $\pi_\theta(a_t \mid s_t)$

determines a_t based on s_t

Q-function

$Q^\pi(s_t, a_t) = \sum_{t\prime=t}^{T} \mathbb{E}_{\pi_\theta}[r(s'_t, a'_t) \mid s_t, a_t]$

determines the total reward from taking a_t in s_t

Value function $V^\pi(s_t) = \mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[Q^\pi(s_t, a_t)]$

defines the total reward from s_t while following policy π_θ

Advantage function $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$

difference between taking a_t and the average return in s_t

Some important functions to consider

Policy function $\pi_\theta(a_t \mid s_t)$

determines a_t based on s_t

Value function $V^\pi(s_t) = \mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[Q^\pi(s_t, a_t)]$

defines the total reward from s_t while following policy π_θ

Q-function

$Q^\pi(s_t, a_t) = \sum_{t'=t}^{T} \mathbb{E}_{\pi_\theta}[r(s'_t, a'_t) \mid s_t, a_t]$

determines the total reward from taking a_t in s_t

Advantage function $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$

difference between taking a_t and the average return in s_t

Some important functions to consider

Policy function $\pi_\theta(a_t \mid s_t)$

determines a_t based on s_t

Q-function

$Q^\pi(s_t, a_t) = \sum_{t\prime=t}^{T} \mathbb{E}_{\pi_\theta}[r(s'_t, a'_t) \mid s_t, a_t]$

determines the total reward from taking a_t in s_t

Value function $V^\pi(s_t) = \mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[Q^\pi(s_t, a_t)]$

defines the total reward from s_t while following policy π_θ

Advantage function $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$

difference between taking a_t and the average return in s_t

Some important functions to consider

Policy function $\pi_\theta(a_t \mid s_t)$

determines a_t based on s_t

Q-function

$Q^\pi(s_t, a_t) = \sum_{t\prime=t}^{T} \mathbb{E}_{\pi_\theta}[r(s'_t, a'_t) \mid s_t, a_t]$

determines the total reward from taking a_t in s_t

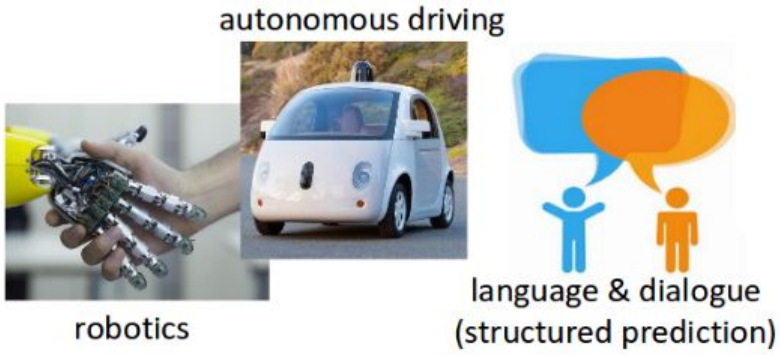
Value function $V^\pi(s_t) = \mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[Q^\pi(s_t, a_t)]$

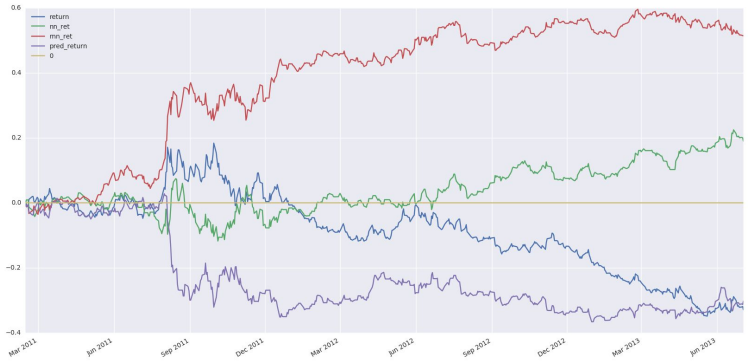
defines the total reward from s_t while following policy π_θ

Advantage function $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$

difference between taking a_t and the average return in s_t

Applications



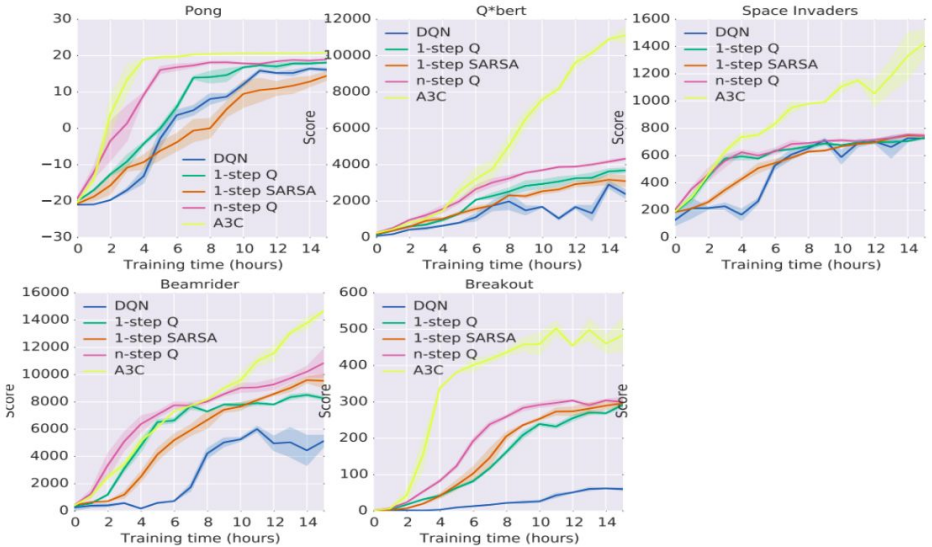


time series prediction

Recent advances

- Asynchronous Advantage Actor-Critic (A3C)
  Volodymyr Mnih et al. “Asynchronous Methods for Deep Reinforcement Learning”. In: (2016).

- Path-Consistency-Learning (PCL)
  Ofir Nachum et al. “Bridging the Gap Between Value and Policy Based Reinforcement Learning”. In: (2017).



