CPE 593 WS - Applied Data Structures and Algorithms

Neural Network Application in Traffic Management

Meet the team

Syed Ahmad Shah

Masters of Applied Artificial Intelligence

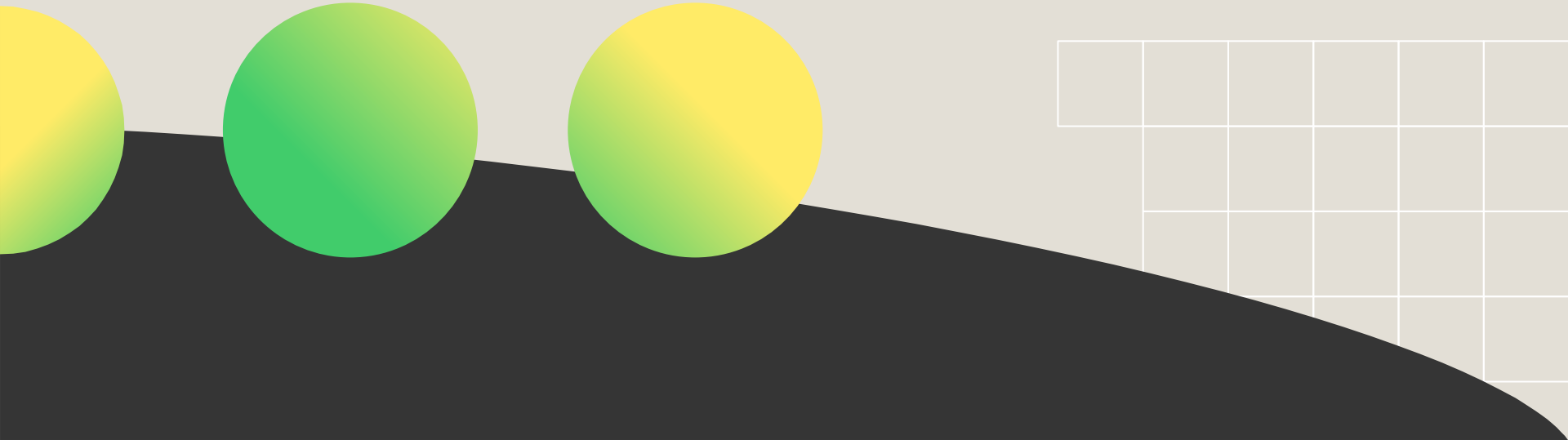
Christina Cuneo

Masters of Applied Artificial Intelligence

What is our Goal?

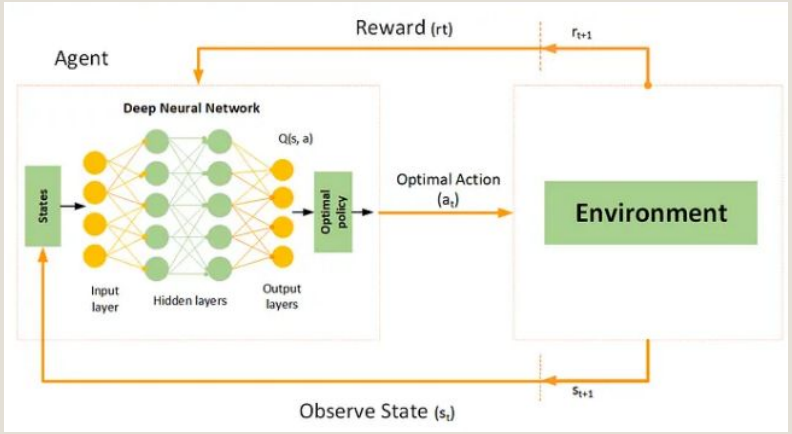
- We are addressing the limitations of **traditional traffic management**, which relies on **fixed timing or rule-based strategies** that cannot adapt to the rapid, unanticipated shifts in demand common in urban environments.
- This is an essential sector for improvement, as **suboptimal timing** leads to **increased vehicle delays, fuel consumption, and gridlock** that may lessen the reliability and preference of car transportation.
- Our idea is to utilize a Deep Q network to develop an adaptive traffic controller, to dynamically manage signal phases and duration
- The objective is to maximize vehicle throughput while minimizing wait times and queue length.

