Playing Starcraft and Saving the World Using Multi-Agent RL!

29 October 2021

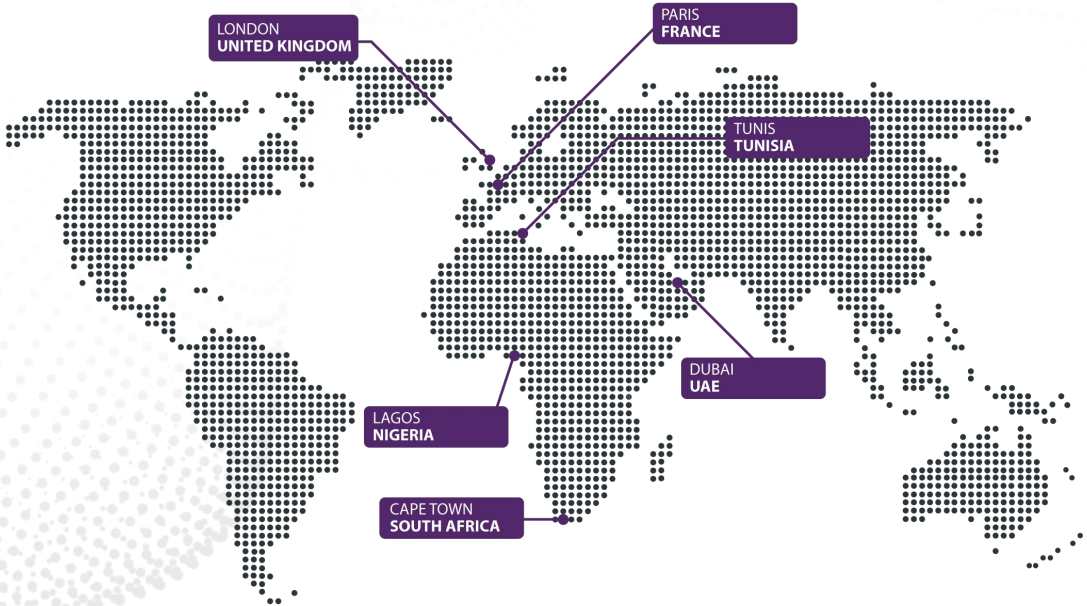
About InstaDeep

One Team: AI Research, Engineering, Software & Infrastructure

20 **Insights & Visualisation**
UX/UI Designers, Unity Developers Storage & CC Engineers

84 **AI Research & Engineering**
ML Engineers Research Engineers Data Scientists

22 **High Performance Engineering**
HPC Engineers Software Engineers DevOps Specialists



InstaDeep™

Our Projects

- Designing novel immunotherapies and world-first tools for biologists.
- Automatically routing circuit boards.
- Optimising train scheduling in Germany.
- Creating open-source libraries for multi-agent reinforcement learning.
- And more...

DeepChain by InstaDeep

MAVA

BIONTECH

DB

NVIDIA ELITE PARTNER

UNIVERSITY OF OXFORD

DeepMind

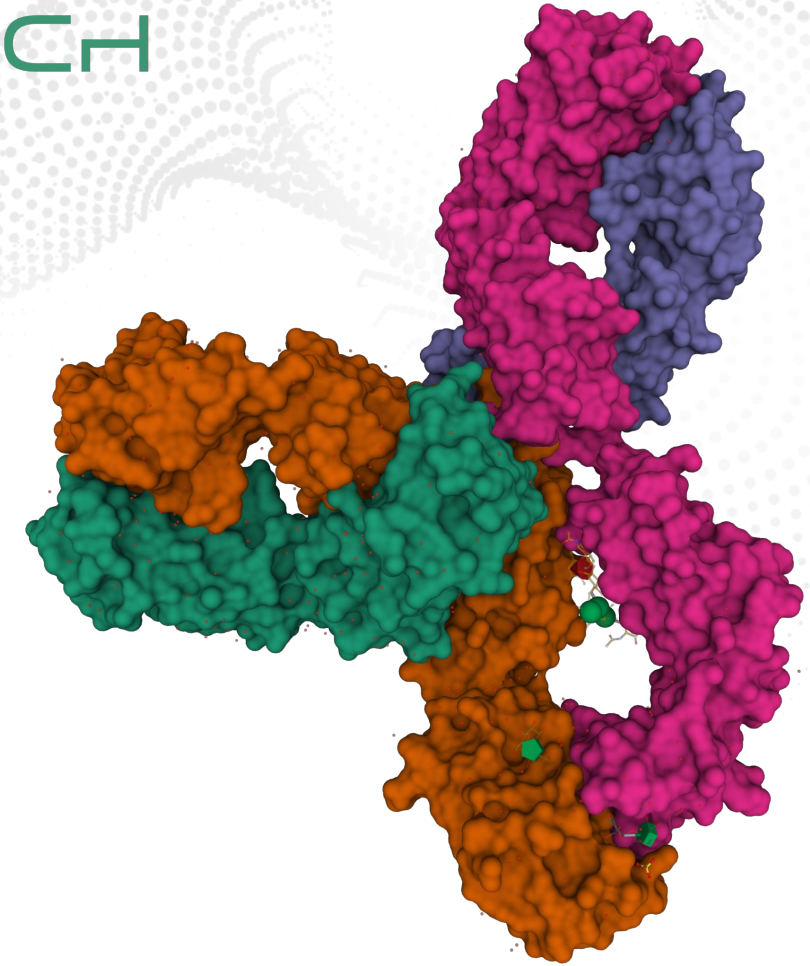
AI 100 CBINSIGHTS 2020

AI 2021 100 CBINSIGHTS

Collaboration with BIONTECH

Developing world-first tools for biologists.

