InstaDeep™

Multi-Agent Reinforcement Learning for Train Scheduling

Alex Laterre, Amine Kerkeni, Joe Phillips, Hui Chen

Nvidia GTC 2020 Tutorial

NVIDIA®

Special Thanks

This tutorial was inspired by our collaboration with **Deutsche-Bahn** around the development of an AI-based decision making and the simulation environment **Flatland** built and open sourced by the **Swiss Federal Railway**

DB

Dr. Irene Sturm, Farhad Safaei, Gereon Vienken

SBB CFF FFS

Erik Nygren, Adrian Egli

Outline

1. Reinforcement Learning: An Introduction

- ❑ Markov Decision Process
- ❑ Policy, Value-Functions
- ❑ Taxonomy of RL

2. Value-Based Method

- ❑ Framing RL as a Regression Problem
- ❑ Deep Q-Networks*

3. Multi-Agent Reinforcement Learning

- ❑ Independent Learners*
- ❑ Centralized Critic Architecture*

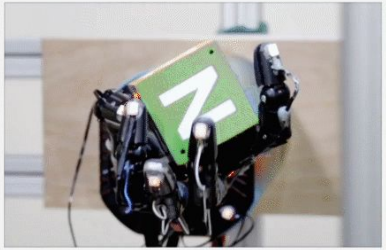
Flatland Multi-Agent Environment*

Reinforcement Learning

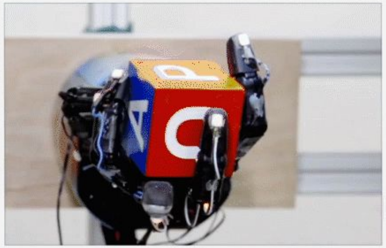
RL is the science of **trial and error**: learn from experience what works (positive reward) and what doesn't (negative reward)

Deep Reinforcement Learning

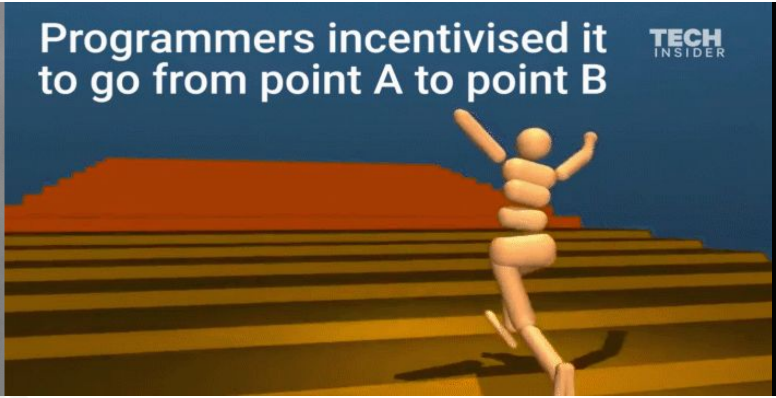
FINGER PIVOTING



SLIDING

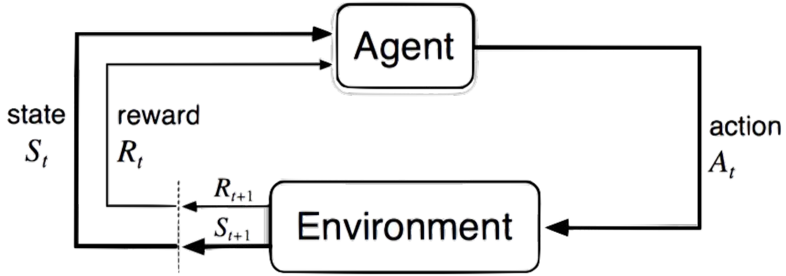


FINGER GAITING



Formalization: Agent-Env / Episode

The agent selects actions and the environment responds to them by presenting new situation and a reward that the agent seeks to maximize over time through the choice of its actions.



The result is a sequence or a trajectory starting like this:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$$

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.

Formalization: Return

The agent's goal is to maximize the expected cumulative reward it receives in the long run. In the simplest case, the return after time step *t* is the discounted sum of the future rewards:

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

The discount rate determines the present value of future rewards:

- A reward received k time steps in the future is worth only γ^{k-1} times what it would be worth if it were received immediately.