Proposed Solution



What is a DQN?

- Deep Q Network is a reinforcement learning (RL) algorithm that utilizes Deep neural Networks to approximate the optimal action-value function.
- DQN is an improvement over the basic Q-Learning algorithm. Which traditionally, Q-Learning defines a table and makes rewards for each possible action, which works great when there is a finite number of possibilities. However for a continuous function, Q-Learning would grow exponentially, thus becomes an infeasible technique.
- DQN is an improvement as rather than keeping a large table, it overcomes this limitation by replacing it with a network that can approximate the Q-values for any state-action pair.



Matrix Factorization

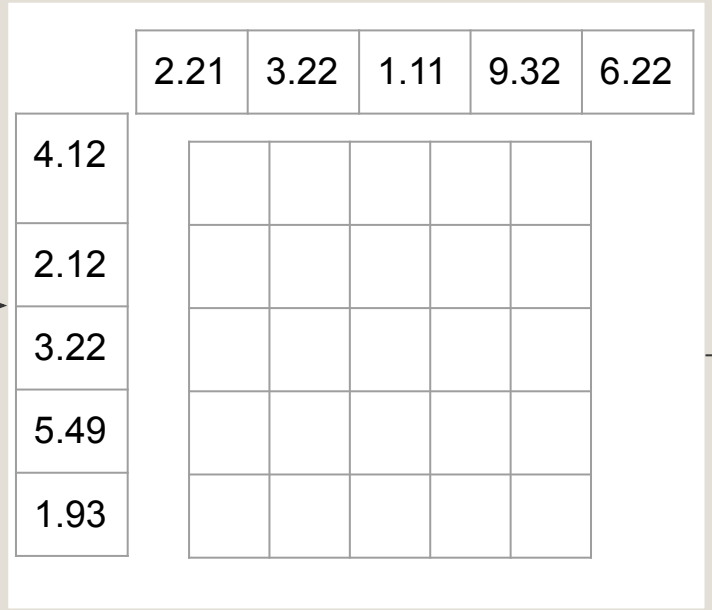
p x d
d x q
Where rank d= 1

Space Complexity:
Full Matrix: O(n^2)
MF: O(nd + dn) = O(2nd) = O(nd)

Matrix Size of 1000x1000:
FM: 1,000,000
MF: 2,000
Factor: 99.8%

n x n

5	3	0	1
4	0	0	1
1	1	0	5
1	0	0	4
0	1	5	4



n x n

5.0272	2.8334	5.7201	0.9964
3.9356	2.2207	4.6045	0.9969
1.1161	0.6835	4.0319	4.9612
0.9371	0.5717	3.2742	3.9749
2.4594	1.4269	4.8650	4.0341

State Representation

Experience Replay Buffer: Dynamic Memory bank that stores the model's past interactions with the environment to be reused for training.

Example:
Should I leave the green light activated for 11 seconds? What happened when I activated the green light for 10 seconds

To implement this we utilize a **"deque"** with a max capacity of **20,000** consisting of **tuples** (s, a, r, s', done) [current state, action, reward, next state, terminated]

Why?
Using a deque provide O(1) complexity for adding new items and removing the oldest actions when we reach the capacity.

27-Dimensional State Vector: The environment is encoded a multidimensional **array (Numpy)**

- **Spatial Data:** 12 lane queues **(4 approaches x 3 lanes)** and 12 lane wait-times
- **Time Data:** Current Phase, time elapsed, last selected duration

Action Space Mapping: We have a discrete action space for the model to choose from, a total of **45 choices**, mapping it to a (Phase, Duration) pair

Training Loop

We train the model in a loop for **1,000 iterations**:

1. **Environment Step:** The “TrafficEnv.step()” function executes an action that simulates vehicle flow (EX: 2 cars for 10 s for straight lanes)

2. **State Transition:** The algorithm adds random vehicles to the current state and updates the wait time counters for non-active lanes

3. **Experience Storage:** We push the information from training into the “deque” buffer

4. **Decay Logic:** The epsilon-greedy exploration rate is updated every iteration.

