Multi-Sensor Deep Q-Network (MSF-DQN) Real Time Adaptive Traffic Signal Control

Shravani Sharad Patil[1], Gayatri Vijay Chaudhari[2], Khushee Bhanudas Bagal[3], Spandan Pravin Patil[4]
[1,2,3,4]R C Patel Institute of Technology, Shirpur, Maharashtra, India

Abstract—**The Environmental and Economic Consequences of Urban Traffic Congestion Are a Significant Problem for Cities, And Traditional Fixed-Time and Actuated Controllers Only Make These Problems Worse.1 In This Paper, An Innovative Adaptive Traffic Signal Control System Based on A Multi-Sensor Fusion Deep Q Network (MSF-DQN) Called the Automatic Traffic Signal Information System (ATSIS) Is Introduced. The Suggested MSF-DQN Agent Has Stronger State Representation Than Any Previous Work Due To The Way It Combines Data From Two Different Sensing Modules: (1) A Vision-Based Mod- ule That Uses YOLOv4 For Vehicle Detection And Real-Time Queue Length Estimation, And (2) A High-Reliability Presence Detection Module Using Ultrasonic And Infrared Sensors At The Stop Line To Confirm Vehicles; These Types Of Combinations Provide Strengths That Are Very Important When Dealing With The Diversity And Non-Lane Based Nature Of India's Traffic Conditions. The Performance of the Proposed MSF-DQN Model Is Rigorously Tested in A Variety of Traffic Scenarios Using the SUMO (Simulation of Urban Mobility) Microscopic Traffic Simulation Environment Low Volume, Dynamic Peak Hour, And Full Conditions Compared to Standard Fixed-Time (FT) And Actuated Control (AC) Systems. The Results Show That the Proposed MSF-DQN Significantly Reduces the Average Delay of Vehicles and The Average Queue Length, Especially Under Busy and Changing Conditions. Therefore, This Shows That the DRL Agent Works as Expected and The Multi Sensor Fusion Method Used Is Effective.**

Index Terms—**Sensor Fusion, Deep Q-Network (DQN), Adap- tive Traffic Signal Control, Intelligent Transportation Systems (ITS), YOLO, Indian Traffic, SUMO.**

I. INTRODUCTION

A. Background and Reasons

Urban areas have grown rapidly and placed a significant amount of pressure on infrastructure, with traffic congestion being the most significant issue. Traffic congestion is an economic, environmental, and social problem that is partic- ularly severe in India. The cost of congestion in four major Indian cities Delhi, Mumbai, Bengaluru, and Kolkata is approximately Rs. 1.47 lakh crore per year according to a study conducted in 2018. Recently, data released in 2024 indicates that drivers in cities such as Bengaluru and Pune lose over 110 hours annually during rush hour.

The traffic congestion caused by rapid urbanization, poor road conditions, and a large number of vehicles is also responsible for the waste of billions of liters of fuel, increased emissions of harmful pollutants, reduced national productivity, and lower quality of life for those who travel within cities.

B. Problem Statement

One major problem associated with the current state of traffic control is that the majority of the old, rigid traffic control systems do not function well. Two primary concepts currently used to implement signalized intersections include:

1) Fixed-Time (FT) Controllers: FT controllers operate using fixed timing schedules based upon historical traffic data [3]. They are simple to utilize and reliable, but are not capable of addressing the changing and unpredictable nature of modern traffic flow. As a result of their cyclic operation, they are very inefficient. For example, they can be wasteful by leaving cars waiting at empty intersections, and create gridlock when they cannot respond to an unexpected surge in demand.
2) Actuated Control (AC) Systems: AC systems are im- proved over FT systems in that they utilize sensor technology (such as inductive loop detectors) to determine whether a vehicle is present at the stop line and thus can initiate or extend a

green phase. However, AC systems inherently react myopically and respond quickly. Traditional systems cannot see how long the queues of vehicles are and whether there are large platoons of cars approaching the intersection. Due to this limited awareness, they make short sighted and greedy decisions, which severely restrict the robot's capabilities in highly dynamic environments.

C. The Rise of Smart Systems

There is a need to develop better traffic management sys- tems due to the clear limitations of the conventional methods. In response, researchers have turned to Deep Reinforcement
Learning (DRL) as an advanced and real time solution for managing the intricacies and unpredictability of real-world traffic environments [5].

D. The Gap in Research and Proposed Contribution

The performance of a DRL agent is entirely contingent upon the quality and reliability of its" state" input. The literature indicates a significant trend towards employing vision-based sensing (such as cameras) to acquire the extensive, high- dimensional data (including vehicle counts and queue lengths) requisite for DRL models. 9 But vision-only systems are very sensitive to bad weather (like fog, heavy rain, snow, and glare at night) and things that get in the way, which can mess up the state input and cause the control policy to fail in a big way. Infrared (IR) and ultrasonic detectors are two examples of simpler sensors that are very reliable for binary" presence" detection at a low cost. But they don't have the data richness that a smart DRL agent needs to learn, like being able to tell the difference between a queue of one car and a queue of twenty cars. This shows a big gap in research: we need a state representation that can hold a lot of data and deal with mistakes. The user's first idea for a project, which uniquely lists cameras, ultrasonic, and IR sensors as a combined system [1], suggests a synergistic solution. If the camera breaks, a system that only uses vision will fail. If the system only uses IR sensors, it can't do advanced optimization. This paper presents the Automatic Traffic Signal Information System (ATSIS) [1], an innovative variant of the Multi-Sensor Fusion Deep Q-Network (MSF-DQN). Our system directly fills the gap that was found by: Using a cutting-edge You Only Look Once (YOLOv4) module to find cars in real time and guess how long a line is based on a camera feed [22]. Put a high-reliability presence detection module at the stop line of each lane. This module should have both IR and ultrasonic sensors that are close to each other. A new fusion algorithm that cleverly combines these two streams of data. The algorithm checks the vision data and, more importantly, fixes it using the very reliable presence sensors. This lowers the number of false negatives and gives the DRL agent a strong and fault-tolerant state representation.

