Original Research Article

# Traffic Control System: Optimization of Signal Efficiency Using Deep Reinforcement Learning

Shaunak Thamke

*The Bronx High School of Science, 75 W 205th St, Bronx, NY 10468, United States*

## ABSTRACT

Traffic congestion remains a major challenge in modern cities. It contributes to delays, fuel waste, pollution, and reduced quality of life. Most intersections still rely on fixed-time signals that do not adapt to real-time traffic demand. This research investigates whether Deep Reinforcement Learning (DRL) can optimize signal timing more effectively than traditional approaches. Historical Kaggle traffic data and real-time New York City (NYC) data was used to generate Poisson-based vehicle arrivals across four-way intersections. Two DRL agents, Proximal Policy Optimization (PPO) and Deep Q Network (DQN), were trained to minimize vehicle wait-time (delays) and queue length (congestion). Although DQN has been widely used in previous traffic-signal studies, PPO remains underexplored. This work provides an evaluation of both methods against two baselines: Fixed-Time Baseline Controller (FBC) that used Kaggle data and Analytical Baseline Controller (ABC) that used real-time NYC data. PPO performed the best and reduced average wait-time by 90.9% compared to FBC and 69.46% compared to ABC. DQN saved 61.32% compared to ABC. PPO consistently delivered greater reductions and stability across runs. Generated heatmaps based on 20 simulations confirmed the adaptability of PPO in maintaining low queue lengths and avoiding severe congestion, common in fixed-time and analytical systems. The DQN heatmap showed reduced queues with occasional variability, while PPO was stable. These findings highlighted the potential of PPO for dynamic, data-driven traffic control which was demonstrated using interactive traffic simulation. This research has great potential in improving traffic flow efficiency, reducing congestion and pollution.

**Keywords:** Analytical Baseline Controller; Congestion; Deep Q Network; Deep Reinforcement Learning; Emergency Vehicles; Fixed-Time Baseline Controller; Proximal Policy Optimization; Traffic Control System

## INTRODUCTION

Traffic congestion and vehicle pollution remain among the most persistent challenges faced by cities that affect billions of people each day. In the United States, the severity of this problem is widely acknowledged. The American Lung Association's *State of the Air 2025* report states that 156 million Americans, nearly 46% of the population, live in areas graded "F" for unhealthy ozone or particulate pollution (1). Furthermore, according to the Texas A&M Transportation Institute's *2025 Urban*

**Corresponding author:** Shaunak Thamke, E-mail: shaunakthamke@gmail.com.



**Accepted** June 15, 2026
https://doi.org/10.70251/HYJR2348.43557567

*Mobility Report* the average American lost approximately 63 hours to traffic delays over the year (2). Intersections are the major bottlenecks that cause long queues, waste fuel, and increased emissions.

Most intersections still operate using fixed-time traffic signals, regardless of actual traffic demand. Fixed-time systems cannot adapt to fluctuating conditions. This lack of adaptability leads to inefficiencies during rush hours, high-volume periods, or sudden congestion spikes. More sophisticated rule-based or analytical controllers use predetermined logic or historical patterns to solve these issues. However, they struggle to adapt to real-time conditions and often fail to minimize delay in unpredictable environments.

Recent advancements in artificial intelligence (AI), particularly in DRL, offer a promising alternative. DRL is a branch of artificial intelligence where an agent learns to make decisions by interacting with its environment as shown in Figure 1. DRL is particularly effective for dynamic, sequential problems like traffic signal control, where each action influences future queue lengths, delays, and overall flow patterns (3).

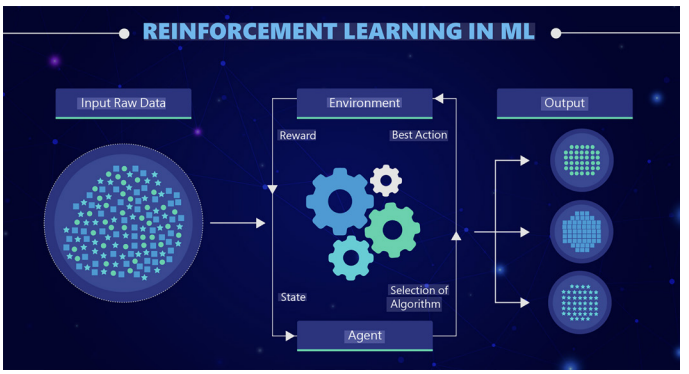



***Figure 1.*** *DRL uses Reinforcement Learning (RL) and deep neural networks. Rather than following explicit instructions, the agent discovers optimal behavior through trial and error. At each time-step, it observes the environment, takes an action, and receives a reward based on the outcome. Over time, the goal is to maximize cumulative reward, which naturally encourages long-term optimal decision-making. Source - United States Artificial Intelligence Institute.*

There are three major categories of DRL methods, value-based, policy-based, and actor-critic. Value-based methods, such as Q-Learning and DQN, learn a value function that estimates future reward for each state–action pair (4). DQN extends traditional Q-Learning by using neural networks. DQN can operate in larger, more complex environments. While DQN can perform well in discrete action spaces, it often struggles in environments with delayed rewards, high stochasticity, or rapidly changing traffic conditions (5). Policy-based methods, such as Reinforce, directly learn a policy that maps states to actions. They can be unstable due to high-variance updates. Actor-critic methods combine both ideas. The "actor" learns the policy while the "critic" estimates a value function to guide the actor's updates. This hybrid approach reduces variance and improves stability in complex, dynamic environments.

Among actor-critic algorithms, PPO has become one of the most widely used and reliable DRL methods. PPO introduces a clipped objective function that limits how drastically the policy can change during each update. It prevents instability and catastrophic forgetting. This makes PPO well-suited for environments where small parameter updates are essential for stable learning (6).

