vehicles

MDPI

Article

A Vehicle Density Estimation Traffic Light Control System Using a Two-Dimensional Convolution Neural Network

Malose John Mathiane *, Chunling Tu, Pius Adewale and Mukatshung Nawej

Department of Computers System Engineering, Faculty of Information and Communication Technology, Tshwane University of Technology, Soshanguve, Pretoria 0001, South Africa; duc@tut.ac.za (C.T.); owolawipa@tut.ac.za (P.A.); nawejmc@tut.ac.za (M.N.)

* Correspondence: mjjohn.malose@gmail.com

Abstract: One of the world's challenges is the amount of traffic on the roads. Waiting for the green light is a major cause of traffic congestion. Low throughput rates and eventual congestion come from many traffic signals that are hard coded, irrespective of the volume of the amount of traffic. Instead of depending on predefined time intervals, it is essential to build a traffic signal control system that can react to changing vehicle densities. Emergency vehicles, like ambulances, must be given priority at the intersection so as not to spend more time at the traffic light. Computer vision techniques can be used to improve road traffic signal control and reduce real-time traffic delays at intersections without the requirement for substantial infrastructure analysis. Long wait times and significant energy consumption are just two of the problems of the current traffic signal control system. To optimal efficiency, the traffic signal's duration must be dynamically changed to account for current traffic volume. To lessen congestion, the approach taken in this research focuses on modifying traffic signal time determined by the density of vehicles at the crossroads. The main purpose of this article is to demonstrate heavy traffic and emergency vehicle prioritization from all directions at the traffic intersection for a speedy passage. Using the Pygame tool, the proposed method in this study, which includes a mechanism for estimating traffic density and prioritization by counting vehicles at a traffic junction, is demonstrated. The vehicle throughput for the adaptive traffic light built using Pygame is compared with the vehicle pass rate for the adaptive traffic light built using Simulation of Urban Mobility (SUMO). The simulation results show that the adaptive traffic light built using Pygame achieves 90% throughput compared to the adaptive traffic light built using SUMO. A Two-Dimensional Convolutional Neural Network (2D-CNN) is implemented using Tensorflow for vehicle classification. The 2D-CNN model demonstrated 96% accuracy in classifying vehicles using the test dataset. Additionally, emergency vehicles, such as ambulances, are given priority for quick passing.

Keywords: intelligent traffic system; lane priority; object detection; artificial intelligence; Pygame; SUMO; 2D-CNN; smart traffic management system

Citation: Mathiane, M.J.; Tu, C.; Adewale, P.; Nawej, M. A Vehicle Density Estimation Traffic Light Control System Using a Two-Dimensional Convolution Neural Network. *Vehicles* **2023**, *5*, 1844–1862. https://doi.org/10.3390/vehicles5040099

Academic Editors: Peter Gaspar and Junnian Wang

Received: 13 November 2023
Revised: 10 December 2023
Accepted: 12 December 2023
Published: 15 December 2023

1. Introduction

The regulation of traffic lights plays a crucial role in every smart traffic management system. The two main factors to be taken into account in traffic light management are the green light series and the period of green light [1]. Traffic congestion in cities is becoming a bigger problem. People commute longer distances to work, school, and shops, attend social events, and deal with traffic jams and accidents [2]. Fixed signal timer intersection traffic control systems are the source of many traffic-related problems. They replicate an identical phase sequence and its length without any modifications. The field of intelligent transport systems offers novel solutions for traffic control in response to the growing demand for road capacity [3].

The timing merely serves as the default setting for managing what might be referred to as ordinary traffic [4]. Although each road junction has a different traffic light timing design, there are traffic lights at the intersections that are always on for a predetermined amount of time [5].

However, many of the current systems use an extremely basic sequence. Currently, three standard systems of traffic control are in use:

- Manual control: As the name implies, traffic control involves human intervention. To manage traffic, traffic police are posted in a specific location;
- Traditional traffic lights with fixed timers: Timers are used to control this. The timer is set to a constant value. The timer value determines when the lights automatically turn from red to green [6];
- Electronic sensors: Another state-of-the-art option is to place proximity or loop detectors beside the road. This sensor collects information about traffic in vehicles. The traffic lights are managed by means of sensor data [7].