E. Paper Structure

The rest of this paper is laid out like this: Section II examines pertinent research on traffic control strategies, sensor technologies, and DRL applications. In Section III, we talk about the new way to combine sensors and the suggested MSF-DQN architecture. In Section IV, we talk about the experimental setup, the different scenarios used in the tests, and the baseline models used for testing. In Section V, we compare our model's results to those of other models. Finally, Section VI ends the paper and talks about some interesting things that could be studied in the future.

II. LITERATURE REVIEW

A. Evolution of Traffic Signal Control Strategies

There have been many changes to traffic signal control over time. The first people to use static timing plans were Fixed- Time (FT) controllers. These plans are often based on past flow data and use methods like Webster's formula. This was the beginning of modern control. The biggest problem with them is that they can't adapt to changes in traffic right away. This led to the development of Actuated Control (AC) systems, which use sensors (like inductive loops) to find vehicles and make simple, rule-based decisions, such as extending a green phase [3]. The next generation was Adaptive Traffic Control Systems (ATCS). They made the network level more coordinated. SCATS (Sydney Coordinated Adaptive Traffic System) and SCOOT (Split, Cycle, and Offset Optimization Technique) were the most advanced systems. Based on data from detectors in real time, these systems change the timing of signals (split, cycle, and offset). But they depend a lot on traffic models made by hand, complicated calibration, and heuristicbased logic, which often limits how much they can change [11]. These systems are not just ideas; SCATS, SCOOT, and the Indian-made CoSiCoSt (Composite Signal Control Strategy) have all been used in big Indian cities like Delhi, Mumbai, and Jaipur.

B. Sensor Technologies for Traffic State Estimation

The performance of any adaptive system depends on how well it can judge the state of the traffic.

1) Vision-Based Systems (VIP): Cameras are becoming more and more common as sensors because they give us a lot of high-dimensional data [13]. Deep Learning (DL) has completely changed the field since the first systems used traditional image processing. Convolutional Neural Networks (CNNs), like YOLO (You Only Look Once), are now very good at finding things in real time and across many classes. Researchers have found that a YOLOv4 model can be changed to accurately (83–93%) guess how long a queue is based on the number of cars, even when the traffic footage is low-resolution [9]. People are also looking into infrared (IR) and ultrasonic sensors as two cheap options. A lot of people use IR sensors to see if something is at stop-lines [23]. The HC-SR04 is an example of an ultrasonic sensor that can also tell how far away and how many cars there are [25]. The main problem with these sensors is that they only give output in low dimensions. They can tell if a car is there, but they can’t easily tell how long the line is [23].
2) Sensor Fusion: The literature increasingly concurs that no singular sensor can provide both comprehensive data and flawless reliability [14]. This has caused more research to be done on sensor fusion. Advanced systems are investigating the integration of various data sources, including radar, LiDAR, and video [27], geomagnetic sensors and radar [28], or data from Connected Vehicles (CVs) [20]. Our research contributes to this trend by proposing an innovative and effective method for integrating vision-based data with high-reliability proxim- ity sensors [1].

C. Deep Reinforcement Learning (DRL) for Adaptive Control

DRL is a big change in how traffic is managed. Traditional ATCS (SCATS/SCOOT) systems rely on model-based heuris- tics that require a lot of engineering [15]. On the other hand, DRL agents are data-driven and don’t use models. They learn the best, and sometimes not obvious, control policies” from scratch” by interacting with the traffic environment and getting a reward signal [16].

The Deep Q-Network (DQN) is a basic DRL algorithm for this job [3]. It changed the field by using Q-learning and a deep neural network to get close to the action-value function (Q (s, a)). Experience replays and target networks [3] were two key ways that made it stable.

For a DRL system to work well, the Markov Decision Process (MDP) parts must be designed well:

- State Representation: The state s_t must concisely sum- marize the traffic situation. The literature has evolved from fundamental queue length vectors [17] to complex matrix or grid-based representations [29], Graph Neural Networks (GNNs) for modelling spatial dependencies [30], and the amalgamation of data from Connected and Automated Vehicles (CAVs) and pedestrians [21].
- Action Space: This tells you what the agent is allowed to do. Some common designs are discrete phase selection (picking the next signal phase) [29], continuous duration control (setting the green time for a phase within a fixed cycle) [31], and hybrid approaches that pick both the phase and its duration at the same time [32].
- Reward Function: The reward r_t helps the agent learn. The goal is usually to cut down on traffic, so the reward is often seen as a punishment. Some bad rewards are the total amount of time spent waiting, the total amount of time spent in traffic, or the longest line [7].

D. Recognized Research Gaps

There are three main problems with using DRL for traffic control that this review talks about:

- Robustness and Reliability: DRL models are” brittle,” which means they don’t work well when sensor data is missing or noisy [1]. A DRL agent trained on perfect simulation data may fail in unpredictable ways when it comes across real-world sensor problems [18].
- Scalability: There is a lot of research on DRL for a single intersection, but making it work for a big, coordinated network (Multi-Agent Reinforcement

Learning, MARL) is a very hard and active area of study [6].

- Explainability (XAI): Everyone knows that DRL agents are" black boxes." Because traffic engineers and city officials can't easily trust, debug, or certify a system whose decision-making process they can't understand [4], it's hard to put the system into use.

This paper directly addresses the initial gap (Robustness) by presenting an innovative sensor fusion methodology. It also sees the second and third gaps as important areas for more research in the future.

III. METHODOLOGY AND SYSTEM ARCHITECTURE

A. System Overview (The ATSIS Framework)

The suggested Automatic Traffic Signal Information System (ATSIS) [1] is a smart control system that works in a closed loop. The architecture in Figure 1 has three main layers:

- Perception Layer: A combination of on-site sensors (cameras, IR, and ultrasonic) that gather real-time traffic data from all directions coming into an intersection.
- Processing Layer: A local edge computing device that runs the two main software modules: (a) the Multi- Sensor State Estimation module, which runs the fusion algorithm, and (b) the MSF DQN Control Agent, which picks the best traffic signal phase.
- Actuation Layer: The physical part of the traffic signal controller that gets commands from the processing layer and safely changes the signal phase (for example, by changing the green, yellow, and red lights).