The time complexity of the DQN is determined by two phases:

- Interaction Phase: O(|A|) to select an action (|A| is the # of possible actions
- Training Phase: O(N x W^2)

This is an optimization problem, as we are looking to **minimize** the **error function**, or in other word **maximize** the **reward function**:

$$R_t = 5P - 0.03W - 0.5Q - 4O - X + \Pi$$

Variable Definitions:

- P: Vehicles passed (Throughput Reward)
- W: Cumulative wait time across all lanes
- Q: Total aggregate queue length
- O: Overtime duration (> 60 seconds)
- X: Maximum single-lane wait (Starvation constraint)
- Π: Phase pressure heuristic

Where we also implement a **pressure penalty** to prevent the model from ignoring the most needed phase.

To derive this value, we calculate the gap between the pressure of the best possible phase (p_max) and the chosen phase

$$\Pi = -0.18 \times \max(0, p_{max} - p_{selected})$$

Algorithm Input/Output

Input

HEAVY A/C STRAIGHT AND RIGHT-DEMAND SCENARIO

App.	LQ	SQ	RQ	TQ	Waits (L,S,R)	Phase / Time
A	1	9	3	13	(18,18,18)	AC_forward / 20
B	0	2	1	3	(4,4,4)	AC_forward / 20
C	2	8	2	12	(16,16,16)	AC_forward / 20
D	1	1	1	3	(5,5,5)	AC_forward / 20

Output:

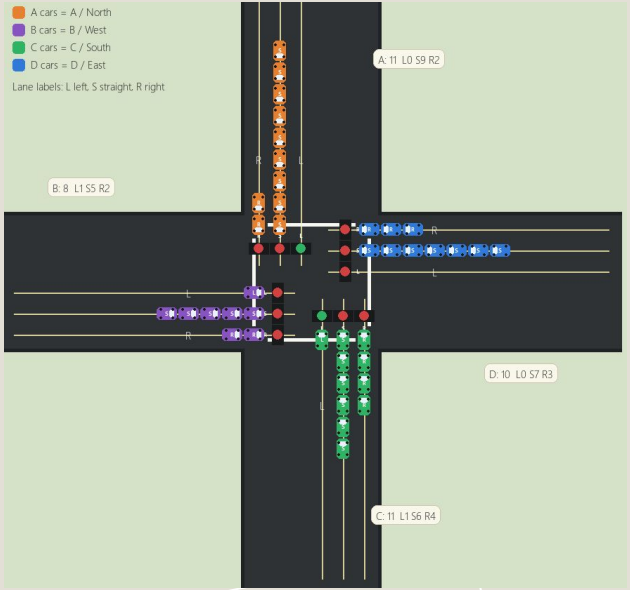
	action_id	phase	duration_s	q_value	selected
40	40	BD_left	50	-7694.384766	True
13	13	AC_forward	50	-7723.174316	False
41	41	BD_left	60	-7761.450684	False
39	39	BD_left	40	-7767.203613	False
11	11	AC_forward	30	-7772.420898	False
12	12	AC_forward	40	-7780.854492	False
42	42	BD_left	70	-7788.079102	False
33	33	AC_left	70	-7799.320312	False
32	32	AC_left	60	-7813.319824	False
14	14	AC_forward	60	-7822.116699	False

