Intersection control

PRESENTED BY
MUKUL CHODHARY & KEVIN OCTAVIAN

EE488F 2023 | KAIST

Overview

BACKGROUND

INTRODUCTION

MDP

AGENT

EQ. CLASSES

MEMORY

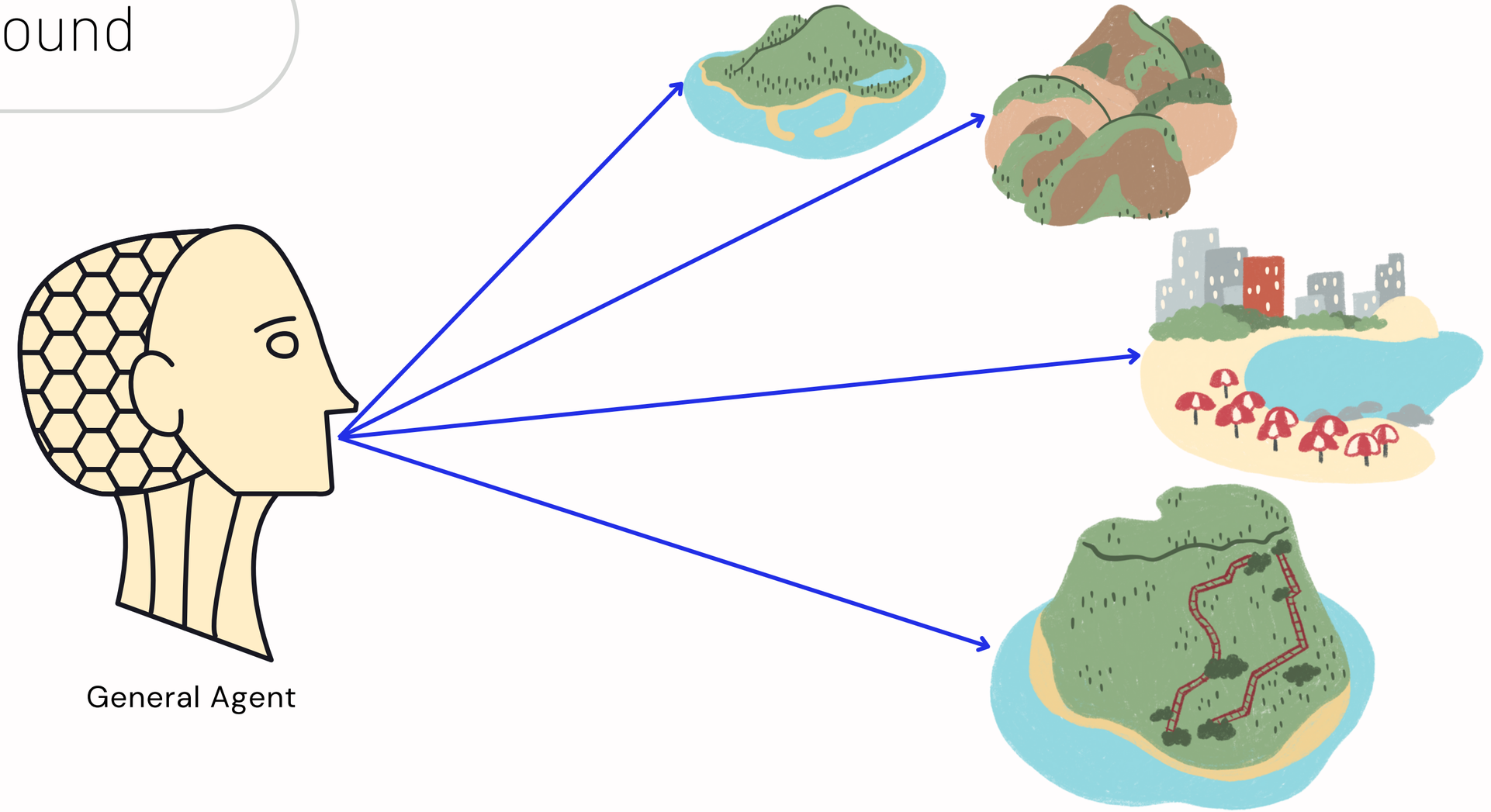
SINGLE AGENT RESULTS

MULTI-AGENT FRAMEWORK

RESULTS

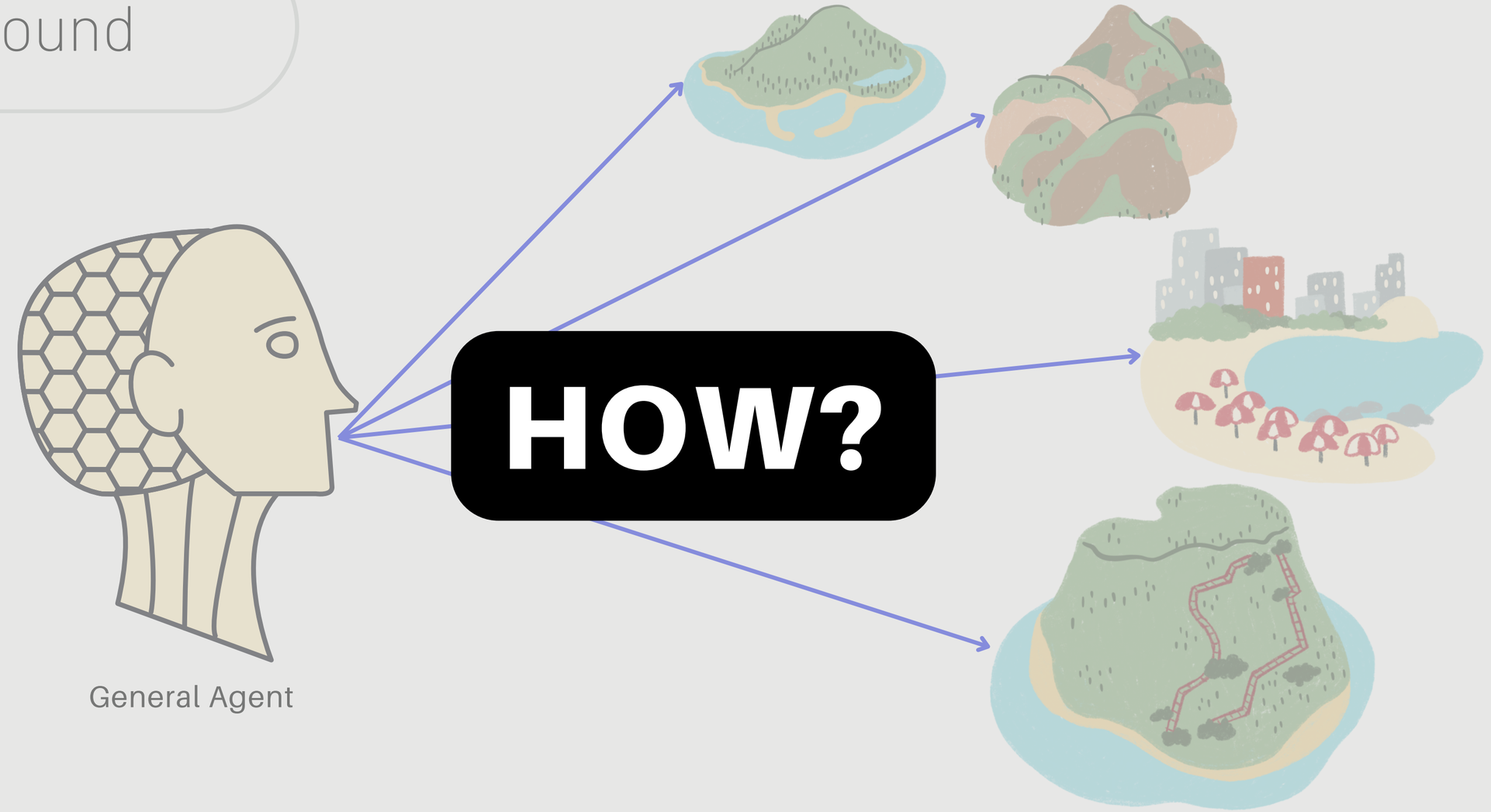
CONCLUSION

Background



General Agent

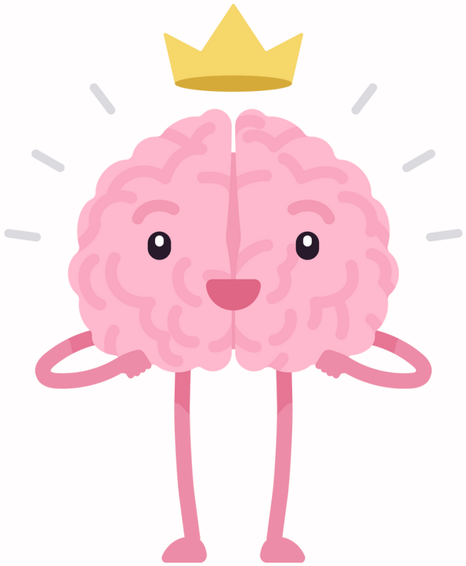
Background



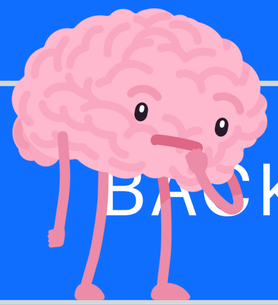
Background

Introduce:

- memory systems to take better advantage of past experiences
- some way to do state abstraction

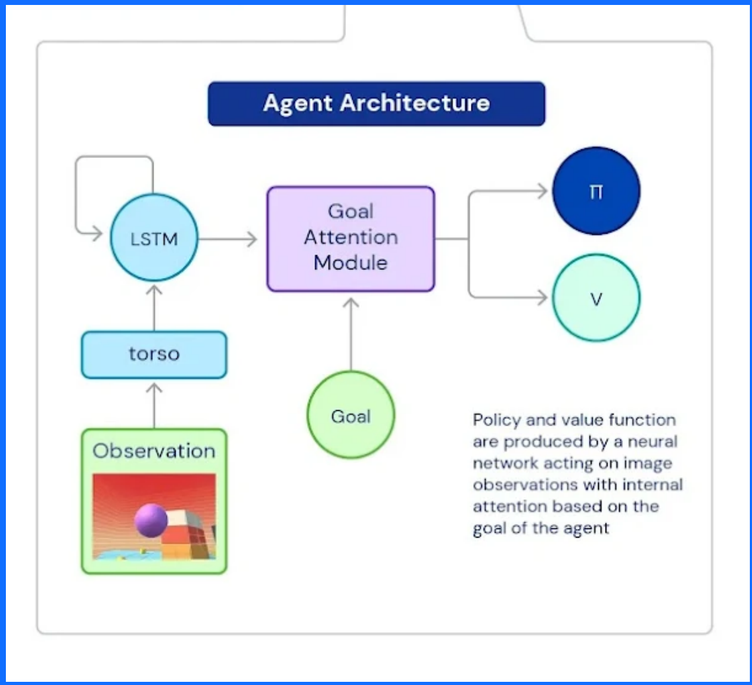


Enviroment



BACKGROUND

Google DeepMind



2017

Neural Episodic Control

Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adrià Puigdomènech Badia, Oriol Vinyals, Demis Hassabis, Daan Wierstra, & Charles Blundell (2017). Neural Episodic Control. CoRR, abs/1703.01988.

2020

Agent57: a general RL agent with episodic control for all Atari games

Badia, A.P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, Z.D. & Blundell, C.. (2020). Agent57: Outperforming the Atari Human Benchmark. Proceedings of the 37th International Conference on Machine Learning, in Proceedings of Machine Learning Research 119:507-517 Available from https://proceedings.mlr.press/v119/badia20a.html.

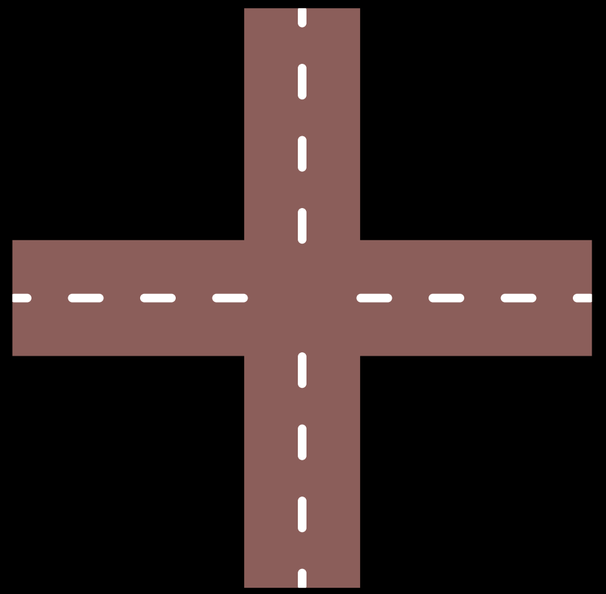
2021

Open-Ended Learning Leads to Generally Capable Agents

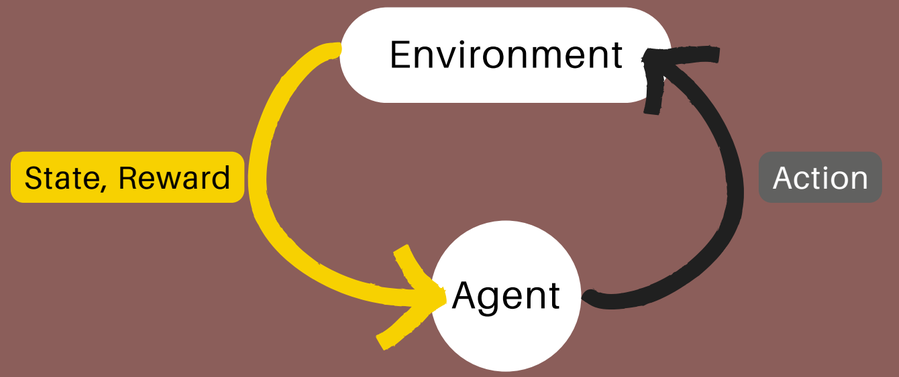
Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michaël Mathieu, Nat McAleese, Nathalie Bradley-Schmieg, Nathaniel Wong, Nicolas Porcel, Roberta Raileanu, Steph Hughes-Fitt, Valentin Dalibard, & Wojciech Marian Czarnecki (2021). Open-Ended Learning Leads to Generally Capable Agents. CoRR, abs/2107.12808.

Introduction

A multi-agent approach to minimise traffic congestion problem (with episodic memory)



MDP



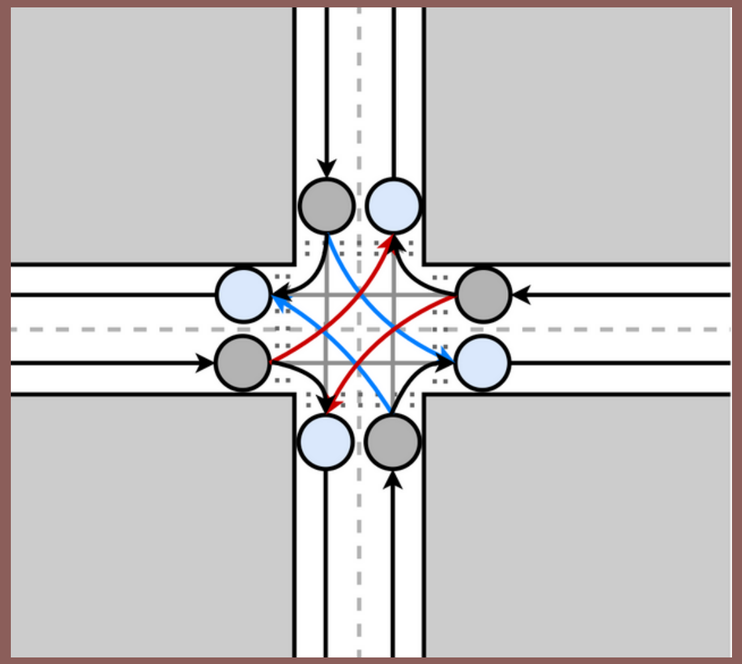
We represent each intersection as a simple Markov Decision Process (MDP), defined as the tuple $\mathscr{M} \triangleq (\mathscr{X}, \mathscr{A}, P, R)$

- $\mathscr{X}$ - **States:**
  - $[x_t(D/i)]$ for all Directions, {E,N,W,S}, and Lanes, {L,F,R}

- $\mathscr{A}$ - **Actions:**
  - be the set of actions taken by traffic lights, i.e. the modes.
  - $\mathscr{A} \triangleq$ {straight line motion} ∪{adjacent motion}
  - $\mathscr{A} \triangleq$ {(E/W, F), (E/W,L), (N/S, F), (N/S, L), (E, F/L), (N, F/L), (W, F/L), (S, F/L)}

- **P - Transition Probability:**
  - arrivals are modelled as a compound poison process
  - the rate of vehicles departing is set to 1.

