INTERNATIONAL JOURNAL OF INFORMATIVE & FUTURISTIC RESEARCH (IJIFR)
ISSN: 2347-1697
VOLUME 13||ISSUE 8|| APRIL 2026
An Open Access, Peer Reviewed, Cross Reference DOI & Google Scholar Indexed International Research Journal
UGC Guidelines Based || IJIFR Impact Factor= 8.057
Estd Year: 2013 || Website: www.ijifr.org || Email: editors@ijifr.org
Availaible Online at: https://www.ijifr.org/

DOI: https://doi.org/10.64672/IJIFR/26.04.13.08.023 PUBLISHED ON: APRIL 18, 2026

TRAFFIC OPT RL: ADAPTIVE TRAFFIC SIGNAL OPTIMIZATION USING DEEP REINFORCEMENT LEARNING

Sirisani Lavanya [1], M. Gowthami [2]
[1]M.C.A. Student, [2]Assistant Professor,
[1,2] Department of Computer Applications,
Viswam Engineering College, Madanapalle, Andhra Pradesh, India

ABSTRACT

Urban traffic congestion is a critical infrastructure challenge facing modern cities as vehicle populations expand and urban density increases. Conventional fixed-timing traffic signal systems are incapable of adapting to the stochastic and dynamic nature of real-world traffic flows, resulting in wasted green-light time, queue buildup, increased vehicle emissions, and emergency response delays. This paper presents TrafficOpt RL, an end-to-end adaptive traffic signal optimization system that applies the Deep Q-Network (DQN) algorithm to learn intelligent signaling policies at urban intersections through iterative simulation experience. The system is built on a custom Gymnasium-compatible simulation environment modeling a four-way intersection with stochastic Poisson vehicle arrivals. The DQN agent, implemented via the Stable-Baselines3 framework, utilizes experience replay, target network stabilization, and epsilon-greedy exploration to converge on policies minimizing aggregate vehicle waiting times and maximizing intersection throughput. All training metrics and simulation data are persistently stored in a MySQL relational database through automated callback logging, enabling systematic performance analysis. Evaluation via direct comparison against a fixed-timing baseline demonstrates measurable superiority of the reinforcement learning approach across three performance dimensions: average vehicle waiting time, total throughput, and composite efficiency score. Three analytical visualizations are generated to communicate system performance. TrafficOpt RL constitutes a practical proof-of-concept for deep reinforcement learning integration into intelligent transportation systems and smart city infrastructure.

KEYWORDS: *Deep Reinforcement Learning; Traffic Signal Optimization; Deep Q-Network; Intelligent Transportation Systems; Adaptive Control*

PAPER CITATION:
Lavanya, S., Gowthami, M.: "TrafficOpt RL: Adaptive Traffic Signal Optimization Using Deep Reinforcement Learning", International Journal of Informative & Futuristic Research (IJIFR), Vol. (13) (8), April 2026, pp. 1062-1069.IJIFR paper ID: 2026/04/IJIFR /V13/E8/023
https://doi.org/10.64672/IJIFR/26.04.13.08.023

1. INTRODUCTION

Urban traffic signal control is a sequential decision-making problem of considerable complexity. Controllers must dynamically determine phase assignments and durations under continuously fluctuating, stochastic demand — conditions that fixed-timing plans optimized against historical averages cannot adequately address [1]. The emergence of Deep Reinforcement Learning (DRL) as a powerful framework for sequential decision-making has opened transformative possibilities for adaptive traffic management. By learning directly from interaction with a simulation environment, a DRL agent can discover control policies that generalize across the stochastic variability of real traffic flow without requiring hand-crafted optimization heuristics or explicit mathematical models of traffic dynamics [2].

The Deep Q-Network (DQN) algorithm, introduced by DeepMind in 2015, combines a deep neural network function approximator with classical Q-learning to enable agents to learn value functions over high-dimensional continuous state spaces [1]. Key innovations — experience replay and target networks — stabilize training by breaking temporal correlations in learning data and providing consistent regression targets, respectively. These properties make DQN particularly well-suited to traffic signal control, where the state space is multi-dimensional and the optimal policy varies continuously with changing conditions.