DQN has been used in several prior studies reviewed during this research. One of the goals of this project was to evaluate whether PPO could offer additional benefits over traditional value-based methods and DQN (7). DQN served as a strong baseline because it is simple, efficient for discrete actions. It provided clear value estimates for understanding the agent's decision-making behavior. However, addressing delayed rewards and non-linear queue dynamics remains challenging for DQN method. This makes it less ideal for long-term deployment. PPO, on the other hand, was chosen as the primary DRL method due to its strong stability, ability to manage sequential decision-making with delayed consequences (8).

This research project explores the use of DRL to develop a Traffic Control System capable of outperforming both fixed-time and analytical controllers. Real data was collected from NYC junctions manually. In addition, real traffic data from Kaggle, with hourly vehicle counts at a four-direction intersection, was preprocessed to create a continuous per-second arrival rate for simulation. A custom environment for the DRL models was built using the Gymnasium library (14). The library queued data in all four directions, with arrival and departure limits during green phases. The DRL agent can either maintain or switch the signal phase, with rewards encouraging the reduction of overall wait-time.

Two DRL algorithms were evaluated: PPO, known for stability in continuous decision-making, and DQN, that has been widely used in prior traffic studies. The models were trained for 200,000 timesteps and compared

against two traditional controllers: FBC with a 15-second cycle and ABC derived from external timing data. Experimental results showed that DRL, especially PPO, achieved substantial reductions in wait-time.

## LITERATURE REVIEW

Traffic signal control has been studied for decades and has evolved from fixed-time schedules to adaptive and intelligent methods. Advances in machine learning, especially RL and DRL, that combine RL with deep neural networks, have created new opportunities. They can help in designing smarter and adaptive traffic control systems. Relevant key literature was reviewed and summarized, understanding their contributions and limitations.

### Traditional & Rule-Based Control Methods

In *Smart Control of Traffic Light Using Artificial Intelligence*, Gandhi, Solanki, Baloorkar, and Daptardar (2020) proposed an AI-based adaptive traffic light system using You Only Look Once (YOLO), a popular object detection algorithm (9). Although the system outperformed fixed timing signals, it remains a snapshot based rather than continuous real-time approach. Additionally, it used a limited dataset quality, simplified modeling assumptions, and the absence of real-world validation.

### Reinforcement Learning & DRL in Traffic Control

*Reinforcement learning based adaptive optimal control model for traffic lights in intelligent transportation systems* (Huang et al., 2024) (10) showed a DQN based adaptive control. It significantly improved traffic efficiency over fixed-cycle signals. It also reduced congestion and improved throughput however, it was still a simulation-based study and lacked real-world validation.

### Adaptive traffic signal control using DRL

In *Intelligent Vehicle Pedestrian Light (IVPL): A Deep Reinforcement Learning Approach for Traffic Signal Control*, Yazdani and Sarvi (2023) proposed a DRL-based adaptive signal system (11) that jointly managed vehicle and pedestrian flows using a DQN framework. Their IVPL model significantly reduced overall user delay compared to traditional vehicle-only control strategies. The limitation was that it used only simulation environments and relied on simplified assumptions for traffic behavior. The study did not include real-world deployment or validation.

In *Deep Reinforcement Learning for Traffic Light Control in Intelligent Transportation Systems*, Zhu, Liu, Borst, and Walid (2023) used DQN and Deep Deterministic Policy Gradient (DDPG) (12). It shows that DRL can learn optimal policies, including emergent "greenwave" coordination. This work relied solely on simulated networks with simplified assumptions lacking real-world validation. It also had limited practical applicability.

In *Deep Reinforcement Learning Based Traffic Signal Control: A Comparative Analysis*, Wu, Kim, and Ma (2023) evaluated several DRL algorithms for adaptive signal control (13). They included DQN, Double Deep Q-Network (DDQN), Dueling DQN, and the policy-based Advantage Actor Critic (A2C) methods, while notably not included PPO. Their results showed that these DRL controllers mainly Q-learning variants reduced delays and queue lengths compared to fixed-time and actuated baselines. Their experiments were limited to a single simulated intersection and did not address real-world conditions.

A review of existing traffic signal research showed several recurring limitations. Most traditional and early RL systems relied on assumptions about perfect sensors or complete network coverage. These inputs are typically unavailable in actual city settings. DRL studies often depended on synthetic or simplified traffic simulations, limiting real-world relevance. Most models struggled with scalability, coordination across intersections, and changing traffic conditions.

Another persistent gap was the scarcity of real-world testing. Most approaches demonstrated performance only in controlled simulation environments. They rarely evaluated robustness under real traffic conditions such as peak-hour surges and randomness in arrivals. Prior work relied on value-based methods like DQN, while this study compared PPO and DQN using real-time historical data from Kaggle and NYC traffic data captured during this research. It provided a more realistic evaluation framework and addressed a major shortcoming identified in literature, i.e., the lack of research grounded in real traffic flows.

## METHODS AND MATERIALS

The research was conducted through multiple steps as shown in Figure 2. First, data acquisition and preprocessing were performed. This involved extracting hourly vehicle counts from Kaggle traffic data and

converting it into per-second Poisson arrival rates. The NYC data was gathered manually in real time, so no processing was required.

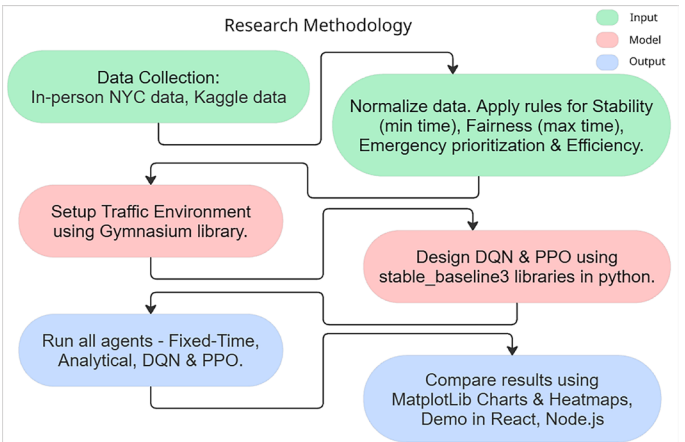



***Figure 2.*** *This methodology is used to design, implement, and evaluate the Traffic Control System. This approach integrates two main components: real-world traffic data from Kaggle and New York City intersection and a custom-built simulation environment. It uses Reinforcement Learning techniques (DQN & PPO) to train the system. It was benchmarked against two established controllers: a Fixed-Time Controller and an Analytical Controller governed by rule-based logic.*

As a part of environment construction, a custom traffic control simulation was built using the Gymnasium library. This environment was designed to accurately model four distinct queues (North-South, South-North, East-West, West-East). It handled real-time arrivals, departures, phase selections, and reward computation.