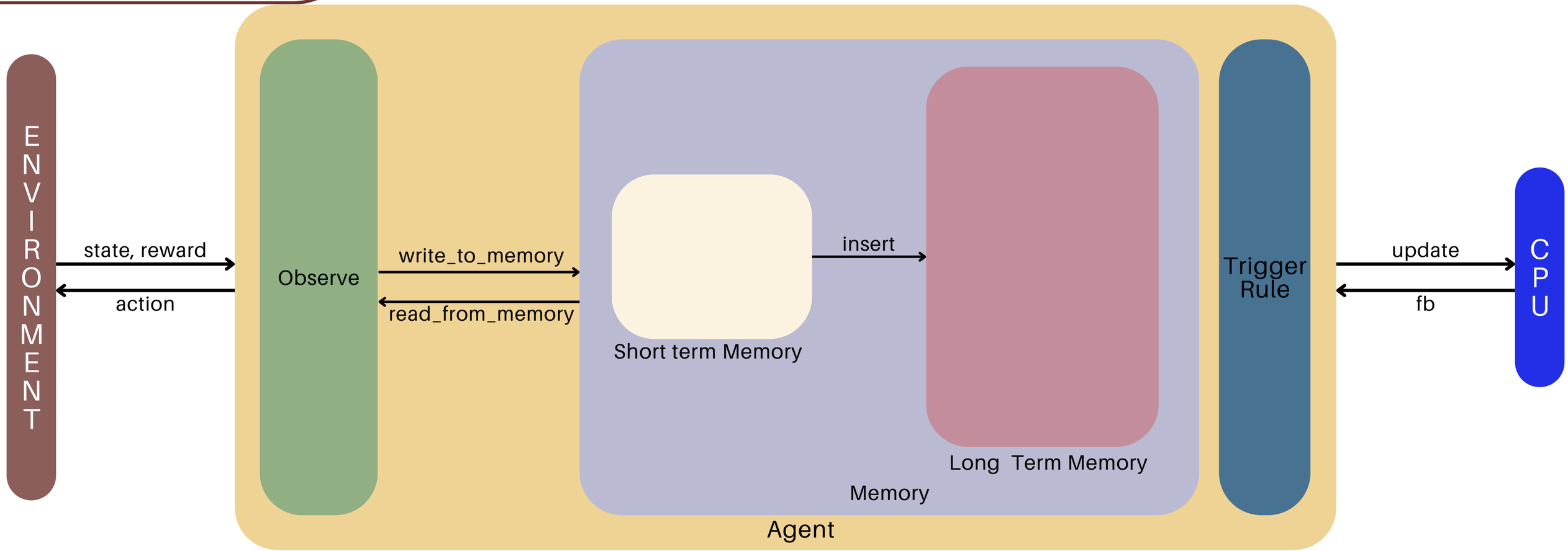
- **R - Rewards:**
  - the reward obtained by transitioning from state x_t to x_{t+1}.
  - Default rewards are proportional to the number of vehicles leaving due to an action. And proportionally negative when the increase in vehicles is too high.



Assumptions

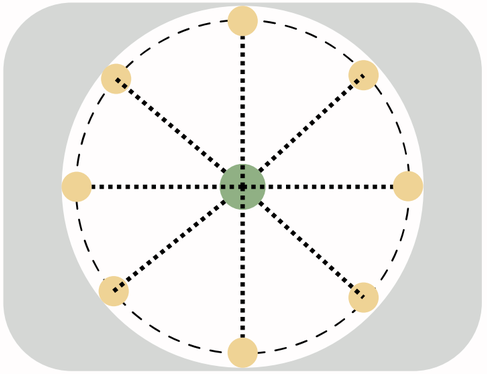
- Discrete number of vehicles leaving an intersection
- Vehicles follow all traffic rules and are allowed to turn right at any time if it’s safe to do so the agent’s action does not directly impact the right-turning vehicles.
- The capacity of a lane is never reached.
- Vehicles are only allowed to enter the grid from the edges, that is there are no “buildings” where vehicles spawn from.
- The vehicles move at an average speed between two nodes and there are no disruptions or random delays, but these can be modelled into the environment if needed.
- There is no time delay to cross an intersection.
- No communication delays.

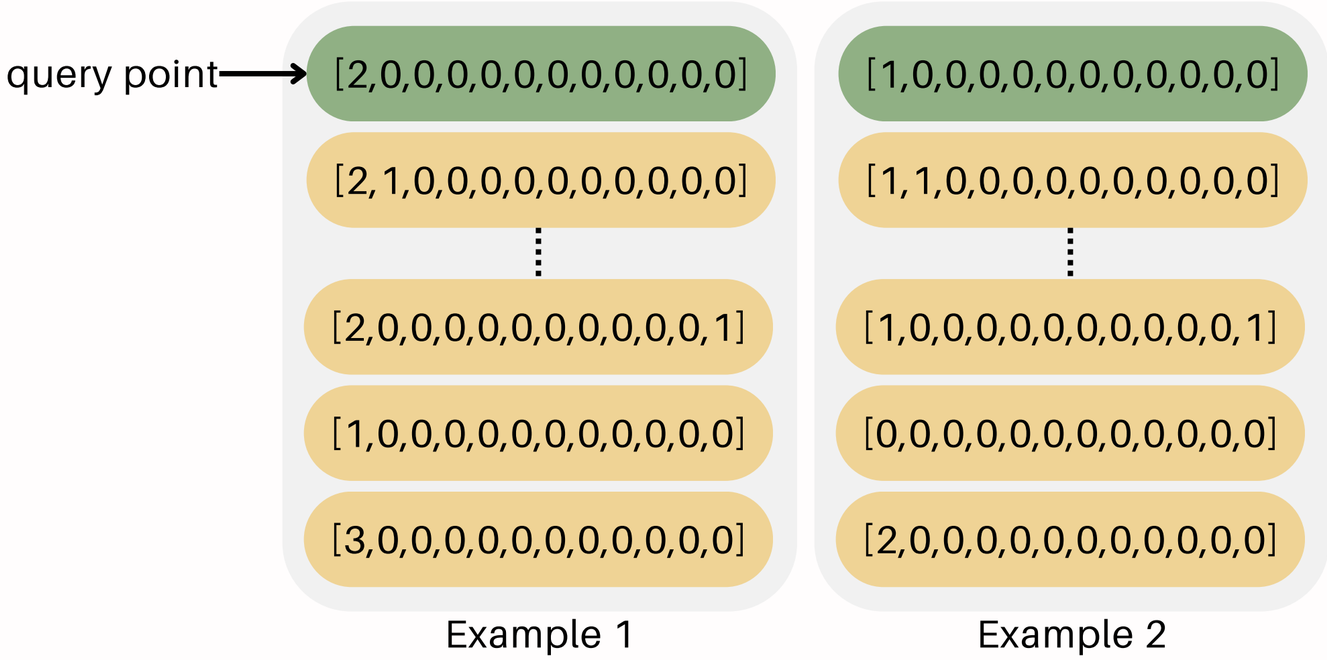
Agent Arch.


ENVIRONMENT
state, reward
action
Observe
write_to_memory
read_from_memory
Short term Memory
insert
Long Term Memory
Memory
Agent
Trigger Rule
update
fb
CPU

Eq. States

We consider all states in 1 unit from the query state to be equivalent.

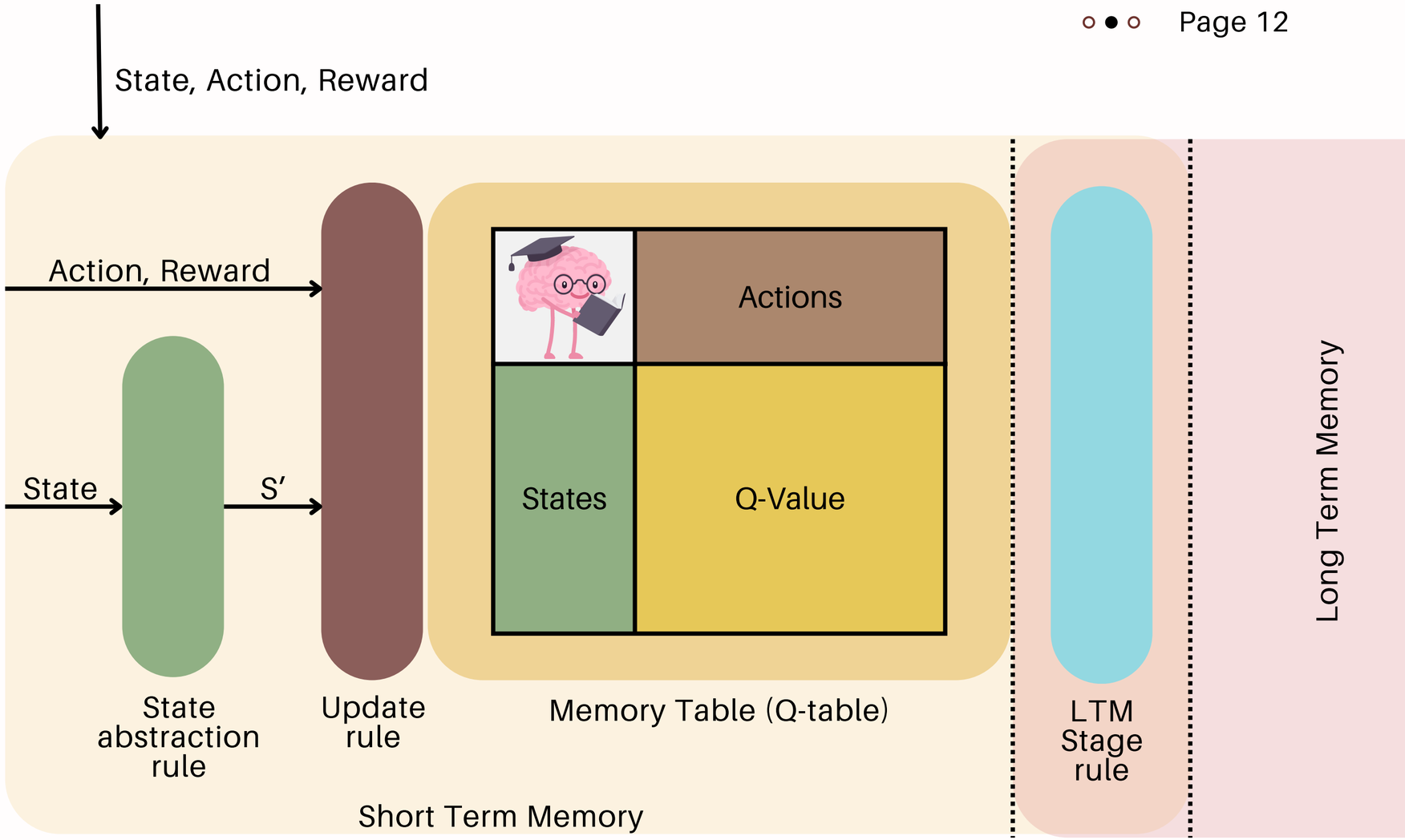




Memory

We treat short-term memory similarly to the Q-table for our SARSA agent.