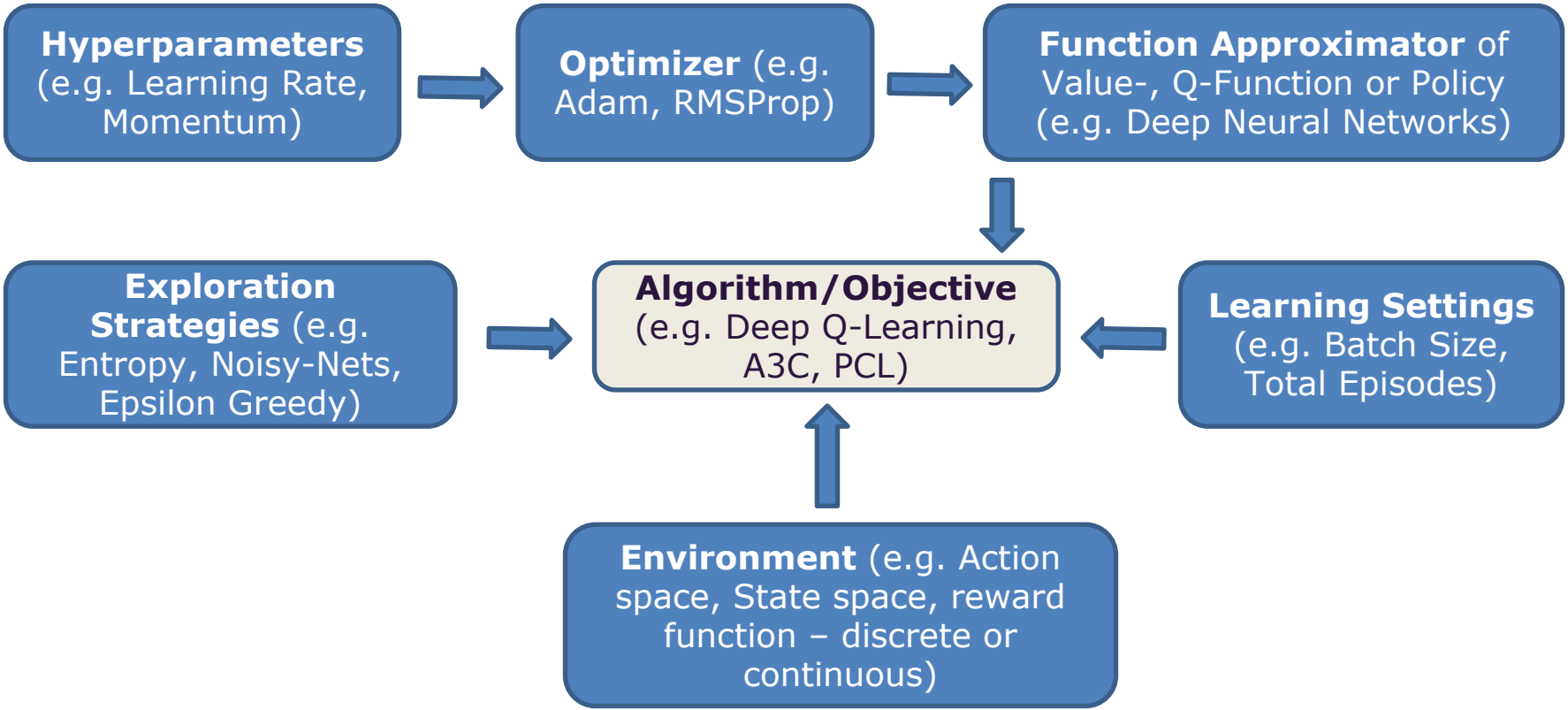
Atari games from OpenAI Gym:

- Environment framework
- Unified API
- Open sourced

Technische Universität München

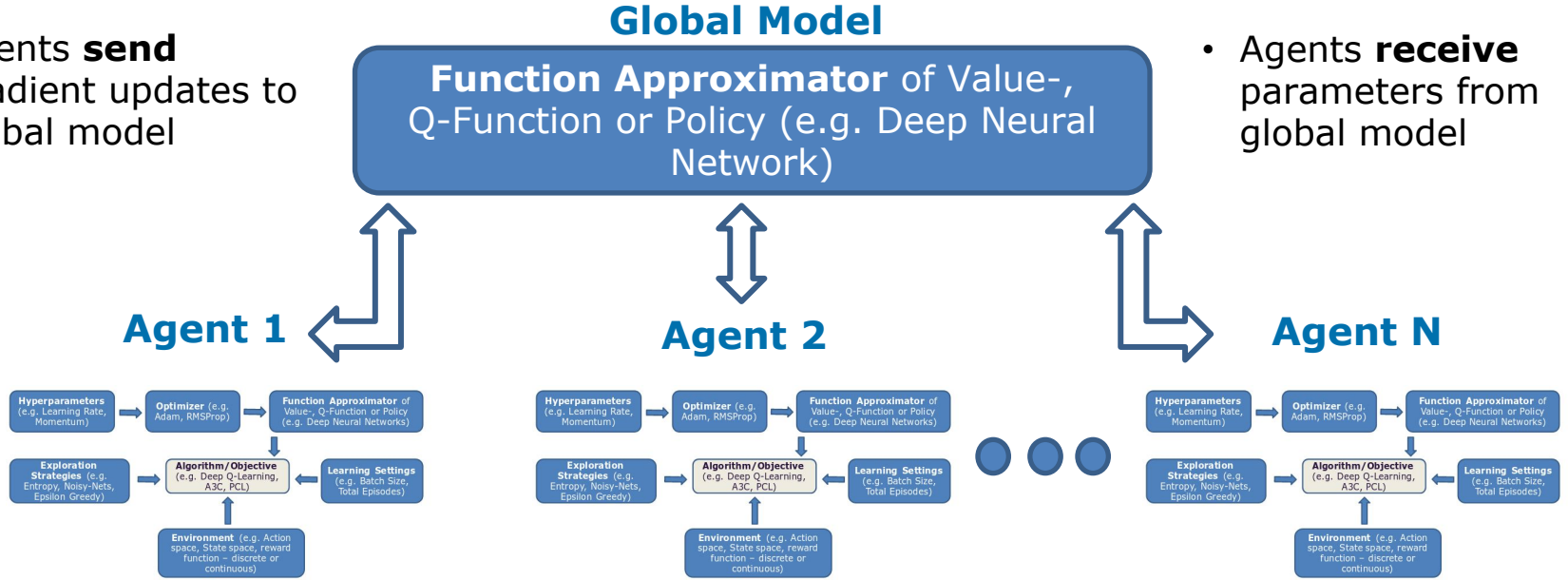
Algorithms

Reinforcement Learning Architecture – Single Agent



Reinforcement Learning Architecture – Multiple Agents

- Agents **send** gradient updates to global model



- Agents **receive** parameters from global model

Advantages:

- Different exploration policies in each agent to maximize diversity
- Parallel updates are likely to be less correlated in time
- Reduction in training time roughly linear in the number of parallel agents

A3C and PCL - Two Reinforcement Learning Algorithms

A3C - (Asynchronous Actor-Critic)

Objective:

- Maximizes the expected sum of the rewards for all possible rollouts τ of $\pi\theta$

$$\theta^{\star} = argmax_{\theta}\mathbb{E}_{\tau\sim\pi_{\theta}(\tau)}\left[\sum_{t=1}^{T} r(s_t, a_t)\right]$$

Advantages:

- Stable and unbiased gradient estimate
- In **asynchronous** setting less correlated updates and better exploration

Disadvantages:

- Sample inefficient due to the on-policy nature of A3C

PCL - (Path Consistency Learning)

Objective:

- Maximizes the expected sum of the rewards for all possible rollouts τ of $\pi\theta$ while keeping entropy of the policy high
- Relates optimal value-function and optimal policy

$$V^*(s_t) - \gamma V^*(s_{t+1}) = r(s_t, a_t) - \tau \log \pi^*(a_t \mid s_t)$$

Advantages:

- On- and Off-Policy updates possible, therefore more sample efficient
- Replay Buffers can be leveraged

Disadvantages:

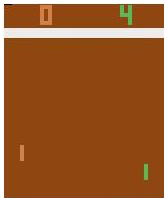
- More Hyperparameters to tune and therefore harder to train

Neural Networks in Reinforcement Learning

Input → **Network Architecture** → **Output**

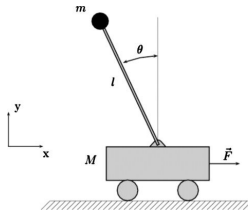
Image based input:

- End-To-End learning through processing of the raw image (e.g. Atari games)



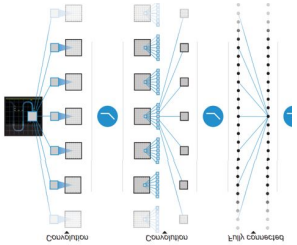
State based input:

- Description of the state through physical characteristics of the system
- Like position, angles, velocity, acceleration



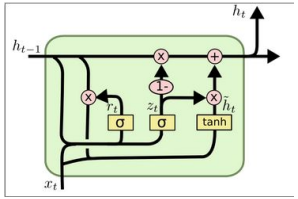
Convolutional Neural Network:

- Space invariant feedforward artificial neural networks
- Can successfully be applied to analyze visual imagery



Long-Short-Term-Memory Networks:

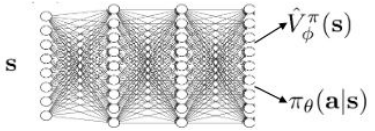
- Extension of classical recurrent neural network (RNN)
- Ability to add and remove information to the cell state based on long term dependencies



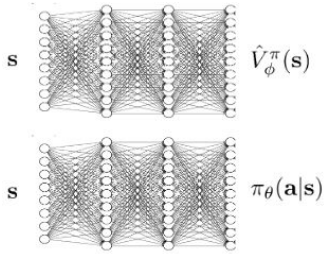
Depending on the algorithm:

- Deep Q-Learning (Q-Function)
- A3C (Value-Function and Policy)
- PCL (Value-Function and Policy)

Networks can either be shared:



or separate:



Project Goals

Modular Design

- Adjustable **algorithm**
- Adjustable **network structure**
- Adjustable **optimizer**
- Adjustable **learning rate**
- Adjustable **environment**
- Adjustable **training settings**
- Adjustable **image preprocessing**

Framework to go:

- **Python + Tensorflow**

Scalability of Computation

- Implementation of the **asynchronous learner** approach
- Leveraging **any** available **hardware** to distribute **computational** workload
  - Multiple threads on the same machine
  - Multiple different machines
  - Multiple threads on multiple machines

Framework to go:

- **Distributed Tensorflow**

Scalability of Simulation

- Implementation of a **flexible** and **scalable** simulation environment
- Leveraging **any** available **hardware** to distribute **simulation** workload

Framework to go:

- **ROS (Robot Operating System)**

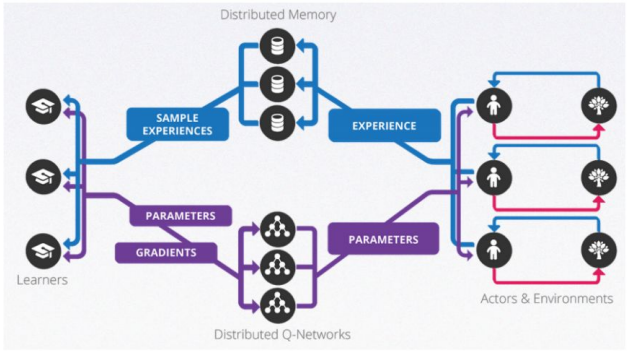
Technische Universität München

Scaling Up Reinforcement Learning

Why should we scale up Reinforcement Learning?

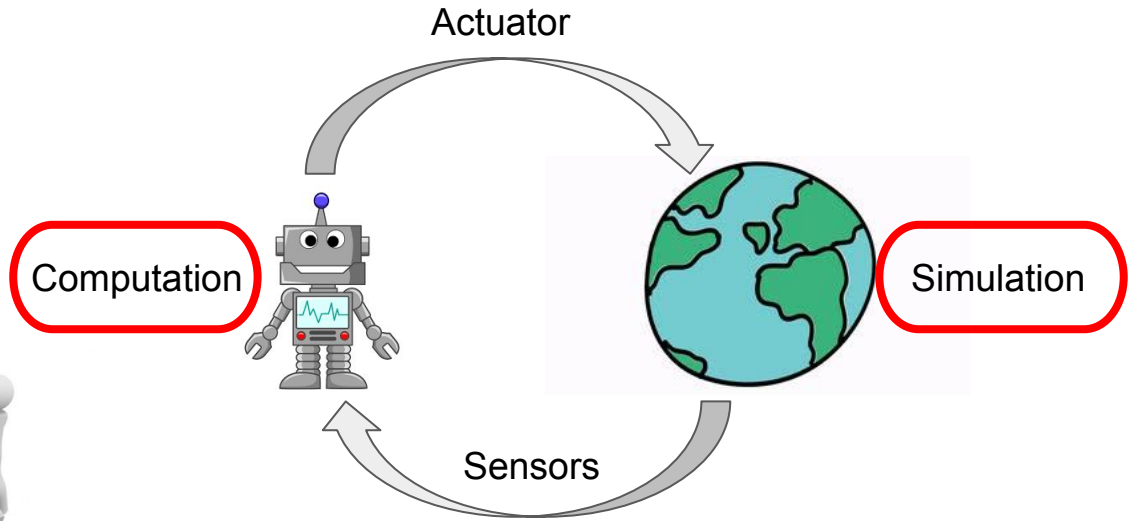
- Exploration
- Sparse Rewards
- Computational Efficiency
- New Research Domains
- GORILA

Gorila (General Reinforcement Learning Architecture)