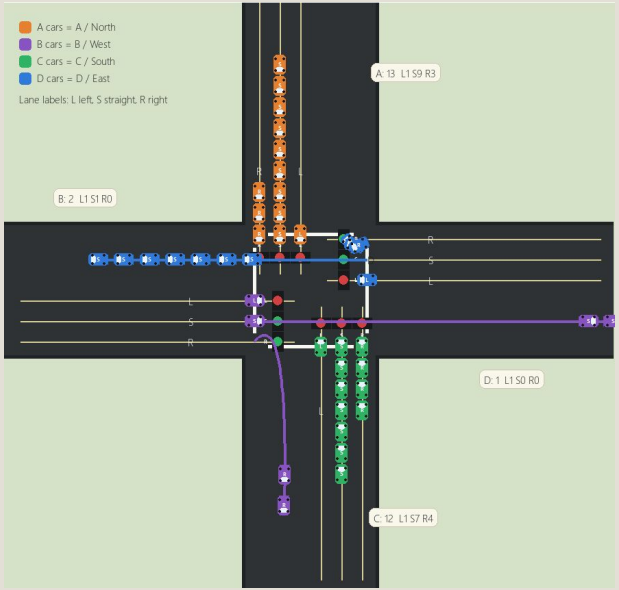
Normalized Input Vector:

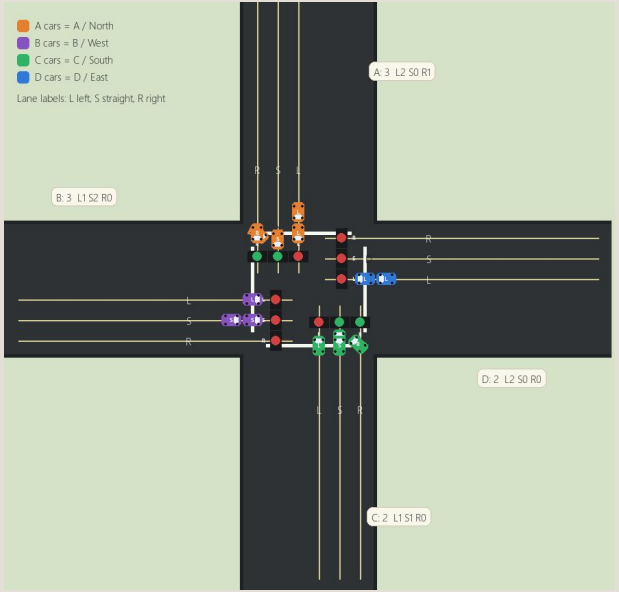
TRAFFIC PHASE AND QUEUE DATA

Parameter	Value
A_left_queue	0.050000
A_straight_queue	0.450000
A_right_queue	0.150000
B_left_queue	0.000000
B_straight_queue	0.100000
B_right_queue	0.050000
C_left_queue	0.100000
...	...
C_right_queue	0.100000
D_left_queue	0.050000
B_right_wait	0.033333
C_left_wait	0.133333
C_straight_wait	0.133333
C_right_wait	0.133333
D_left_wait	0.041667
D_straight_wait	0.041667
D_right_wait	0.041667
current_phase	0.200000
time_in_phase	0.222222
last_duration	0.111111

Real-Time Simulation







Baseline Comparisons

To put the DQN's performance in context, we evaluated it against three non-learning controllers over 50 episodes of 120 steps each.

Each represents a different class of controller: open-loop scheduling, reactive greedy, and analytical optimization. None of them learn.

Baseline Controllers

Fixed-Time Controller (FTC)

- Preset times, no learning or adaptation
  - cycles all phases on a fixed 30-second timer, no awareness of current traffic state
- Algorithm notes: basically just looping
- Data Structures: lists for phases

Webster's

- Based on classical traffic engineering formula
  - $$C^* = \frac{(1.5L + 5)}{(1 - Y)}$$
  - Computes the optimal cycle length from observed demand, then splits green time proportionally across phases

Max Pressure Control (MPC)

- Adaptive, picks phase based on current state to relieve most pressure
- at each step:
  - scores every phase by summing the queues it would serve and picks the highest
  - green duration is set proportional to that phase's pressure share
- Algorithm notes: greedy, No learned behavior