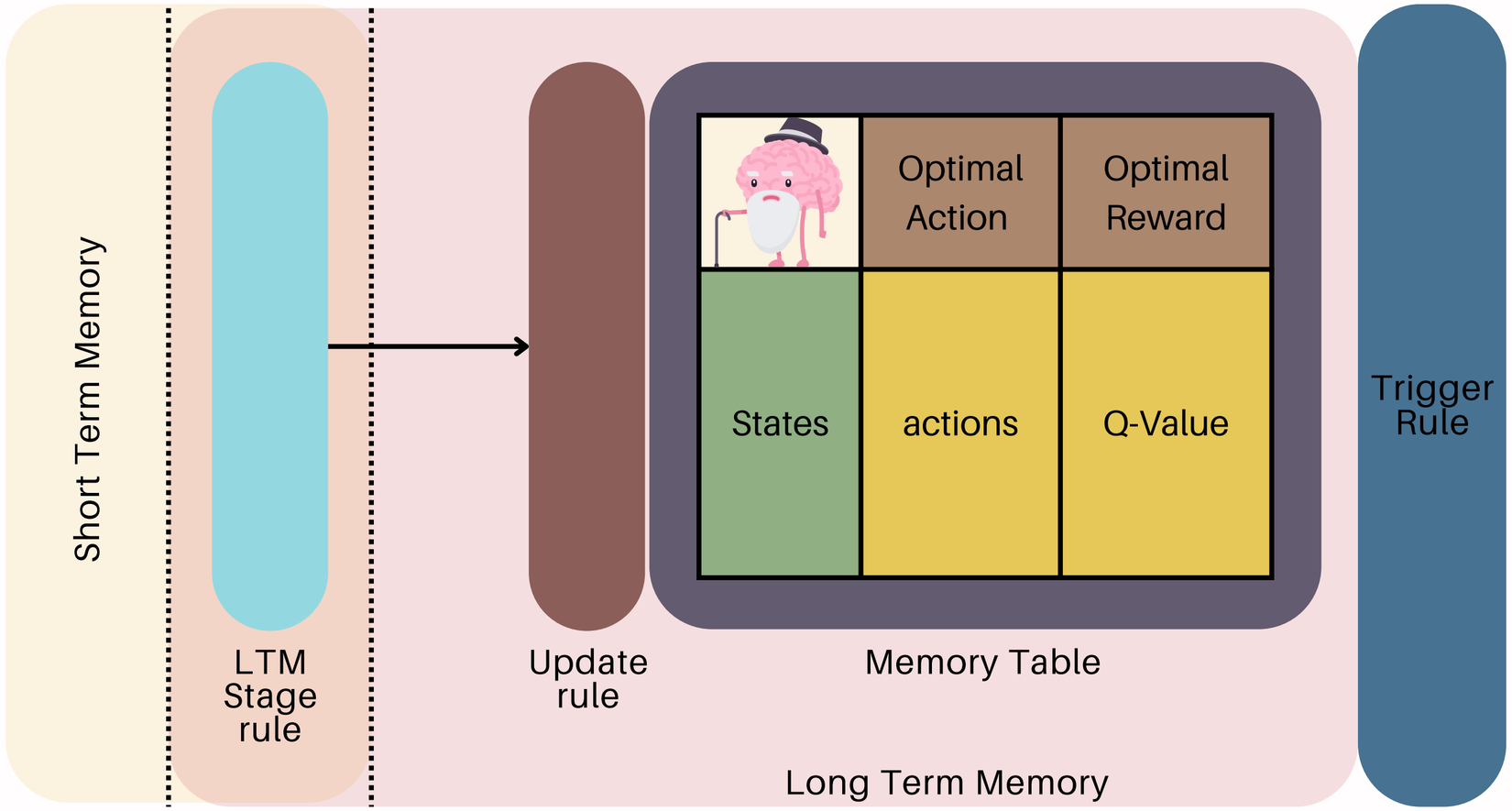
- State abstraction rule:
  - Get an equivalence class that exists in the memory.
  - If it doesn't, S'=S
- Update rule:
  - Append the abstracted state S' or add to S' if it exists to the memory table in a circular fashion overriding the oldest insert.
  - Calculate the Q-value from reward, current state, and next state.
- Long Term Memory(LTM) stage rule
  - When the memory table is full, stage the m/2 states with the highest Q values to be added to Long Term Memory. The m is the size of the short-term memory table.
  - Empty the short-term memory



Memory

Long Term Memory stores the optimal action and reward for the abstracted state, s'.

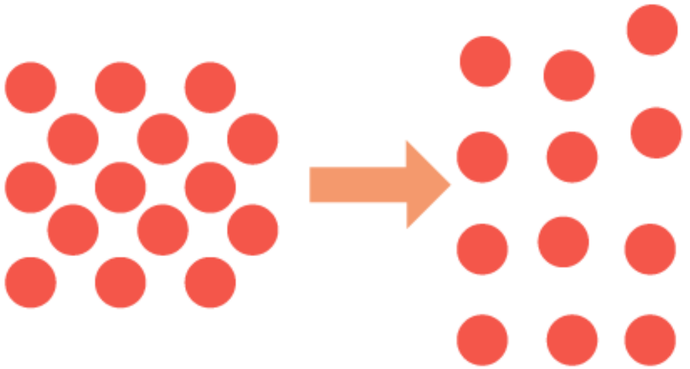
- Update rule:
  - if a state is present in the memory table, update the optimal action when reward for this new action is greater than some threshold.
  - if a state is not present, simply append it to memory table.



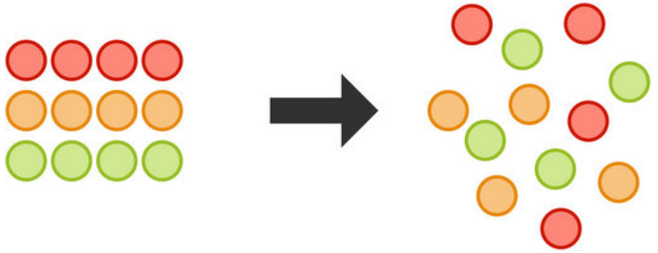
Reward Functions

- **Default**:
  - The default reward function is evaluated by finding the (neg) change in a number of vehicles for direction, and lanes in the action.
- **Diffusion**:
  - We treat an intersection as a medium for vehicles (particles) to pass through. The aim of this system should be to maximise the diffusion.
  - The Diffusion is calculated by the mean signed squared displacement of the particles
  - diffusion = $-\text{mean}(\text{sign}(d) \times d^2)$, d = next state - current state
- **Entropy**:
  - Similarly, the entropy of the whole system should be high. Low entropy implies congestion in a lane.
  - The entropy for our system is calculated by the difference between current and next state weighted entropy.
  - $e = w_{i+1} - w_i$
  - weighted entropy of a state,
    - $w_i = -\sum_j (s_{ij} \times \ln(s_{ij}))$

Diffusion



Entropy

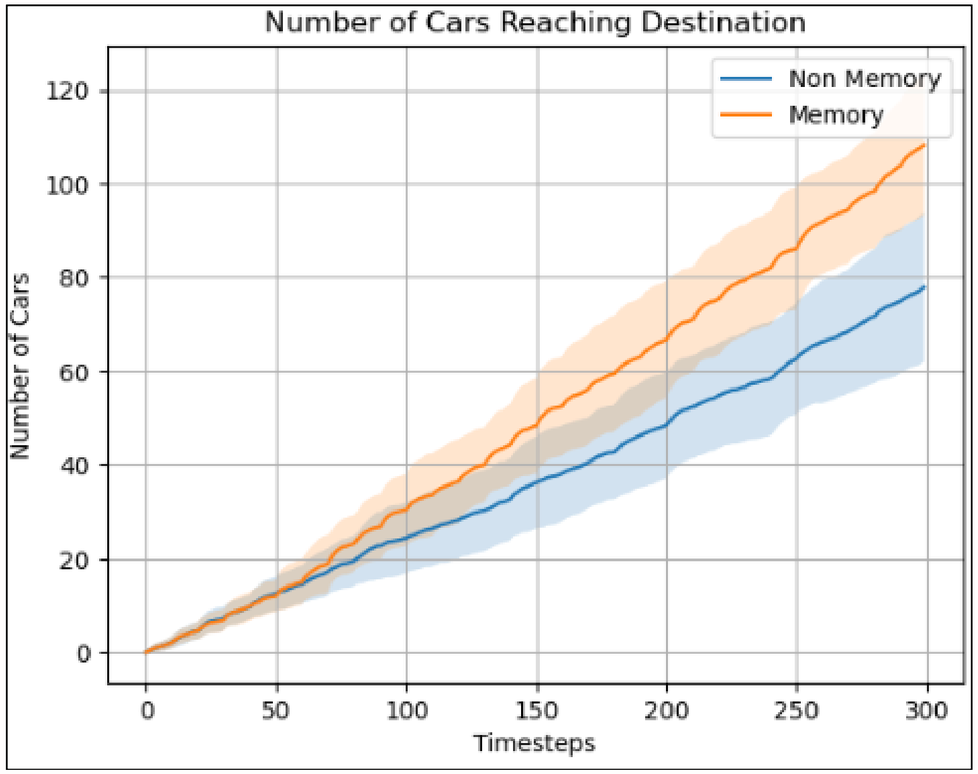


Single Agent Results

Average Waiting Time (W)

