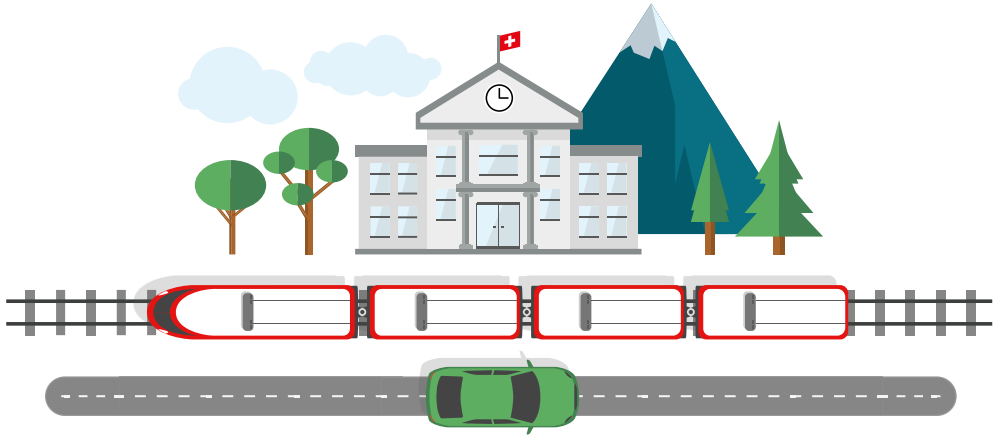


FLATLAND

ALESSANDRO LOMBARDI
FIORENZO PARASCANDOLO

DEEP LEARNING COURSE

M.SC. IN ARTIFICIAL INTELLIGENCE
UNIVERSITY OF BOLOGNA
2019-2020

WORKFLOW

Challenge description: official site

Preliminary phases:

- Understand the environment
- Work on single agent setting to get used with tools, theories and techniques
- Read some literature on Multi-Agent Deep Reinforcement Learning (MADRL)

Implementation:

- Study PyTorch and implement algorithms
- Improve the environment (rewards, deadlocks, statistics and metrics)
- Test and evaluate

THE ENVIRONMENT

The **Agent Step routine** represents a crucial to understand the internals of the Flatland environment. A readable pseudo-code can be found on the report.

Rewards (present in the Flatland baseline):

- The step penalty is equal to the speed of the agent
- When an agent reaches its destination the reward is 0

Malfunctions and speeds (present in the Flatland baseline)**:**

- Modeled as probability distributions

Deadlocks detection (not present in the Flatland baseline)

- Implemented by us and used both as a performance index and for Rewards Shaping

METRICS

Normalized_score	$\frac{score}{max\_steps \cdot n\_agents}$
Accumulated_normalized_score	$\frac{\sum normalized\_score}{N}$
Completion_percentage	$100 \cdot \frac{tasks\_finished}{n\_agents}$
Accumulated_completion	$\frac{\sum completion\_percentage}{N}$
Deadlocks_percentage	$100 \cdot \frac{n\_deadlocks}{n\_agents}$
Accumulated_deadlocks	$\frac{\sum deadlocks\_percentage}{N}$

MADRL

Flatland: **Fully cooperative** and given non-global observations can be modelled as a **Decentralized POMDP**

Challenges:

- Non-Stationarity
- Partial Observability
- Scalability
- Information structure
- Multi-agent credit assignment

Approaches and solutions:

- Communication techniques
- Recurrent networks
- Parallel environments
- Centralized/Decentralized and centralized learning for decentralized execution
- Meta-learning
- Rewarding: shaping and global vs local

REWARDS SHAPING

Penalties/Bonus:

- Stop_penalty
- Invalid_action_penalty
- Deadlock_penalty
- Shortest_path_penalty_coefficient
- Done_bonus
- Uniform_reward

Pros:

- Accelerate learning
- Learn complex behaviors

Cons:

- Cobra effect

"Historically, the government tried to incentivize people to assist them in ridding the area of cobras.
If citizens brought in a venomous snake they had killed, the government would give you some money.
Naturally, people started breeding venomous snakes."