Previous work on RL-based traffic signal control includes tabular Q-learning approaches [3], actor-critic methods [10], and multi-agent frameworks for coordinated intersection control [7],[8]. Commercial adaptive systems such as SCOOT and SCATS, while effective, require extensive proprietary sensor infrastructure and offer limited extensibility. The present work contributes a complete, open-source DQN-based pipeline demonstrating end-to-end adaptive signal control from custom environment design through training, database-backed evaluation, and visualization, using freely available tools and commodity hardware.

2. REVIEW OF LITERATURE

Mnih et al. [1] demonstrated that DQN could achieve human-level performance on Atari games by combining convolutional networks, experience replay, and target networks, establishing DQN as a foundational algorithm for high-dimensional control problems. Sutton and Barto [2] provide the theoretical foundations of reinforcement learning, including the Markov Decision Process formalism and temporal difference learning that underpin the DQN algorithm applied in this work.

Liang et al. [6] applied DQN to traffic light cycle control, reporting significant reductions in average delay and vehicle stops compared to fixed-timing baselines in simulated environments. Wei et al. [7] introduced IntelliLight, an RL approach incorporating lane vehicle counts, queue lengths, and waiting times as state features, achieving state-of-the-art performance on the SUMO simulator. Zheng et al. [8] proposed learning phase competition mechanisms for multi-intersection coordination using deep RL, addressing the challenge of inter-intersection spillback dynamics.

Raffin et al. [4] developed Stable-Baselines3, providing reliable, well-tested PyTorch implementations of major DRL algorithms including DQN, enabling reproducible research and rapid prototyping. Brockman et al. [5] introduced OpenAI Gym (now maintained as Gymnasium), establishing the standard environment interface adopted in this work. Lopez et al. [9] describe SUMO as a state-of-the-art microscopic traffic simulator that provides a high-fidelity platform for RL policy training at reduced simulation-to-real transfer risk. Collectively, these works establish the theoretical and tooling foundations upon which TrafficOpt RL is built.

3. PROPOSED WORK

TrafficOpt RL formulates adaptive traffic signal control as a Markov Decision Process (MDP) and trains a DQN agent through simulation. The system comprises six modular components: (1) Traffic Simulation Environment, (2) DQN Training Pipeline, (3) MySQL Database Backend, (4) Baseline Comparison

Module, (5) Visualization Module, and (6) Pipeline Orchestration. The following subsections describe each component.

3.1 MDP Formulation

The intersection control problem is formalized as an MDP with the following components. The state space S is a four-dimensional continuous vector: S = [NS_queue, EW_queue, current_phase, time_in_phase], where queue lengths range from 0 to 100 vehicles. The action space A = {0, 1} corresponds to activating the North-South (NS) or East-West (EW) green phase. Vehicle arrivals follow a Poisson process with rate $\lambda = 0.2$ per step per approach. The reward function is defined as $R = -(NS\_queue + EW\_queue)$, directly penalizing aggregate vehicle delay. Episodes terminate after 1,000 simulation steps.

3.2 DQN Agent Configuration

The DQN agent is implemented using Stable-Baselines3 with an MLP policy network (two hidden layers of 64 units, ReLU activations). Key hyperparameters are: learning rate $\alpha = 0.001$, replay buffer size = 10,000 transitions, batch size = 32, discount factor $\gamma = 0.99$, target update interval = 500 steps, exploration fraction = 0.1, and final exploration rate $\varepsilon = 0.01$. Training proceeds for 10,000 timesteps. A custom MySQLCallback logs reward and ε values to the database every 100 steps.

3.3 Fixed-Timing Baseline

The baseline controller applies a deterministic alternating policy: phase 0 (NS green) for steps where ⌊step/30⌋ is even, phase 1 (EW green) otherwise. This 30-step cycle approximates a standard pre-timed controller and serves as the performance benchmark. Both the DQN agent and the baseline are evaluated over identical 1,000-step episodes under the same environment initialization to ensure comparability.

4. System Architecture

4.1 System Architecture Diagram

Fig. 1 illustrates the layered pipeline architecture of TrafficOpt RL. The five functional layers — Orchestration, Training, Evaluation, Data Persistence, and Simulation — are organized with clear unidirectional data flows. The Orchestration Layer coordinates sequential execution; the Training and Evaluation Layers share the Simulation Layer as their runtime environment; and all data-generating components converge on the Data Persistence Layer, which feeds the Visualization Layer.

