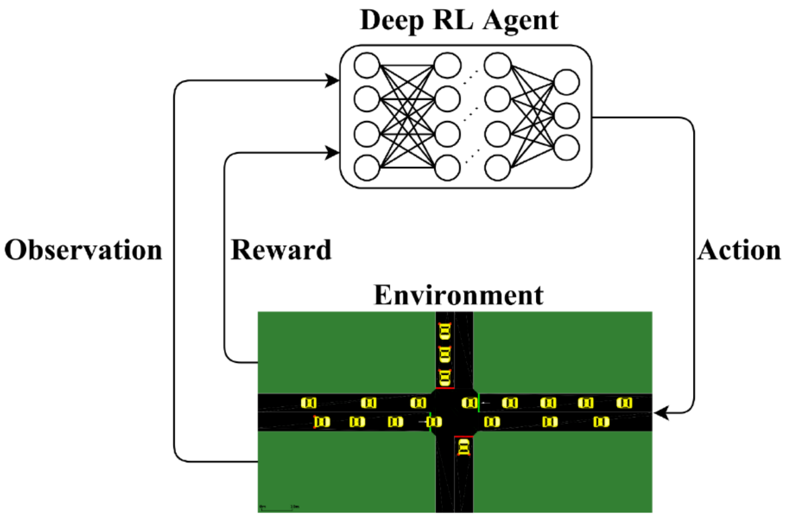
A Deep Reinforcement Learning Approach to Adaptive Traffic Lights Management

Andrea Vidali, Luca Crociani, Giuseppe Vizzari, and Stefania Bandini

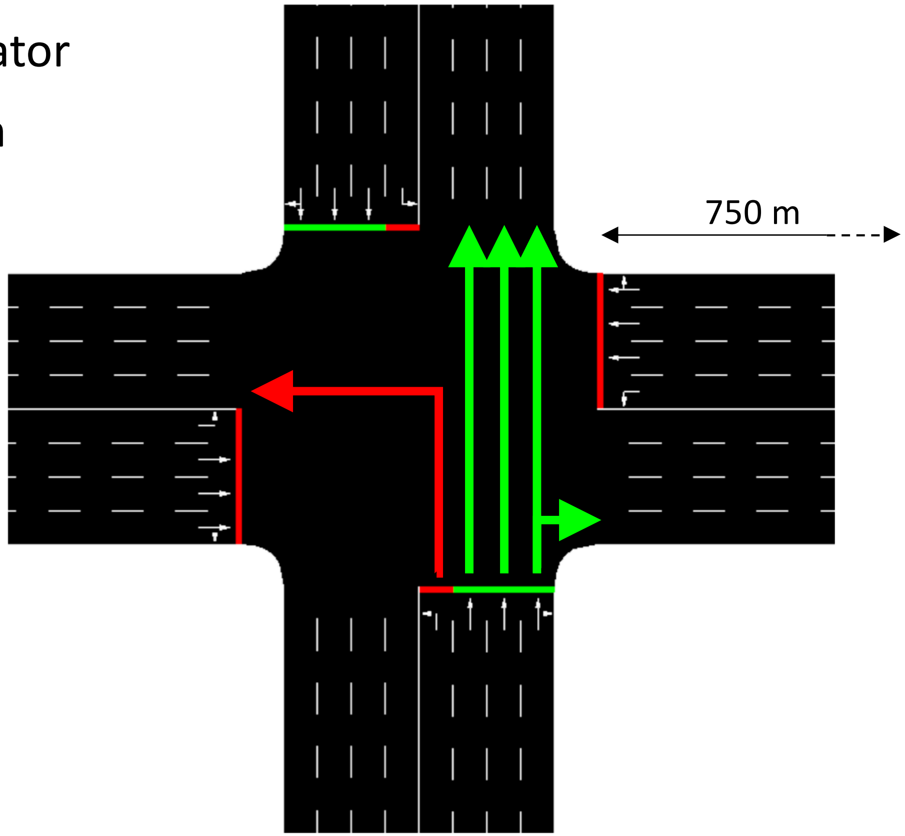
Introduction



- **Context:** traffic lights management in a single four-way intersection
- **Goal:** design, experiment and evaluate a deep reinforcement learning agent for this task employing a plausible experimental setting
- **Reinforcement learning:** machine learning area dealing with studying how **agents** choose **actions** in an **environment** to maximise the cumulative **reward**, that supposedly leads to achieving a given objective

Environment

- **A four-way intersection**
- **Implementation:** SUMO microscopic traffic simulator
  - Reproduces realistically the traffic dynamics in the intersection
  - Can be accessed and controlled via a well-defined API
  - Simulation step: 1 second (not necessarily the same timestep of TL agent decision!)
- **TL agent** manages the traffic lights, whereas SUMO agents manage individual vehicles
- **TL agent goals:** choose the most appropriate semaphore phase (1 among a fixed set of allowed configurations), in order to maximise the efficiency of the intersection