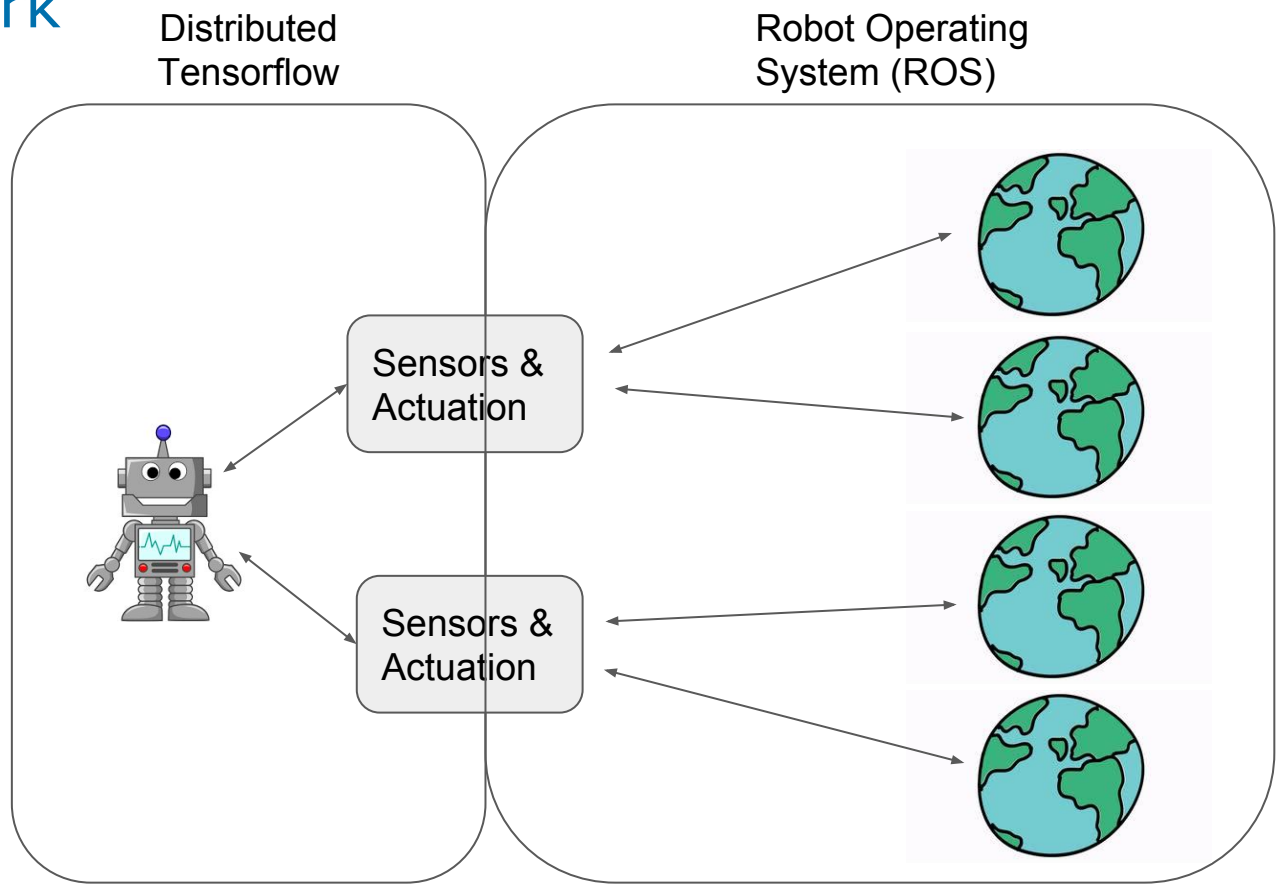


- 10x faster than Nature DQN on 38 out of 49 Atari games
- Applied to recommender systems within Google

The Framework

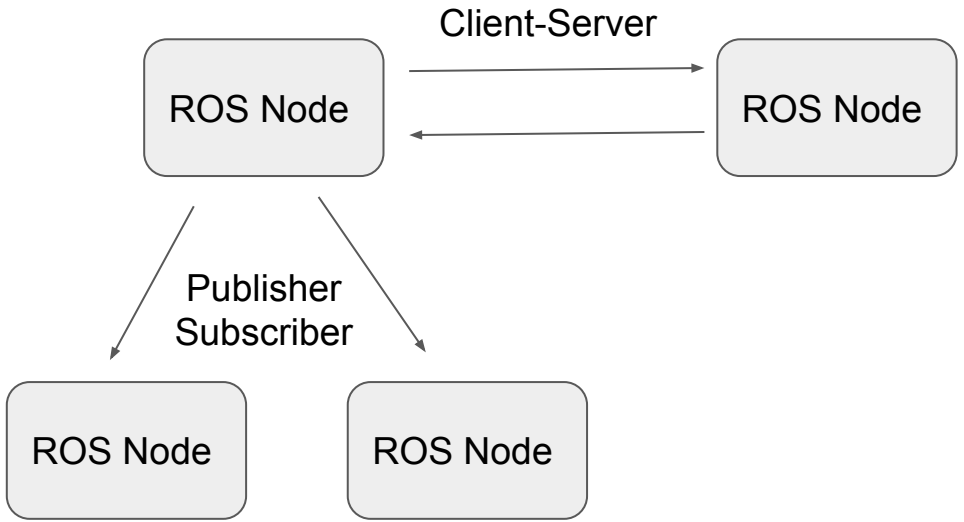


The Framework



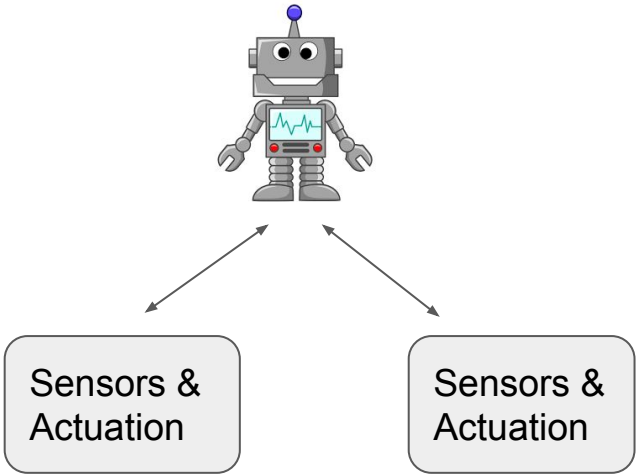
Robot Operating System (ROS)

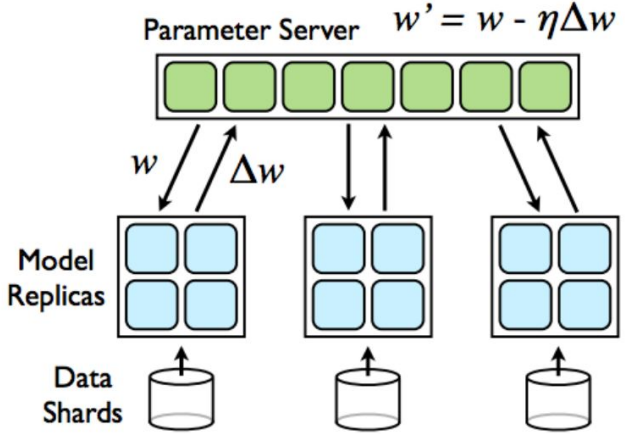
- Open Source platform for Robotic environments
  - Nodes
  - Communication framework
- Programming Language Agnostic
  - C++/Java/Python/C#
- Communication Paradigms
  - Client Server
  - Publisher Subscriber