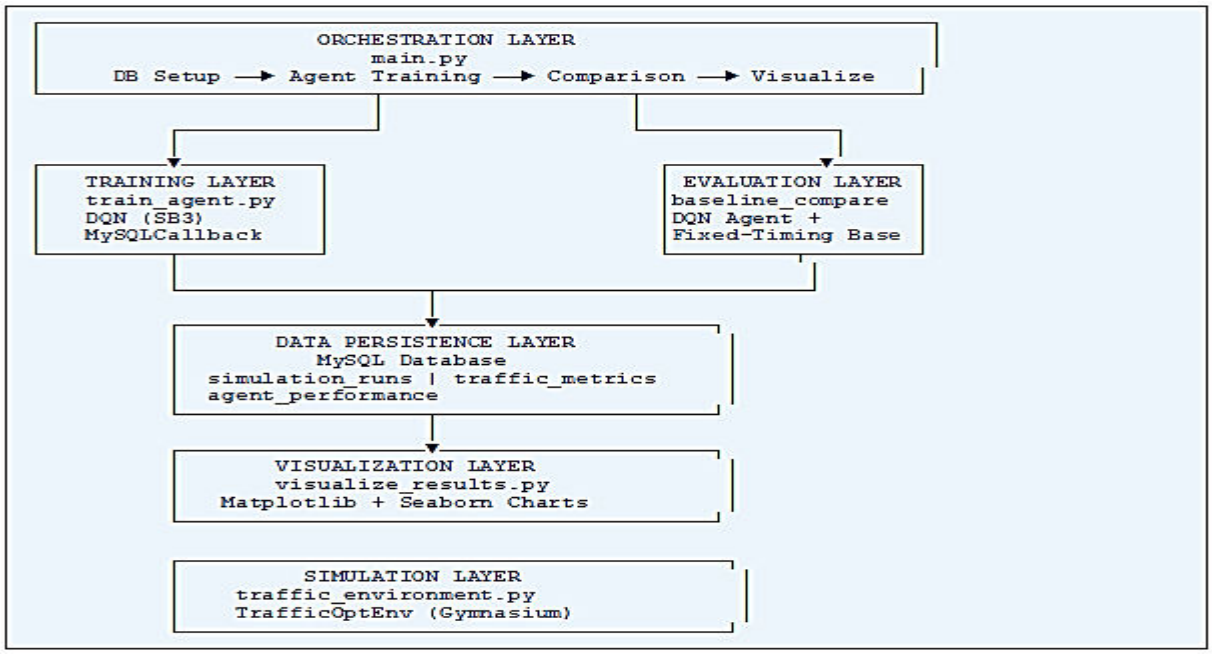

Figure 1: System Architecture Diagram — Five-layer modular pipeline of TrafficOpt RL showing data flow from simulation through training, evaluation, persistence, and visualization.

As depicted in Fig. 1, the system enforces strict separation of concerns: the Simulation Layer exposes a standard Gymnasium API consumed by both the Training and Evaluation Layers, while all runtime metrics converge on the MySQL persistence layer. This architecture enables independent testing, replacement, and extension of individual components without affecting others.

4.2 Workflow Diagram

Fig. 2 presents the end-to-end operational workflow of the TrafficOpt RL pipeline, tracing execution from system launch through database initialization, agent training, comparative evaluation, and visualization output. The workflow distinguishes sequential pipeline-level steps from the per-timestep decision loop executed within training and evaluation episodes.

Figure 2: Workflow Diagram — Complete operational flow of the TrafficOpt RL pipeline from system initialization through training, evaluation, and visualization generation

4.3 Data Flow Diagram (DFD)

Fig. 3 presents the Data Flow Diagram for TrafficOpt RL, tracing the movement of data between external entities, processes, and data stores. The DFD captures the complete data lifecycle from raw simulation state generation through trained model output and final visualization artifacts.