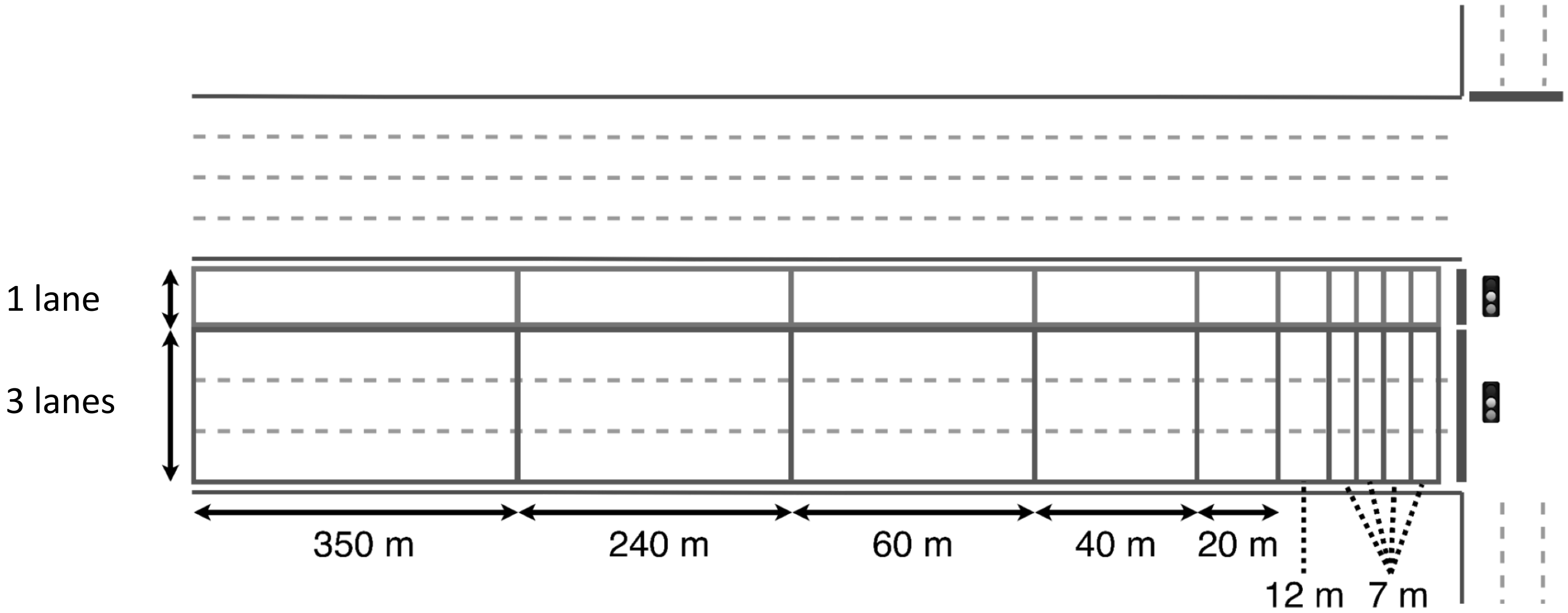


State of the environment

- Discretization of the **environment**
- Modelling choices are plausible considering actual implementation limits…
  - … some papers in the literature use **SUMO UI as an input** to the traffic light agent!
  - Our parsimony could even be excessive