Extreme values:

- If = 1: the agent maximizes the sum of the rewards independently of when the rewards are received.
- If = 0: the agent is "myopic" in being concerned with only maximizing the immediate reward.

Policies and Value Functions

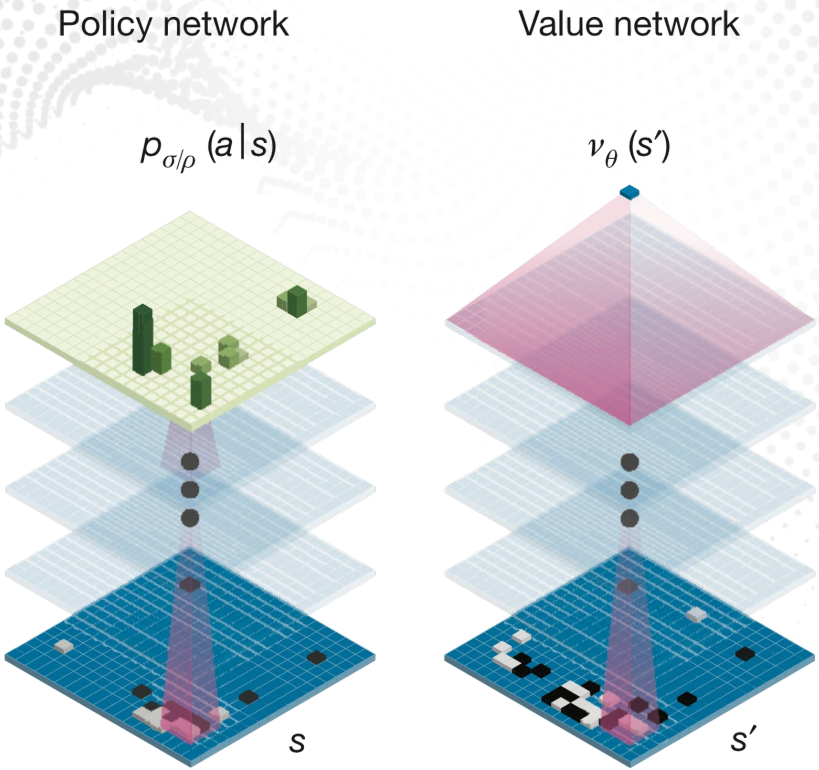
A **policy** is a mapping from states to probabilities of selecting each possible action.

The **value function** of a state under a policy is the expected return when starting in the state and following the policy thereafter.

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right], \text{ for all } s \in \mathcal{S},$$