The next step involved Controller Design. Four separate controllers were implemented. FBC which used a 15-second duration per direction; ABC which operated on a set of rules for minimum and maximum phase times; and two distinct Reinforcement Learning agents, namely a DQN agent and PPO agent.

Following design, the Reinforcement Learning agents were trained over 200,000 timesteps using baseline3 library (15). All four controllers were then rigorously evaluated across multiple random seeds and various sampled traffic hours.

Finally, for comparison metrics, the primary measures of performance were the Total Wait Time (TWT) that represent delays and Queue Length that represent congestion. TWT was specifically quantified as the cumulative sum of all vehicles wait-time while Queue Length was number of vehicles queued over the entire simulation time.

**FBC using Kaggle Traffic Dataset**

The Kaggle traffic data set utilized for FBC contained hourly traffic counts for a four-direction intersection as shown in Figure 3 (16). *This dataset contains 48.1k (48120) observations of the number of vehicles each hour in four different directions: 1) DateTime, 2) Direction, 3) Vehicles, 4) ID (16).* After loading, timestamps were converted to datetime, missing hours were filled using interpolation, and traffic counts were pivoted into Vehicles_NS, Vehicles_SN, Vehicles_EW, Vehicles_WE. These values were used to compute arrival rates per second for each simulation hour. A simple 15-second

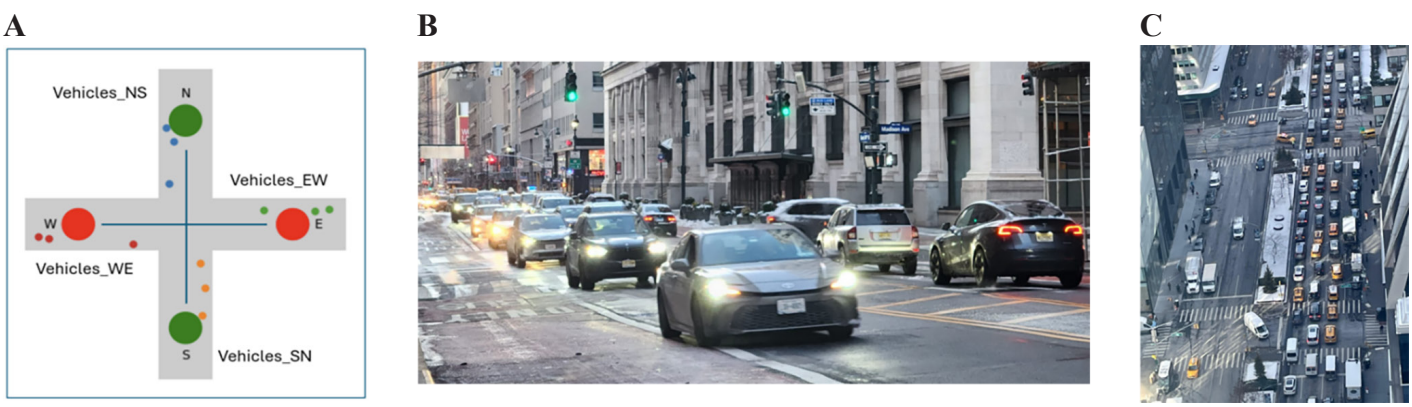



***Figure 3.*** *As shown in (A), the four-direction intersection modeled in this research was used by all controllers: FBC, which used Kaggle data; ABC, which used NYC data; and the DQN and PPO models. The Vehicles_NS, Vehicles_SN, Vehicles_EW, and Vehicles_WE values were used to compute arrival rates per second for each simulation hour. (B) shows the side view, while (C) shows the top view of real-time traffic data captured by Author in New York City.*

alternating schedule was implemented for FBC. It did not adapt to traffic conditions and served as the simplest benchmark.

**ABC using NYC Data**

For the ABC, NYC data was collected on the 4-direction intersection of Park Avenue and 86th Street on Tuesday November 4th, 2025, at the peak time of 9 am Eastern Time. While most corporate offices in NYC operate on a three-days-per-week in-office schedule, Tuesday is typically the busiest commuting day. Therefore, a daytime morning slot was selected to align with peak traffic pattern. The first five minutes of real-time traffic (300 time-steps at 1 second per step) data observed at a NYC intersection data were collected. The data included the number of vehicles arriving across the intersection and their wait-times. The pictures of the data captured are shown in Figure 3. After this initialization period, a structured synthetic timeline of 5,000 time-steps was generated to simulate extended system behavior.

The simulation dataset contained the three main columns. Time_step Ranging from 0 to 5000, queue_NS and queue_EW are simulated queue lengths following predefined rules for stability, fairness, and forward-looking optimization. NS_green as binary indicator, where 0 means NS is green, & 1 means EW is green.

**Rules for ABC**

The ABC was built using rules to ensure three core principles. Stability, fairness, and forward-looking optimization. For stability, the controller enforces a minimum 5-step green to prevent rapid phase toggling that would create confusion and disrupt flow.

For fairness, it enforces a maximum 10-step green duration to prevent any direction from monopolizing the signal. This ensured balanced service so even lower-volume lanes received movement without facing prolonged delays. Emergency vehicles were given the highest priority until they passed the intersection.