- Machine learning models to develop **novel immunotherapies** for a range of cancers and infectious diseases.
- **Drug Design** - e.g. mRNA vaccines and therapeutics
- **Protein Engineering**



Designing a Prospective COVID-19 Therapeutic with Reinforcement Learning

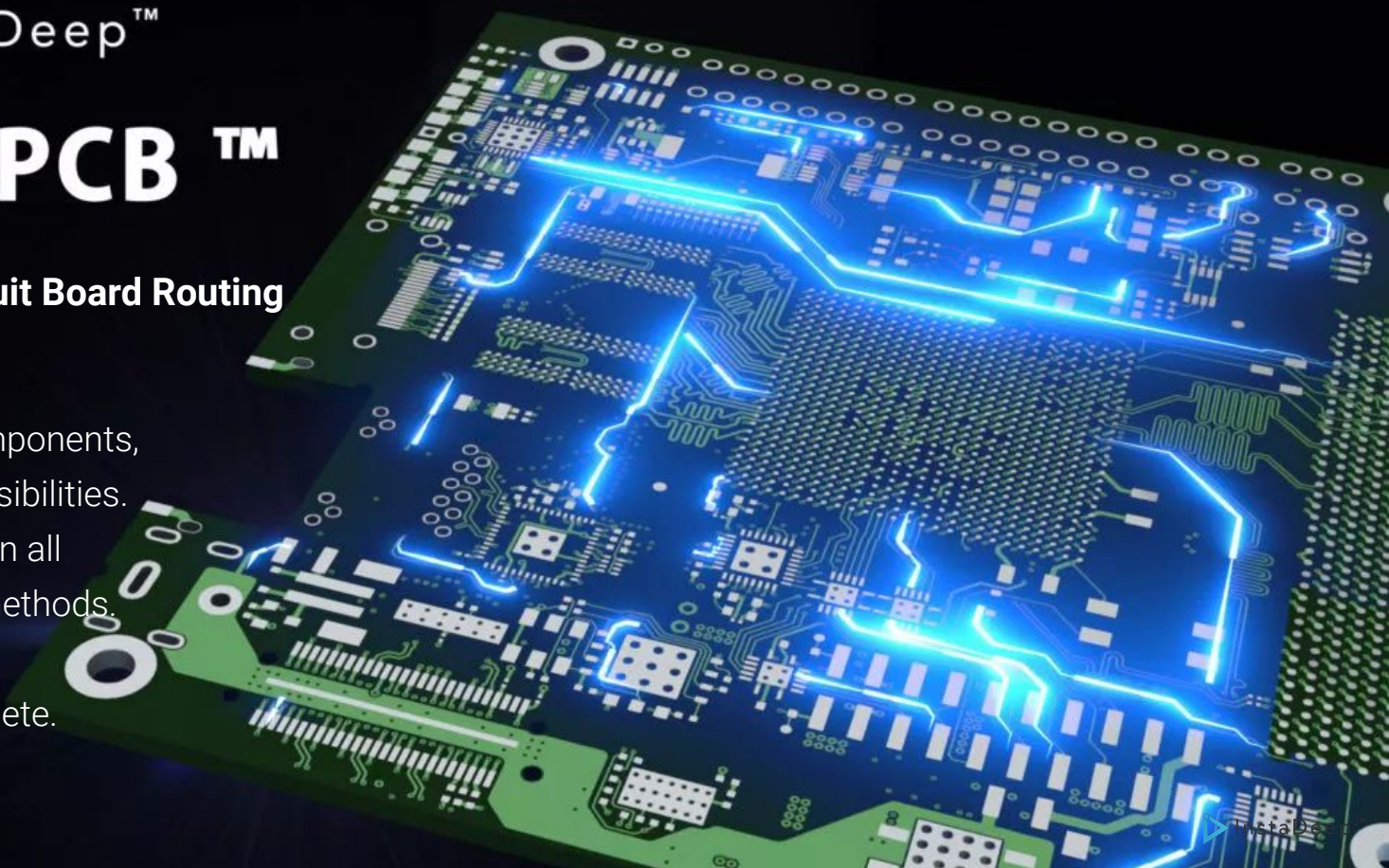
Marcin J. Skwark* InstaDeep
Nicolás López Carranza InstaDeep
Thomas Pierrot InstaDeep
Joe Phillips InstaDeep
Slim Said InstaDeep
Alexandre Laterre InstaDeep
Amine Kerkeni InstaDeep
Uğur Şahin BioNTech
Karim Beguir InstaDeep

InstaDeep™

DeepPCB ™

Solving Circuit Board Routing using AI

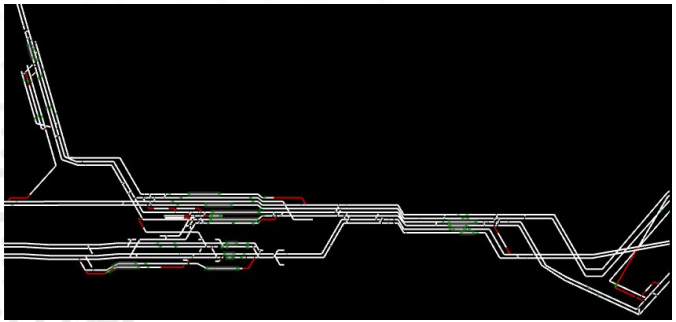
- Many components, many possibilities.
- Faster than all existing methods.
- Routing is NP-Complete.



Train Scheduling with DB BAHN

Solving Train Scheduling using Multi-agent Reinforcement Learning

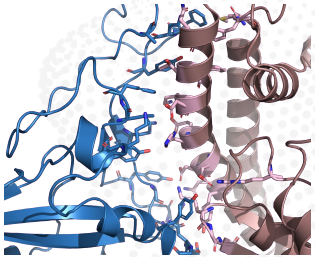
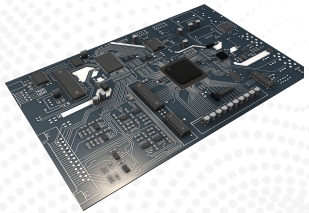
- Key Challenge: **How to efficiently manage dense traffic on complex railway networks?**
- Why this problem is difficult:
  - Many trains and potential routes - solution must **scale**
  - Must accommodate delays, train breakdowns, etc - solution must be **robust**
  - Trains must be on time - solution must be **efficient**
- **MARL** provides a way to achieve this
  - Each train represents an agent
  - Agents must **coordinate** to ensure all trains arrive on time



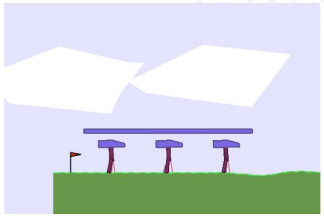
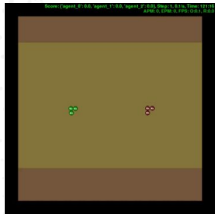
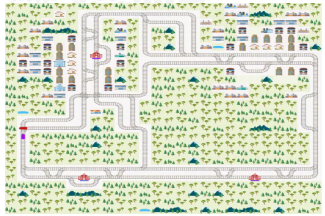
Mava: A Research Framework for Multi-Agent RL

Speed up research in Multi-Agent RL with useful architectural and environment integrations.