DQN

- trained over 1,000 episodes
- feeds the 27-dim state vector through the network and picks action with the best Q-value
- Selects both phase and duration
- Algorithm notes: Value-based reinforcement learning
- Data Structures: replay buffer (deque), neural network weights

Controlled Scenario Probing

1. Balanced light demand

2. Heavy A/C straight and right-turn demand

3.Heavy B/D straight and right-turn demand

4. A/C left-turn pocket pressure

5. B/D left-turn pocket pressure

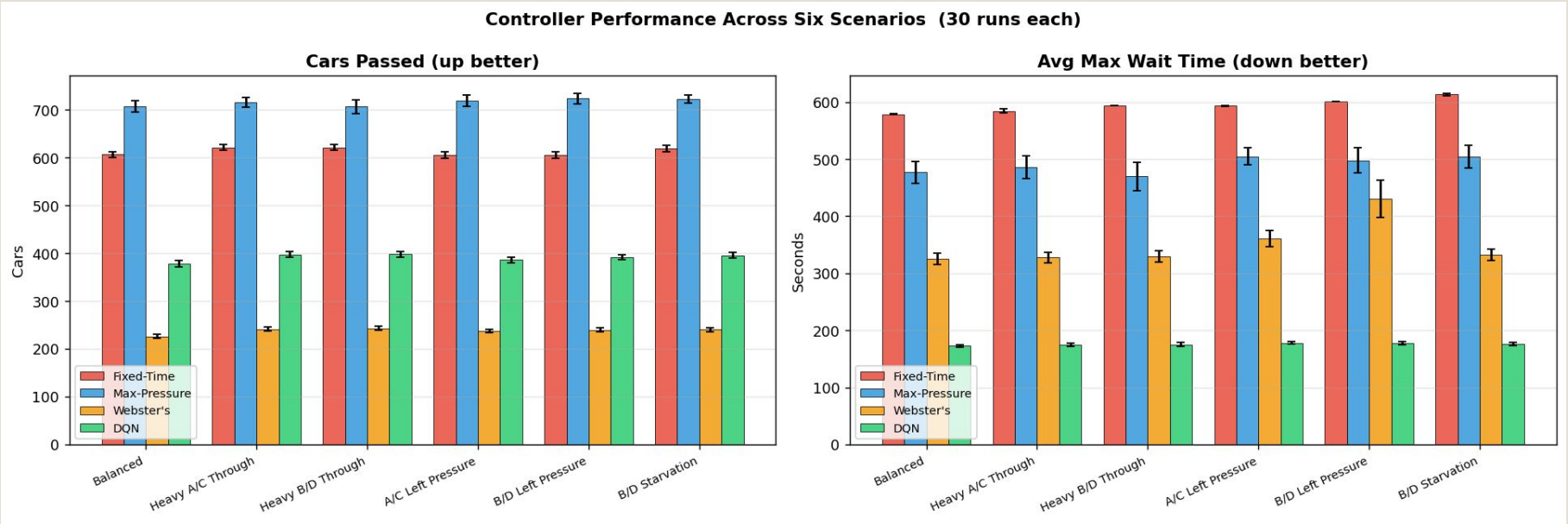
6. B/D starvation (modest queue, long wait)

Approach:
one of the four directions entering the intersection (A/B/C/D), each with three lanes

Pocket pressure:
queue building specifically in the left-turn lane, separate from the main approach queue

Starvation:
an approach keeps getting skipped; queue stays small but wait time grows