For Forward-Looking Optimization, the controller used predictive queue-based logic; when queues tie, it simulated both options and selected the one with the lower expected future wait.

**DRL-Based Controller (DQN)**

In this traffic-light research, the DQN controller learns a value that represents how good a traffic-light action is for reducing future congestion and wait-time. Here, the state includes the current queue length, which direction has the green signal, and how long the light has been in that phase. The actions are simple. Switch the light or stay with the current green. After each action, the agent receives a reward based on how the queues changed. DQN updates its estimates using formula given below.

$$Q_{new}(s, a) = Q(s, a) + \alpha\,[r + \gamma \max_{a'} Q(s', a') - Q(s, a)] \quad (1)$$

$r$ reflects how much the total wait-time improved; $\gamma$ determines how much the agent cares about future congestion, and $\max_{a'} Q(s', a')$ estimates the best future action from the next state.

**DRL-Based Controller (PPO)**

The PPO agent observes queue lengths, current phase, and time-in-phase, and outputs an action (switch or stay). The reward function was designed to reduce overall wait- times.

**Modeling PPO**

PPO Traffic environment consists of a

- **State Vector ($s_t$)** mainly queue length in each direction since last change in the signal.

$$s_t = [q_{NS}, q_{SN}, q_{EW}, q_{WE}, \ phase, \ time\_in\_phase] \quad (2)$$

$q_i$ = *queue length in each direction*;
$i \in \{NS, SN, EW, WE\}$

phase $\in \{0, 1\}$ (0 = *NS Green*, 1 = *EW Green*)

*time_in_phase = duration since last change*

- **Actions** (can be either keep the current signal state or switch.

$a_t \in \{0, 1\}$ (0 = *keep current phase*, 1 = *switch phase*) (3)

- **Rewards ($r_t$)** - The rewards are given based on the total number of vehicles waiting at each duration of time interval. Thus, PPO agent gives higher reward when the current green side has more vehicles waiting than the red side, and lower (even negative) reward when it is serving the "wrong" side.

Current total number of vehicles waiting at time $t + 1$:

$$W_{tot}(t+1) = q_{NS}(t+1) + q_{SN}(t+1) + q_{EW}(t+1) + q_{WE}(t+1) \quad (4)$$

$$W_{tot}(t+1) = W_{NS\_group}(t+1) + W_{EW\_group}(t+1) \quad (5)$$

Cumulative total wait-time:
$TotalWait(t+1) = TotalWait(t) + W_{tot}(t+1)$

The reward at time $t$ in terms of wait-time

$$r_t = \begin{cases} W_{NS\_group}(t+1) - W_{EW_{group}}(t+1), \text{ if } phase_t = 0 \\ W_{EW\_group}(t+1) - W_{NS_{group}}(t+1), \text{ if } phase_t = 1 \end{cases} \quad (6)$$

Where:

$$W_{NS_{group}}(t+1) = q_{NS}(t+1) + q_{SN}(t+1) \quad (7)$$

$$W_{EW_{group}}(t+1) = q_{EW}(t+1) + q_{WE}(t+1) \quad (8)$$

**Evaluation Framework**

Both DRL and baseline controllers were evaluated under identical traffic arrivals by setting the same random seeds. Metrics were aggregated across 20 randomized test runs. The hyper-parameters used for DQN and PPO models are as shown in Table 1.

***Table 1.*** *Summarizes the DQN and PPO hyperparameters used for traffic signal control. Trained in over 200,000-time steps, both agents used a two-action discrete space: keep or switch the signal phase, and a six-dimensional observation space containing queues, phase, and time-in-phase. As shown in (A), DQN used off-policy learning with replay buffer, ε-greedy exploration, and target network updates. PPO, as shown in (B), used on-policy learning with rollout steps, clipping, and generalized advantage estimation. These settings allow both models to be compared under the same traffic environment.*

**A**

| DQN Hyperparameter | Value |
|---|---|
| Algorithm | DQN |
| Policy Network | MLP |
| Hidden Layers | 2 × 64 neurons |
| Activation | ReLU |
| Learning Rate | 0.001 |
| Discount Factor (γ) | 0.99 |
| Replay Buffer Size | 100,000 |
| Batch Size | 64 |
| Learning Starts | 1,000 steps |
| Target Network Update | Every 1,000 steps |
| Training Frequency | Every 4 steps |
| Exploration Strategy | ε-greedy |
| Initial ε | 1 |
| Final ε | 0.05 |
| Epsilon Decay | Linear over first 20,000 steps |
| Optimizer | Adam |
| Gradient Clipping | Max norm = 10 |
| Total Training Timesteps | 200,000 |
| Action Space | Discrete(2): keep or switch |
| Observation Space | 6-dimensional: queues, phase, time-in-phase |

**B**

| PPO Hyperparameter | Value |
|---|---|
| Algorithm | PPO |
| Policy Network | MLP |
| Network Architecture | MLP, 2 hidden layers × 64 neurons |
| Activation Function | Tanh |
| Learning Rate | 0.0003 |
| Discount Factor, γ | 0.99 |
| Batch Size | 64 |
| Rollout Steps, n_steps | 2048 |
| Replay Buffer | Not used; PPO is on-policy |
| Clipping Parameter, clip_range | 0.2 |
| GAE λ | 0.95 |
| Number of Epochs per Update | 10 |
| Entropy Coefficient | 0 |
| Value Function Coefficient | 0.5 |
| Max Gradient Norm | 0.5 |
| Target KL | None |
| Epsilon Schedule | Not used; PPO uses stochastic policy exploration, not ε-greedy |
| Total Training Timesteps | 200,000 |
| Action Space | Discrete(2): keep or switch |
| Observation Space | 6-dimensional: queues, phase, time-in-phase |