Non Memory	Memory
18.6 +- 13.6	26.2 +- 9.5

- Acting greedily based on the memory maximize the number of cars passing the intersection
- While the non-memory approach try to minimize the waiting time of vehicles in the intersection



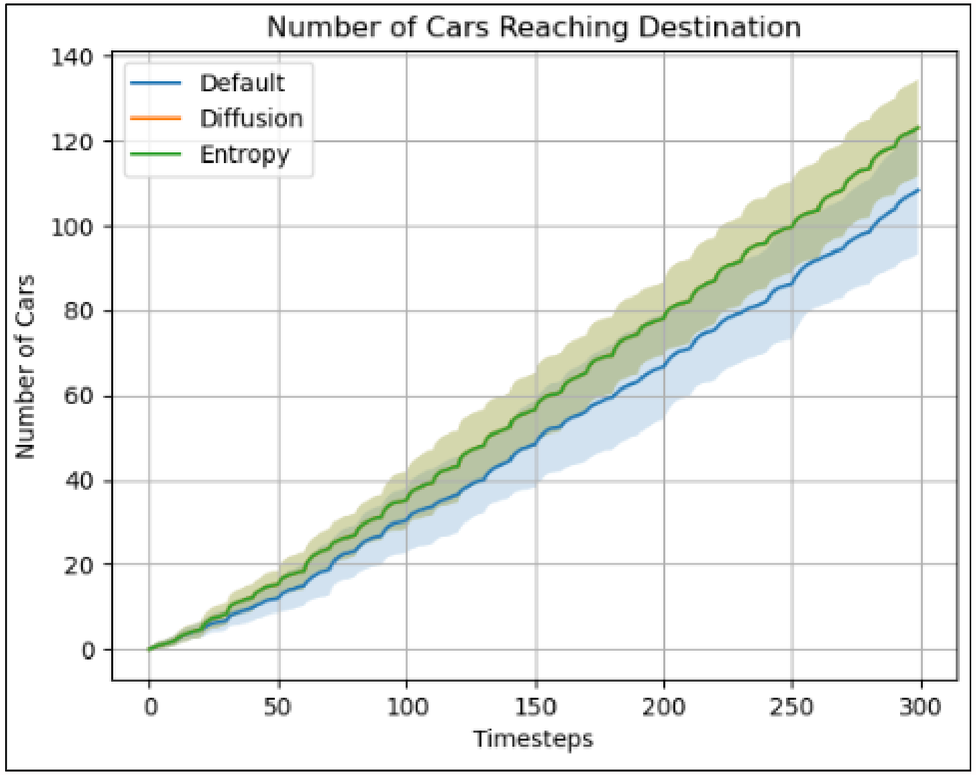
Single Agent Results

Average Waiting Time (W)

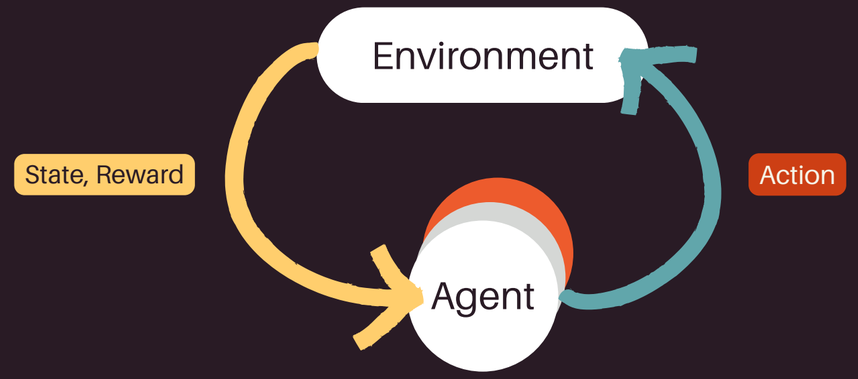
Default	Diffusion	Entropy
26.2 +- 9.5	19.4 +- 6.4	19.4 +- 6.4

Using diffusion and entropy as the reward give same results:

- Diffusion and Entropy have different absolute magnitude but have similar relative magnitude
- The traffic control problem is not sensitive to reward scaling (unlike Atari games where the agents learn differently based on the reward scaling)

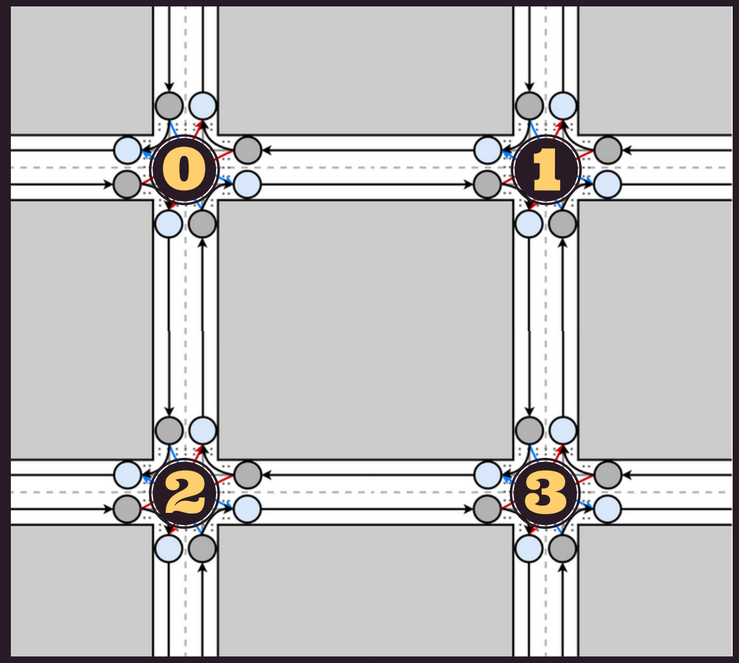


Decentralized MDP



Previous MDP formulation can be extended to multi agents settingby considering all agents, which defined as $\mathcal{M} \triangleq (I, \mathcal{X}, \mathcal{A}, P, R)$