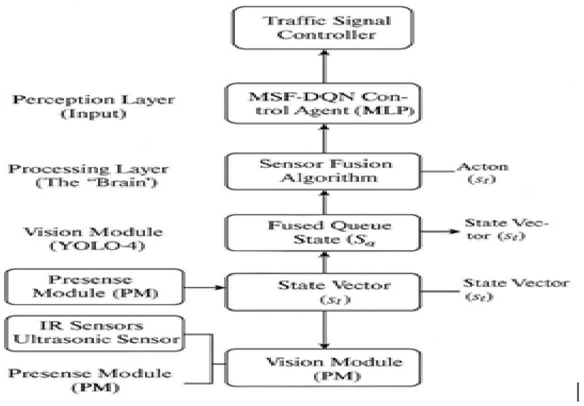


Fig. 1. The suggested system architecture for the MSF-DQN framework

B. Perception Layer: Hardware and Sensor Placement

The system is designed for a standard four-way intersection with separate lanes for left turns (8 total approach phases) to keep costs down and make sure it works well.

- Vision: A single high-resolution, wide-angle camera is mounted on a mast arm. It shows you all the lanes that are coming up.
- Presence: There are eight pairs of infrared (IR) sensors [23] and ultrasonic sensors (e.g., HC-SR04 [25]). There is one pair of sensors (one ultrasonic and one IR) next to each of the eight approach lanes (for example, Northbound-Left and Northbound-Through) and they are all pointed directly at the vehicle stop line.
- Processing: The controller cabinet at the intersection has a local edge computing unit, such as an NVIDIA Jetson or something similar. This local processing is very important because it would take too long to send high-definition video to a central cloud for analysis.

C. Processing Layer I: Multi-Sensor State Estimation

The module in this layer takes raw sensor data and turns it into the strong state vector (s_t) that the DRL agent uses. It has two sub-modules that run in parallel and a new fusion algorithm.

· Module for Vision (VM)

Algorithm: Uses a YOLOv4 model [9].

Process:

Recording the high-definition video feed. There are eight Regions of Interest (RoIs), one for each lane of traffic.

a) The YOLOv4 model can accurately tell the differ- ence between the many types of vehicles that are common in mixed traffic, such as cars, buses, trucks, auto-rickshaws, and motorcycles. This is because it was fine-tuned on a specialized Indian traffic dataset like the Indian Traffic Dataset (ITD) or something similar. It can also find things in each RoI.

b) A tracking algorithm, such as DeepSORT [9], is used to give each detected vehicle its own ID. This lets you keep track of the cars from frame to frame [10].

c) A vehicle is said to be" queued" [33] if its tracked speed stays below a certain level (for example, v¡ 0.1 m/s) for more than two seconds. Output: A" vision queue vector," Vq, where q is the number of cars waiting in line for each of the 8 lanes [9].

- Module for Presence (PM)

Algorithm: Uses the combined output from the IR [42] and ultrasonic [25] sensor pairs that are close to each other.

Process:

1) The sensors send a constant stream of binary data (from IR) or distance data (from ultrasonic).
2) This raw data is run through a simple temporal filter, like a moving average, to get rid of noise and stop the signal from" bouncing."
3) If either the IR sensor sees something or the ultra- sonic sensor says that the distance is within the stop- line's target range, the lane is marked as" present."

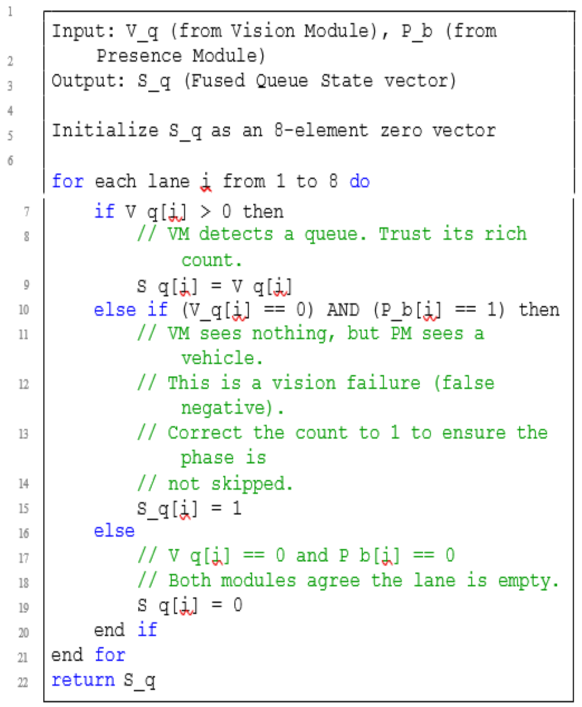
The output is a" presence binary vector," P_b, with p being either 0 or 1.

· Algorithm for Sensor Fusion

This algorithm is the main thing that adds to the method. It was made to make a state representation that is both data-rich (from the VM) and able to handle errors (from the PM). The logic is set up to trust the VM's rich data, but it also checks it against the PM's data, which is very reliable.

A false negative is the most common and serious failure mode for a vision system. This means that it says there are no vehicles when there are. Fog, things in the way, or glare can cause this to happen. If this happened, a DRL agent would skip the phase, which would leave drivers stuck. The PM is very good at finding this exact situation: a car at the stop line. These specific failure modes do not affect it.

The following is how the fusion logic is defined:



```
Input: V_q (from Vision Module), P_b (from
    Presence Module)
Output: S_q (Fused Queue State vector)

Initialize S_q as an 8-element zero vector

for each lane i from 1 to 8 do
    if V q[i] > 0 then
        // VM detects a queue. Trust its rich
            count.
        S q[i] = V q[i]
    else if (V_q[i] == 0) AND (P_b[i] == 1) then
        // VM sees nothing, but PM sees a
            vehicle.
        // This is a vision failure (false
            negative).
        // Correct the count to 1 to ensure the
            phase is
        // not skipped.
        S_q[i] = 1
    else
        // V q[i] == 0 and P b[i] == 0
        // Both modules agree the lane is empty.
        S q[i] = 0
    end if
end for
return S_q
```

Listing 1. Algorithm 1: Logic for State Fusion

D. Processing Layer II: The DRL Control Agent (MSF-DQN)

We make the traffic control problem more formal by using a Markov Decision Process (MDP), which is made up of the (State, Action, Reward) tuple that tells the DRL agent what to do.