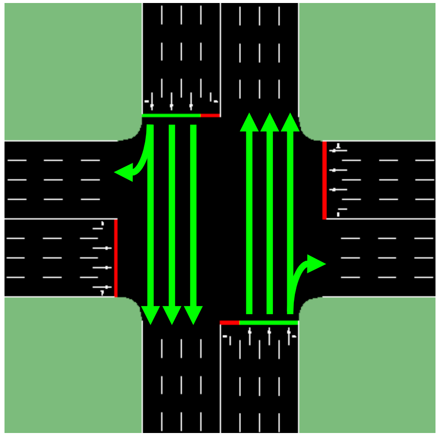
Vehicle presence cell – Total 80 cells

- **1** - at least one vehicle is present
- **0** - otherwise

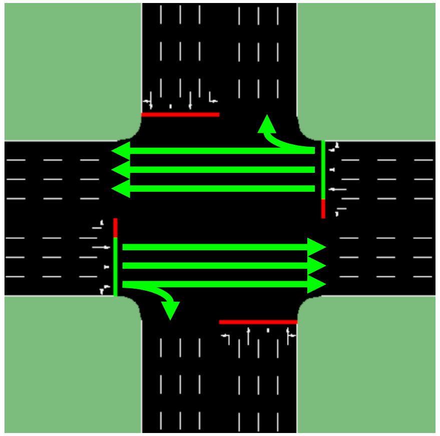


Actions

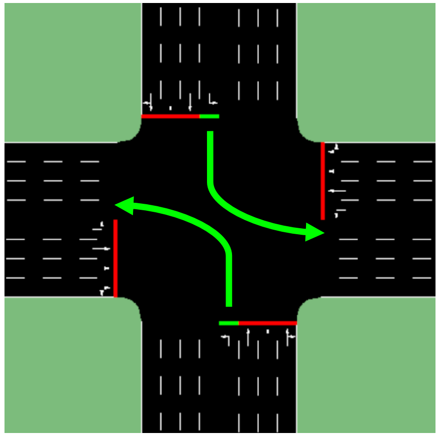
North-South



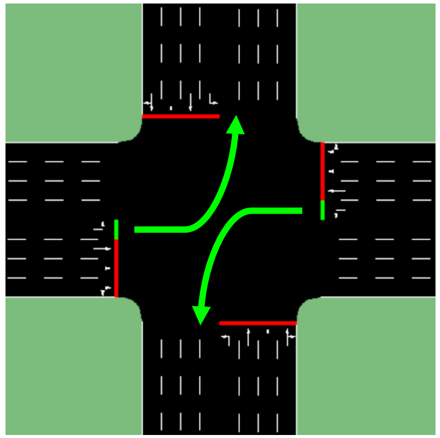
East-West



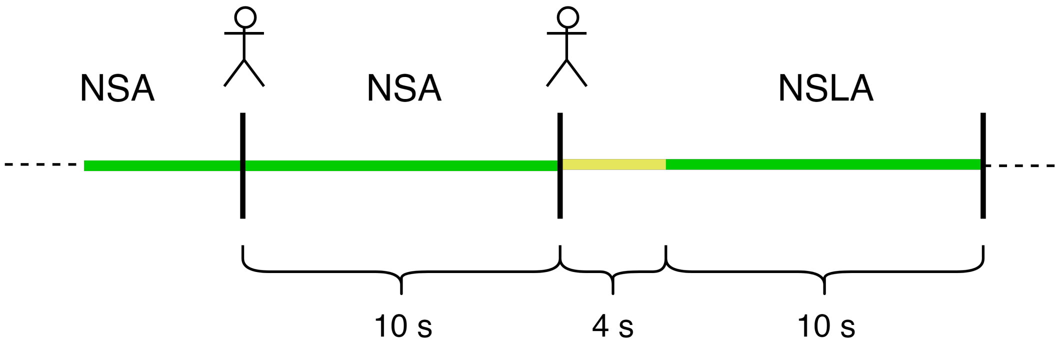
North-South left turn



East-West left turn



- Green light: 10 seconds
- Yellow light: 4 seconds



Reward

Used metric: *total waiting time*

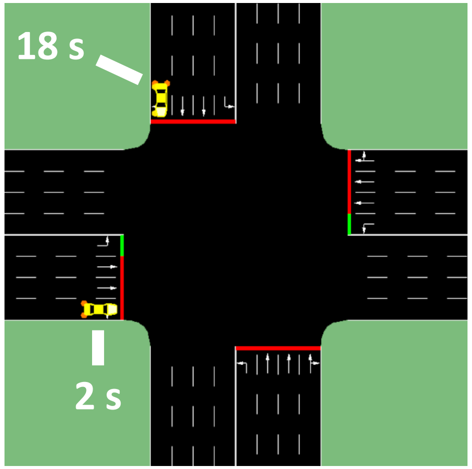
$$twt_t = \sum_{v'} wt_{(v',t)}$$

v': Vehicles that are waiting

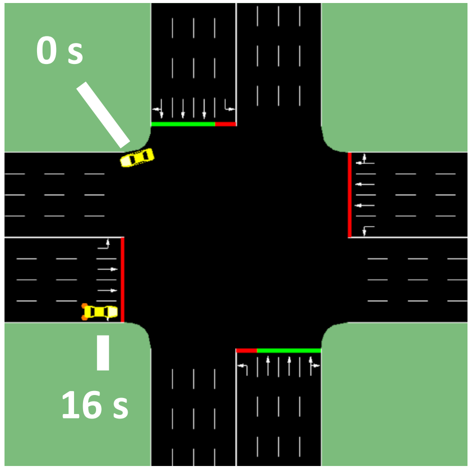
Baseline (literature) reward function:

$$\boldsymbol{r_t = 0.9 * twt_{t-1} - twt_t}$$

Timestep: $t-1$



Timestep: t



twt_{t-1} = 18 + 2 = **20** s

twt_t = **16** s

r_t = 0.9 $*$ 20 - 16 = **2**

Issue detected within the experimentation phase:

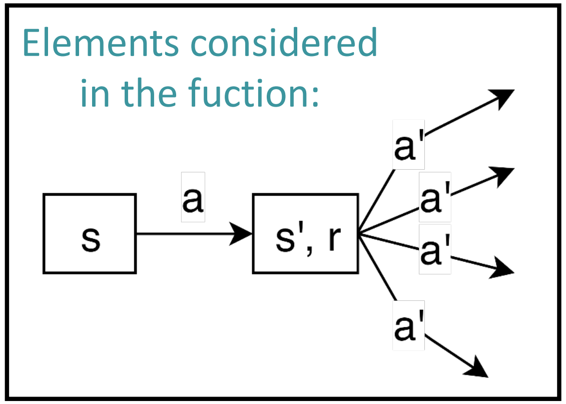
- Total waiting time for vehicles is provided by SUMO via its API;
- SUMO's interpretation is to compute it since the last stop of the vehicle… but if the queue is long, the vehicle will stop even several times waiting to cross the intersection
- We introduced an additional metric (*accumulated total waiting time - atwt*), considering the time spent by a vehicle within a scenario moving with a velocity lower than a given threshold (for the present work 0.1 m/s)

Alternative reward function: $\boldsymbol{r_t = atwt_{t-1} - atwt_t}$

Q-Learning

- **Q-value =** value of an action at a given time
- **Action choice criterion:** every timestep, **choose the action $\boldsymbol{a}$ maximizing $\boldsymbol{Q(s,a)}$**

$$\underbrace{Q(s,a)}_{\substack{\text{Expected value of the}\\ \text{execution of action } a \text{ in}\\ \text{state } s}} = \underbrace{r}_{\substack{\text{Immediate}\\ \text{reward}}} + \underbrace{\varphi}_{\substack{\text{Future reward}\\ \text{discount factor}\\ [0:1]}} \underbrace{max_{a'}\, Q'(s',a')}_{\substack{\text{Maximum value of}\\ \text{actions in the next}\\ \text{state}}}$$