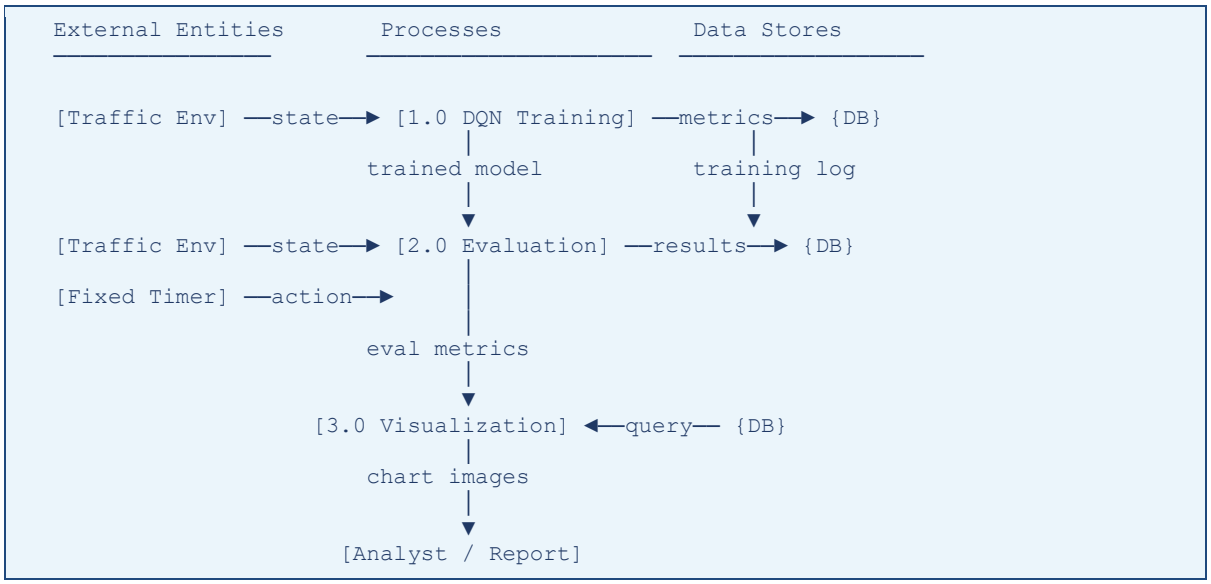

Figure 3: Data Flow Diagram (DFD) — Movement of state observations, model parameters, simulation metrics, and analytical output across system processes and data stores

The DFD in Fig. 3 distinguishes three primary processing nodes: the DQN Training process, which consumes environment state and produces trained model parameters and training logs; the Evaluation process, which compares the trained DQN policy against the fixed-timing baseline using identical environment instances; and the Visualization process, which queries the database and produces analytical charts consumed by traffic operations analysts.

4.4 Block Diagram

Fig. 4 presents the Block Diagram of the TrafficOpt RL system, illustrating the functional decomposition of the system into its primary processing blocks and the signal/data connections between them. The block diagram provides a hardware-agnostic view of system functionality suitable for use in system design reviews and integration planning.