PS-PPO

PARAMETER-SHARING PROXIMAL POLICY OPTIMIZATION

- Policy-based
- On-Policy
- Model-Free
- Shared vs non-shared neural network vs shared recurrent
- Actor-critic
- Centralized learning for decentralized execution
- Disposable trajectories
- Parameter sharing and agent ID

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right],$$

$$L(s, a, \theta_k, \theta) = -\min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a), clip\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon\right) A^{\pi_{\theta_k}}(s,a)\right)$$

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017

Jayesh Gupta, Maxim Egorov and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning, 2017

PS-PPO: IMPLEMENTATION

Neural Network

2 nns according to **Actor-Critic** paradigm

- Actor controls how our agent behaves, to predict the probability of an action given the state (s).
- Critic measures how good the action taken is.

Possibility to set the shared portion of the network as a RNN using a **Long Short-Term Memory (LSTM)**.

Training loop

1. Collect set of trajectories by running the current policy (memory is updated every time an agent does an effective decision)

2. Compute advantage estimates based on the current value function

3. Update the policy by maximizing the PPO-clip objective when the memory is full with mini-batch gradient descent

Advantages: $A_t^{(N)} = V_t^{(N)} - V^\pi(s_t)$

- **N-steps**

$$V_t^{(N)} = \sum_{i=t}^{t+N-1} \gamma^{i-t} r_i + \gamma^N V^\pi(s_{t+N})$$

- **Generalized-advantage-estimation (GAE)**

$$V_t^{(GAE)} = (1-\lambda) \sum_{N>0} \lambda^{N-1} A_t^{(N)}$$

$$\delta_t = r_t + \lambda V^\pi(s_{t+1}) - V^\pi(s_t)$$

$$A^{(GAE)} = \sum_{N>0} (\lambda\gamma)^N \delta_{t+N}$$

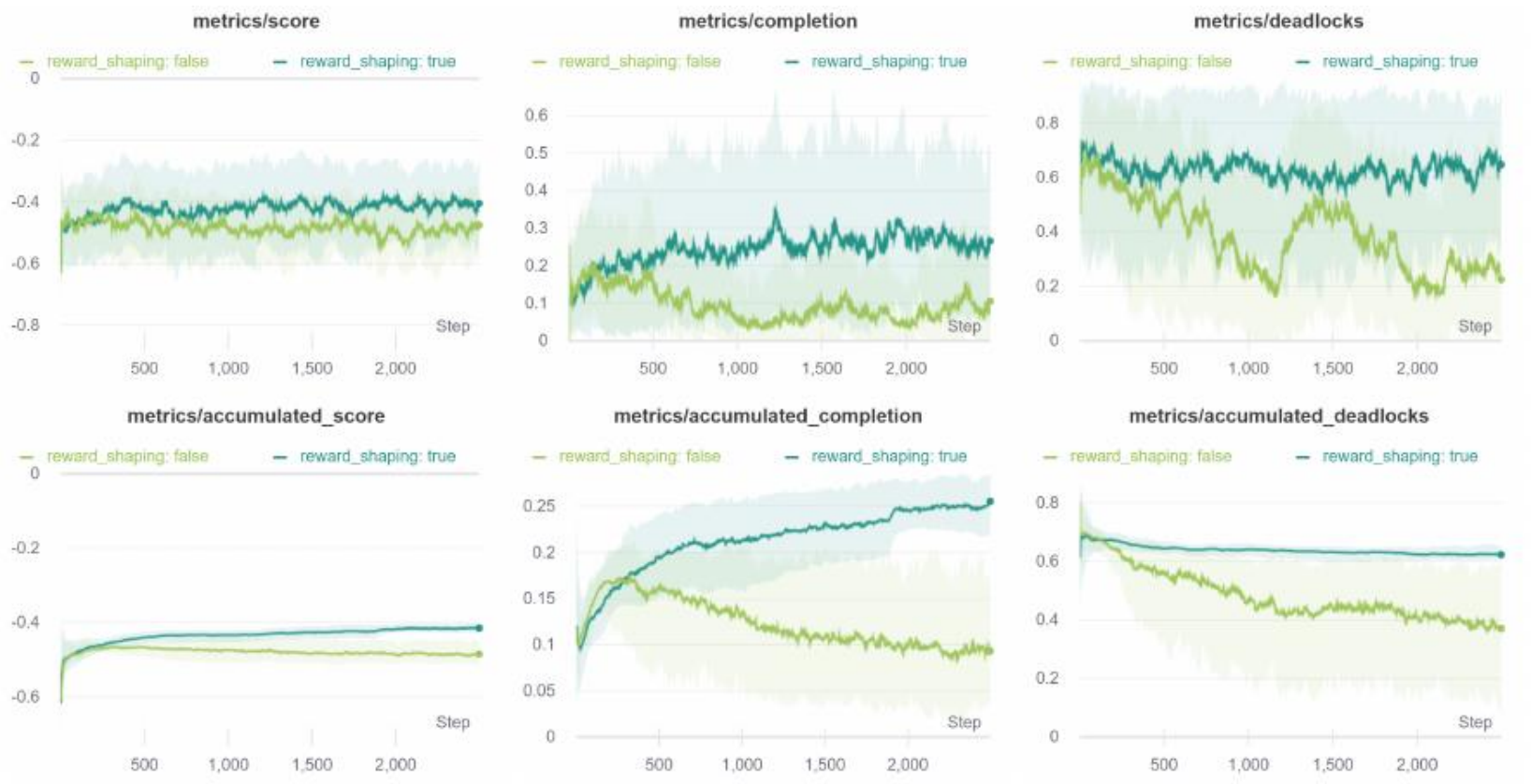
- The parameters $\gamma, \lambda \in [0,1]$ adjust the bias-variance trade-off.