These traditional approaches have several shortcomings. It takes a lot of work to operate the manual regulating mechanism. Handling all the traffic in a city or town by hand becomes very difficult when resources are scarce. This calls for the need for an upgraded traffic management system. Static traffic conflicts are handled by a traffic signal with a pre-programmed timer for each phase that does not respond to real traffic on the route. [8]. Accuracy and coverage are typically at odds when using electronic sensors, such as proximity sensors or loop detectors, and a limited budget will limit the number of facilities available because high-quality data collection typically requires expensive and complex hardware. Furthermore, a large number of sensors are usually needed to give comprehensive coverage throughout a network of facilities due to the restricted effective range of most sensors [9].

Deep learning-based models have been widely introduced for traffic management systems in recent years due to their superior object detection accuracy compared to traditional machine learning algorithms, such as support vector machines and image processing algorithms [10]. Convolutional activity constitutes one of CNN's foundations.

There is less need for humans to perform certain tasks thanks to the widespread automation of processes and goal-achieving capabilities of modern technologies. A computer can undoubtedly complete tasks faster and more accurately than a human, despite the fallibility of humans [11]. Robots can now learn on their own instead of needing to be programmed with every possible action and outcome thanks to this technological advancement [12] Convolution neural networks, TensorFlow, and Keras were utilized in this study because they were necessary [13].

The foundation of a convolutional neural network is the perceptron model. As a result, photos are now classified differently. A CNN, or ConvNets, is an example of a deep learning architecture [14]. Among many other applications, they work best in facial recognition and object detection. Self-driving cars have also benefited from them. Due to their exceptional ability to identify meaningful patterns within images by comprehending the spatial structure of the relevant input, these networks are helpful for image analysis and classification. CNNs are designed to take advantage of the spatial data, in contrast to traditional neural networks that ignore spatial structures, such as pixels, that are further apart or closer together [15]. Figure 1 shows the basic architecture of a CNN.

The fundamental objective of the newly implemented method in this article is to increase the overall number of cars that can pass through a junction while simultaneously reducing the amount of time that each track of vehicles must wait for a signal, given the mathematical function used to compute waiting times. Thanks to this system's capacity to instantly adapt to the flow of traffic there, vehicles should be able to drive through the intersection as quickly as feasible. Since traffic density varies depending on the time of day (morning, during the day, in the afternoons or evenings), the adaptive intelligence light system analyses fluctuations in traffic volume. The suggested framework aims to provide a crossroad signal that can adapt to the movement of traffic. This aims to increase the

amount of time between green signals and minimize delays, congestion, and waiting times by moving traffic through the system considerably more quickly than in a static system. Additionally, the system detects and modifies the traffic green light timing to allow for a quicker pass through at junctions, and special duty vehicles, like ambulances, are given priority at the intersection using the 2D CNN model for vehicle classification. This article includes the following sections. The literature review discusses the contributions of other experts on the subject, details the components of current traffic management systems, and is followed by methodology sections that demonstrate the proposed method and model, in which traffic lights are mentioned as one of the main contemporary sources of congestion, and the problem is defined in terms of its history and effects. All of the traffic density-reduction strategies' simulated models and procedures are discussed. The simulation and experimental findings are then analyzed, and the logical parts of traffic systems, such as the traffic simulation setting technique, are explained. The section compares and analyzes data gathered from the simulation programs, as well as data visualization. Finally, there is a conclusion followed by future work, which presents the summary, conclusion taken, limitations, and contributions to knowledge.