Controlled Scenario Probing



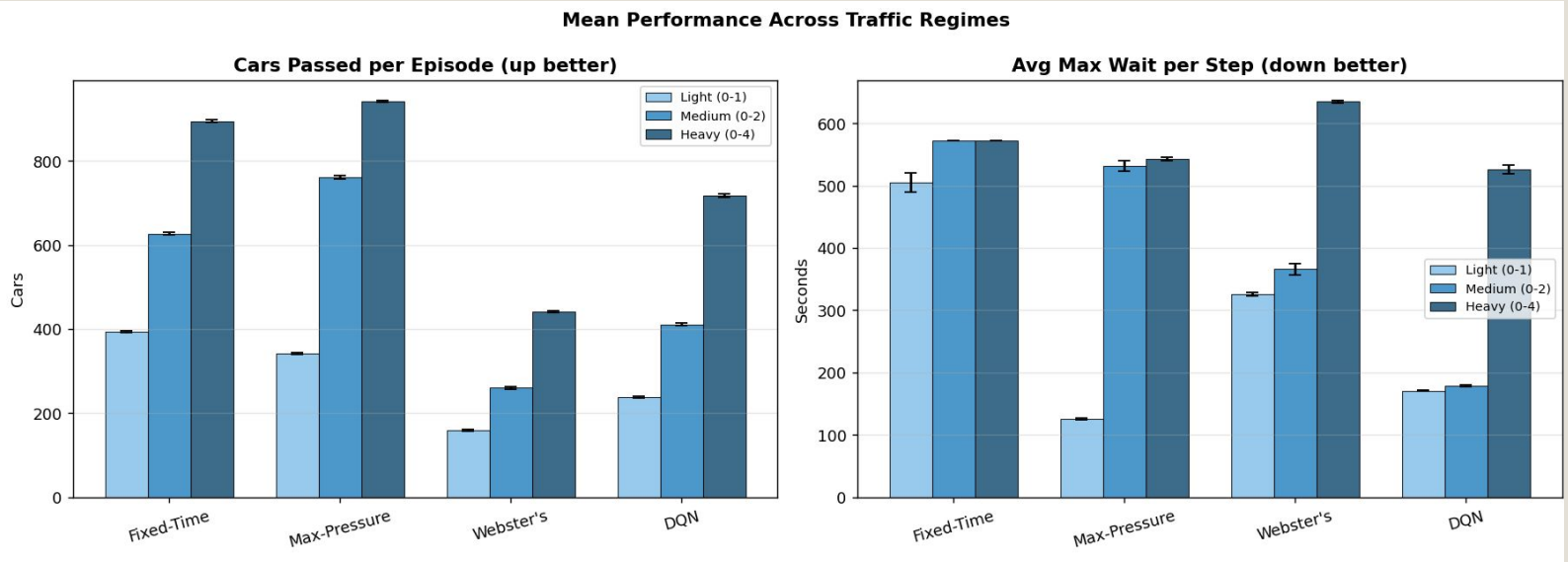
1. Balanced light demand
2. Heavy A/C straight and right-turn demand
3. Heavy B/D straight and right-turn demand
4. A/C left-turn pocket pressure
5. B/D left-turn pocket pressure
6. B/D starvation (modest queue, long wait)

Results

The DQN doesn’t pass the most cars, but it has much smaller average wait times, and the smallest final queue.

Controller	Cars Passed	Reward	Avg Max Wait (s)	Final Queue
Fixed-Time	624.5	−40,485	573.0	146.8
Max-Pressure	689.0	−29,869	458.9	82.3
Webster's	257.6	−9,465	363.4	33.6
DQN	**408.7**	**−5,397**	**180.8**	**32.4**