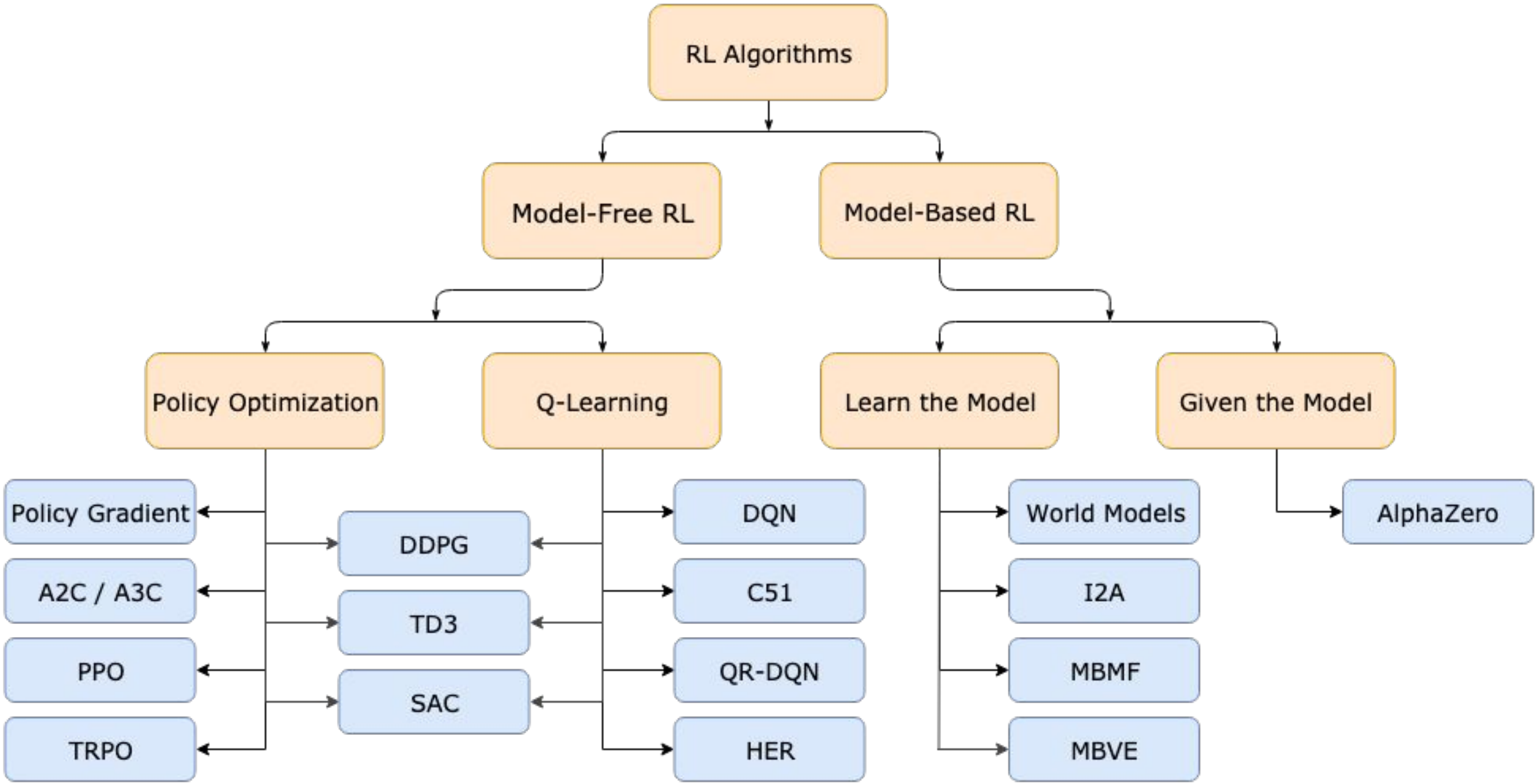
The **action-value function** of a state and action under a policy is the expected return when starting in the state, taking the action and following the policy thereafter.

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right]$$



Source: DeepMind

Taxonomy of RL Algorithms



source : Spinning-Up OpenAI

Value-Based and Policy-Based Learning

$$v(s) \;/\; Q(s, a)$$

Value-Based Learning

The agent optimize the state-value function, that it uses to select the action to take at each step, e.g. the action with the highest value estimate

$$\pi(a|s)$$

Policy-Based Learning

The agent optimizes its policy right away without passing through a value function. The agent takes the action with the highest probability.

Value-Based Learning

The agent interacts with the environment and collects trajectories. The accumulated experience can be used to learn the value function.

Monte Carlo: When an episode ends, the agent looks at the cumulative return it obtained to see how good it did, then can update its value-estimates for the states encountered and make better decisions next time.

$$\underbrace{V(S_t)}_{\text{Maximum expected future reward starting at that state}} \leftarrow \underbrace{V(S_t)}_{\text{Former estimation of maximum expected future reward starting at that state}} + \underbrace{\alpha}_{\text{learning rate}}[\underbrace{G_t}_{\text{Discounted cumulative rewards}} - \underbrace{V(S_t)}]$$

Value-Based Learning

The agent interacts with the environment and collects trajectories. The accumulated experience can be used to learn the value function.

Temporal-Difference Learning: Update the value estimates in part based on other estimates: “*Learning a guess from a guess*”.

$$V(S_t) \leftarrow \underbrace{V(S_t)}_{\text{Previous estimate}} + \alpha\big[\underbrace{\underbrace{R_{t+1}}_{\text{Reward t+1}} + \underbrace{\gamma V(S_{t+1})}_{\text{Discounted value on the next step}}}_{\text{TD Target}} - \underbrace{V(S_t)}_{}\big]$$

Deep Q-Networks

- Represent state-action value function by a deep neural network
- Define the objective function by the **mean-squared error in Q-values**:

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

- Leading to the following **Q-learning Gradient**:

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

- Optimise the objective **end-to-end by SGD** using this gradient definition.

nature Human-level control through deep reinforcement learning, V. *Mnih et al*, *nature* 2015

Stability Issues with Deep Q-Networks

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
- ❑ Scale of rewards and Q-values is unknown
  Naive Q-learning gradients can be large unstable when backpropagated.