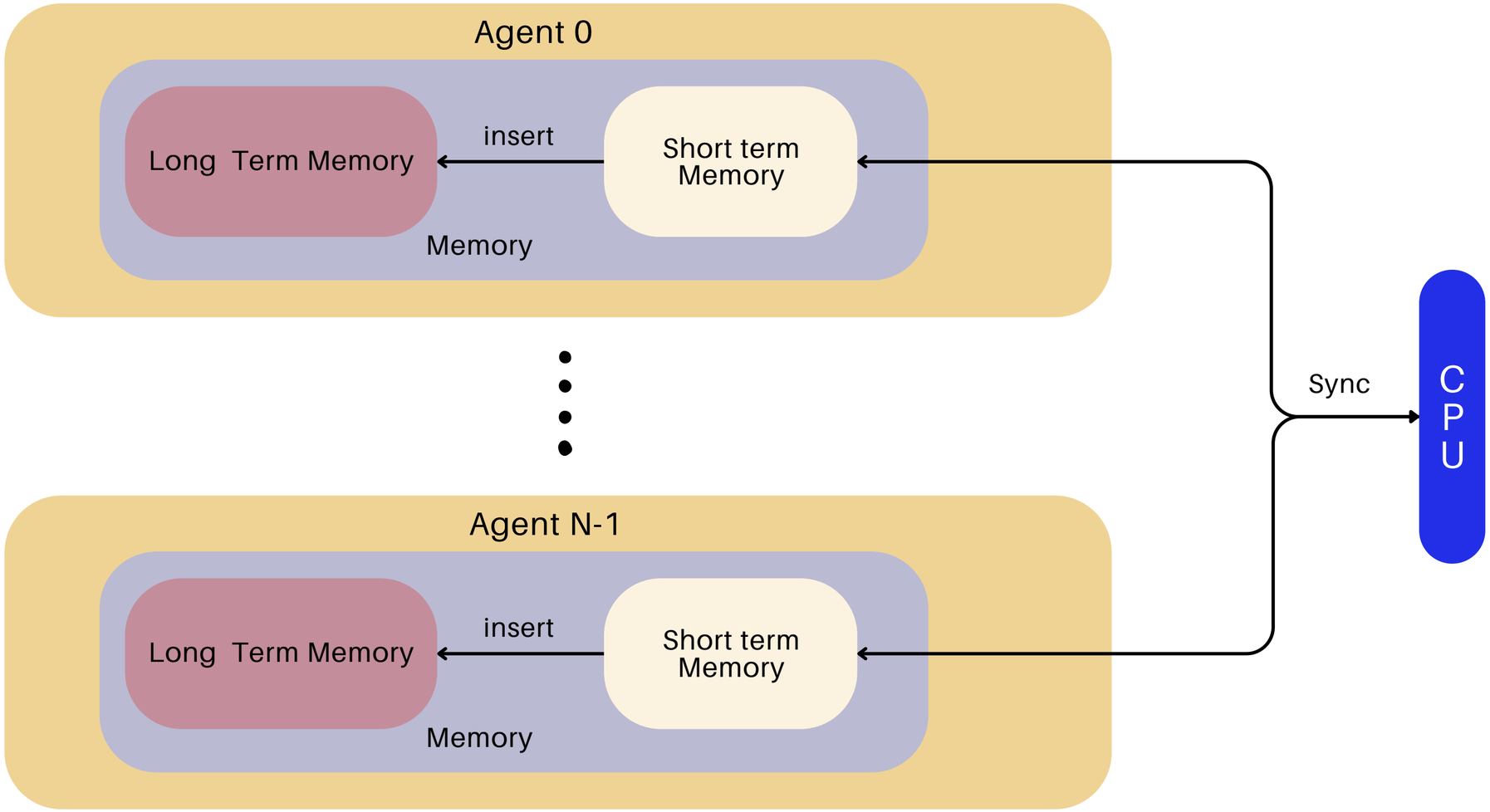
- I - Agent set:
  - The set of agents indexed 0, ..., N-1.
- $\mathcal{X}$ - States:
  - $[x_t(D/i)]$ for all Directions, {E,N,W,S}, and Lanes, {L,F,R}
- $\mathcal{A}$ - ACTIONS:
  - be the set of actions taken by traffic lights, i.e. the modes.
  - $\mathcal{A} \triangleq$ {straight line motion} ∪{adjacent motion}
  - $\mathcal{A} \triangleq$ {(E/W, F), (E/W,L), (N/S, F), (N/S, L), (E, F/L), (N, F/L), (W, F/L), (S, F/L)}
- P - TRANSITION PROBABILITY:
  - arrivals are modelled as a compound poison process
  - the rate of vehicles departing is set to 1.
- R - REWARDS:
  - the reward obtained by transitioning from state x_t to x_{t+1}.
  - Default rewards are proportional to the number of vehicles leaving due to an action. And proportionally negative when the increase in vehicles is too high.



Multi-agent Framework

CPU layer will synchronize and keep the Q-tables (short-term memory) of all agents

- Implementation details:
  - No delay or interference in the data passing between memories and CPU
  - Synchronization occurs every x timesteps
  - Only seen states that will be updated on each memory (partial update)



Grid Implementation

For this project, we assume a Manhattan grid structure with all in-between roads of the same length.

This configuration can easily be changed with little modification as the grid structure is initialised with a dictionary. This will allow us in future to test our multi-agent setup on different configuration of the intersection-grids.

```python
# Define the graph structure
graph_structure: Dict[Any, Tuple[int, List[Tuple[Any, int, str]]]] = {
    'A': (4, [('in1', 0, 'N'), ('B', 1, 'E'), ('D', 2, 'S'), ('in2', 0, 'W')]),
    'B': (4, [('in3', 0, 'N'), ('C', 3, 'E'), ('E', 3, 'S'), ('A', 1, 'W')]),
    'C': (4, [('in4', 0, 'N'), ('in5', 0, 'E'), ('F', 3, 'S'), ('B', 3, 'W')]),
    'D': (4, [('A', 2, 'N'), ('E', 3, 'E'), ('in6', 0, 'S'), ('in7', 0, 'W')]),
    'E': (4, [('B', 3, 'N'), ('F', 3, 'E'), ('in8', 0, 'S'), ('D', 3, 'W')]),
    'F': (4, [('C', 3, 'N'), ('in9', 0, 'E'), ('in10', 0, 'S'), ('E', 3, 'W')])
}
```

Fig: sample grid structure example

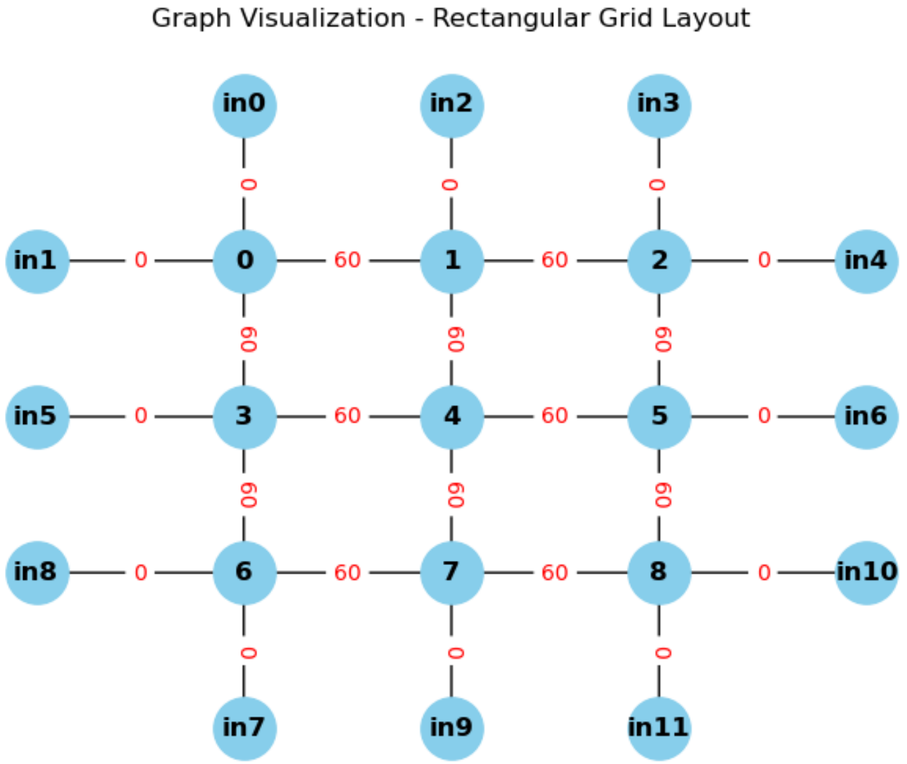
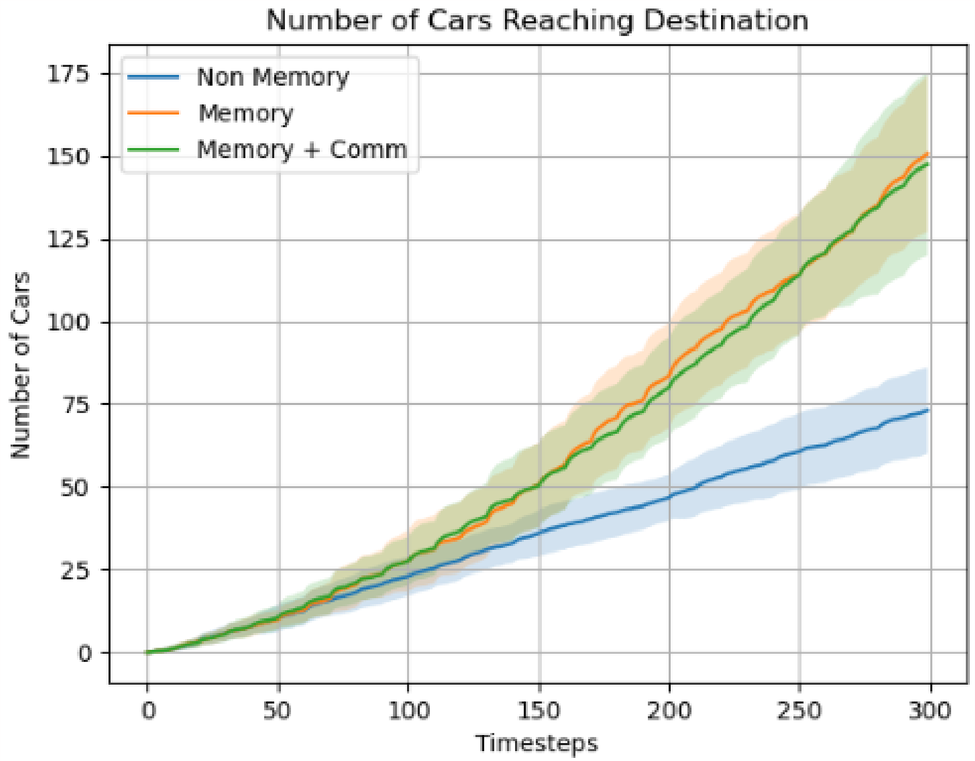


Fig: Grid Structure used in simulation

Multi-agents Results



- The gap between memory and non-memory approach widen in the multiple intersection setting
- Communication (CPU) layer reduces the variation of waiting time slightly

Average Waiting Time (W)