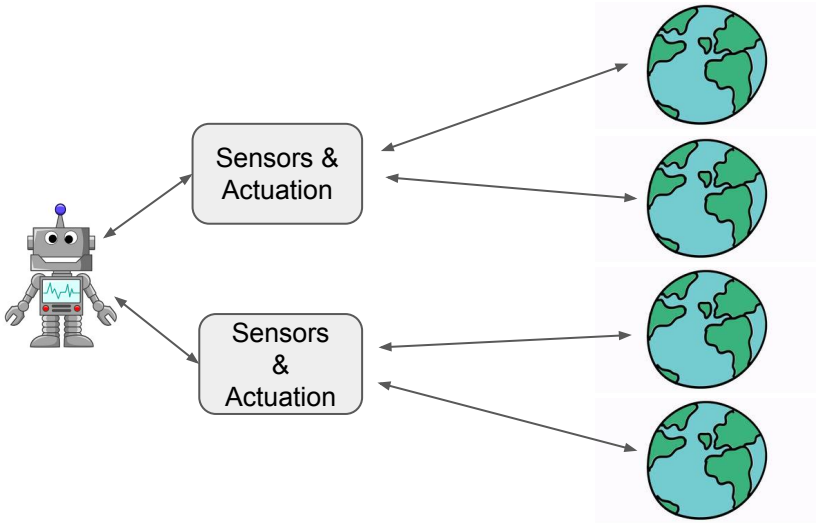
Distributed Tensorflow

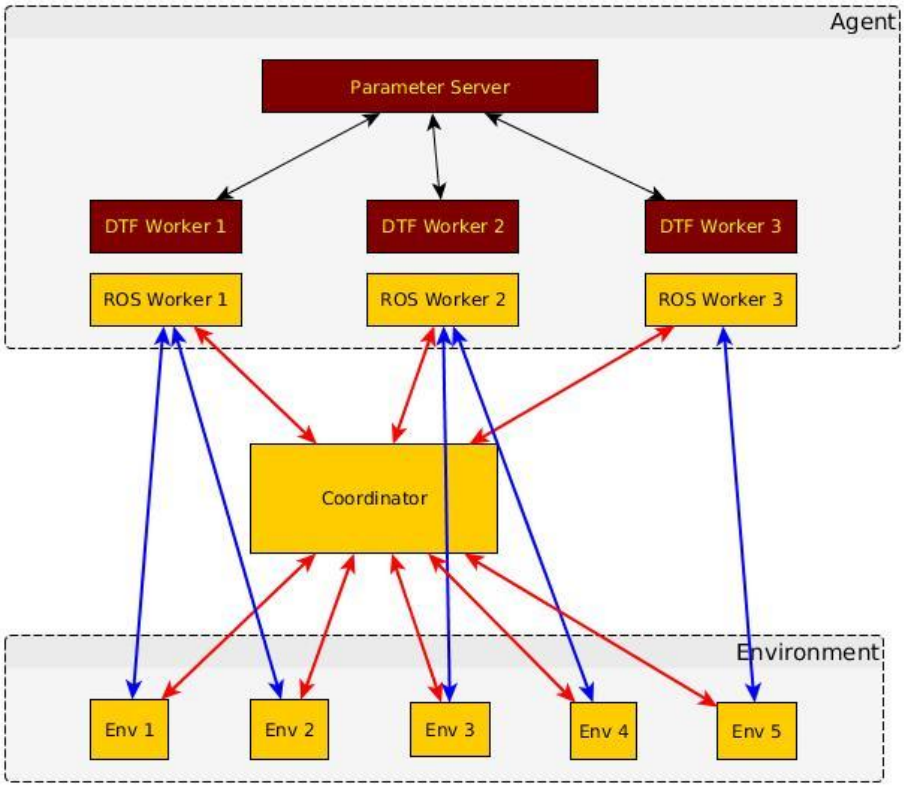
- Client Server architecture
  - Parameter Server
  - Worker Nodes