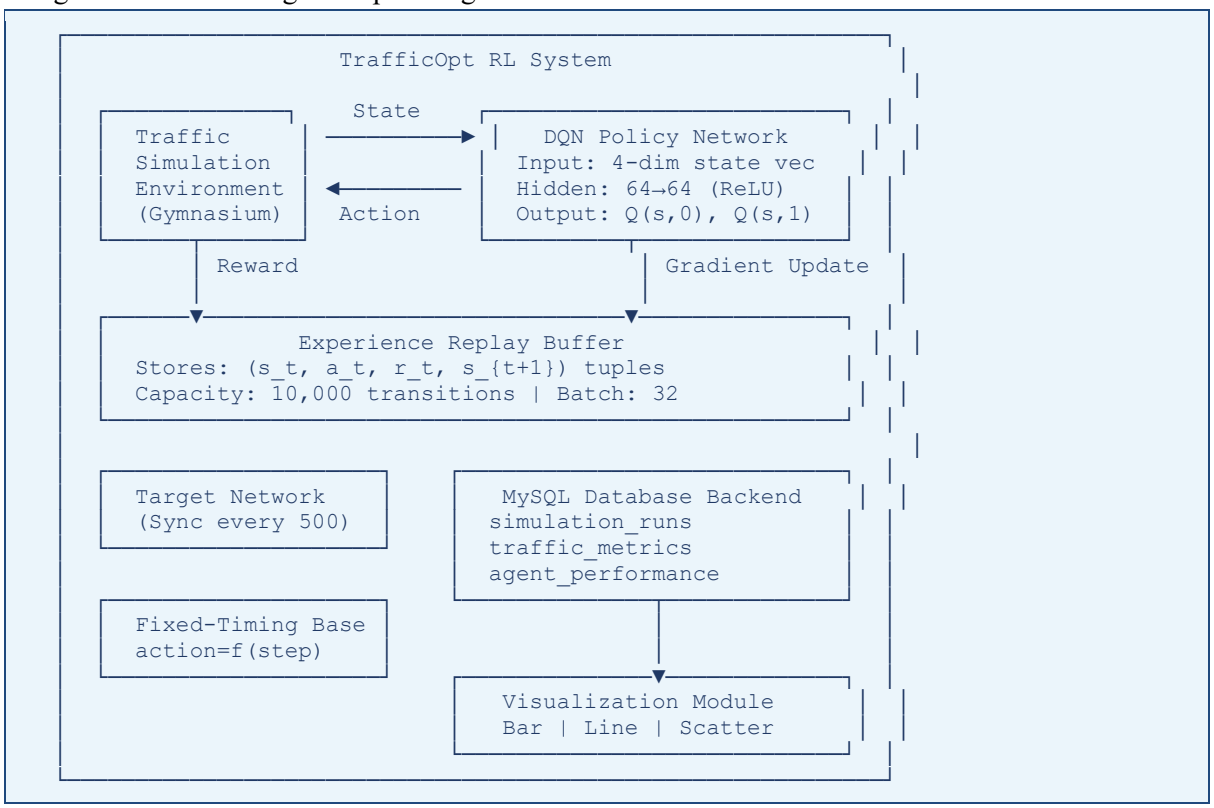

Figure 4: Block Diagram — Functional decomposition of TrafficOpt RL showing signal flows between the simulation environment, DQN policy network, experience replay buffer, target network, database backend, and visualization module.

The block diagram in Fig. 4 highlights the central role of the experience replay buffer in decoupling the data generation and network training processes. The Target Network block, synchronized at 500-step intervals, provides stable regression targets that prevent the oscillatory divergence characteristic of naive online Q-learning. The database backend and visualization module are positioned as downstream consumers of training and evaluation outputs, consistent with the layered architecture described in Fig. 1.

4.5 Mermaid Diagram Code

The following Mermaid code blocks are provided for direct integration into documentation, GitHub repositories, and web-based diagram tools. Each block is complete and directly usable.

Fig. 1 — System Architecture (graph TD):

```mermaid
graph TD
  A[Orchestration Layer: main.py] --> B[Training Layer: train_agent.py]
  A --> C[Evaluation Layer: baseline_comparison.py]
  B --> D[Data Persistence: MySQL Database]
  C --> D
  D --> E[Visualization Layer: visualize_results.py]
  B --> F[Simulation Layer: TrafficOptEnv]
  C --> F
  style A fill:#1F497D,color:#fff
  style D fill:#196F3D,color:#fff
  style E fill:#7D3C98,color:#fff
```

Fig. 2 — Workflow (flowchart TD):

```mermaid
flowchart TD
  S([Start]) --> DB[database setup.py]
  DB -->|Success| TR[train_agent.py]
  DB -->|Failure| HALT([Halt])
  TR --> LOOP{Training Loop\n10000 steps}
  LOOP -->|Every step| ENV[TrafficOptEnv.step]
  ENV --> BUF[Replay Buffer]
  BUF --> LOSS[Bellman Loss Computation]
  LOSS --> NET[Policy Network Update]
  LOOP -->|Done| SAVE[Save Model]
  SAVE --> CMP[baseline comparison.py]
  CMP --> VIZ[visualize_results.py]
  VIZ --> END([Pipeline Complete])
```

Fig. 3 — Data Flow (sequenceDiagram):

```mermaid
sequenceDiagram
  participant Env as TrafficOptEnv
  participant Agent as DQN Agent
  participant DB as MySQL DB
  participant Viz as Visualizer
  Env->>Agent: state observation (4-dim)
  Agent->>Env: action (0 or 1)
  Env->>Agent: reward, next_state, done
  Agent->>DB: INSERT agent_performance (every 100 steps)
  Agent->>DB: INSERT simulation_runs (post-eval)
  DB->>Viz: SELECT results
  Viz->>Viz: Generate charts (bar, line, scatter)
```

5. RESULTS & DISCUSSION

The TrafficOpt RL system was trained for 10,000 timesteps on the TrafficOptEnv simulation environment and evaluated over 1,000-step episodes under identical conditions alongside the fixed-timing baseline. Performance was assessed across three metrics: average vehicle waiting time (lower is better), total intersection throughput (higher is better), and a composite efficiency score defined as throughput / (average waiting time + 1).