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

- ❑ Use experience replay
  Break correlations in data, bring us back to iid setting
  Learn from all past policies
- ❑ Freeze target Q-network
  Avoid oscillations
  Break correlations between Q-network and target
- ❑ Clip rewards or normalize network adaptively to sensible range
  Robust gradients

nature Human-level control through deep reinforcement learning, V. *Mnih et al*, *nature* 2015

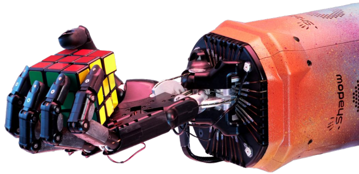
Proximal Policy Optimization

How can we take the **biggest** possible improvement step on a policy using the data we currently have, without stepping so far that we accidentally cause performance collapse?

- ❑ **TRPO**[1] tries to solve this problem with complex second-order methods.
- ❑ **PPO**[2] is a simplification of TRPO that uses first-order methods and <u>a few tricks</u> to keep the policy close to the old one.

<u>Comments:</u>

- Go-To algorithm for Reinforcement Learning
- Behind most of the latest successes in RL (DotaFive, Robotic Manipulation, etc.)
- Requires large mini-batch size and a distributed training approach (GPUs).



<u>References</u>

1. Trust Region Policy Optimization, *J. Schulman, S. Levine, Moritz, M. I. Jordan, P. Abbeel* - ICML 2015
2. Proximal Policy Optimization Algorithms, *J. Schulman, P. Dhariwal, A. Radford, O. Klimov* - 2017

Multi-Agent Reinforcement Learning

Context

- We consider the competitive-cooperative problem, in which a system of several learning agents must jointly optimise either a single reward signal – the team reward – accumulated over time or their individual reward.

What if we train independent agents and consider the other agents as part of the environment?

- The environment dynamic effectively changes as teammates change their behaviour.
  An agent faces with a non-stationary learning problem.

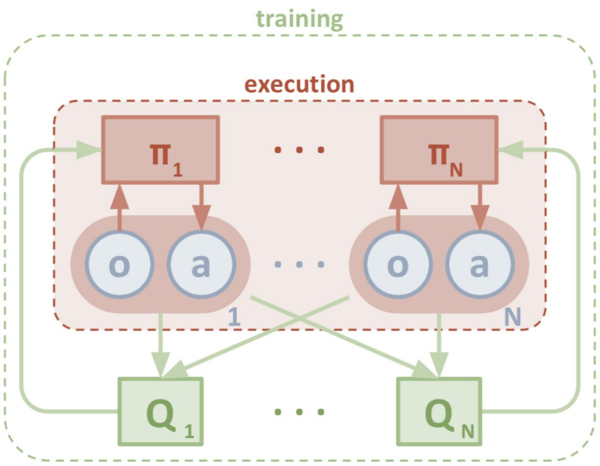
- The agent can't explain its own observed reward as it depends on other agents actions which are hidden in the environment dynamic.

<u>Conclusion</u>: Training the agents independently from each others is often unsuccessful. Let's Try!

Multi-Agent Reinforcement Learning

Paradigm

"*Centralised learning, Decentralised Execution*"



Let's consider an Actor-Critic Architecture

- ❑ Actor: Defines the agent's behaviour, its policy
- ❑ Critic: Estimates the State-Value Function, Q(a,s) or V(s)

What does it solve?

- ❑ Conditioned on the other agents' action, the environment becomes stationary from the agent's perspective.
- ❑ Facilitate the Credit Assignment

Reference:

Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments, *R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, I. Mordatch* - NeurIPS 2017

Flatland Multi-Agent Environment

A high-performance simulator which simulates the dynamics of train traffic as well as the railway infrastructure.

Objective:

❑ Make all agents (trains) arrive at their target destination with a minimal travel time.

❑ We want to minimize the time steps (or wait time) that it takes for each agent in the group to reach its destination.