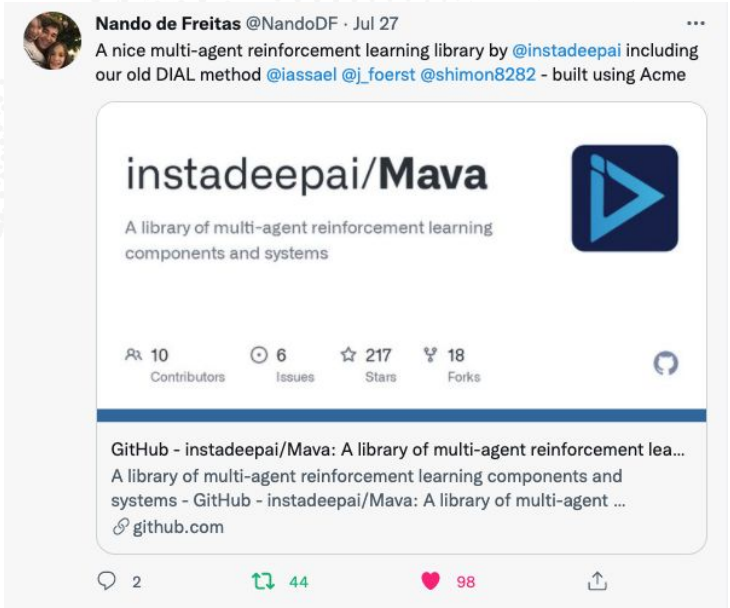
- Design with **research** and **production** in mind.


- Mava integrates with many popular **research environments** and Mava systems have achieved **SOTA performances** on some of them.

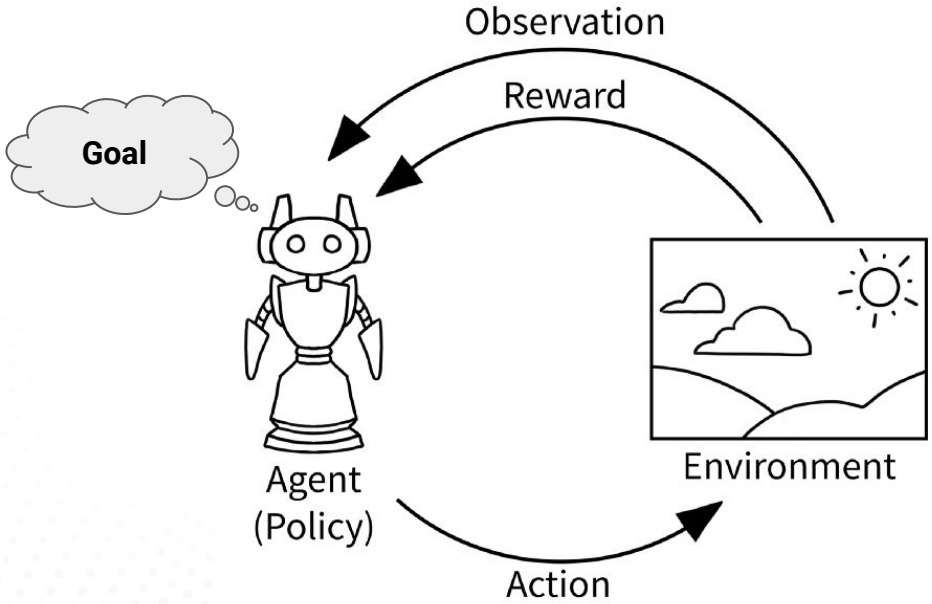




Recognised by top researchers at DeepMind including **Marc Lanctot** and **Nando de Freitas**.



Reinforcement Learning (RL) Basics

Reinforcement Learning



RL is **goal-directed learning from interaction** (trial and error).

Learn - **what to do** (*how to map situations to actions*, as to *maximize a numerical reward*).

Source: Image Source

Elements of RL - Policy π

Policy: Mapping from (*perceived*) states to actions.

Deterministic:

$$a = \pi(s)$$

Stochastic:

$$\pi(a|s) = P[A_t = a|S_t = s]$$

$$a \sim \pi(A|s)$$

In Deep RL - policies are parameterized by the weights of Neural Network π_θ

Source: Image Source

Elements of RL - Reward & Return - R & G

Reward: Scalar feedback - how good our current state is.

$$R_t = R(s_t, a_t) \quad \text{or} \quad R_t = R(s_t, a_t, s_{t+1})$$

Return: Expected cumulative reward over time.

Formulation (infinite-time horizon):

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

, where $\gamma \in [0,1]$ is a discount factor, that penalise rewards in the future.

Elements of RL - Value functions

Value: What is good in the **long run**.

Value of state *(s)* /state-action *(s,a)*: How good is a *(s)* or *(s,a)* pair, i.e. the expected return *(G_t)* if you start at *(s)* or *(s,a)* and then act according to your policy.

Formally,
State-value function:

$$V^{\pi}(s) = E_{\pi}\left[G_t \mid S_t = s\right]$$

Action-value function (Q-value function):

$$Q^{\pi}(s, a) = E_{\pi}\left[G_t \mid S_t = s, A_t = a\right]$$

Efficiently estimating values is critical to RL.

Elements of RL - Markov Decision Process (MDP)

MDPs: Formal way to describe sequential decision-making (RL).

Markov Property: Transitions only depend on the most recent state and action, and no prior history ~ *current state contains all necessary information*.