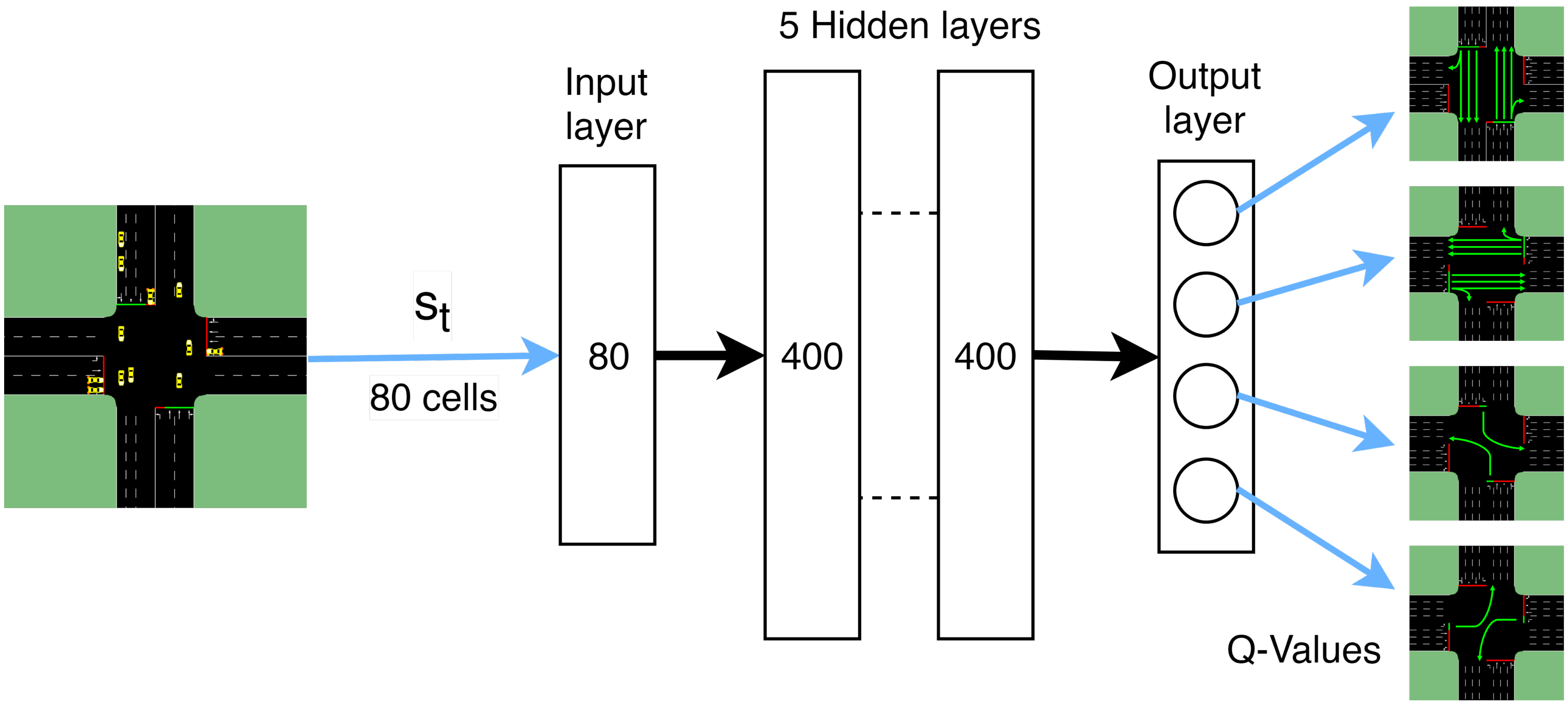


Maximise $Q(s,a)$ → follow the best line of action that was learned so far

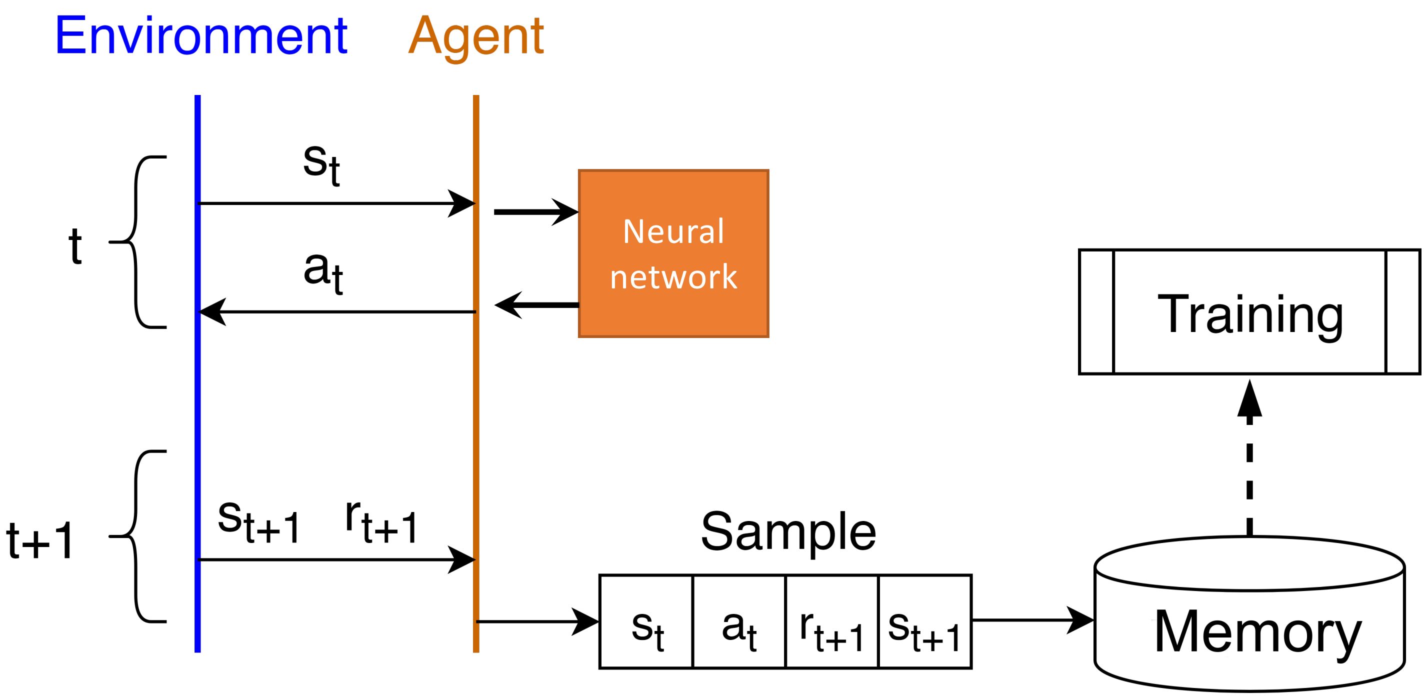
Action selection policy actually based on ε-greedy exploration policy (gradually switch from ***exclusively exploring*** the effects of actions to ***exclusively exploiting*** the acquired information)