## RESULTS & DISCUSSION

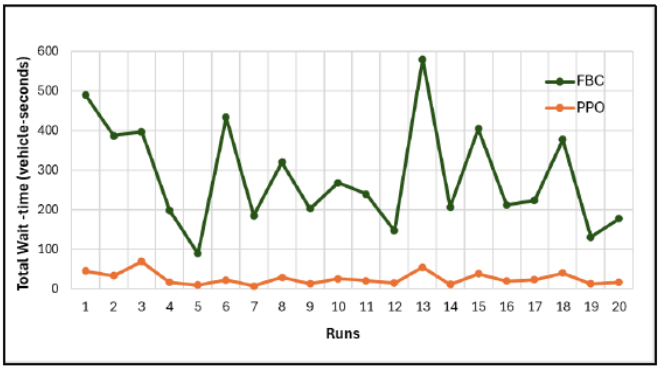
### Testing PPO against FBC

Comparison of PPO Model was performed against FBC. Identical arrival patterns were input to both models from real Kaggle data. Same random seeds and random hour for each run ensured consistency of input data. Each simulation lasted 3,600 seconds, during which total wait-time of the cumulative queued vehicles was measured. FBC switched North-South to East-West every 15 seconds and vice versa. On the other hand, PPO agent decided in real time whether to switch or stay based on observed queues. It consistently achieved much lower wait-times as shown in Figure 4, created using Matplotlib (17). It demonstrated stronger performance over the FBC signal strategy across 20 randomized hour-long tests.

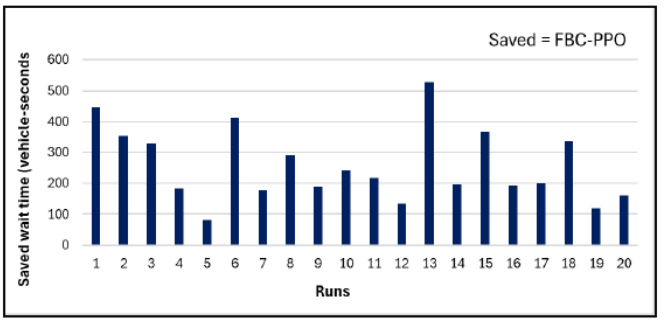
A

| Run | Total Wait Time (Vehicle-Seconds) (FBC) (A) | Total Wait Time (Vehicle-Seconds) (PPO) (B) | Saved Total Wait Time (Vehicle-Seconds) C = (A) - (B) |
|---|---|---|---|
| 1 | 491 | 45 | 446 |
| 2 | 387 | 33 | 354 |
| 3 | 397 | 69 | 328 |
| 4 | 198 | 16 | 182 |
| 5 | 89 | 9 | 80 |
| 6 | 434 | 22 | 412 |
| 7 | 185 | 7 | 178 |
| 8 | 320 | 29 | 291 |
| 9 | 203 | 13 | 190 |
| 10 | 268 | 25 | 243 |
| 11 | 239 | 20 | 219 |
| 12 | 147 | 14 | 133 |
| 13 | 581 | 54 | 527 |
| 14 | 207 | 11 | 196 |
| 15 | 405 | 38 | 367 |
| 16 | 212 | 19 | 193 |
| 17 | 223 | 23 | 200 |
| 18 | 378 | 40 | 338 |
| 19 | 131 | 13 | 118 |
| 20 | 177 | 16 | 161 |
| Average | 283.6 | 25.8 | 257.8 |
| Std Dev | 132.6 | 16.3 | 119.5 |

B




C




D

========== SUMMARY ==========
Average Total Wait-time (FBC): 283.60
Average Total Wait-time (PPO): 25.80
Saved Wait-time (FBC - PPO): 257.80
Saved Wait-time (%): 90.90%
==========

***Figure 4.*** *(A) presents the performance of PPO compared with FBC across 20 simulation runs. The total average wait time for PPO was 25.80 vehicle-seconds, compared with 283.6 vehicle-seconds for FBC. Thus, the average saved wait time was 257.8 vehicle-seconds across the 20 runs, representing a 90.9% improvement. The standard deviation for PPO was 16.3 vehicle-seconds, much lower than FBC's 132.6 vehicle-seconds, indicating that PPO produced more consistent wait-time performance across the simulations, as visually shown in (B). (C) visually shows that PPO reduced wait time in every run, while (D) illustrates the overall 90.9% reduction in wait time compared with FBC.*

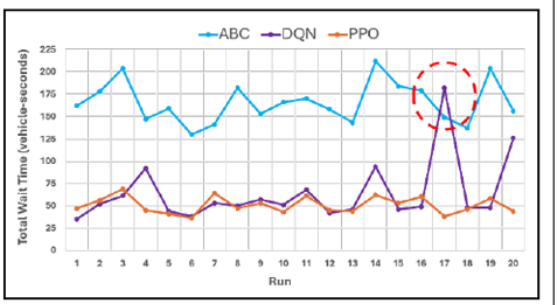
**Testing DQN and PPO against ABC**

DQN and PPO models were tested against ABC using NYC real-time traffic data. As shown in Figure 5, both DRL models outperformed ABC across 20 independent runs, even though ABC incorporated engineered rules, minimum and maximum phase durations, and forward-looking tie-breaking logic. DQN and PPO maintained substantially lower total wait times than ABC in almost every run. PPO delivered the most stable performance, with wait times mostly between 40 and 65 vehicle-seconds, while DQN also performed well but showed greater variation, including one outlier in Run 17 where it underperformed ABC. In contrast, ABC's wait times remained significantly higher, typically between 140 and 200 vehicle-seconds, reflecting its limited ability to adapt to real-time queue fluctuations.