A MDP consists in a tuple (**S**, **A**, **P**, **r**, γ) where:

- **S** is a finite set of states.
- **A** is a finite set of possible actions.
- **P** : S x A x S -> [0,1] , is the transition probability.

$$P(s'|s,a) = P[S_{t+1} = s' | S_t = s, A_t = a]$$

- **R** : S x A -> R , is the reward function.
- γ: the discount factor.

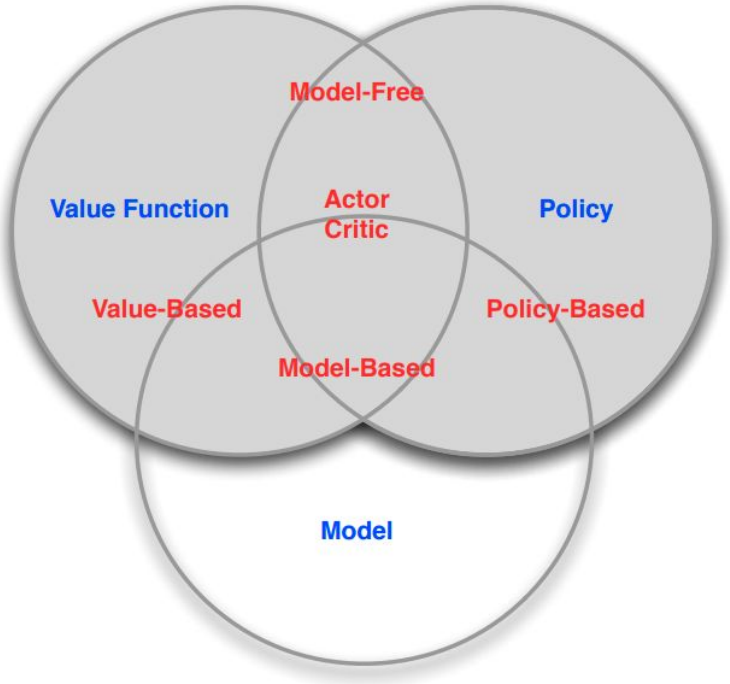
Elements of RL - Trajectory (τ)

Trajectory: Sequence of states, actions and rewards in the world.

$$\tau = s_0, a_0, r_1, s_1, a_1, \ldots$$

We use R_{t+1} instead of R_t to denote that the reward due to A_t because it emphasizes that the next reward R_{t+1} and next state S_{t+1} are jointly determined. Unfortunately both conventions are widely used in the literature.

RL Agent Taxonomy



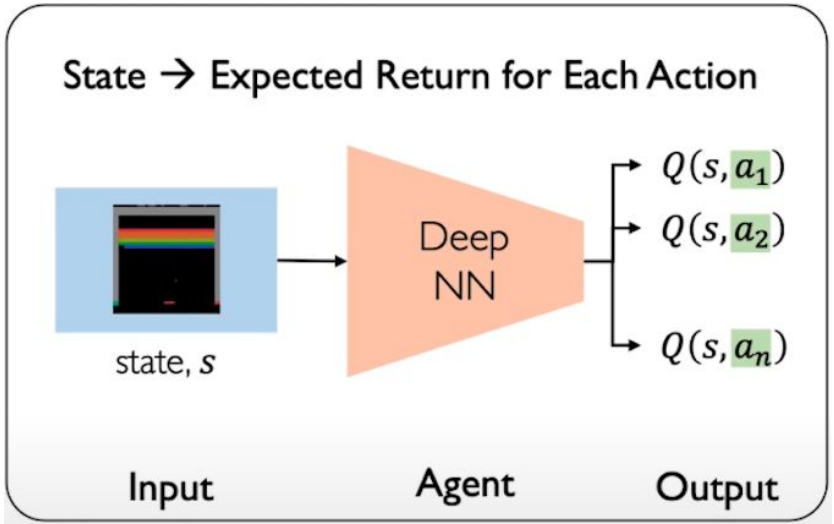
Source: David Silver RL Lecture Notes

Kinds of RL Algorithms

	Value-Based Methods	Policy-Based Methods	Actor-Critic Methods
LEARN	$Q_\theta(s,a)$	$\pi_\theta(a\|s)$	Actor $\pi_\theta(a\|s)$ Critic $Q_w(s,a)$
ACT	$a = \underset{a}{argmax}\ Q(s,a)$	$a \sim \pi_\theta(a\|s)$	$a \sim \pi_\theta(a\|s)$
E.G.	DQN.	Reinforce.	A2C/A3C, DDPG, PPO, etc.

Deep Q-Learning (DQN)

First RL algorithm that was shown to work directly from raw pixels, in a variety of environments.



DQN Learning

Target

Predicted

$$\mathcal{L}(w) = \mathbb{E}\left\{\left(r + \max_{a' \in \mathcal{A}} Q_w(s', a') - Q_w(s, a)\right)^2\right\}$$

DQN Acting

$$a \leftarrow \begin{cases} \underset{a}{argmax}\ Q(s, a) & \text{with probability } 1 - \varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$

Souce: Human-level control through deep reinforcement learning , MIT 6.S191: Reinforcement Learning

Deep Q-Learning (DQN) - Main Breakthroughs

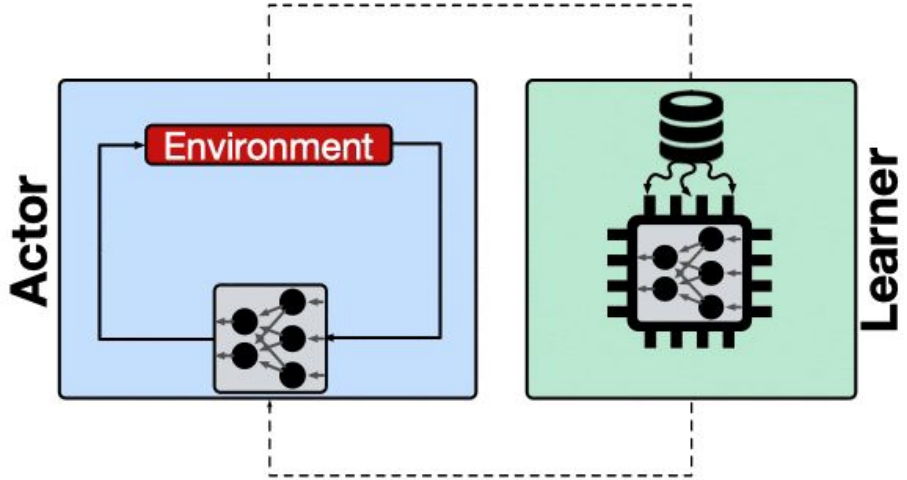
Naive Q-Learning oscillates or diverges with neural networks:

- ❏ Data is sequential:
  Successive sample are correlated, non-iid.
- ❏ Policy changes rapidly with slight changes to Q-values

DQN provides a stable solution to deep value-based RL:

- ★ Use experience replay
  Break correlations in data, bring us back to iid setting.
- ★ Freeze target Q-network
  Avoid oscillations
  Break correlations between Q-network and target

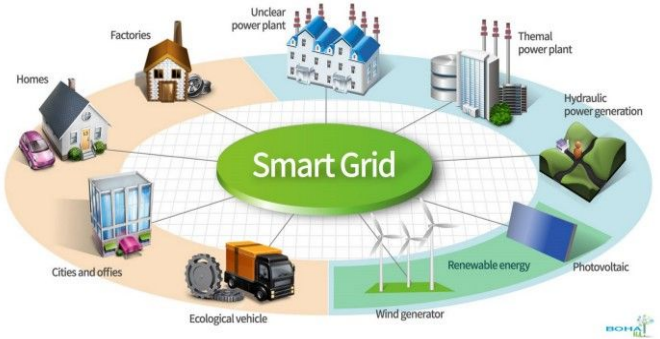
Distributed RL

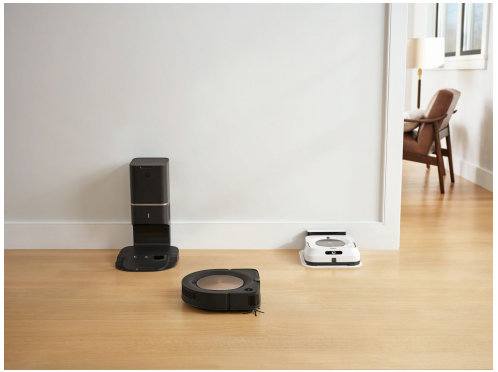


Source: [Massively Large-Scale Distributed Reinforcement Learning with Menger]

Introduction to Multi-Agent Reinforcement Learning (MARL)

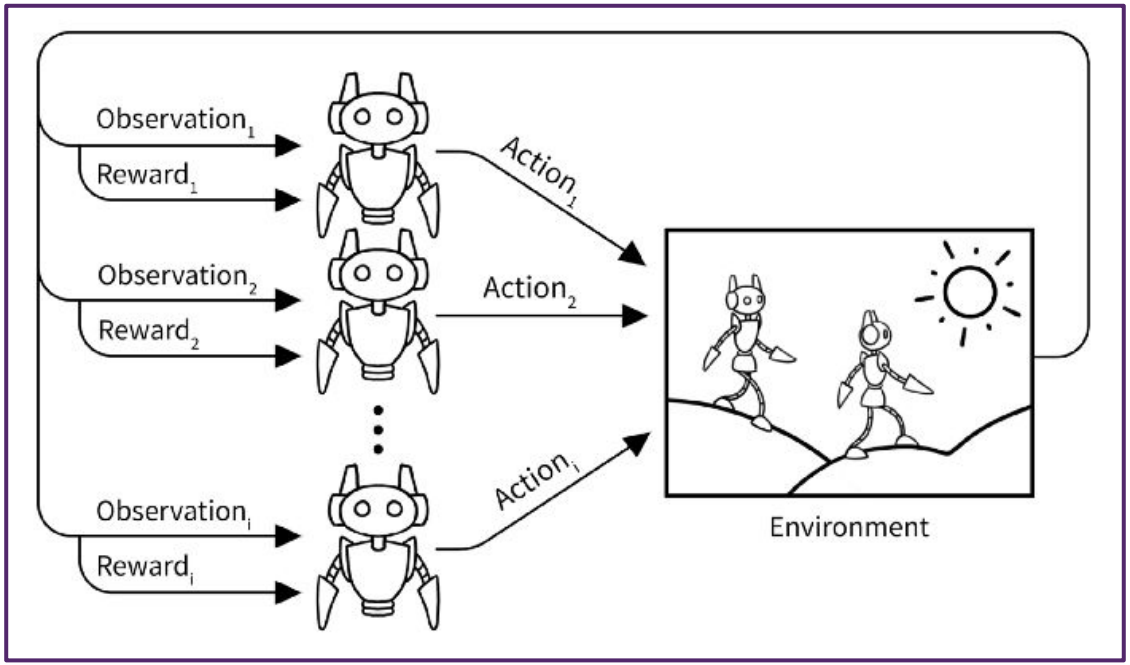
Why MARL - Multi-Agent Problems are everywhere





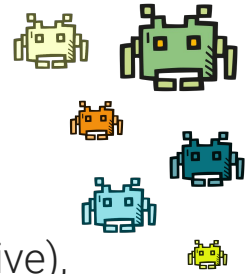
Source: Railway Smart Grid Traffic Home Robots

MARL



Source: [Image Source]

MARL - Considerations

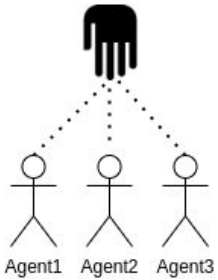


Reward - shared team reward (cooperative), individual adversarial rewards (competitive), mixed (combination).

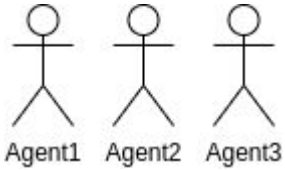
Observations - Partial or fully observability.

Policies or Value Function - Centralized or decentralized.

Centralized



Decentralized



Training - Centralized or decentralized.