Deep neural network

- The state space is very large ➔ **Deep neural network (fully connected)**
- **Goal:** approximate $\boldsymbol{Q(s, a)}$

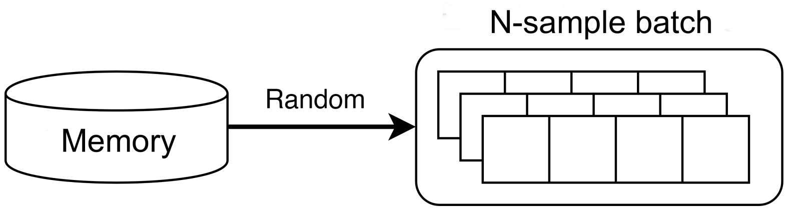


Information acquisition for training



- **Problem**: environment states highly correlated among them, training with sequential information (with this network architecture) is not effective
- **Solution:** train using acquired experience (experience replay), not immediately acquired episodes. A memorization mechanism is required

Actual training phase



- Memory capacity: 50000 samples
- Oldest sample removed to accommodate the new one
- Training instance: **random sampling the memory**
  - Takes place every step
  - Batch size: 100 samples

Training: for each sample, expected Q-values are updated using the information present in the sample

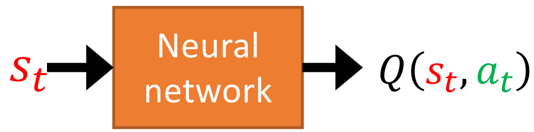
Sample

s_t	a_t	s_{t+1}	r_{t+1}

Q-values update

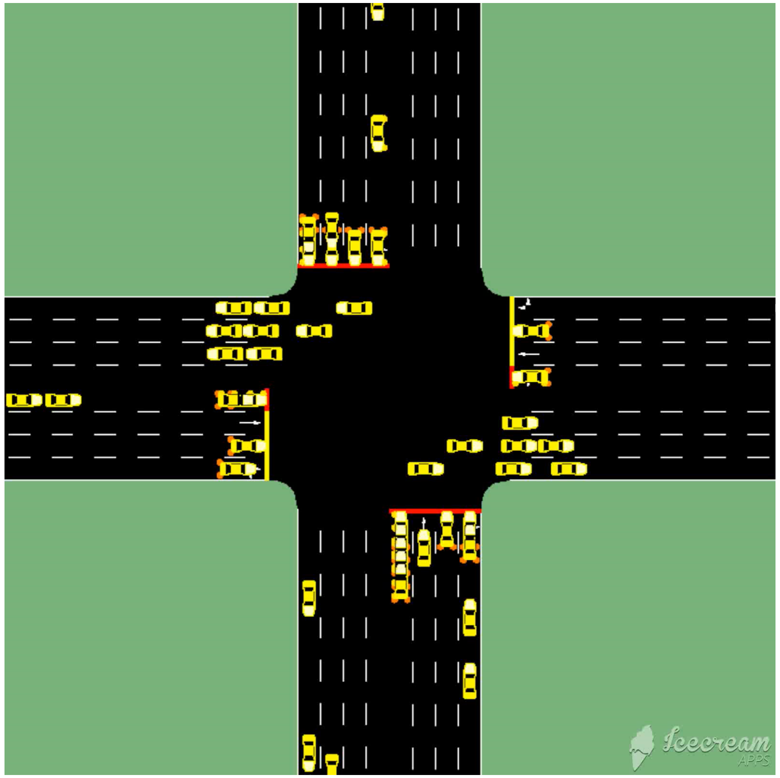
$$Q(s_t, a_t) = r_{t+1} + \gamma * max(\, Q'(s_{t+1}, a_{t+1}\,))$$