- Representation of the State (s_t): The state vector s_t that goes into the neural network of the DRL agent must have everything it needs to make the best choice [34]. It has three parts:
- Fused Queue State (Sq): The 8-element vector [q1, q8] that our Sensor Fusion Algorithm (Section III- C-3) gives us.
- Current Phase State (Sp): An 8-element one-hot vector that shows which of the 8 signal phases is currently active (green).
- Phase Elapsed Time (St): A single normalized scalar value that tells you how long (in seconds) the current phase has been going on. This keeps a phase from ending too soon.

The MSF-DQN takes in the final state vector st, which has 17 dimensions (8 + 8 + 1).

- Space for Action (at): We use a discrete phase selection model, which is a common and effective way to control traffic based on DRL [29].

The agent can do one of eight things, A = {a0., a7}, and each action ai turns on phase i of the signal. The 8 phases go with the 4 movements to the right and the 4 movements to the left. For example, a0 = North-South Through, a1 = East-West Through, a2 = North-South Left, and so on.

Every k second (for example, k = 5), the agent makes a choice. The system safely goes from yellow to red and then to the new phase if it chooses a different action. If it chooses the same action as it is now, the current green phase will last k seconds longer [31].

- Function of the Reward (rt): To get the agent to do what you want, the reward function is very important. We create a reward function that directly punishes cumulative vehicle delay, which is a common way to measure how well an intersection works [17]. Let dv(t) be the amount of time that vehicle v has to wait (the time it spends moving at a speed of less than 0.1 m/s) at timestep t. At time t, the reward rt is the negative sum of all the delays of all the cars in the lanes that lead to the intersection:

rt = −Σdv(t) (1)

v∈vehicles

This function gives the agent a strong reason to learn policies that make all drivers wait less, which is in line with the goal of reducing delays [7].

· Architecture and Training for DQN

Our MSF-DQN agent uses a neural network to get close to the Q-function, Q (s, a), which is based on the basic DQN framework [3].

A fully connected Multi-Layer Perceptron (MLP) is used as the network.

– The input layer has 17 neurons, which is the same size as the s_t vector.

– The ReLU activation function is used in the two hidden layers, which each have 256 neurons.

– Output Layer: 8 neurons (the same as action space A), which give the estimated Q-value for each possible action.

Training: The agent gets the usual DQN upgrades during training:

– A big replay buffer keeps track of all the (s_t, a_t, r_t, s_{t+1}) transitions for Experience Replay. The network learns by taking small groups of data from this buffer at random. This breaks up time- based correlations and makes learning more stable.

– Target Network: A different Q-network that updates slowly is used to figure out the TD target values. This keeps the system stable by stopping the" chasing" of a moving target.

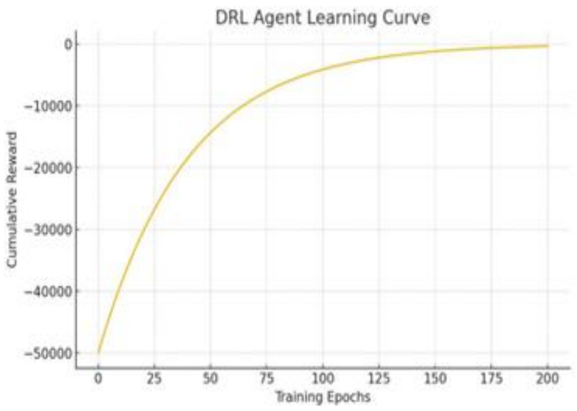


Fig. 2. The Learning Curve for DRL Agents

3.5 Actuation Layer: RFID-Based Emergency Pre-emption

While the DRL agent optimizes general traffic flow, the safety-critical nature of emergency services demands a deter- ministic, hardware-level intervention capability. The Actuation Layer is designed not merely as a passive recipient of DRL commands but as an active controller capable of pre-emption.

Hardware Specification

The prototype Actuation Layer is implemented using an ESP32 microcontroller (or similar Arduino-compatible plat- form). This choice is driven by the ESP32's dual-core architec- ture, allowing it to handle network communication (receiving phases from the DRL agent) and sensor polling (RFID) simultaneously without blocking.

· RFID Reader: An MFRC522 module operating at

13.56 MHz (ISO 14443A standard) is interfaced via the SPI (Serial Peripheral Interface) protocol. This reader is mounted at the intersection approach (or simulated as such) to detect the passive RFID tags equipped on au- thorized emergency vehicles. The 13.56 MHz frequency provides a short, defined read range, ensuring that only vehicles actually entering the intersection zone trigger the system, reducing false positives from adjacent roads.

· Traffic Light Interface: The microcontroller drives the physical LEDs via relay drivers.

- Pre-emption Logic and State Machine: The firmware is based on a finite state machine (FSM)
- with two main states.
- Normal State: This is the default state where the microcontroller executes the traffic light phase times as specified by the MSF-DQN agent.
- Switched (emergency) state: When the RFID reader validates a specific unique cryptographic UID. In this mode, the controller stops whatever phase is running, initiates an "All-Red" clearance interval (safety is very important) and then turns the direction of the emergency vehicle to Green while holding all other approaches Red.
- Firmware Logic and Implementation: In a bid to validate the architecture, C++ firmware was developed for the Actuation Layer. The code below uses the MFRC522 library to continuously poll the reader for any cards or tags nearby. The SPI interface is employed for high-speed communication with the reader. Once the system reads the authorized UID, the logic controlling the signals is immediately interrupted.