Multi-Agent MDP

A MDP consists in a tuple (S,**U**,P, r, $\mathbf{I}$, γ) where:

- S is a finite set of states.
- **U** is joint action space,

$$u^i \in U, \text{ where } i \in I$$
$$\text{e.g. } u^1 \rightarrow \text{ action of agent 1.}$$

- P : S x **U** x S -> [0,1] , is the transition probability.
- R : S x **U** -> R , is the reward function.
- $\mathbf{I}$ is a finite set of agents.
- γ: the discount factor.

Equivalent to MDP with factored/high dimensional action space.

Dec-POMDP - Popular Formulation

Partially Observable:

- Observation function: conditions on global state **s** and agent index **i**.

$$O(s,i) : \mathrm{S\ x\ I} \rightarrow Z$$

- Recurrence important - need to remember past states/memory[1].

Recurrence + Decentralised Policies:

- Policies are conditioned on an agent's action-observation history:

Action-observation history (for agent i): $\tau^i \in T \equiv (Z \times U)$

Decentralised policies: $\pi^i(u^i|\tau^i) : T \times U \rightarrow [0,1]$

In general, Dec-POMDP do not have the Markov property [2].

Sources: 1]Deep Recurrent Q-Learning for Partially Observable MDPs , 2] Decentralized POMDPs

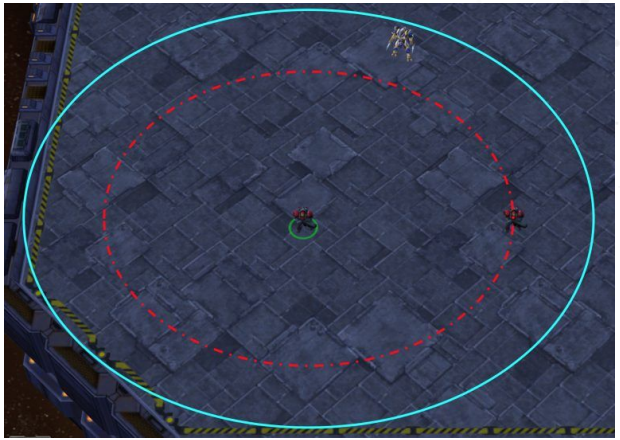
Decentralization

Natural: Systems are naturally decentralized e.g. distributed sensors.
Artificial: Improsed decentralization e.g. by forcing agents to have partial view of global state to improve tractability of learning.

Starcraft II

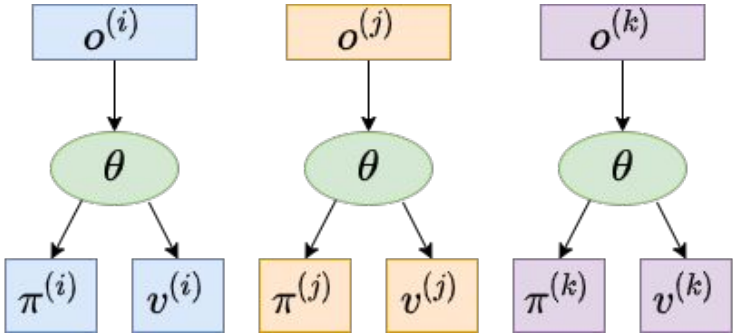
Global State

Observation



Training MARL Systems - Centralised Training & Decentralised Execution (CTDE)

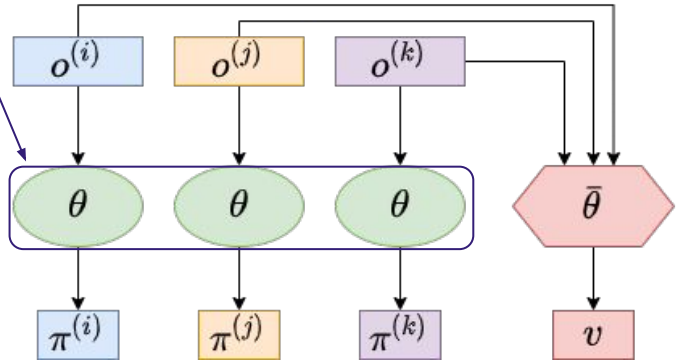
Decentralised training & execution



Problems
- Non-stationary.
- Hard to coordinate.

Centralised training & decentralised execution

Same network when using **parameter sharing**.



Problems
- Learning centralized V function over complex action space - challenging.

∴ Learn factored Value functions - VDN [Sunehag et al] , QMIX [Rashid et al], etc.

Single Agent RL vs MARL

Single agent

Pros

Full observability.

Cons

Does not scale.

Multi agent

Pros

Scalable.

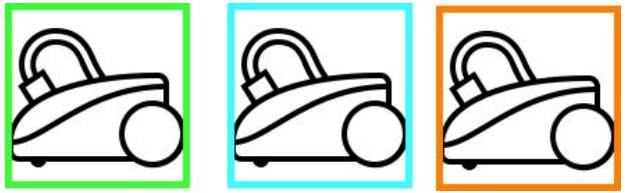
Decompose problem.

Cons

Partial observability.

Non-stationarity.

Single Agent RL vs MARL - Toy Example

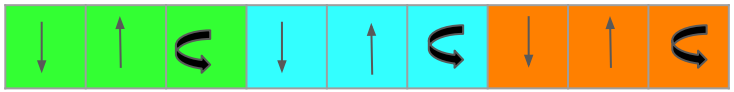


Three robots want to clean a house.
$s \in R^3$, $a \in R^3$.

Single agent

State

Actions

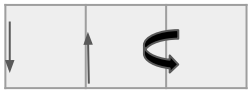


Multi agent

With CTDE.