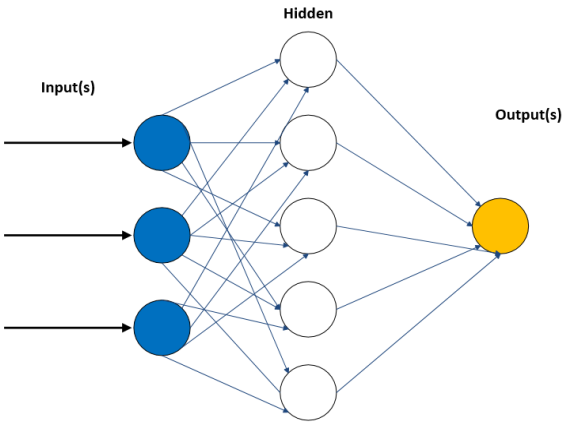

Figure 1. Artificial intelligence model of a basic perceptron.

2. Related Work

When traffic is heavy, critical response times could be shortened by putting the recommended smart traffic management system into practice. In the past, manual methods were insufficient to control traffic as needed. The number of cars on the road today is too many for the infrastructure that currently exists. In metropolitan areas, this problem affects a larger population. A lot of research has been performed on traffic management employing the intelligent approach of using traffic data to determine the state of the green light. Intelligent traffic control is a broad topic, and Table 1 lists the methods that have been used recently.

Table 1. Different intelligent traffic control methodologies.

Reference Number	Methodology	Main Contribution	Limitations
[16]	A signal timing optimization model based on bus priority.	The classic approach to a signal timing optimization problem was combined with the idea of bus priority. A new methodology that specifically considered passenger delay during the signal timing optimization process was presented as a solution to this issue.	The traffic data collected in a fixed period, such as peak time, is the only factor considered by the optimization model when updating the signal timing scheme. This may not be feasible for traffic data collected in near real-time.
[17]	Intelligent monitoring technology of traffic flow based on computer vision.	The method uses vehicle identification and localization to provide real-time, accurate, and robust traffic flow data collection on road segments.	The technique needs to be improved in terms of processing data collected on the road segments, and it is limited when it comes to high-resolution images.

Table 1. *Cont.*

Reference Number	Methodology	Main Contribution	Limitations
[13]	An adaptive traffic congestion control approach with emergency vehicle protocol.	To allow for minimal traffic congestion, the divider is adjusted in accordance with the number of vehicles on the road. When there is traffic congestion, it is challenging to move when an ambulance is passing on the road. Using an RFID reader and an RFID tag, this concept avoids this issue.	The system counts the number of cars and recognizes an ambulance using image processing techniques. Machine learning algorithms may be used in the future to monitor different kinds of emergency vehicles, such as police cars and fire trucks, and to increase the accuracy of vehicle identification.
[18]	An IOT-based traffic controlling system.	A suggested technique for priority-based vehicle identification makes use of techniques from the picture processing industry. If an emergency vehicle is identified, that lane will take precedence over all other lanes.	The vehicle count mechanism that would prioritize other lanes with heavier traffic in addition to emergency vehicles is absent from the proposed system.
[10]	Emergency vehicle type classification using a convolutional neural network.	The pre-trained model in this work, VGG-16, had a smaller convolutional layer and filter size. The experiment yielded a 95% accuracy rate for the suggested method.	The module may have learned from the color of the feature and needs some tweaks because it recognizes a typical red car as a firetruck and a white car as a police car.
	A bidirectional vehicle platooning-based intelligent transportation system.	This work proposes an intelligent transportation system that can monitor nearby vehicles and signals to monitor other vehicles to prevent accidents and shorten wait times at busy intersections by giving drivers access to pertinent data.	The proposed methodology does not have intelligence in controlling the traffic lights, and no emergency vehicles are given priority at the intersection.

An intelligent vehicular traffic management and control system is suggested in this research [19]. Using an algorithm based on several criteria and determining the optimal routes where there is the least amount of congestion, the suggested strategy seeks to decrease the problem of traffic congestion and make traffic signals function effectively, eliminating traffic jams. Other forms of communication technology, such as cellular networks and Geographic Positioning Systems (GPS), must be integrated with the proposed system [19]. The junction control issue in smart cities with smart transportation systems was examined in this work [20]. In contrast to current designs, the suggested method divides the parked cars in a lane into various groups. Through V2I communications, the okay to pass an intersection is then sent in terms of vehicle groupings [20]. Finding the right groupings based on current traffic circumstances presents the greatest challenge. The approach for the online collection of vehicle status on a road in an urban setting employing clustering in vehicular ad hoc networks is provided in this research [21]. To automatically schedule the traffic within its field of vision without contacting other intersections or a central control system, the information acquired is sent to the traffic light at the intersection [19]. Results obtained reveal that the suggested strategy significantly reduces the average global delay and increases the frequency of non-stopped vehicle movements.