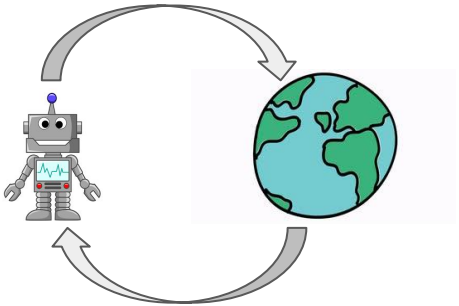


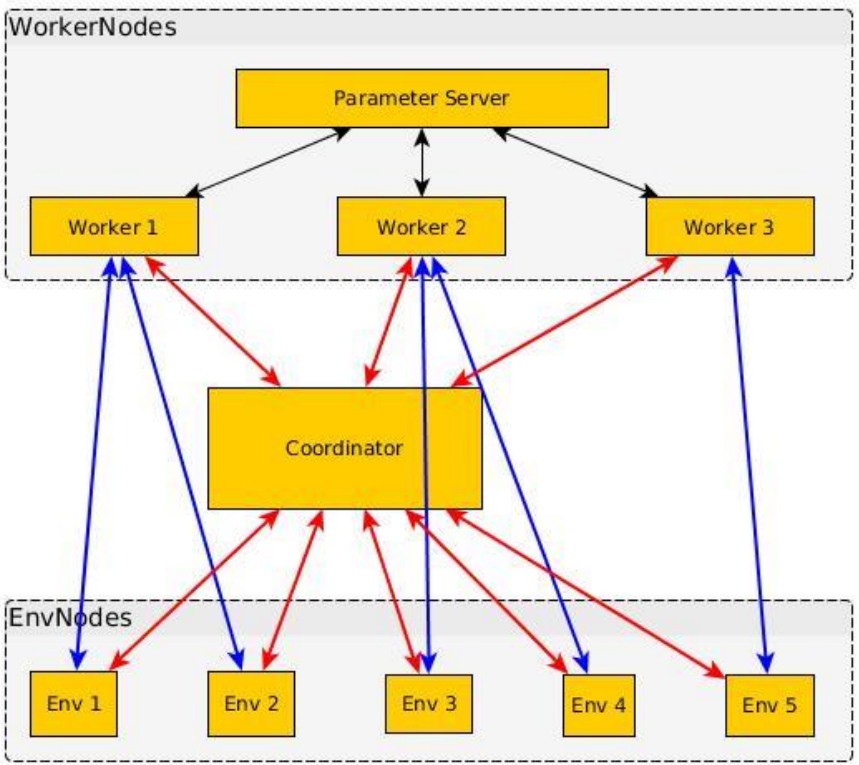
Putting it Together



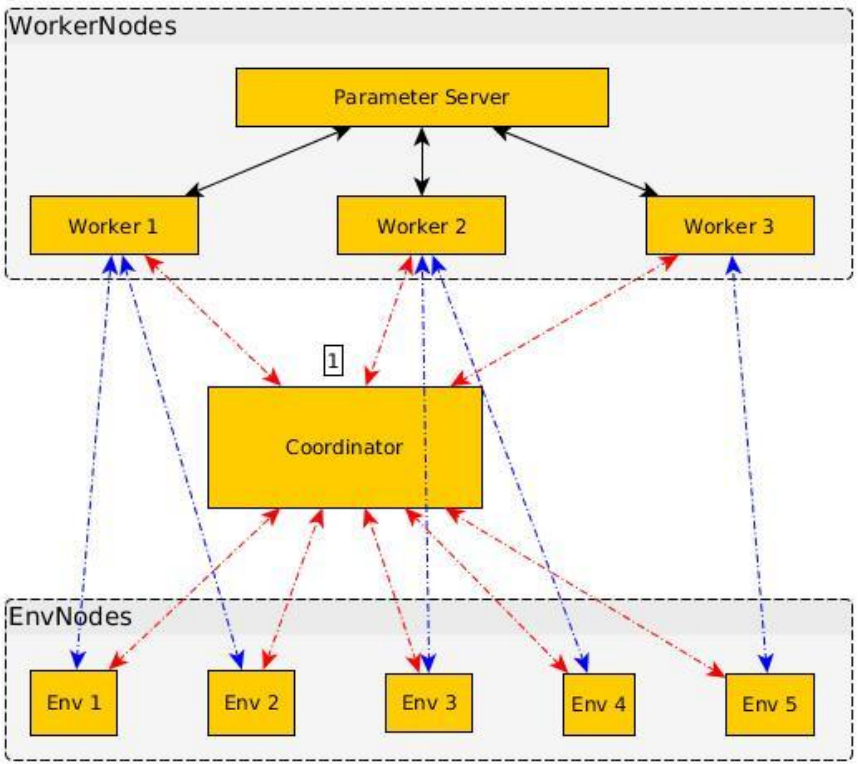


Sequence of Operations

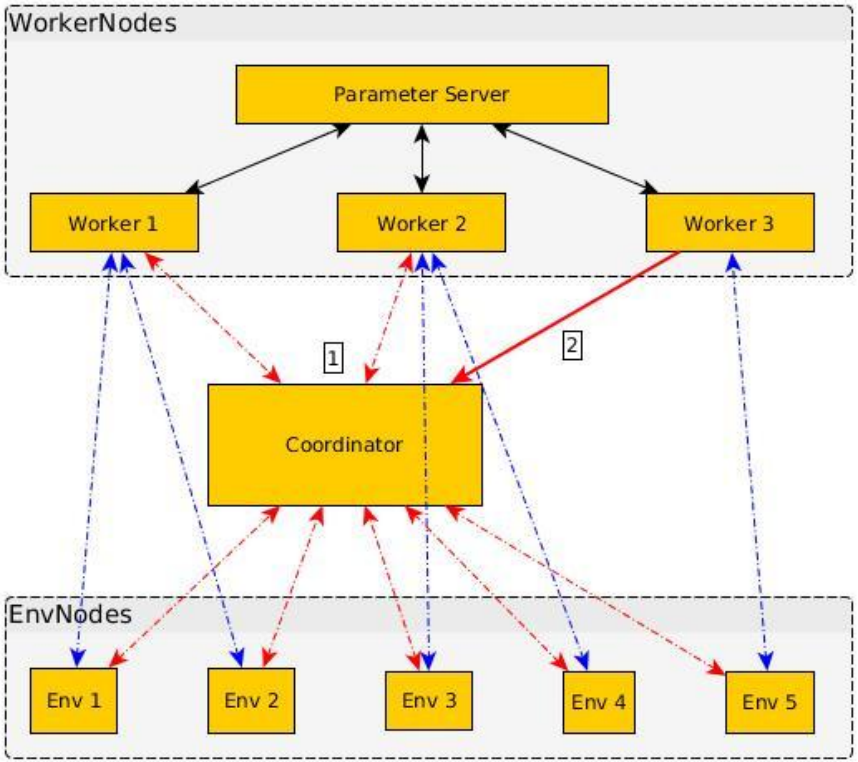




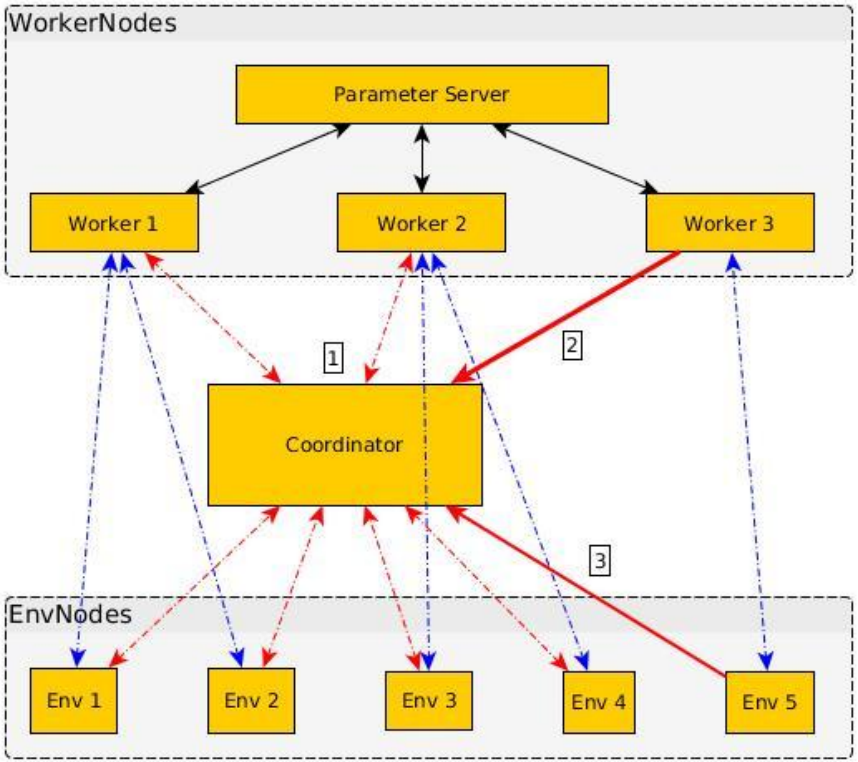
Sequence of Operations



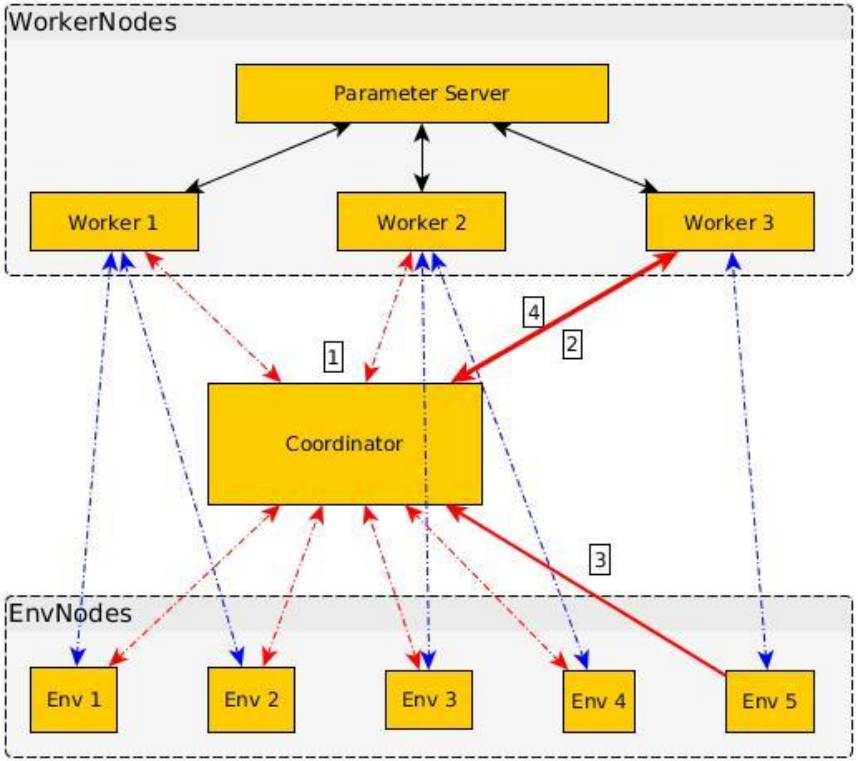
Sequence of Operations



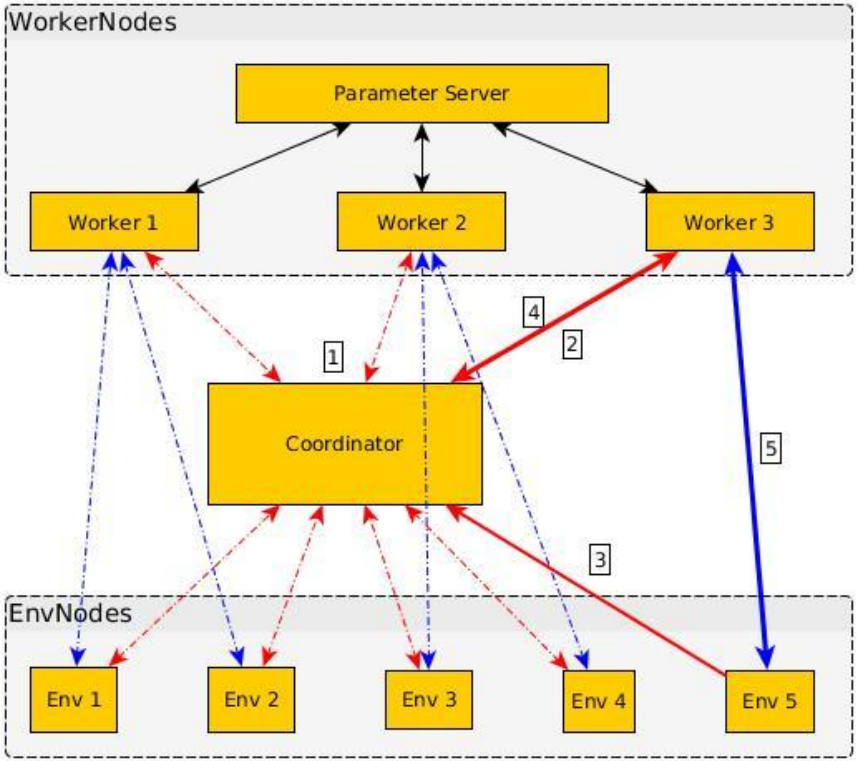
Sequence of Operations

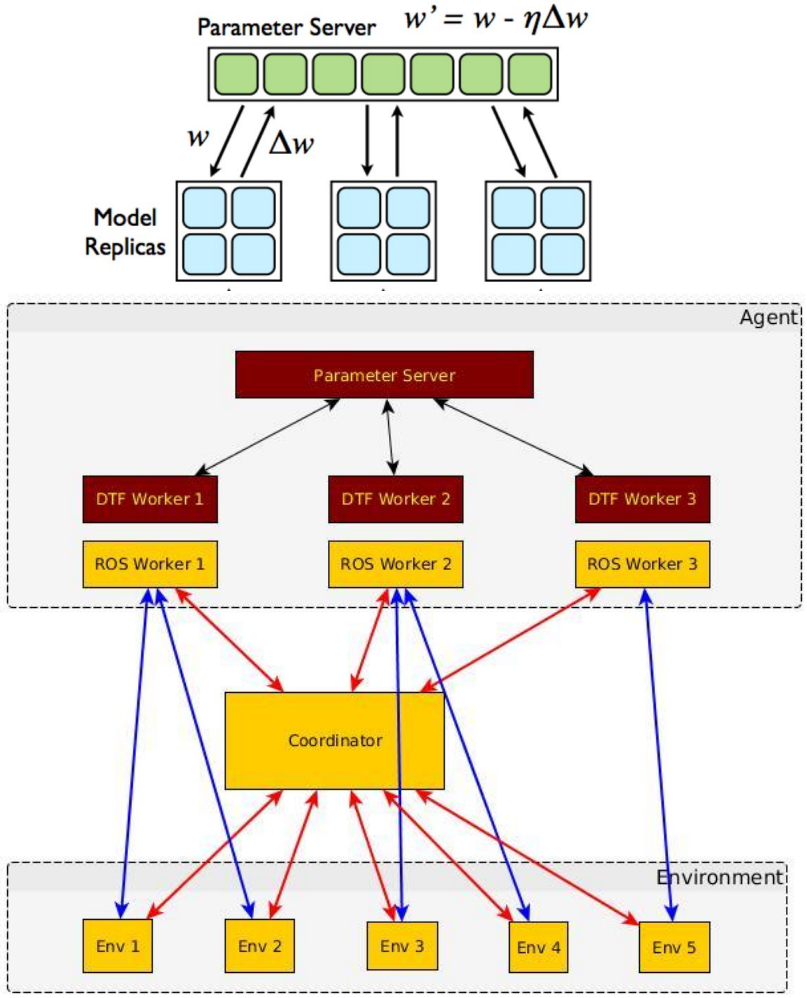


Sequence of Operations



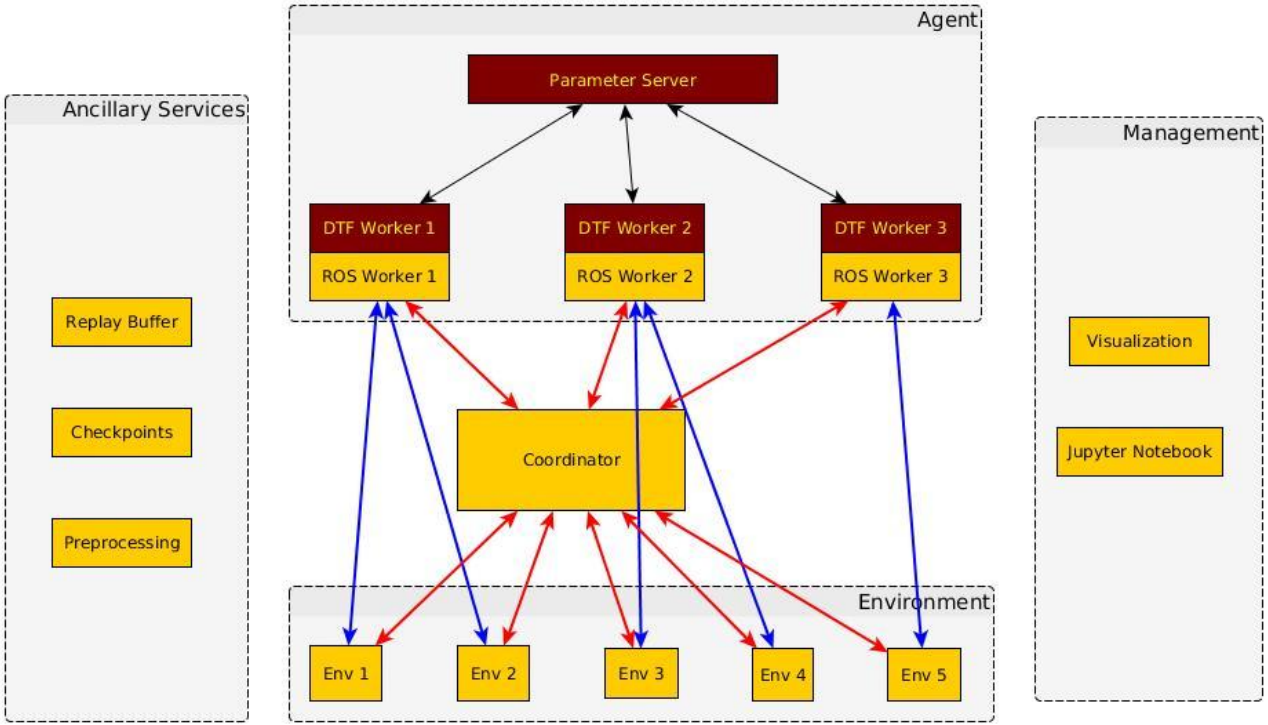
Sequence of Operations




Parameter Server
$w' = w - \eta\Delta w$
w
Δw
Model Replicas
Agent
Parameter Server
DTF Worker 1
DTF Worker 2
DTF Worker 3
ROS Worker 1
ROS Worker 2
ROS Worker 3
Coordinator
Environment
Env 1
Env 2
Env 3
Env 4
Env 5

Ancillary Services

- Visualization
- Remote Access
- Replay Buffers
- Checkpoints
- Preprocessing



The Way Forward

- Complete Framework for Distributed RL
  - Multi Agent Reinforcement Learning
  - Game Theoretic Research Problems
- Arbitrary scaling → Internet scale?
- Native integration with Robotic platforms