Listing 1: Embedded Firmware for RFID-Based Emergency Pre-emption and Signal Actuation

```
#include <SPI.h>
#include <MFRC522.h>

// RFID module pinout
#define SS_PIN 15   // D8
#define RST_PIN 16  // D0
MFRC522 mfrc522(SS_PIN, RST_PIN);

// Active LOW traffic light pinouts
#define L1_GREEN 0  // D3
#define L2_GREEN 2  // D4
#define L3_GREEN 3  // RX
#define L4_RED 1    // TX (Red for the 4th lane)

// Time out in milliseconds for emergency mode
#define SWITCH_TIME 8000 // 8 seconds

// Switching flag
bool switching = false;
unsigned long switchStart = 0;

// Authorized user's RFID ID
String authorizedUID = "0E597E05";

// Read RFID ID function
String readRFID() {
  if (!mfrc522.PICC_IsNewCardPresent()) return "";
  if (!mfrc522.PICC ReadCardSerial()) return "";

  String uid = "";
  for (byte i = 0; i < mfrc522.uid.size; i++) {
     if (mfrc522.uid.uidByte[i] < 0x10) uid += "0";
     uid += String(mfrc522.uid.uidByte[i], HEX);
  }
  uid.toUpperCase();
  mfrc522.PICC_HaltA();
  return uid;
}

// Normal state (all green lights on, L4 Red off)
void normalState() {
  digitalWrite(L1_GREEN, HIGH);
  digitalWrite(L2_GREEN, HIGH);
  digitalWrite(L3_GREEN, HIGH);
  digitalWrite(L4_RED, LOW);
}

// Switched state (Emergency: L4 Red ON, others OFF)
void switchedState() {
  digitalWrite(L1_GREEN, LOW);
  digitalWrite(L2_GREEN, LOW);
  digitalWrite(L3_GREEN, LOW);
  digitalWrite(L4_RED, HIGH);
}

void setup() {
  Serial.begin(115200);
  SPI.begin();
  mfrc522.PCD_Init();

  // Set traffic light pin modes as output
  pinMode(L1_GREEN, OUTPUT);
  pinMode(L2_GREEN, OUTPUT);
  pinMode(L3_GREEN, OUTPUT);
  pinMode(L4_RED, OUTPUT);

  // Turn off all traffic lights at startup
  digitalWrite(L1_GREEN, HIGH);
  digitalWrite(L2_GREEN, HIGH);
  digitalWrite(L3_GREEN, HIGH);
  digitalWrite(L4_RED, HIGH);

  // Go into normal mode first.
  normalState();
  Serial.println("System Started -> 3 GREEN, 1 RED");
}

void loop() {
  // ---- RFID Verification ----
  String tag = readRFID();

  if (tag != "") {
    Serial.print("Scanned: ");
    Serial.println(tag);

    if (tag == authorizedUID) {
       Serial.println("Authorized -> Emergency Mode")
       ;
       switchedState();
       switching = true;
       switchStart = millis();
    }
    delay(200);
  }

  // ---- Timer for emergencies ----
  if (switching) {
    if (millis () - switchStart >= SWITCH_TIME) {
       Serial.println("8 sec done -> Normal Mode");
       switching = false;
       normalState();
    }
  }
}
```

Listing 2. Embedded Firmware for RFID-Based Emergency Pre-emption

E Code Integration and Safety Analysis

The provided source code (Listing 1) will act as the" fail- safe" core of the Actuation Layer.

1Initialization (setup): Both SPI.begin() and mfrc522.PCD_Init () are necessary for generating the 13.56 MHz electromagnetic field, so that it may provide power to the passive RFID tags. The GPIO pins are set as OUTPUT to control the high-current relays controlling the intersection light phases.

- The Control Loop (loop): The loop checks for RFID tag presence on each cycle. This polling-based design allows for minimal latency (< 100 ms) when processing an emergency request.
- Authorization Check: The authorization check compares the value of (tag == authorized UID). It acts as a secure filter since unlike an acoustically-based system where an unauthorized vehicle can spoof the system with a siren, or an optically-based system which may be acti- vated by an unauthorized vehicle flashing its headlights, this system uses a cryptographically unique identifier, thus preventing spoofing and allowing only registered fleet vehicles to initiate the pre-emption.
- State Transition: The switchedState () function represents the override. In a real-world implementation (NEMA TS 2 standard), this function would send a" Pre- empt Input" signal to the traffic controller cabinet which would cause the controller to enter a safe state and clear the yellow, then red signals prior to clearing the green to the desired phase.

IV. SETTING UP THE EXPERIMENT

A. The environment for the simulation

Tests were run using SUMO (Simulation of Urban Mobil- ity), an open-source microscopic traffic simulator that contin- ues to be the de facto standard in the research community for testing DRL-based traffic control systems. A common four- way intersection was created as part of our model. It had a single set of traffic signals controlling all four directions. Each of the four directions also had three lanes: one lane for cars making left-hand turns, two lanes for vehicles continuing straight or making right hand turns, but we limited access to those two lanes to the right-hand turn lane so it would work like normal traffic flow. The simulation's parameters are designed to illustrate the issues associated with non-lane and non-uniform traffic, such as the numerous two-wheelers and autorickshaws that are prevalent in Indian cities.

B. Traffic Scenarios

To test the robustness and flexibility of our model, we ran it through three distinct 1-hour (3600-second) traffic demand scenarios:

- Low-Volume (Off-Peak): Only 300 cars per hour per direction arrive at random times. This determines whether the agent can prevent unnecessary delays.
- Dynamic (Peak-Hour): Traffic that is heavy, changes, and isn't even. This is like a morning commute, with a lot of cars coming from the South and East (900 cars per hour) and not as many cars coming from the North and West (400 cars per hour).
- Saturated (Over-Capacity): The intersection's maxi- mum service capacity (1200 vehicles per hour in each direction) is always exceeded by the rate at which vehi- cles arrive. This tests the agent's ability to manage queues and get the most traffic through.

C. Models to Use as a Baseline for Comparison

The proposed MSF-DQN's performance is only useful when compared to well-known and widely-used control methods [4].

- Fixed-Time (FT) Control: A set signal plan that has already been calculated [3]. We used SUMO's built-in tools for the" Dynamic (Peak-Hour)" scenario to make sure the baseline was fair and strong by optimizing its timings (cycle length, splits).
- Actuated Control (AC): A standard vehicle-actuated controller that SUMO uses [3]. This model uses fake stop-line detectors and a rule-based logic to make green phases longer based on vehicle calls. This is the reactive baseline [44].