Neural network training

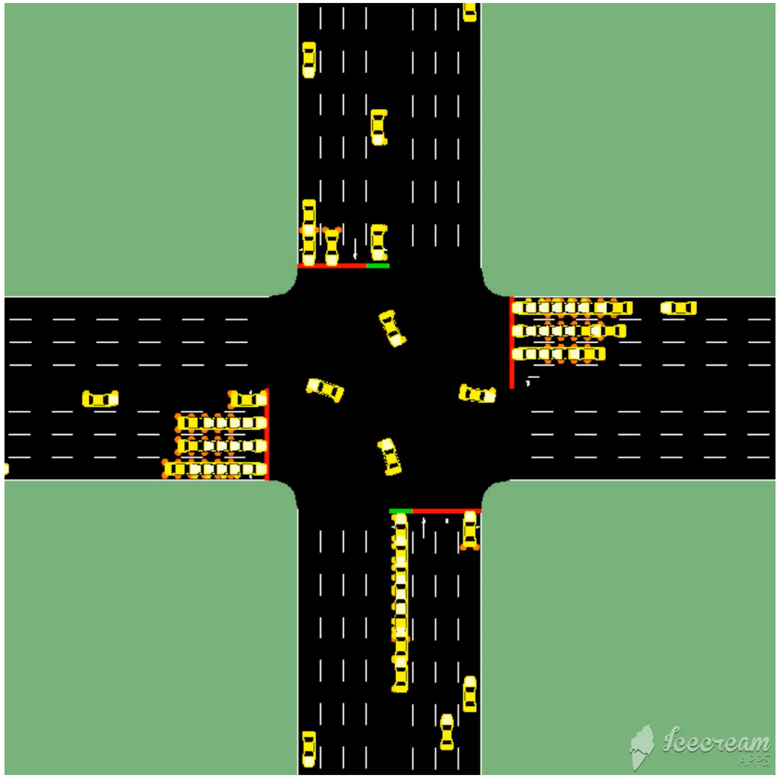


Qualitative results

RL Agent



Static Traffic Light

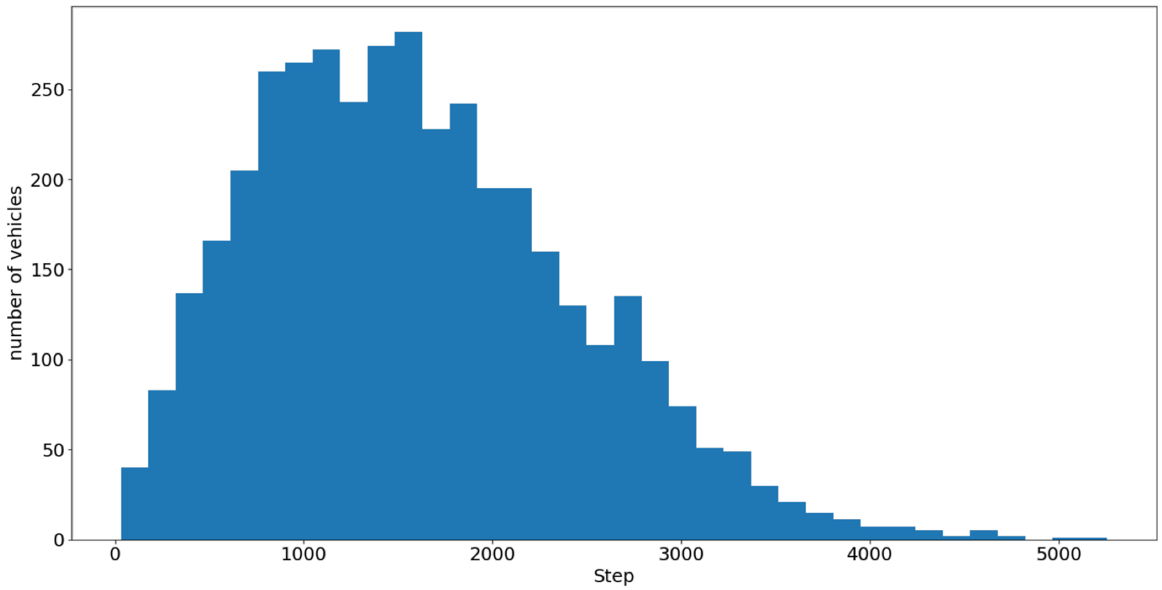


Quantitative results: simulation setup

- **Episode** = 1 h 30 min
- **Total episodes** = 1600
  - Overall time equivalent= 100 days
  - Training duration about 8 hours
    - Can be improved significantly…

4 Traffic scenarios considered

- **High** Traffic – **4000** vehicles
- **Low** Traffic – **600** vehicles
- **North-South** Traffic – **2000** vehicles
- **East-West** Traffic – **2000** vehicles



- Cyclic switching of scenarios
- Vehicle origin and destination randomly chosen
- Timing of generation of vehicles within an episode according to Weibull distribution

Quantitative results: performance evaluation

Static traffic light (STL)

Phase	Duration (s)
North-South	30
North-South left turn	15
East-West	30
East-West left turn	15
Yellow	4