Batch-mode

- **Classical**: individual transitions in the same mini-batch are preserved
- **Shuffle**: individual transitions in the same mini-batch are not preserved (using such samples causes learning instability)

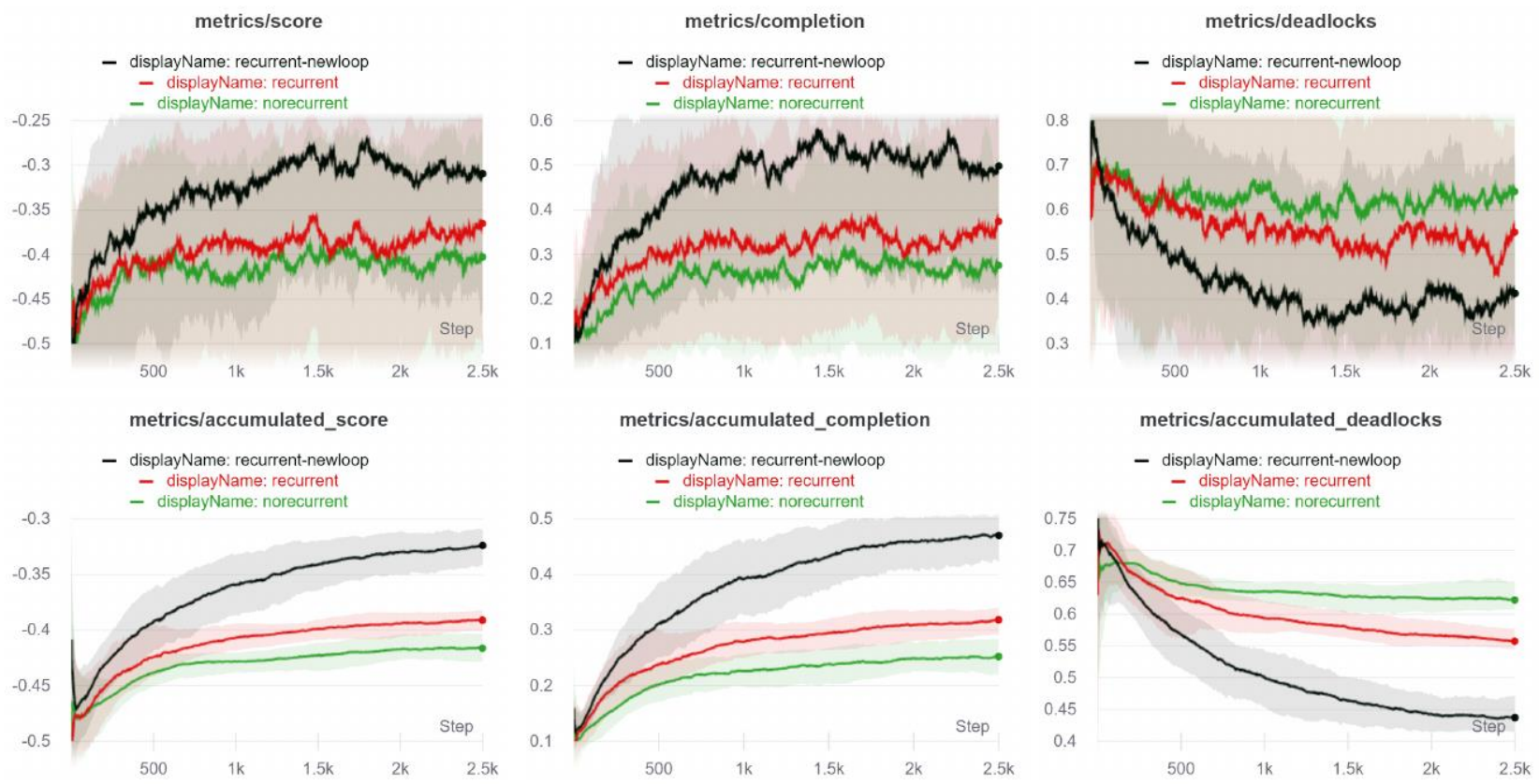
PS-PPO: RESULTS

- Grid search (sweep)
- Environment with malfunctions and different speeds
- Rewards Shaping vs no Rewards Shaping (wandb project)



PS-PPO: RESULTS

- Recurrent vs no Recurrent
- New loop to add at most one deadlock observation in memory for each agent when it occurs (also for agents with different speeds)



D3QN

DUELING DOUBLE DEEP Q NETWORK

- Value-based
- Off-Policy
- Model-Free
- Shared vs non-shared neural network
- Centralized learning for decentralized execution
- Uniform Experience Replay or Prioritized Experience Replay
- Fingerprints to solve non-stationarity

$$\delta = r + \gamma Q\left(s', argmax_{a'} Q\left(s', a'; \theta_k\right); \theta_k^-\right) - Q(s, a; \theta_k)$$

$$L(\theta_k, \theta_k^-) = E_{(s,a,r,s')\sim D}\left(\delta\right)^2$$

Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip H. S. Torr, Pushmeet Kohli, Shimon Whiteson. Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning (2017)

D3QN: IMPLEMENTATION

Neural Network

2 nns, called **Online** and **Target**

- Online: used as a policy to control agents, to predict the action to perform in the next state (s') and the Q value in the current (s).
- Target: computes the Q value of s' and is **soft updated** using the parameters of the online one.

Dueling structure, the network(s) computes the advantage A(s, a) and the state value V(s) and computes the Q value from them by the formula

$$Q(s,a) = V(s) + (A(s,a) - \frac{1}{|\mathcal{A}|}\sum_{a} A(s,a))$$

Training loop

- Epsilon-greedy for exploration vs exploitation trade-off
- Memory is updated every time an agent does an effective decision
- Every n agent's memory updates the network is improved
- Mini-batch gradient descent

Experience replay

- **Uniform Experience Replay**
- **Prioritized Experience Replay** implemented from scratch using a **Sum Tree** data structure to hold transitions and priorities and make updates efficient.

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \qquad p_i = |\delta| + \epsilon \qquad w_i = \left(\frac{1}{N}\frac{1}{P(i)}\right)^\beta$$

- **P(i)** = priority of a sampled transition
- $\mathbf{w_i}$ = **importance-sampling** weights used to correct the bias introduced by the PER sampling