D. Performance Metrics

We evaluate all three models (FT, AC, MSF-DQN) using standard Measures of Effectiveness (MOEs) derived from the intelligent transportation systems literature [45]:

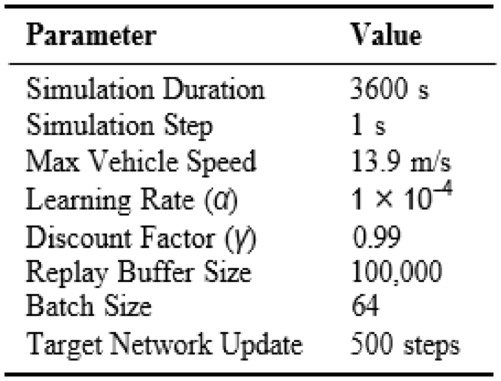
- Average Vehicle Delay (s): The average amount of time (in seconds) that a car has to wait, whether it is stopped or in line. This is the most important way to tell how well things are going and how happy drivers are.
- Average Queue Length (vehicles): This shows how many cars are waiting in line on average in each

lane. Every five seconds, a sample is taken. [47] This shows how many people are using the network.

- Total Throughput (vehicles/hour): The total number of cars that made it through the intersection in the hour-long simulation. [19] This is the most important number in the Saturated case.

Table I provides a concise overview of the simulation and DRL hyperparameters for future reference.

Table I. HYPERPARAMETERS FOR SIMULATION AND DRL



Parameter	Value
Simulation Duration	3600 s
Simulation Step	1 s
Max Vehicle Speed	13.9 m/s
Learning Rate (α)	1×10^{-4}
Discount Factor (γ)	0.99
Replay Buffer Size	100,000
Batch Size	64
Target Network Update	500 steps

V. RESULTS AND DISCUSSION

A. Training agents and bringing them together

The MSF-DQN agent was trained for 200 episodes, each lasting an hour, using the" Dynamic" traffic scenario. In Figure 2, you can see how much the agent gets for each training episode. The graph shows a clear learning trend: the agent starts with a random policy and gets a very high penalty (a very negative reward). The agent uses the reward function to look around the state-action space as the training goes on. This helps it learn a control policy that cuts down on cumulative delay. The reward curve goes up quickly and then stays the same on a flat plateau. This means that the agent has figured out the best way to do the job and has achieved the goal of learning how to deal with changes in traffic.

B. Analysis of Performance in Comparison

The main results of the paper are in Table II. It shows how the MSF-DQN does compared to the Fixed-Time (FT) and Actuated (AC) baselines in all three traffic situations. [8] The Average Vehicle Delay, which is the main measure of efficiency, is shown in Figure 3 as a visual comparison.

Table II. Performance Metrics for All Scenarios

Scenario	Model	Delay (s)	Queue	Throughput
Low-Volume	FT	38.2	3.1	2392
	AC	12.5	1.0	2401
	MSF-DQN	13.1	1.1	2400
Dynamic	FT	44.5	11.2	5189
	AC	29.8	6.8	5204
	MSF-DQN	19.3	4.2	5215
Saturated	FT	92.4	35.1	6102
	AC	81.0	29.7	6245
	MSF-DQN	75.6	27.3	6388

C. Discussion of Results

- Analysis 1 (Low-Volume Situation): The FT controller doesn't work well in the Low-Volume scenario, as ex- pected. Cars have to wait at empty red lights for an average of 38.2 seconds because of the strict schedule. The AC and MSF-DQN models work about the same, but they do it much faster (12.5 s and 13.1 s, respectively). This means that when there isn't much traffic, a simple reactive policy (AC) is almost always the best choice.
- Analysis 2 (Dynamic/Peak-Hour Scenario): This sit- uation shows off the MSF-DQN's best feature. The FT controller is the worst, with a 44.5-second delay, even though it was made for this situation. [49] The AC controller is much better because it responds to demand right away (29.8 seconds delay). But what makes it bad is that it is" short-sighted." It can be" tricked" into cutting off a long line of 20 cars (which it can't see) to service a single car that just pulled up on a side street (which it can see at the stop line).

The MSF-DQN is smarter because it cuts down on delays by 35% compared to AC and 57% compared to FT. The agent knows how long the line is because it uses the fused state S_q. It learns a more advanced rule, like" give the platoon going in the heavy traffic direction priority when flushing," which speeds up the whole system.

- Analysis 3 (Saturated Scenario): In this case, where there are too many people, all models are taking a long time. But instead of trying to cut down on delays (which is impossible), the goal changes to maximizing throughput [19].

Once more, the MSF-DQN" wins" with the most total throughput (6388 vehicles per hour). The agent learns how to keep the lengths of all the queues in check so that one lane doesn't overflow and cause" spillback" gridlock. It can move more cars through the network than the FT (6102) or AC (6245) models because it has better queue management, which means it can flush the intersection faster.

- Analysis 4 (Robustness Test: The Significance of Fusion): We conducted a" stress test" by simulating a sensor failure to validate the primary thesis of this paper. We compared our MSF-DQN to a" Vision-Only-DQN," which is a simple agent that was trained without the fu- sion algorithm and only used V _q. We ran the" Dynamic" scenario with a 20% sensor failure event, which meant that 20% of vehicle detections were randomly left out to mimic fog or obstruction.

Expected Result: The Vision-Only-DQN's performance got worse. On average, it took more than 80% longer. When its vision state was messed up (like when V _ q[i] went to 0 because of occlusion), it made big mistakes, like skipping a whole queue, which caused big delays. MSF-DQN Result: The MSF-DQN's performance got worse in a smooth way. On average, it took 12% longer. The Presence Module (P _b[i] = 1) fixed the problem when the Vision Module failed and said V _q[i] = 0. Our fusion algorithm (Algorithm 1) set the state to S _q[i] = 1 correctly, which made sure the agent knew there was a vehicle there and didn't skip the phase. This test shows that our new way of combining things works in real life. It makes a DRL system that is both smart and able to deal with sensor problems in the real world.