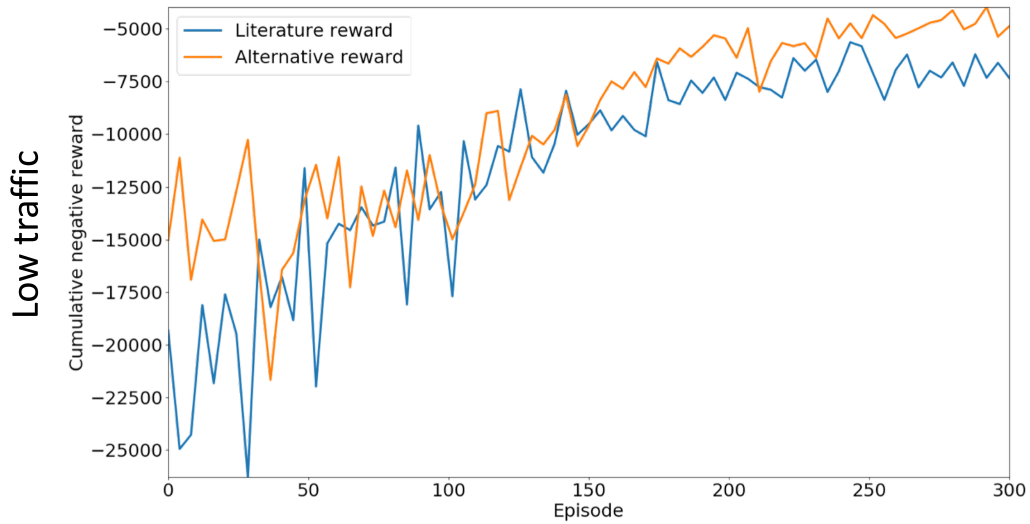
Evaluation metrics

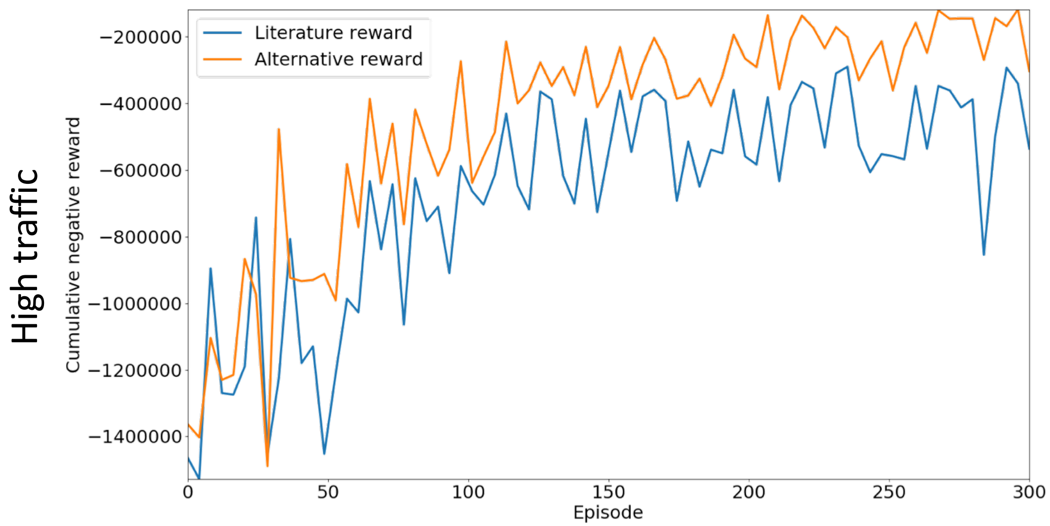
5 episodes for evaluation

- Overall results averaged out from more evaluation runs

- twt - **Total wait time**
  - Sum of all *waiting times* for all vehicles in a given episode
- awt/v - **Average wait time / vehicle**

Quantitative results





	Literature reward agent	Alternative reward agent
	Low-traffic scenario	
cwt	-30	-47
awt/v	-29	-45
	High-traffic scenario	
cwt	+145	+26
awt/v	+136	+25
	NS-traffic scenario	
cwt	-50	-62
awt/v	-47	-56
	EW-traffic scenario	
cwt	-65	-65
awt/v	-59	-58

- The RL agent is able to opportunistically choose appropriate actions in low to medium demand situations
- In high traffic, (and especially long) fixed cycles actually outperform the RL agent
- The choice of a proper reward function has dramatic implications

Conclusions

RL agent is able to outperform the baseline static traffic light

The choice of a proper reward function has potentially impressive implications on the achieved results

- Modelling experience and knowledge is beneficial or even necessary even when employing ML techniques
- … in particular it avoids making unreasonable assumptions on the environment state representation

This work is a good starting point for further explorations…

- To improve the RL approach (improve the neural network, improve the state representation – we’ve been pretty conservative, additional information would improve results significantly, explore alternative reward functions…)
- To extend the studied context (towards a MAS, multiple intersections…)
- To experiment the approach in a real-world scenario (still in silico, first)
- To study the co-evolution of an overall system in which both the traffic lights and the vehicles can adapt to perceived changes!

Thanks for your attention!

Giuseppe Vizzari
University of Milano-Bicocca, Milano, Italy
giuseppe.vizzari@unimib.it