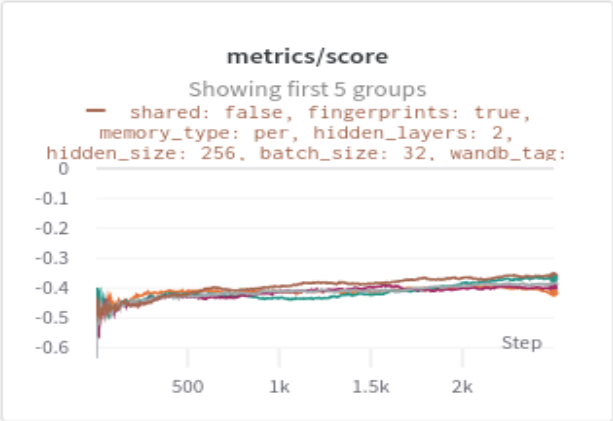
Fingerprints

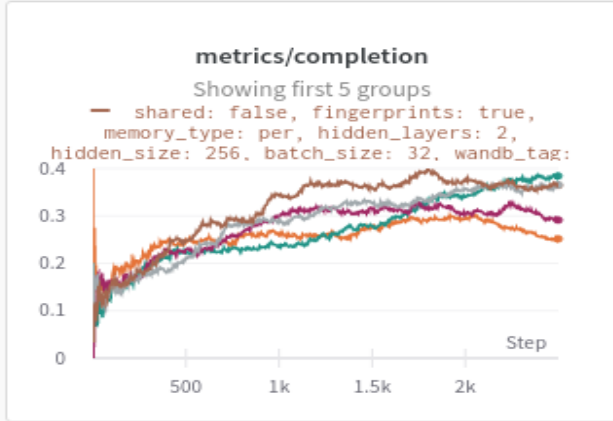
- Different methods, epsilon + step, epsilon + episode etc.. Using the step seems to negatively influence the performance. A good compromise is using epsilon + episode

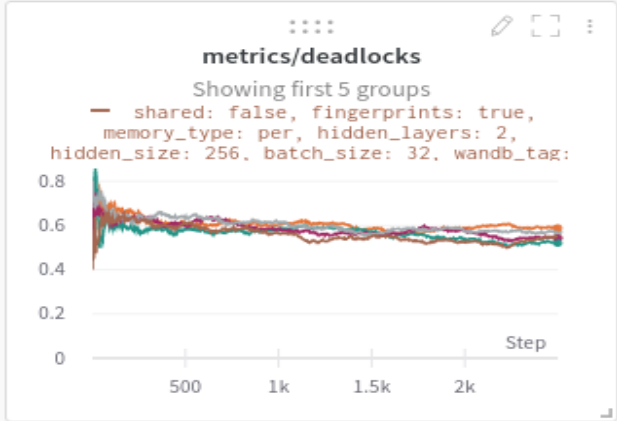
D3QN: RESULTS

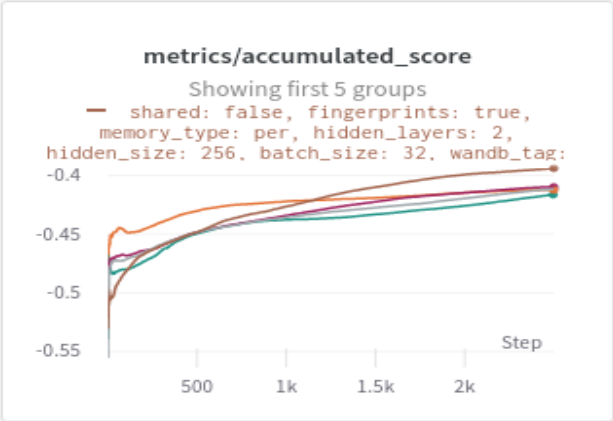
[D3QN sweeps](#)

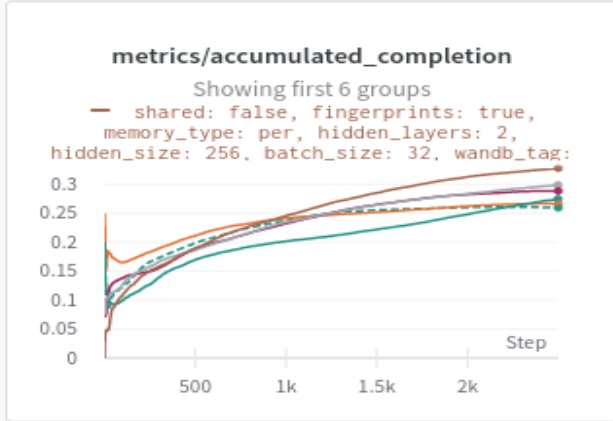
[experiments on Experience Replay, Fingerprints, Network sizes and structure](#)