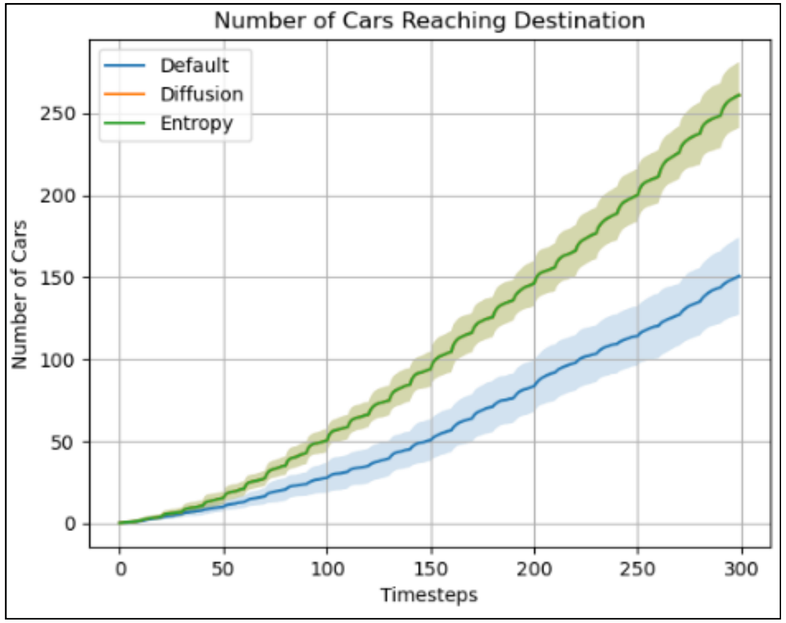
Non Memory	Memory	Memory + Communication
16.8 +- 7.6	25.1 +- 4.3	26.7 +- 4.4

Average Waiting Time per Node

Node	Non Memory	Memory	Memory + Communication
0	16.8 +- 11.7	24.0 +- 5.3	26.3 +- 5.7
1	15.8 +- 9.7	23.7 +- 5.5	26.1 +- 5.4
2	18.2 +- 13.0	28.5 +- 7.5	28.8 +- 6.6
3	18.5 +- 10.2	23.7 +- 4.7	25.7 +- 5.5
4	13.4 +- 7.8	26.7 +- 14.3	26.3 +- 8.2
5	16.9 +- 9.1	24.2 +- 5.2	26.7 +- 5.2
6	16.6 +- 10.1	28.7 +- 7.0	28.4 +- 6.8
7	14.7 +- 8.1	25.5 +- 8.0	25.2 +- 7.5
8	16.5 +- 9.0	23.8 +- 5.0	26.6 +- 5.6

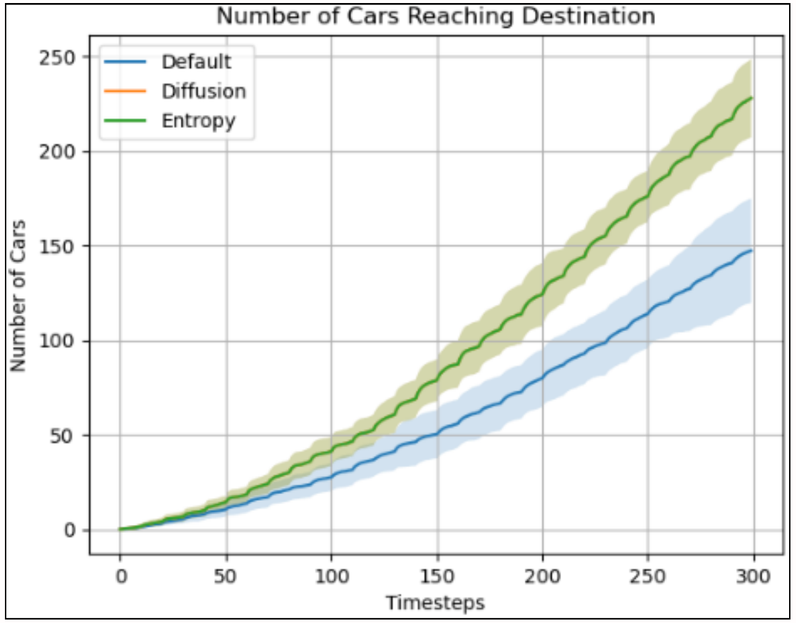
Multi-agents Results

Memory



Default	Diffusion	Entropy
25.1 +- 4.3	18.7+- 2.0	18.7+- 2.0

Memory + Communication



Default	Diffusion	Entropy
26.7 +- 4.4	23.6 +- 2.2	23.6 +- 2.2

Multi-agents Results

Average Waiting Time per Node

Memory

Node	Default	Entropy	Diffusion
0	24.0 +- 5.3	19.4 +- 2.5	19.4 +- 2.5
1	23.7 +- 5.5	19.3 +- 2.8	19.3 +- 2.8
2	28.5 +- 7.5	19.7 +- 3.0	19.7 +- 3.0
3	23.7 +- 4.7	18.0 +- 2.7	18.0 +- 2.7
4	26.7 +- 14.3	20.2 +- 5.3	20.2 +- 5.3
5	24.2 +- 5.2	17.6 +- 3.7	17.6 +- 3.7
6	28.6 +- 7.0	19.4 +- 3.2	19.4 +- 3.2
7	25.5 +- 8.0	19.1 +- 3.7	19.1 +- 3.7
8	23.8 +- 5.0	16.8 +- 3.2	16.8 +- 3.2

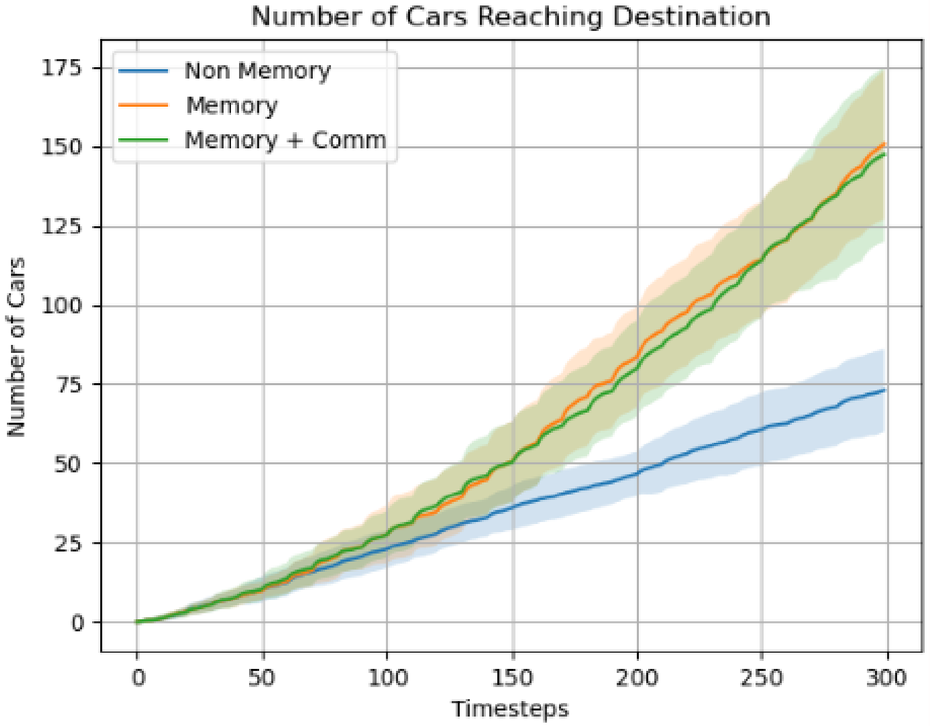
Memory + Communication

Node	Default	Entropy	Diffusion
0	26.3 +- 5.7	25.2 +- 3.8	25.2 +- 3.8
1	26.1 +- 5.4	25.0 +- 3.9	25.0 +- 3.9
2	28.8 +- 6.6	26.3 +- 4.0	26.3 +- 4.0
3	25.7 +- 5.5	22.6 +- 3.2	22.6 +- 3.2
4	26.3 +- 8.2	26.8 +- 5.3	26.8 +- 5.3
5	26.7 +- 5.2	22.0 +- 3.2	22.0 +- 3.2
6	28.4 +- 6.8	24.2 +- 3.5	24.2 +- 3.5
7	25.2 +- 7.5	21.8 +- 3.5	21.8 +- 3.5
8	26.6 +- 5.6	20.6 +- 3.1	20.6 +- 3.1

- DIffusion and entropy yield the same results in multi intersection setting
- Communication (CPU) layer does not affect the variances of waiting time for entropy and diffusion-based reward