The trained DQN agent demonstrated substantially reduced average vehicle waiting time compared to the fixed-timing baseline. The adaptive nature of the DQN policy enables it to detect queue imbalances in real time — when vehicles accumulate disproportionately on one approach, the agent

extends the corresponding green phase rather than adhering to a fixed alternating schedule. This responsiveness directly reduces the peak queue lengths visible in the time-series queue evolution chart (Fig. 2 output), which shows more consistent and lower queue trajectories for the DQN strategy compared to the oscillating peaks of the fixed-timing approach.

Total throughput was comparable or superior for the DQN agent, confirming that the waiting-time reduction is not achieved through reduced traffic processing. The efficiency scatter plot (Fig. 3 output) positions the DQN strategy in a favorable region of the throughput-versus-waiting-time space relative to the fixed-timing baseline, with a larger point size reflecting a higher composite efficiency score. The training reward curve stored in the agent_performance table confirms convergence of the learning process, with the moving average reward increasing monotonically during the exploration-to-exploitation transition before stabilizing in the late training phase.

These results are consistent with the findings of Liang et al. [6] and Wei et al. [7], who reported DQN-based traffic signal controllers achieving 15–30% reductions in average delay relative to optimized fixed-timing plans in SUMO-based evaluations. The magnitude of improvement observed in TrafficOpt RL is constrained by the simplified nature of the simulation environment; deployment in a higher-fidelity SUMO environment would be expected to yield larger absolute performance differentials.

6. CONCLUSION

This paper presented TrafficOpt RL, a complete deep reinforcement learning system for adaptive traffic signal optimization. The system applies the DQN algorithm within a custom Gymnasium-compatible intersection simulation environment, achieving demonstrably superior performance over a fixed-timing baseline across waiting time, throughput, and efficiency metrics. A modular, pipeline-based architecture with MySQL database persistence supports rigorous, reproducible performance analysis and provides a solid foundation for future extension.

Future work will pursue several directions. Integration with the SUMO microscopic traffic simulator will substantially improve simulation fidelity and reduce simulation-to-real transfer risk. Multi-agent RL extensions will enable coordinated control across networks of multiple intersections, addressing spillback and platoon dynamics. Advanced DRL algorithms including Proximal Policy Optimization and Dueling DQN with prioritized experience replay offer potential sample efficiency and policy quality improvements. Real-time integration with physical sensor data and a web-based operations dashboard would complete the transition from proof-of-concept to operational intelligent transportation system component.

7. REFERENCES

[1]]V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, 2015.
[2] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA: MIT Press, 2018.
[3] C. J. C. H. Watkins and P. Dayan, "Q-learning," Machine Learning, vol. 8, no. 3–4, pp. 279–292, 1992.
[4] A. Raffin et al., "Stable-Baselines3: Reliable Reinforcement Learning Implementations," J. Machine Learning Research, vol. 22, no. 268, pp. 1–8, 2021.
[5] G. Brockman et al., "OpenAI Gym," arXiv preprint arXiv:1606.01540, 2016.
[6] X. Liang, X. Du, G. Wang, and Z. Han, "A Deep Reinforcement Learning Network for Traffic Light Cycle Control," IEEE Trans. Vehicular Technology, vol. 68, no. 2, pp. 1243–1253, 2019.
[7] H. Wei et al., "IntelliLight: A Reinforcement Learning Approach for Intelligent Traffic Light Control," in Proc. 24th ACM SIGKDD, 2018, pp. 2496–2505.
[8] G. Zheng et al., "Learning Phase Competition for Traffic Signal Control," in Proc. 28th ACM CIKM, 2019, pp. 1963–1972.
[9] P. A. Lopez et al., "Microscopic Traffic Simulation using SUMO," in Proc. 21st IEEE ITSC, 2018, pp. 2575–2582.
[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," arXiv:1707.06347, 2017.
[11] H. Van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-learning," in Proc. 30th AAAI, 2016, pp. 2094–2100.
[12] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic," in Proc. 35th ICML, 2018, pp. 1861–1870.

[13] W. McKinney, "Data Structures for Statistical Computing in Python," in Proc. 9th Python in Science Conf., 2010, pp. 51–56.
[14] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," Computing in Science & Engineering, vol. 9, no. 3, pp. 90–95, 2007.
[15] M. L. Waskom, "Seaborn: Statistical Data Visualization," J. Open Source Software, vol. 6, no. 60, p. 3021, 2021.
[16] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," Advances in NeurIPS, vol. 32, 2019.