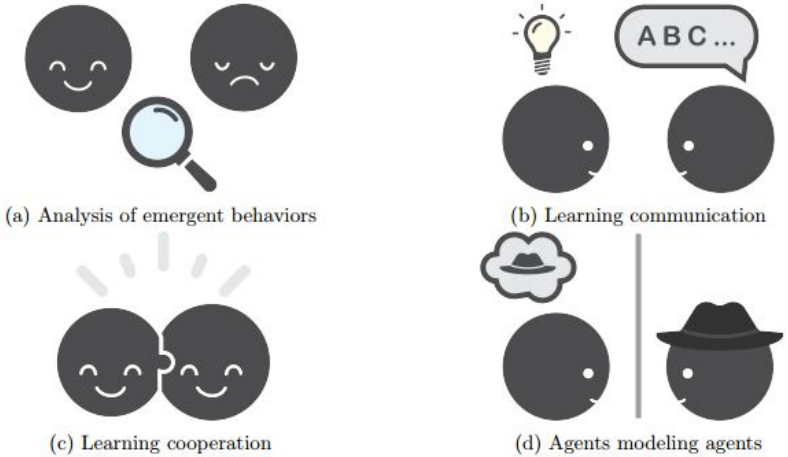
State

Actions



What happens with 1000 agents?

Other aspects in MARL



(a) Analysis of emergent behaviors

(b) Learning communication

(c) Learning cooperation

(d) Agents modeling agents

Source: A Survey and Critique of Multiagent Deep Reinforcement Learning

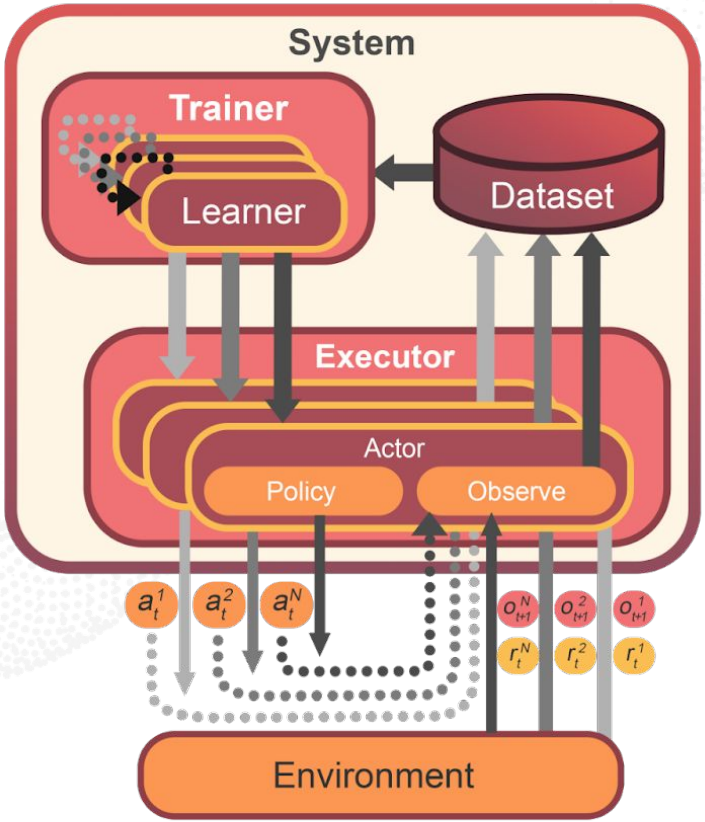
Mava: a research framework for distributed multi-agent reinforcement learning

InstaDeep™

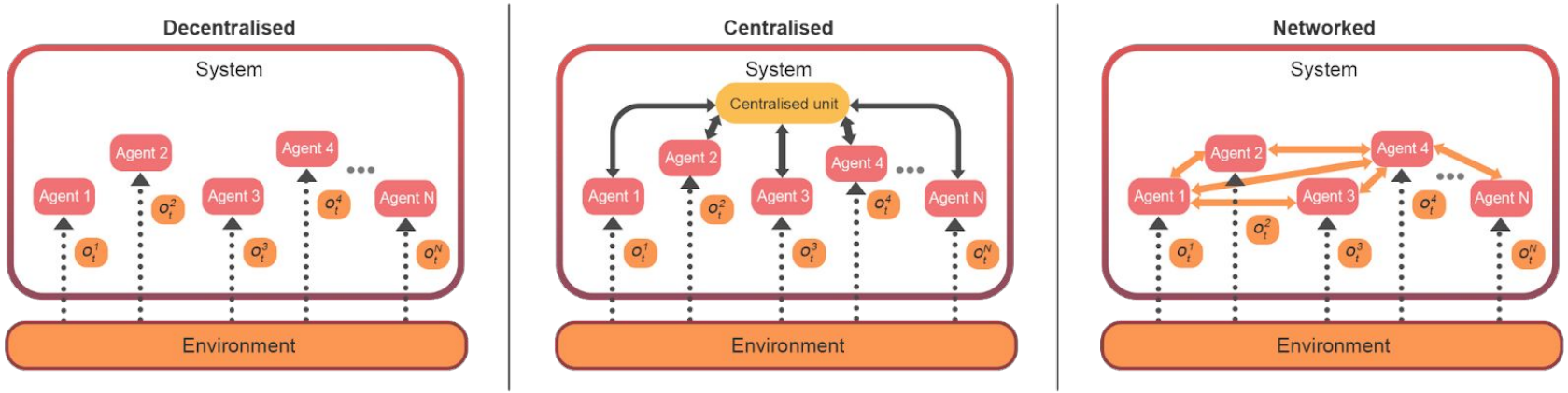
Mava in a nutshell

System design

- Executor
  - Multiple actors
- Trainer
  - Multiple learners
- Dataset
  - Interface between executor and trainer