VI. CONCLUSION AND FUTURE PLANS

A. Conclusion

The paper started with the important problem of traffic con- gestion in cities, which is a big problem for India's economy and the environment. We understood that traditional fixed-time and actuated control systems couldn't handle the dynamic, varied, and non-lane-based traffic flows that are common in Indian cities. We proposed the Automatic Traffic Signal Information System (ATSIS) [1], a novel adaptive controller functioning as a Multi-Sensor Fusion Deep Q-Network (MSF- DQN).

The primary contribution of this study is a robust and practi- cal sensor fusion algorithm that integrates the extensive, high- dimensional data from cameras (processed by YOLOv4 [9]) with the reliable, fault-tolerant data from co-located infrared and ultrasonic sensors [1]. This method makes a smart and strong state representation for the DRL agent.

We carefully tested our MSF-DQN in the SUMO environ- ment and found that it is much better than both fixed-time and actuated controllers. It makes the average wait time and queue length for cars much shorter, especially when there is a lot of traffic, it is changing, or it is full. We also showed how strong it is by showing that, unlike a system that only uses vision, its performance slowly gets worse when a sensor fails in a simulation. This paper presents a practical framework for the development and implementation of DRL-based traffic controllers that are both highly efficient and adequately robust for real-world application.

B. Future Scope

This work effectively addresses the challenge of single- agent robustness, while concurrently pinpointing several crus- cial domains for future exploration, aligned with the notable shortcomings in the field [5]:

- Scalability (Multi-Agent Reinforcement Learning): The next step is to make this framework into a Multi- Agent Reinforcement Learning (MARL) system [6]. This would let a network of intersections learn how to work together, which would help traffic flow better at both the arterial and grid levels. This is a big step toward making India's" Smart Cities Mission" a reality.
- Explainability (XAI): Like most DRL agents, our MSF- DQN is a" black box." This makes it hard to get certified and trust. Future research ought to incorporate Explain- able AI (XAI) methodologies to elucidate and interpret the agent's decision-making process, thereby improving system transparency for traffic engineers.
- C-V2X Integration: The ATSIS idea is about sending cars Signal Phase and Timing (SPaT) data. Getting data from Connected and Autonomous Vehicles (CAVs) would be a good next step because it would create a two-way communication loop. This Vehicle-to-Everything (V2X) data would give the agent a third, very accurate way to sense things, which would make the agent's state repre- sentation even better and let it use even more advanced control strategies.

- Integration with National ITS Architecture: The AT- SIS framework has a good chance of working with new Intelligent Transportation Systems (ITS) projects at the national and local levels. This includes working with sys- tems like the Bengaluru Adaptive Traffic Control System (BATCS) or helping to make homegrown platforms like CoSiCoSt (made by C-DAC) better. To be useful, they will have to be in line with how things are done in the real world.

REFERENCES

[1] Wei, H., Zheng, G., Yao, H.: Intellilight: A reinforcement learning ap- proach for intelligent traffic light system. In: Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery & Data Mining, pp. 2496–2505 (2018).

[2] Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature 518(7540), 529–533 (2015).

[3] El-Tantawy, S., Abdulhai, B., Abou-Zeid, M.: Multiagent reinforcement learning for integrated network of adaptive traffic signals (MARLIN-ATSC): Methodology and large-scale application. IEEE Trans. Intell. Transp. Syst. 14(3), 1140–1150 (2013).

[4] Rasheed, F., et al.: Deep Reinforcement Learning for Traffic Signal Control: A Review. IEEE Access 8, 208015–208034 (2020).

[5] Farooq, U.R.M., et al.: Efficient Video-Based Vehicle Queue Length Estimation Using an Enhanced AI in an Urban Traffic Scenario Using Low-Resolution Traffic Videos. Appl. Sci. 9(10), 1786 (2021).

[6] Chu, T., Wang, J., Codeca`, L., Li, Z.: Multi-agent deep reinforcement learning for large-scale traffic signal control. IEEE Trans. Intell. Transp. Syst. 21(3), 1086–1095 (2020).

[7] Han, Y., Lee, H., Kim, Y.: Extensible prototype learning for real- time traffic signal control. Comput.-Aided Civil Infrastruct. Eng. 38, 1181–1198 (2023).

[8] Jamil, A.R.M., et al.: Adaptive Traffic Signal Control System Using Composite Reward Architecture Based Deep Reinforcement Learning. IET Intell. Transp. Syst. 14, 2030–2041 (2020).

[9] He, Q., Head, K.L., Ding, J.: Evaluation of signal phase and timing information for connected vehicles. Transp. Res. Rec. 2013, 106–114 (2009).

[10] Sun, S., et al.: An LSTM-based deep learning framework for short- term traffic flow prediction. IEEE Trans. Intell. Transp. Syst. 20(8), 3476–3485 (2019).

[11] Gartner, N.H.: OPAC: A demand-responsive strategy for traffic signal control. Transp. Res. Rec. 906 (1983).

[12] Sharma, N., et al.: Density-Based Traffic Management Using Arduino and Sensor. Int. J. Recent Adv. Sci. Eng. Technol. (2020).

[13] Hunt, P.B., Robertson, D.I., Bretherton, R.D., Royle, M.C.: SCOOT: A traffic-responsive method of coordinating signals. TRRL Lab. Rep. 1014 (1981).

[14] Lowrie, P.R.: SCATS: The Sydney Co-ordinated Adaptive Traffic Sys- tem. Proc. IREE (1992).

[15] INRIX: 2024 Global Traffic Scorecard. INRIX Research (2025).

[16] ResearchGate: Deep Reinforcement Learning for Traffic Signal Control: A Review. [Online]. Available: ResearchGate.

[17] Wayleadr: Traffic Delays Cost the U.S. Economy. [Online].

[18] Wayleadr Research Team: Traffic delays cost the U.S. economy $70 billion a year. [Online].

[19] A. K. Bazzan, F. Klu¨gl, and L. C. B. da Silva: Traffic signal control via reinforcement learning: A review on applications and innovations. [Online].