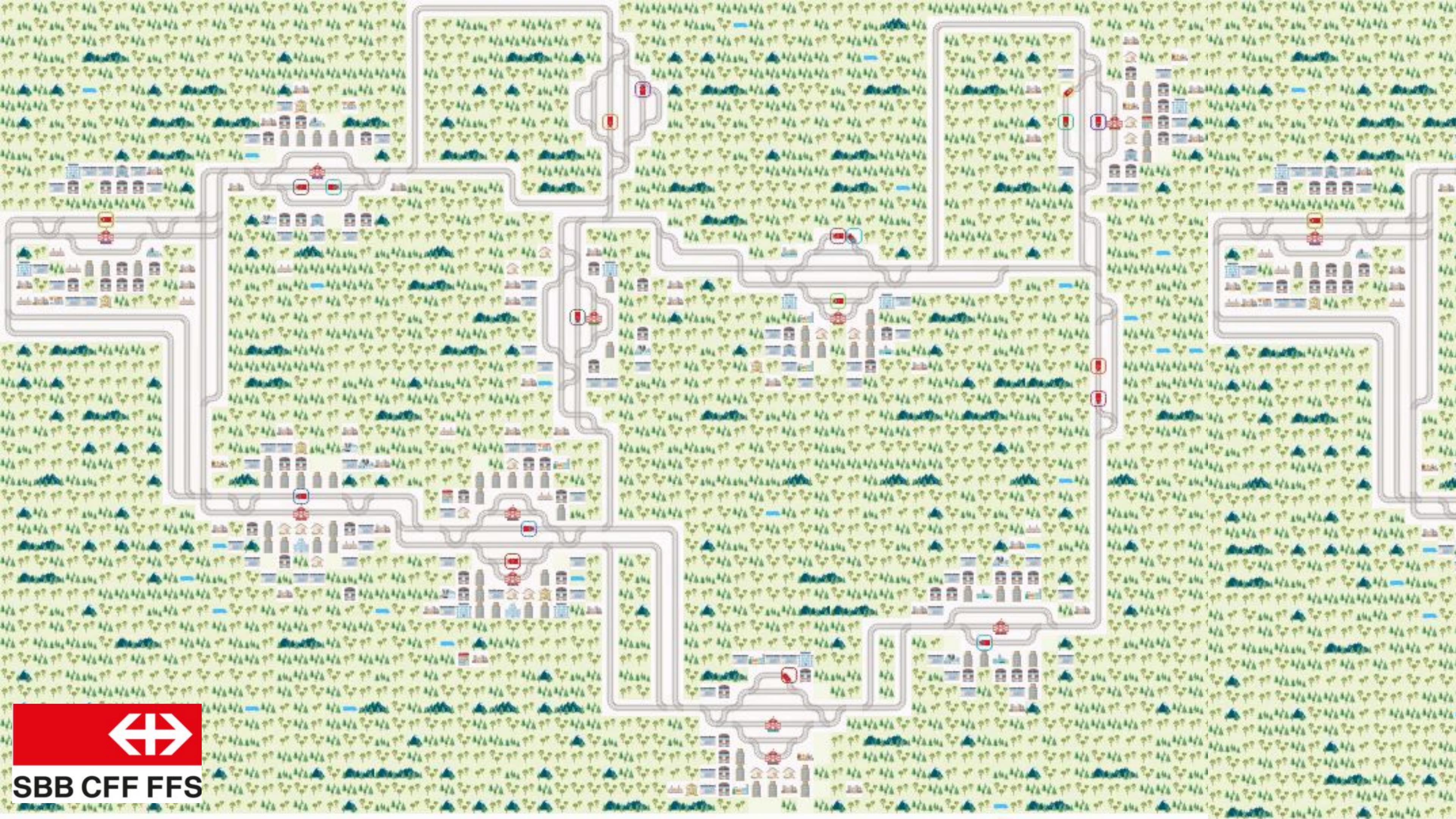
Boardgame:

❑ A 2D grid environment with restricted transitions between neighboring cells to represent railway networks.

Let's Play!! 😃

SBB CFF FFS

InstaDeep™



SBB CFF FFS

Get in Touch

facebook.com/InstaDeepAI

twitter.com/instadeepai

linkedin.com/company/instadeep