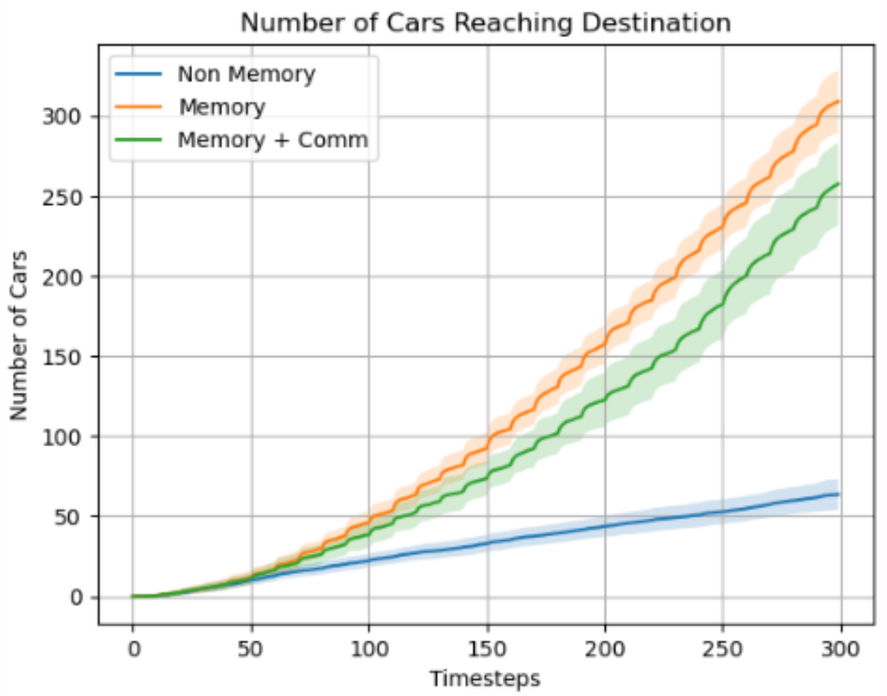
Multi-agents Results

3x3



Non Memory	Memory	Memory + Communication
16.8 +- 7.6	25.1 +- 4.3	26.7 +- 4.4

5x5

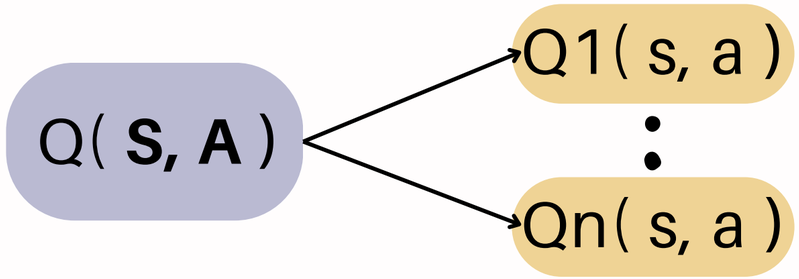


Non Memory	Memory	Memory + Communication
10.8 +- 5.1	17.7+- 1.4	23.2 +- 2.5

Further Analysis

Why independent agents perform better than the semi-centralized agents?

- Independent agents setting learn the decomposed Q-functions which induce more noise (more exploratory)



- Recent results also show such behaviour on certain cases

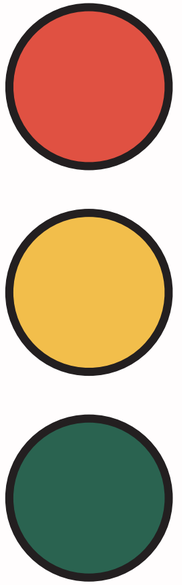
Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge?

Map	corridor	$6h\_vs\_8z$	$3s5z\_vs\_3s6z$	$3m$	$2m\_vs\_1z$
IPPO	80	60	90	100	100
MAPPO	0	8.95	80	99.875	100

Median Win Rate

Future works

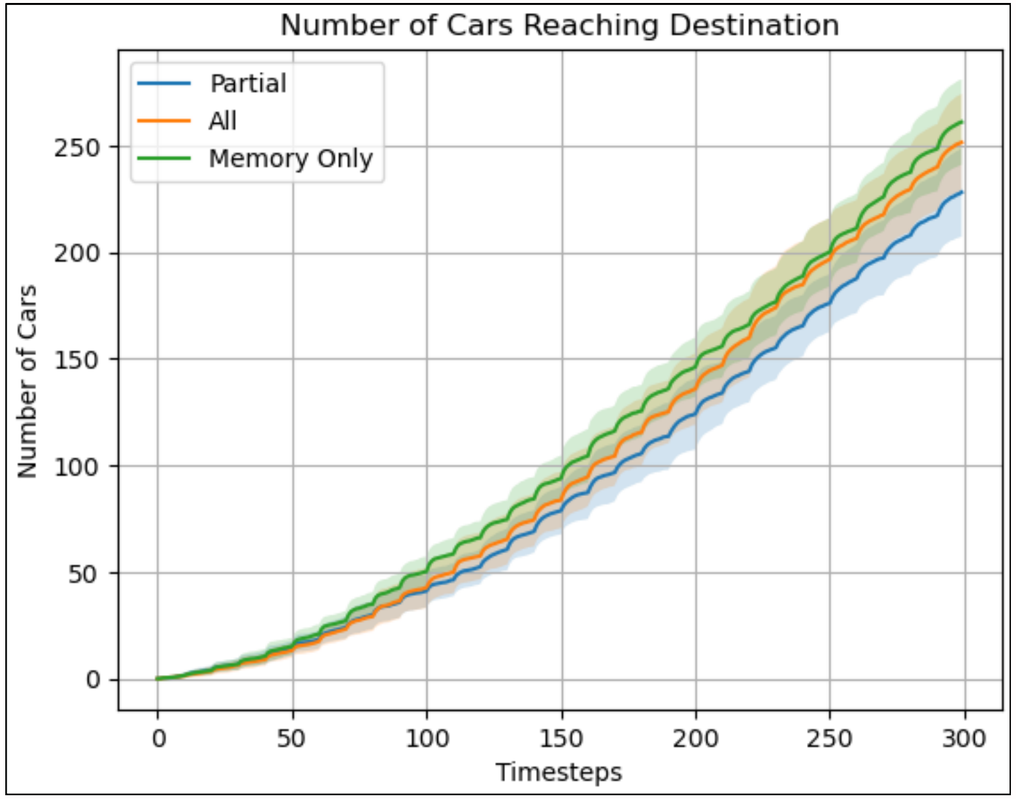
1. Embed green light duration in the action space -> more adaptive traffic control
2. Analyze the effect of CPU-memory delay in the traffic control
3. Synchronise long-term memory with the CPU instead of short-term memory for a more stable state-action pair
4. Experiment with different LTM stage rules, e.g. periodic staging vs frequency-based staging.
5. Testing on non-manhattan grid structures.
6. Using a more extreme state abstraction algorithm.
7. Testing the overall structure in a different environment setting, e.g. atari games.



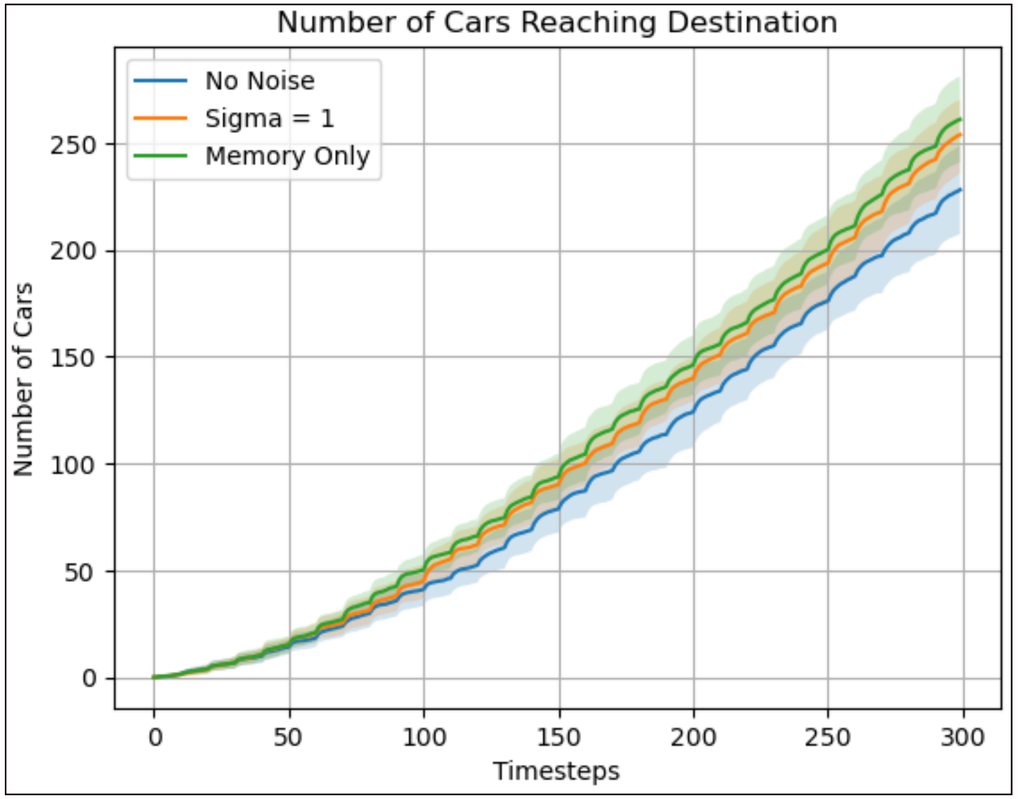
Appendix

Appendix

Memory Update



CPU with Noise



Thank you

MUKUL CHODHARY
KEVIN OCTAVIAN
EE488F 2023|KAIST