A flexible traffic signal-regulated method was suggested for this study [4] to decrease the average time spent waiting at an intersection and increase traffic throughput. In contrast to a fixed-time control algorithm and an actuated control approach, the experimental results demonstrated that the proposed formula might result in higher throughput and shorter vehicle waiting times. The suggested system [20] ensures that by adapting the green signal timing according to the volume of traffic at the light, the direction with more traffic obtains a green signal for a longer amount of time than the way with less traffic. As a result, there were fewer unwelcome delays, less traffic, and less waiting time, all of which would result in less fuel use and pollution. A traffic light scheduling algorithm (ITLC) for isolated scenarios is presented in this paper [22]. Additionally, for open-network settings, an ATL regulating algorithm was created. Vehicle ad hoc communications technology is used by the ITLC algorithm to collect real-time traffic information on all competing traffic patterns at every intersection with a signal [20]. Future studies will look at these traffic signal

scheduling algorithms' fault tolerance, dependability, and partial relationship with vehicle problems [23].

The topic of this study [24] is the application of edge computing and Internet of Things sensors to an optimal control method for emergency vehicle priority. Throughout this construction, emergency vehicles will always take precedence, causing no disruption to normal traffic. The real-world trial with the Edge server and Dashboard Unit (DBU) is complete. There is a need for an improvement in the waiting for vehicles. In this study [25], a suggestion for allocating emergency vehicles to traffic has been offered. The system combines vehicle counting, timely warning transmission through the sensor network, and visual sensing methods to gauge the separation between intersections and the emergency vehicle. The range measurement in adverse weather and busy traffic conditions needs more study. The main objective of this paper's [26] design is to minimize the amount of time that EVs must wait at the crossing and to stop accidents there, which can lower the number of fatalities and the damage of priceless property. No driver interaction is necessary in this instance for the system to deliver requests from SRM to the RSU. Thanks to DSRC's dependable communication, transmission and reception happen significantly faster than they would have using the GSM technique [15]. A dynamic web or mobile application interface that permits more intelligent automatic and manual control of the system may be developed to enhance it [14].

In this study [27], a basic CNN model was put up and meticulously trained utilizing data augmentation techniques. The model's performance in measuring traffic density was excellent. However, training the CNN for a new site takes a significant amount of computational power and human labor. In the current work, a unique method for accurately quantifying traffic density at the aggregate level for traffic control and management was demonstrated by the counting of vehicles on a road section. The strategy was successful in accurately counting the number of vehicles.

The methodology provided in this research includes the calculation of density and prioritization of intersections with significant traffic. In addition, emergency vehicles are prioritized using a 2D-CNN for vehicle categorization. The proposed method is compared to one of the VANET methodologies to illustrate its performance and correctness.

3. Proposed Methodology and Model

The method of determining traffic density at a traffic intersection by counting cars and modifying traffic signal timing in response to vehicle density is illustrated using the Pygame tool. The goal of this approach is to manage the traffic by prioritizing lanes with more vehicles. Ambulances are given priority for quick passing at crossings when approaching the traffic intersection after they are identified using the 2D-CNN vehicle categorization. This study's development takes the following factors into account:

- Emergency vehicles enter the intersection from both directions at the same time. In this case, the suggested system is designed in such a way that it prioritizes lanes with heavy traffic, and there is no emergency vehicle preemption, which is accomplished through the program;
- Intersection accidents. The proposed system lacks an accident detection method and instead focuses on vehicle density and emergency vehicle preemption. It was designed with the assumption that no accidents would ever occur at the intersection and that traffic would flow freely.