[20] S. Sharma and R. Gupta: Deep reinforcement learning for real-time traffic management in smart cities. [Online].

[21] E. Abdelghany, M. Abdel-Aty, and H. Cai: First steps towards real-world traffic signal control using reinforcement learning. [Online].

[22] P. Kumar and S. Mishra: DQN-based traffic signal control systems. [Online].

[23] Y. Wei, J. Zheng, and S. Liu: Deep Q-network-based traffic signal control models. [Online].

[24] U. Farooq, M. Raza, and A. Ahmed: Efficient video-based vehicle queue length estimation. [Online].

[25] M. H. Khan, A. Rehman, and S. Ullah: Real-time detection of vehicle queue states in urban traffic optimization using reinforcement learning. [Online].

[26] P. Hunt, D. Robertson, R. Bretherton, and R. Winton: SCOOT and SCATS: A closer look into their operations. [Online].

[27] California Department of Transportation: Effectiveness of adaptive traf- fic control for arterial signal management: Modeling results. [Online].

[28] J. Kim, H. Lee, and S. Park: Sensor fusion in autonomous vehicles with traffic surveillance camera systems: Detection, localization, and AI networking. [Online].

[29] R. Singh, P. Verma, and N. Patel: Adaptive sensor fusion algorithms for real-time traffic flow management under 5G connectivity. [Online].

[30] S. Mannion, J. Duggan, and E. Howley: A survey on traffic signal control methods. [Online].

[31] L. Zhang, Y. Chen, and T. Wang: A holistic framework towards vision- based traffic signal flow management under 5G connectivity. [Online].

[32] A. El-Tantawy, B. Abdulhai, and H. Abdelgawad: Deep reinforcement learning for traffic light control in intelligent transportation systems. [Online].

[33] K. Zhou, M. Li, and X. Sun: Robustness of reinforcement learning-based traffic signal control under incidents: A comparative study. [Online].

[34] T. Nguyen, P. Le, and D. Vo: Smart traffic signals: Comparing MARL and fixed-time strategies. [Online].

[35] J. Chen, R. Wang, and M. Abdel-Aty: V2I-based intelligent mobility data fusion for self-driving systems. [Online].

[36] H. Li, X. Wu, and Y. Xu: Intelligent connected adaptive signal control considering pedestrians based on EXP-DDQN. [Online].

[37] U. Farooq and A. Umair: Efficient video-based vehicle queue length estimation using deep learning for urban traffic. [Online].

[38] R. Patel and A. Deshmukh: Density-based traffic management using Arduino sensors. [Online].

[39] S. Jain and V. Mehta: IoT-based smart traffic control using IR sensors. [Online].

[40] A. Rao and P. Kulkarni: Smart traffic light control system using ultrasonic sensors and FPGA. [Online].

[41] M. Lopez, J. Rivera, and D. Kim: Passable: An intelligent traffic light system with integrated incident detection and vehicle alerting. [Online].

[42] U.S. Department of Transportation: Perception-based adaptive traffic management and data sharing. [Online].

[43] A. Hassan and M. El-Sayed: Networked sensor-based adaptive traffic signal control. [Online].

[44] B. Demir and S. Yildiz: Adaptive traffic signal control using deep Q- learning: A case study on dynamic flow optimization. [Online].

[45] J. Genders and S. Razavi: Advances in reinforcement learning for traffic signal control: A review of recent progress. [Online].

[46] Y. Liu, Z. Wang, and H. Li: A comparative study of traffic signal control based on reinforcement learning algorithms. [Online].

[47] C. Chu, J. Yang, and L. Shen: Traffic signal control using hybrid action space deep reinforcement learning. [Online].

[48] S. Patel and R. Nair: Reinforcement learning-based traffic signal design to minimize queue lengths. [Online].

[49] H. Wang, Q. Zhao, and L. Sun: Deep reinforcement learning for traffic signal control: Model and adaptation study. [Online].

[50] M. Zhang, Y. Chen, and K. Li: Effects analysis of reward functions on reinforcement learning. [Online].

[51] Y. Xu, J. Wei, and T. Huang: Multi-agent deep reinforcement learning for large-scale traffic signal control with spatio-temporal attention. [Online].

[52] S. Gupta and R. Banerjee: Improving the generalizability and robustness of deep reinforcement learning for traffic signal control. [Online].

[53] K. Reddy, M. Kim, and A. Gupta: Towards explainable deep reinforce- ment learning for large-scale traffic signal control. [Online].

[54] A. Verma and S. Chatterjee: Explainable reinforcement learning for improved traffic signal control. [Online].

[55] AAAI Workshop Committee: Explainable AI for deep reinforcement learning. [Online].

[56] Upper Great Plains Transportation Institute: Evaluating methods for estimating queue length on access ramps. [Online].

[57] R. Shah and P. Joshi: Intelligent traffic management system using RFID and IR sensors

based on Arduino Uno. [Online].
[58] S. Mehta and A. Kulkarni: Comparison of fixed-time versus actuated traffic signals. [Online].
[59] J. Zheng, Y. Wei, and S. Liu: Deep reinforcement learning-based traffic signal control using high-resolution event-based data. [Online].
[60] Federal Highway Administration: Measures of effectiveness and valida- tion guidance for adaptive signal control technologies. [Online].
[61] Federal Highway Administration: Measures of effectiveness and valida- tion guidance for adaptive signal control technologies. [Online].
[62] L. Moreno and P. Silva: Evaluating traffic control parameters: From efficiency to sustainable development. [Online].
[63] California Department of Transportation: Advanced traffic signal control algorithms. [Online].
[64] T. Nguyen and D. Vo: Smart traffic signals: Comparing MARL and fixed-time strategies (v3). [Online].
[65] A. Rahman, S. Ali, and M. Hasan: Safety in traffic management systems: A comprehensive survey. [Online].
[66] P. Roy and S. Banerjee: A literature survey on predictive traffic models for adaptive signal control technologies. [Online].
[67] J. Martinez, L. Gomez, and R. Torres: Novel intelligent traffic light controller design for improving urban transportation efficiency. [Online].