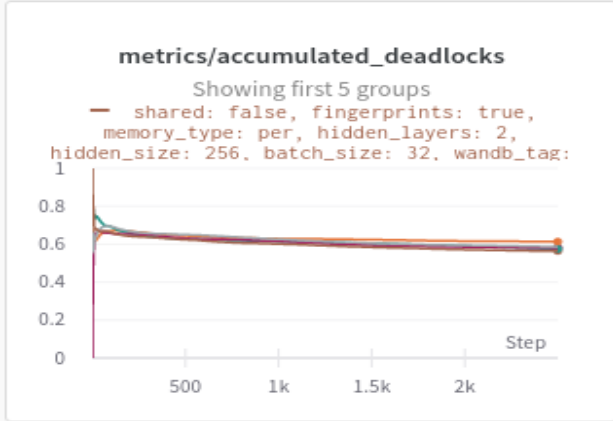












PROS AND CONS

	PS-PPO	D3QN
Sample efficient	No	**Yes**
More focus on discrete environments	No	**Yes**
Convergence speed	**Faster**	Slower
Adaptability to different environments	**Yes**	No
Easy transfer learning and robust training	**Yes**	No

CURRICULUM LEARNING

A **Task** represents an objective the agent should learn.

Based on three questions:

- How to generate tasks?
- How to order tasks?
- How to transfer?

Manual curriculum (Unity ML-agents)
A file specifies how each level is created

Semi-automatic curriculum
Linear and simple functions to model how certain parameters of the environment change during time

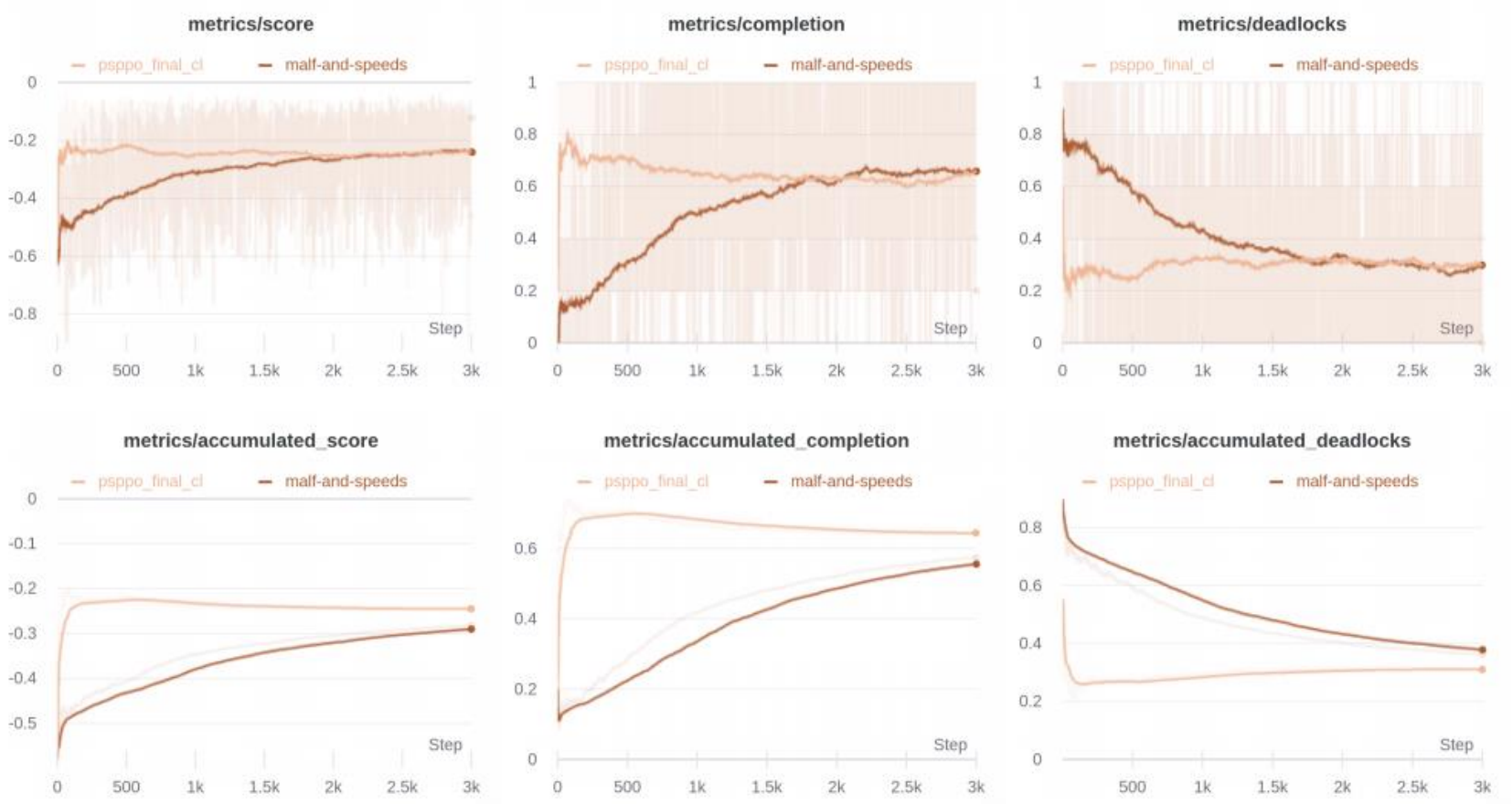
Pros:

- The agent effectively learns better and learning curves of agents pre-trained are remarkably better

Cons:

- Not that easy to tune. It introduces a lot of choices and in complex environments such as Flatland it is not easy to generate environments with increasing and stable difficulty.
- Longer training
- Generalization to unseen environments is not always successful

CURRICULUM LEARNING: RESULTS



curriculum tests

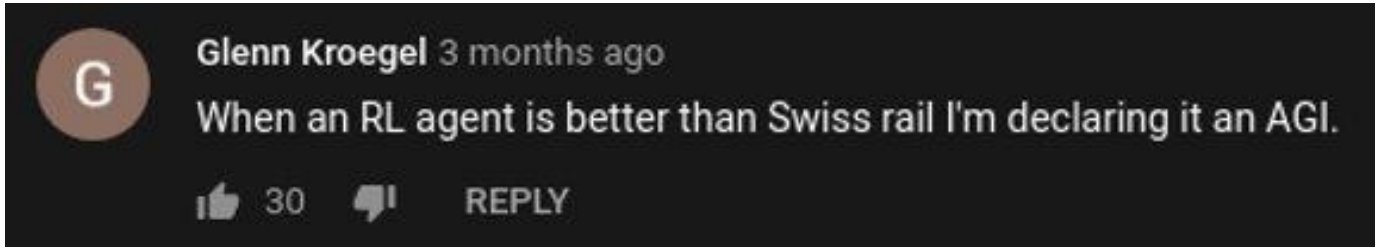
FINAL RESULTS

Metrics	No malfunctions - single speed (1.0)					
	Medium size			Big size		
	3 agents	5 agents	7 agents	5 agents	7 agents	10 agents
accumulated_completion	0.9193	0.8134	0.7263	0.8700	0.7933	0.6721
accumulated_deadlocks	0.0760	0.1770	0.2630	0.1206	0.1951	0.3080
accumulated_score	-0.167	-0.262	-0.341	-0.223	-0.294	-0.398

Metrics	Malfunctions - 4 speeds (0.25)					
	Medium size			Big size		
	3 agents	5 agents	7 agents	5 agents	7 agents	10 agents
accumulated_completion	0.8493	0.7098	0.6321	0.7242	0.6231	0.5071
accumulated_deadlocks	0.1213	0.2616	0.3304	0.2246	0.3299	0.4583
accumulated_score	-0.174	-0.226	-0.257	-0.227	-0.266	-0.308

final wandb project

CONCLUSIONS AND FUTURE WORKS



Other ideas and approaches

- Meta-Reinforcement Learning
- Hierarchical Learning
- Global rewards using images
- Action skipping, Sparse Rewards with curiosity and agents ordering