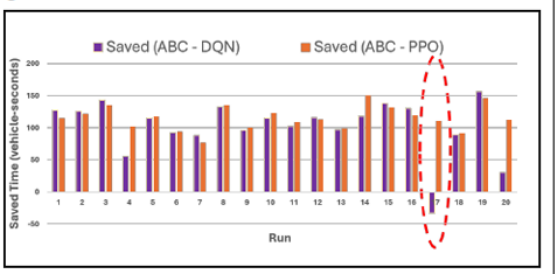
A

| Run | Total Wait Time (Vehicle-Seconds) (ABC) (A) | Total Wait Time (Vehicle-Seconds) (DQN) (B) | Total Wait Time (Vehicle-Seconds) (PPO) (C) | Saved Total Wait Time (Vehicle-Seconds) D=(A)-(B) | Saved Total Wait Time (Vehicle-Seconds) E=(A)-(C) |
|---|---|---|---|---|---|
| 1 | 162 | 35 | 47 | 127 | 115 |
| 2 | 178 | 52 | 56 | 126 | 122 |
| 3 | 204 | 61 | 69 | 143 | 135 |
| 4 | 147 | 92 | 45 | 55 | 102 |
| 5 | 159 | 44 | 41 | 115 | 118 |
| 6 | 130 | 38 | 36 | 92 | 94 |
| 7 | 141 | 53 | 64 | 88 | 77 |
| 8 | 182 | 50 | 47 | 132 | 135 |
| 9 | 153 | 57 | 53 | 96 | 100 |
| 10 | 166 | 51 | 43 | 115 | 123 |
| 11 | 170 | 68 | 61 | 102 | 109 |
| 12 | 158 | 42 | 45 | 116 | 113 |
| 13 | 143 | 46 | 44 | 97 | 99 |
| 14 | 212 | 94 | 62 | 118 | 150 |
| 15 | 184 | 46 | 53 | 138 | 131 |
| 16 | 179 | 49 | 60 | 130 | 119 |
| 17 | 149 | 182 | 38 | -33 | 111 |
| 18 | 137 | 48 | 46 | 89 | 91 |
| 19 | 204 | 48 | 58 | 156 | 146 |
| 20 | 156 | 126 | 44 | 30 | 112 |
| Average | 165.7 | 64.1 | 50.6 | 101.6 | 115.1 |
| Std Dev | 23.2 | 35.5 | 9.3 | 43.4 | 18.6 |

B




C




D

```
======== SUMMARY ==========
Average Total Wait-time (ABC):  165.70
Average Total Wait-time (DQN):   64.10
Saved Wait-time (ABC - DQN):    101.60
Saved Wait-time (%):             61.32%
==================================

==========SUMMARY ==========
Average Total Wait-time (ABC):  165.70
Average Total Wait-time (PPO):   50.60
Saved Wait-time (ABC - PPO):    115.10
Saved Wait-time (%):             69.46%
```

***Figure 5.*** *(A) compares ABC, DQN, and PPO across 20 simulation runs. PPO achieved the lowest average total wait time at 50.60 vehicle-seconds, compared with 64.10 for DQN and 165.70 for ABC. This represents a 69.46% wait-time reduction for PPO and a 61.32% reduction for DQN compared with ABC. As shown in (B), PPO also had the lowest standard deviation, indicating the most consistent performance. (C) compares saved wait time across runs, showing that in Run 17, DQN produced a negative saved wait time, which indicates greater instability compared with PPO. (D) summarizes the overall percentage improvements of DQN and PPO relative to ABC.*

Across all runs, ABC exhibited the highest cumulative delay. DQN achieved a 61.32% reduction in wait time, while PPO performed better with a 69.46% reduction. PPO also showed lower variability, with a standard deviation of 9.3 compared with 35.5 for DQN, indicating greater consistency and robustness. While DQN generally saved wait time compared with ABC, it showed one instance of degradation in Run 17. PPO, however, consistently outperformed ABC in every run. Overall, these results demonstrate that adaptive DRL-based traffic signal control, especially PPO, provides more reliable and significant improvements than traditional analytical strategies under realistic traffic conditions.

**Total Queue Length Heatmaps for DQN & PPO against ABC**

Besides total wait-time, total queue length across the intersection was also tested. The heatmaps shown in Figure 6 compared total queue length across 20 runs for ABC, DQN, and PPO controllers. ABC showed several dark purple regions, with runs exceeding 200 vehicle-seconds, which indicated high congestion and variability. DRL methods significantly reduced queue lengths, but their stability differed. DQN maintained low values in most runs (around 35–68 vehicle-seconds). However, it noted some instability through occasional spikes, such as Run 17 reaching 182. PPO showed consistent results, with all runs close to 36–60 vehicle-seconds. Overall, PPO provides the most reliable congestion reduction, outperforming both ABC and DQN.