The methodology section is divided into two major sections: traffic density estimation and emergency vehicle preemption using the 2D-CNN. The simulation's main method is traffic density estimation and prioritization, and vehicles are counted as they pass, approach, and pass the intersection. Figure 2 shows the proposed system model.

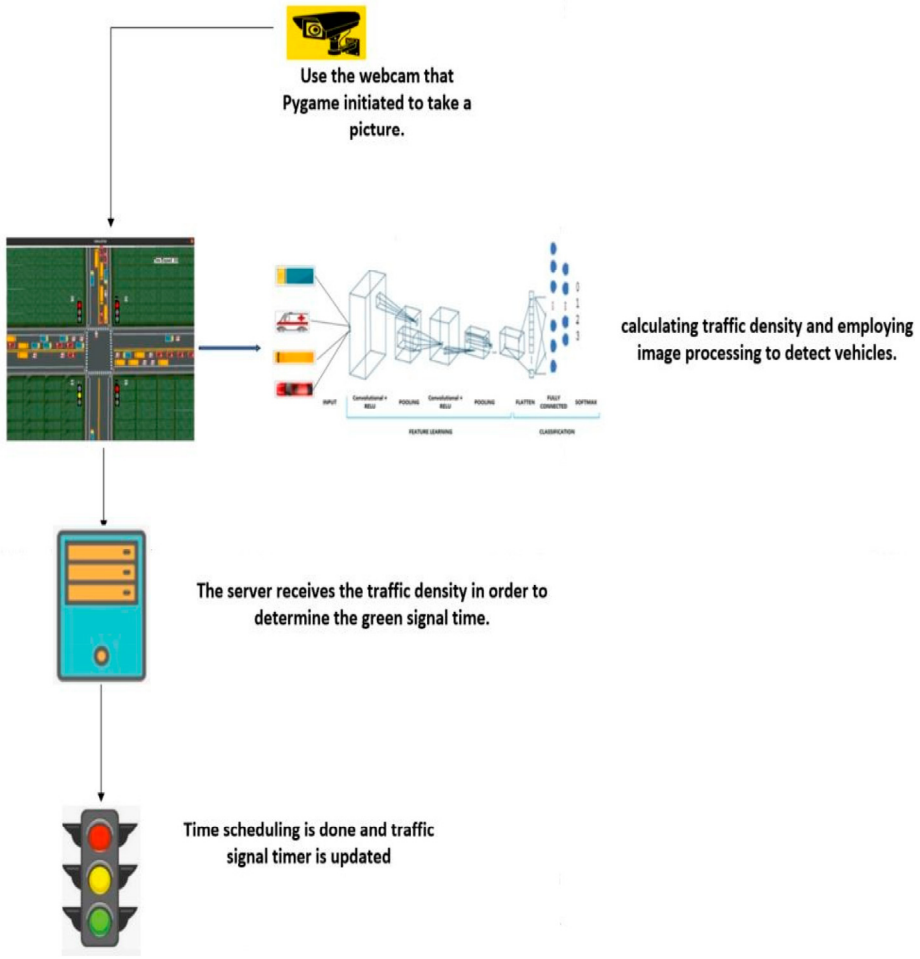

Figure 2. The proposed system model.

As shown in Figure 3, the emergency vehicle preemption program acts to interrupt the main program, and when an emergency vehicle is detected approaching the intersection using the 2D-CNN, the traffic light changes to green for a faster passage.