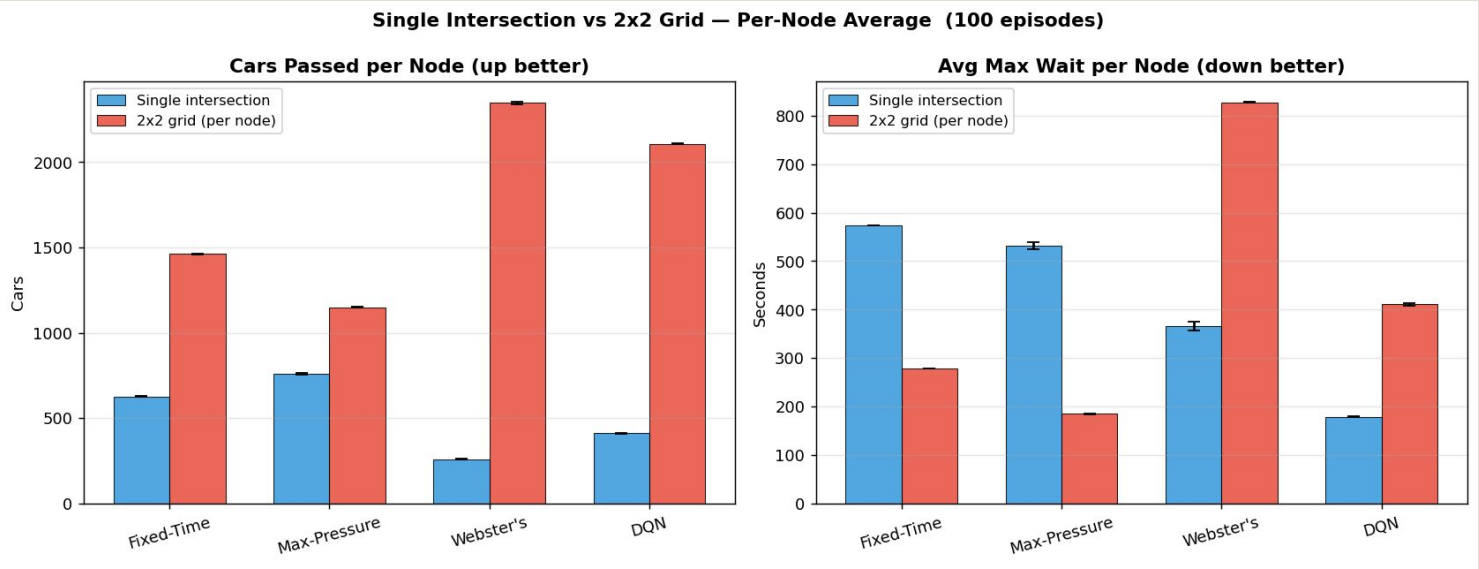
Results



Mean average maximum wait time and final queue length for each controller across light (0–1), medium (0–2), and heavy (0–4) arrival rates.

Each bar is the mean over 200 episodes; error bars show 95% confidence intervals. The DQN was trained only on the medium regime.

Results - Single Intersection vs. 2×2 Grid



Cars passed and average maximum wait time per node, comparing single-intersection performance against the 2×2 grid across all four controllers (100 episodes). The same DQN weights were used at each grid node without retraining.

References:

[1] S. Amin, "Deep Q-Learning (DQN)," Medium, Sept. 14, 2024, medium.com/@samina.amin/deep-q-learning-dqn-71c109586bae. Accessed 13 May 2026.

[2] "Traffic Simulation Optimization Considering Driving Styles," ScienceDirect, www.sciencedirect.com/science/article/pii/S2772424725000216. Accessed 28 Apr. 2026.

[3] "Traffic Flow Theory," FHWA, www.fhwa.dot.gov/publications/research/operations/tft/chap10.pdf. Accessed 28 Apr. 2026.

[4] "Energy Consumption and Fault Analysis of Data Center Power System," IEEE Xplore, ieeexplore.ieee.org/document/10245678/. Accessed 28 Apr. 2026.

[5] "Research on Optimization Algorithm for Urban Traffic Flow Based on Computer Simulation," ResearchGate, www.researchgate.net/publication/375849235. Accessed 28 Apr. 2026.

[6] X. Deng et al., "Traffic Flow Simulation of Modified Cellular Automata Model Based on Producer-Consumer Algorithm," PeerJ Computer Science, Sept. 2022, pmc.ncbi.nlm.nih.gov/articles/PMC9575854/.

[7] P. Varaiya, "Max pressure control of a network of signalized intersections," Transportation Research Part C, vol. 36, pp. 177–195, 2013.

[8] F. V. Webster, Traffic Signal Settings, Road Research Technical Paper No. 39, HMSO, London, 1958.