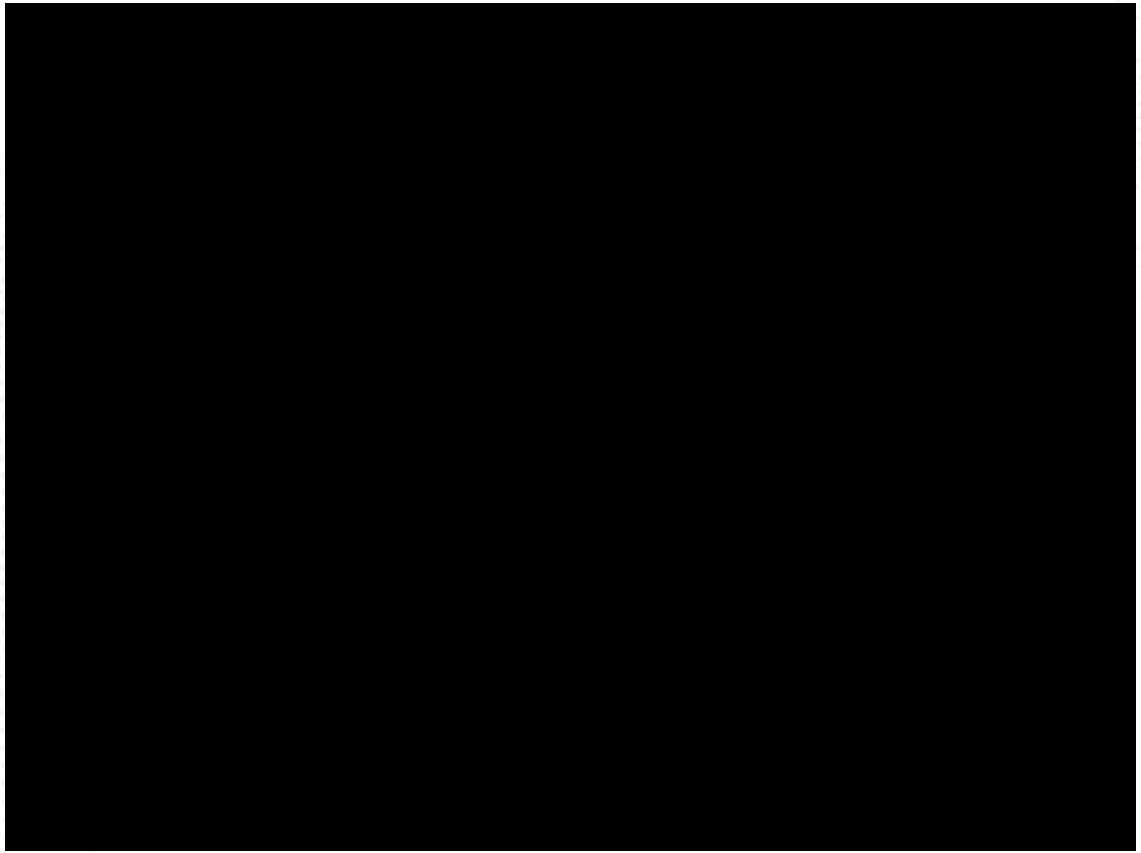
Mava: System Architectures



More Components

- ❏ Communication
- ❏ Replay Stabilisation (fingerprints)
- ❏ Value-decomposition (mixing)
- ❏ Prioritized experience replay
- ❏ And many more.

Mava in a nutshell



Mava in a nutshell

What does Mava bring to the table?

1

Designed to be flexible for research and production.

2

Integrates with DeepMind RL ecosystem: Acme, Reverb, Launchpad

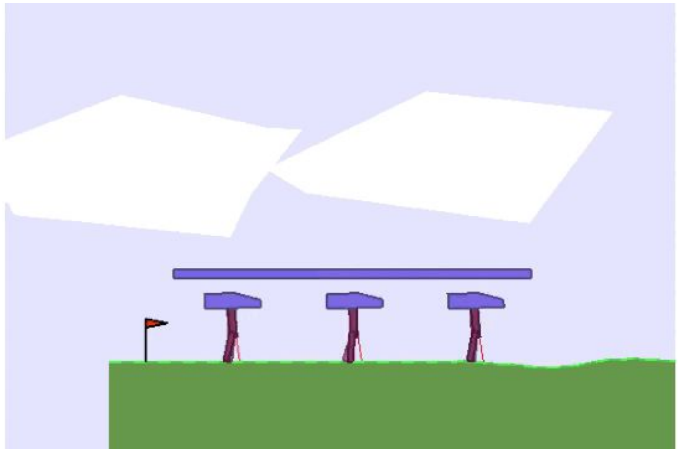
3

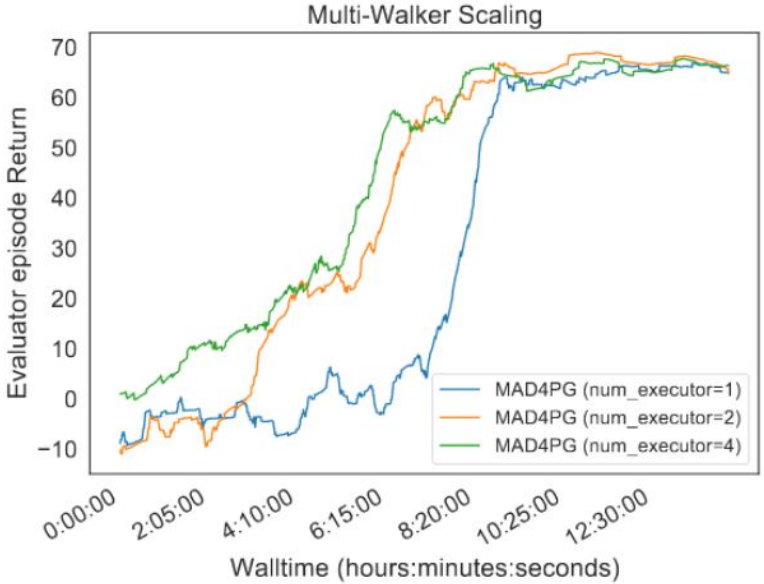
Open-source: active development and dedicated team of maintainers

4

Built with scaling mind for real-world applications

Mava: SOTA on Multi-Walker





Questions?

The only stupid question is the one you were afraid to ask but never did. -Rich Sutton

Sources

- Reinforcement Learning: An Introduction - Richard S. Sutton and Andrew G. Barto
- Spinning Up in Deep RL
- Factored Value Functions for Cooperative Multi-Agent Reinforcement Learning - Shimon Whiteson
- MIT 6.S191: Reinforcement Learning MIT 6.S091: Introduction to Deep Reinforcement Learning (Deep RL)
- Learning to Communicate with Deep Multi-Agent Reinforcement Learning - Jakob Foerster
- Survey Papers - Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms , A Review of Cooperative Multi-Agent Deep Reinforcement Learning
- Some slides adapted from on Hands-on Multi-Agent Reinforcement Learning using Mava

Let’s train some agents!