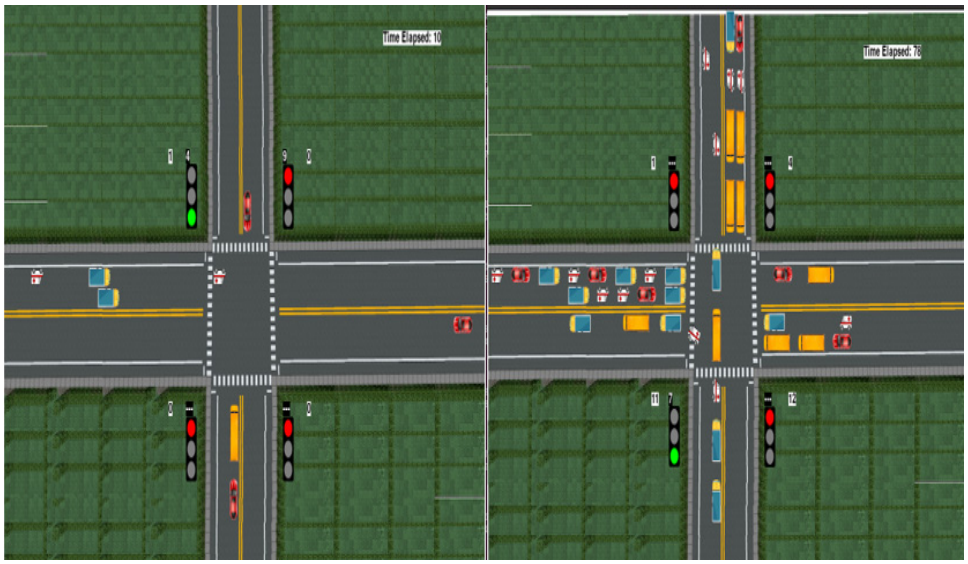

Figure 3. Vehicle count and traffic volume at an intersection.

3.1. Traffic Density Estimation and Prioritization

The created system employs vehicle detection to determine the current traffic load using data generated from traffic junction simulation in Pygame. Each class of vehicle, including cars, buses, trucks, and ambulances, has a number that is counted to determine the traffic load. The green signal timer is configured by the signal-switching algorithm using this load and a few other variables [28]. Additionally, emergency vehicles are given priority at the intersection using the 2D-CNN for vehicle classification. To show how well the system works and to contrast it with an existing adaptive traffic system built with SUMO, a simulation is produced.

A signal-switching algorithm modifies the red signal timers of other lights and correctly sets the green signal timer based on the traffic density data provided by the vehicle detecting module. Additionally, depending on timings, it alternates between the signals. The algorithm receives data about the autos that the detection module has identified. The total number of cars is then determined using this data. The detection model validates the output (0,1,2,3) of the 2D-CNN using images taken from Pygame via the webcam launched by the tool for the purpose of emergency vehicle preemption. Once the green signal time has been established and allotted, the red signal times of the additional signals are modified as needed. Any number of signals at a junction can be added or lowered using this technique. Following the traffic intensity assessment algorithm's processing time, the duration of the green light is taken into consideration. The number of lanes and the overall count of automobiles in every classification, such as cars, ambulances, and trucks, are used to compute traffic density (TD), as indicated in Equation (1):

$$\text{Traffic Density (TD)} = \frac{\sum \text{AutomobileGroup}(\text{NoOfAutomobile}_{\text{AutomobileGroup}})}{(\text{NoOfTrafficLanes} + 1)} \quad (1)$$

where:

- TD is the traffic density calculated using the simulation's current load of vehicles;
- The vehicle detection module determines the length of the string NoOfAutomobile-Group, which encodes the number of cars from each class present at the signal;
- The intersection's lane count is given by the parameter NoOfTrafficLanes.

The method sets the default time for the first signal of the first cycle when it is first run, and it uses traffic density (TD) to determine the timings of all subsequent signals as well as all other signals in the first cycle. The countdown to the red light of the next green signal is approaching five seconds, or the timer for the green signal is approaching five seconds. Detecting vehicles traveling in both directions is the responsibility of a second thread, while the main thread controls the timing of the current signal. Once the outcome has been processed, the next green signal timer is programmed. The adjusted traffic light timer shown in Figure 3 makes it possible for vehicles to move through the intersection faster when they are seen approaching.