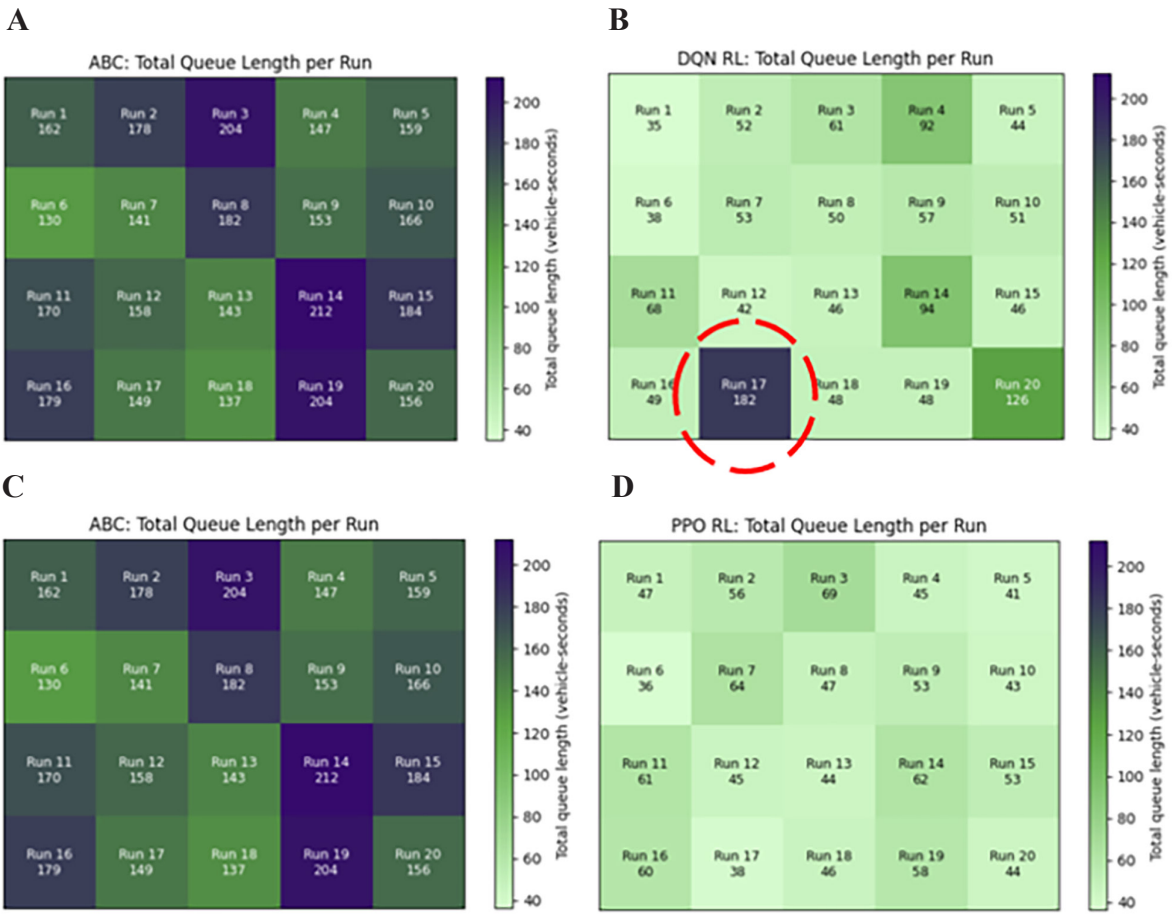



***Figure 6.*** *The heatmaps compare total queue length across 20 runs for the ABC, DQN, and PPO controllers. In (A) and (C), ABC displays several dark regions, with some runs exceeding 200 vehicle-seconds, indicating heavier congestion and higher variability. The DRL-based controllers reduced queue lengths substantially, shown by the lighter green regions. In (B), DQN kept most values low, around 35–68 vehicle-seconds, but showed occasional instability, such as Run 17 reaching 182. In (D), PPO produced the most consistent results, with all runs close to 36–60 vehicle-seconds, demonstrating the strongest and most reliable congestion reduction.*

**Real-life Demonstration**

A demo system was built with a front-end interface using React and TypeScript for interactive traffic simulations as shown in Figure 7. Vite, a Node.js-based build tool, facilitated fast development and bundling. The UI visualized reinforcement learning optimizations (PPO vs FBC) and (ABC vs DQN/PPO), displaying real-time queue lengths, wait times, and emergency vehicle prioritizations. Scalable Vector Graphics (SVG) animations like blinking emergency vehicle, enhanced the simulation experience (18).

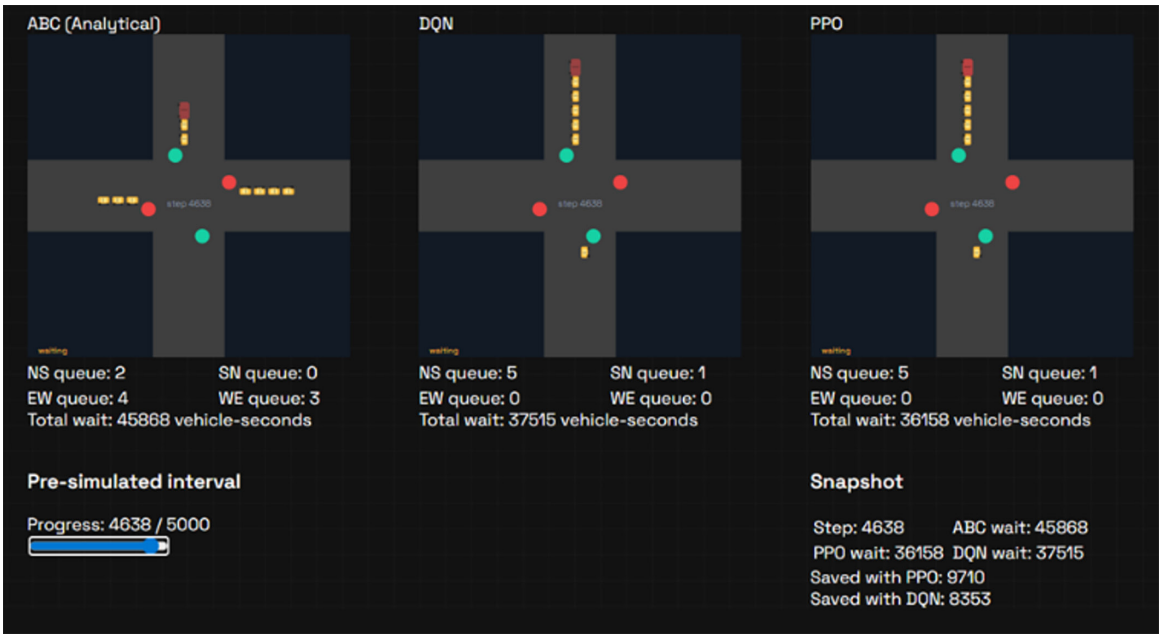



***Figure 7.*** *Simulation demo with sample step = 4638/5000 showing PPO outperforming ABC and DQN by reducing total wait times and minimizing total queue lengths while ensuring emergency vehicles receive the highest priority.*

**Limitations and Potential Error**

The traffic model used a simplified four-way intersection. Hence it may not consider all real-world traffic situations such as pedestrians, bicycles, or lane priorities, etc. The DRL model can be sensitive to hyperparameters, training time, and randomness hence more iterations could be useful. Although the study showed strong results, more detailed datasets across NYC and field testing are needed.