Start
LIVE DEMO

Technische Universität München

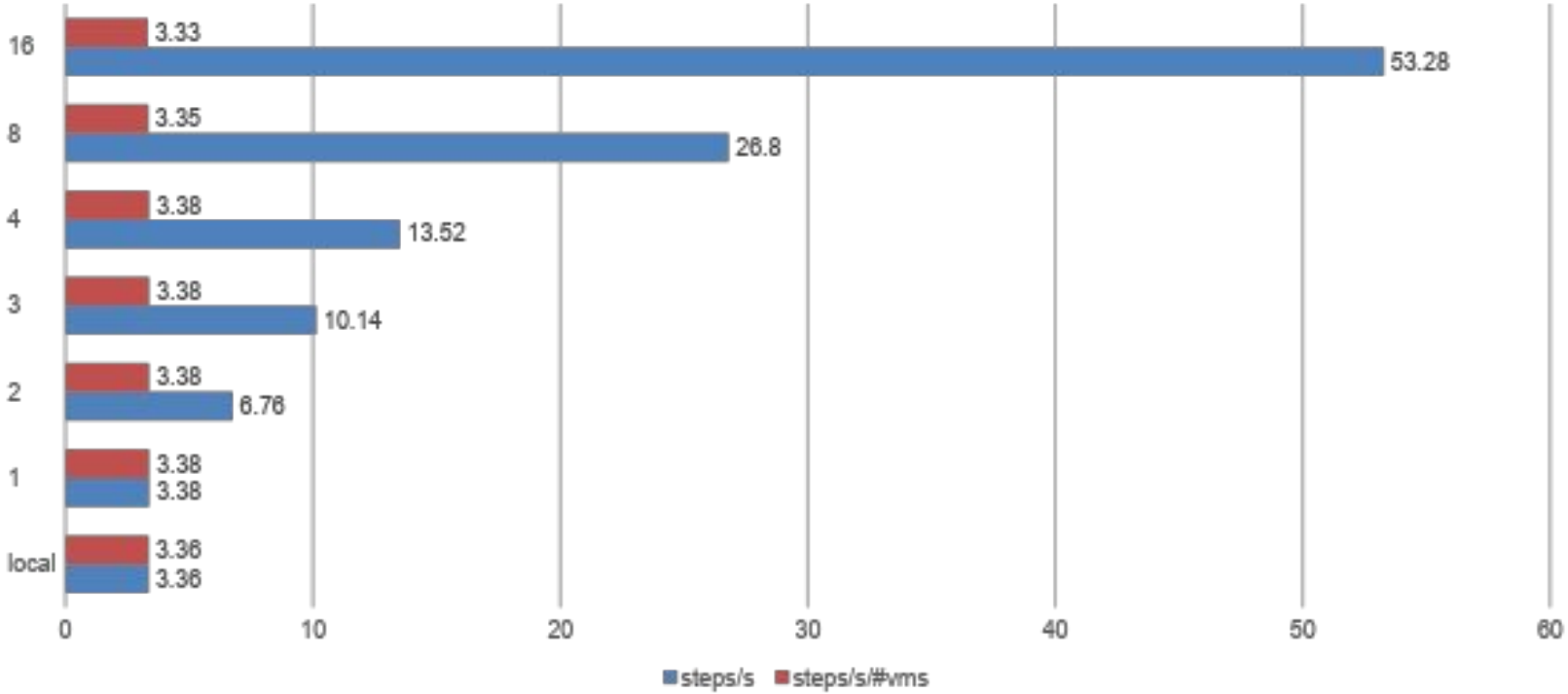
Runtime and Algorithm Tests

Scalability test: Setting

- Test of distribution efficiency on the LRZ cloud
- Compared average trainsteps/second on equal batch size
- 1,2,3,4,8,16 vms (4-cpu each) with one 2-cpu parameter server

38

Scalability test: Result



→Almost no loss in computational speed using 16 machines!

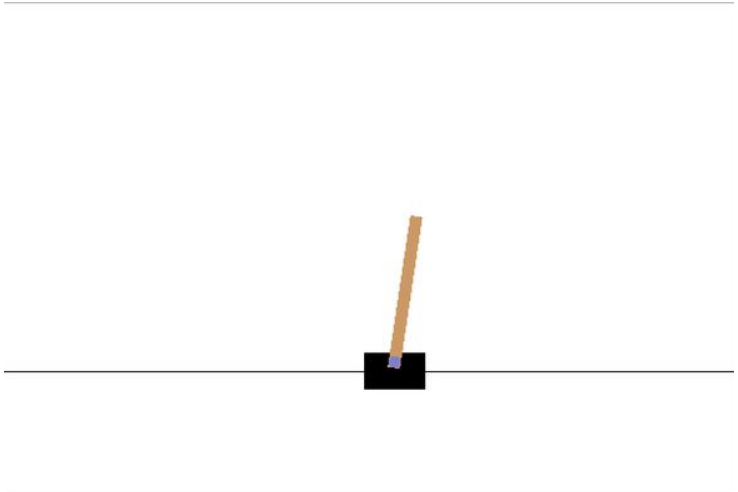
Hyperparameter Study: Challenges

- Finished Implementation contains a lot of Parameters
- Feasible Parameter combinations for successful training are rare
- Successful training runs take several hours for hard environments
- A singular test run is not representative as there is also randomness involved
- Impact of Hyperparameters is dependent on each other and on environment

40

Hyperparameter Study: Setting

- Study conducted on Cartpole
- Balance problem from OpenAI Gym
- Less compute intensive environment allows for more tests
- Different settings were averaged over 15 runs



Hyperparameter Study: Learning rate

- Learning rate has big impact on algorithm stability
- Too small learning rate may lead to being stuck in small local optima and learning is slower
- Too big learning rate version may not be stable in global optimum

First maximum reward	199	182	252	2587	1636	>5000	>5000	>5000	211	189	2706	555	2858	180	1544
Solved After	429	209	4451	>5000	1749	>5000	>5000	>5000	261	214	3952	844	3270	230	1757

Cartpole A3C Algorithm runs with learning rate=0.02

42

Hyperparameter Study: Learning rate

- Learning rate has big impact on algorithm stability
- Too small learning rate may lead to being stuck in small local optima and learning is slower
- Too big learning rate version may not be stable in global optimum

Learning rate	0.003	0.005	0.01	0.02
Solved	93%	80%	100%	86%
∅ Solved After	1389	1047	585	1635

Average over 15 runs with different learning rates

Conclusion

We achieved:

- Workbench for state of the art reinforcement algorithms
- Computation efficient and fully automated scalability
- Scalable environments through ROS and Distributed Tensorflow

44

We are ready for your questions:)

Adrian Sieler

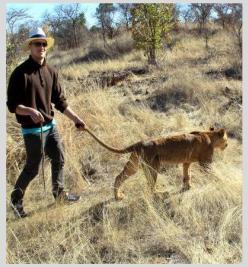
- Master Mathematics in Data Science
- Expert in Big Data related Buzzwords
- Loves scrambled egg

Haris Jabbar

- Master CSE Student
- Roboticist
- Wantrepreneur
- #LikeHumansDo

Christian Holland

- Master Mathematics
- AI enthusiast
- Love to travel
- Tea connoisseur



Egor Labintcev

- first-year Master student in CSE
- ML/DS
- CNN part, general coding
- loves bretzels

45