**Future Work**

In future, more work can be done on extending this research from single agent to multi-agent systems for overall traffic management. Future studies could also include: how can PPO's reward function be enhanced to incorporate pedestrian safety, fuel use, and emissions? Real-time traffic data is currently sparse. So live data collection from sensors, cameras, or GPS can be prioritized. The findings can inform policies such as special-purpose lanes, and flyovers in congested areas.

## CONCLUSION

The study shows the effectiveness of PPO based DRL in optimizing signal efficiency unlike traditional methods. Compared to prior studies that largely used simulated data, training and testing using real-time NYC data make the results more credible for real-life implementation. By dynamically learning and adapting to the queue patterns, PPO agent achieved significant reductions in total vehicle wait-time to the extent of 70–90% in comparison to traditional approaches. These results validated that AI-driven traffic control systems can adapt more effectively to real-time traffic conditions and manage congestion better. The research provides a strong foundation for future multi-agent, city-scale traffic optimization systems. It also emphasizes the valuable role DRL can play in shaping the next generation of smart, sustainable, and efficient urban transportation networks.

## ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my Bronx Science Research teacher, Dr. Vladimir Shapovalov, for his invaluable support, guidance, and encouragement throughout this project. I would also like to thank my parents for their continuous support and motivation.

## FUNDING SOURCES

The author declares that no external funding was received for the conduct of this research or the preparation of this article.

## CONFLICT OF INTEREST

The author declares no conflicts of interest related to this work.

## REFERENCES

1. American Lung Association. *New report: Nearly half of people in U.S. exposed to dangerous air pollution levels*. American Lung Association. 2025. https://www.lung.org/media/press-releases/state-of-the-air-2025
2. Schrank D, Lasley P, Albert L, Jha K. Texas A&M Transportation Institute. *Traffic hits record high as commuters rewrite the rush hour.* Texas A&M University. 2025. https://tti.tamu.edu/2025/10/traffic-hits-record-high-as-commuters-rewrite-the-rush-hour/
3. Wei H, Zheng G, Gayah VV & Li Z. *A survey on traffic signal control methods.* arXiv. 2019. https://doi.org/10.48550/arXiv.1904.08117
4. Haydari A and Yilmaz Y. Deep reinforcement learning for intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems (TITS).* 2020. https://arxiv.org/pdf/2005.00935
5. Mnih V, et al. Human-level control through deep reinforcement learning. *Nature*. 2015; 518 (7540): 529–533. doi: 10.1038/nature14236. https://web.stanford.edu/class/psych209/Readings/MnihEtAlHassibis15NatureControlDeepRL.pdf
6. Schulman J, et al. Proximal Policy Optimization Algorithms. arXiv:1707.06347, 2017. https://arxiv.org/pdf/1707.06347
7. Kozlica R, Wegenkittl S and Hirländer S. Deep Q-Learning versus Proximal Policy Optimization: Performance Comparison in a Material Sorting Task. arXiv:2306.01451, 2023. https://arxiv.org/pdf/2306.01451, https://doi.org/10.1109/ISIE51358.2023.10228056
8. Shabestary MA, et al. Traffic signal control system. U.S. Patent 11,783,702 B2, 2023. https://patents.google.com/patent/US11783702B2
9. Gandhi M, Solanki D, Daptardar R & Baloorkar NS. Smart control of traffic light using AI. *IEEE*. 2020. https://doi.org/10.1109/ICRAIE51050.2020.9358334
10. Huang Z. Reinforcement learning based adaptive control method for traffic lights in intelligent transportation. *Alexandria Engineering Journal.* 2024; 106: 381–391. https://doi.org/10.1016/j.aej.2024.07.046
11. Yazdani M, Sarvi M, Asadi Bagloee S, Nassir N, Price J & Parineh H. Intelligent vehicle pedestrian light (IVPL): A deep reinforcement learning approach for traffic signal control. *Transportation Research Part C: Emerging Technologies*. 2023; 149: 103991. https://doi.org/10.1016/j.trc.2022.103991
12. Zhu M, Liu X-Y, Borst S & Walid A. Deep Reinforcement Learning for Traffic Light Control in Intelligent Transportation Systems. *IEEE*. 2023. https://arxiv.org/pdf/2302.03669
13. Wu C, Kim I & Ma Z. Deep Reinforcement Learning Based Traffic Signal Control: A Comparative Analysis. *Procedia Computer Science.* 2023; 220: 275–282. https://doi.org/10.1016/j.procs.2023.03.036
14. Gymnasium. (n.d) gymnasium is an open-source Python library used as a standard API for developing, comparing, and testing reinforcement learning (RL) algorithms. Towers M, *et al*., "Gymnasium: A Standard Interface for Reinforcement Learning Environments," arXiv preprint arXiv:2407.17032, 2024. https://gymnasium.farama.org/
15. Stable Baselines3. (n.d) stable_baseline3 is a set of reliable implementations of reinforcement learning algorithms in PyTorch. A. Raffin et al., Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*. 2021; 22: 1–8. https://stable-baselines3.readthedocs.io/
16. Soriano F. *Traffic Prediction Dataset*. Kaggle. 2021. https://www.kaggle.com/datasets/fedesoriano/traffic-prediction-dataset/data?select=traffic.csv
17. Matplotlib. (n.d.) matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Hunter JD. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering.* 2007; 9 (3): 90–95. https://matplotlib.org/, https://doi.org/10.1109/MCSE.2007.55
18. Guo M, Wang P, Chan C-Y and Askary S. A reinforcement learning approach for intelligent traffic signal control at urban intersections. arXiv preprint arXiv:1905.07698, 2019. https://arxiv.org/pdf/1905.07698, https://doi.org/10.1109/ITSC.2019.8917268