3.2. Emergency Vehicle Preemption Using a 2D-CNN

Ambulances are given the ability to control traffic lights at crossings thanks to a practice called emergency vehicle preemption. This vehicle is supposed to cross and have priority at the intersections fast and securely, even though other vehicles are still subject to a red traffic light. Pygame is used to demonstrate the operation of the traffic light's signal switching and emergency vehicle preemption system in the flowchart in Figure 4:

The procedure starts when an emergency vehicle approaches the traffic light. When an emergency vehicle is detected using the detection model and the 2D-CNN for vehicle classification, the computer extends the green light to allow it to safely cross the intersection. According to Table 2, every vehicle has a "type" allocated to it with the program (0—car, 1—bus, 2—truck, 3—ambulance).

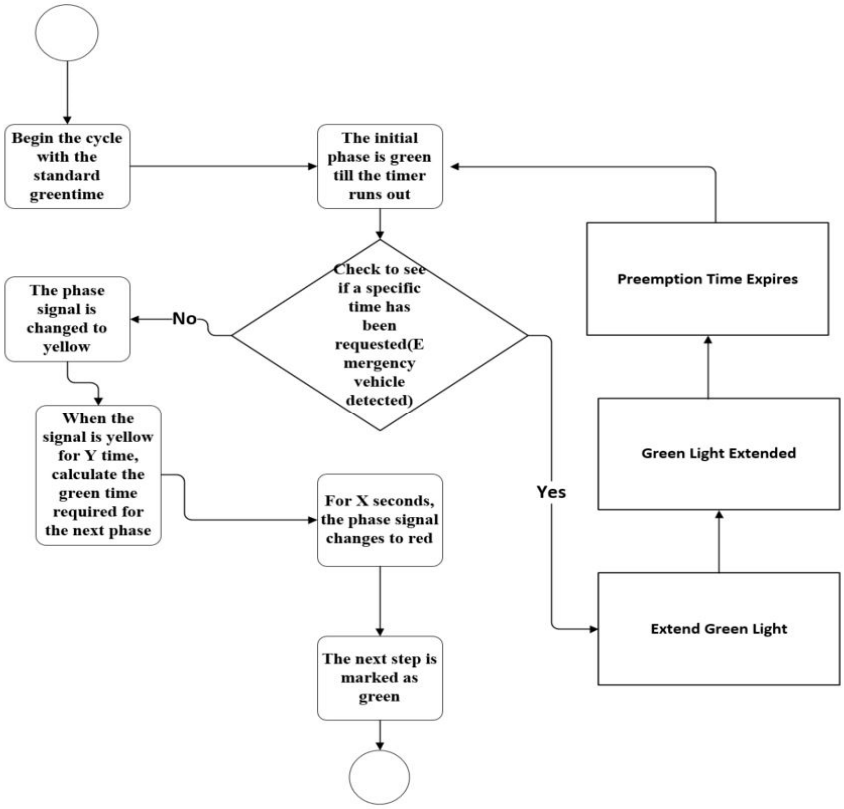

Figure 4. Diagram for emergency vehicle preemption and Pygame-based traffic light signal switching.

Table 2. The Pygame simulation engine's categorization of vehicles.

Vehicle Type	Vehicle Groups	Average Speed
0	car	2.25 m/s
1	bus	1.8 m/s
2	truck	1.8 m/s
3	ambulance	2.5 m/s

Based on the output of the 2D-CNN model displayed in Figure 5, the computer can recognize emergency vehicles using the vehicle type that has been provided to it (3-ambulance) and adjust the traffic signal as necessary. The system enters the "Green Light Extended" state when the green light has been prolonged, as seen in Figure 4.

The collection of photos of various vehicles, including cars, trucks, buses, and ambulances, is divided into training datasets and test datasets. To ensure uniformity, every image is scaled to a fixed size and changed to grayscale for simpler processing. As shown in Figure 5, the CNN model is fed different image labels as input (e.g., car label 0, bus label 1, truck label 2, and ambulance label 3), and it is then trained to learn over time using Keras and OpenCV as part of the supervised learning technique The model was trained using 100 images with 100 epochs, and the detection algorithm built using Pygame uses the output of either 0, 1, 2, or 3 from the model after it has been compiled to control the traffic logic and extend the time of the traffic green light. Figure 5 displays the CNN for picture categorization.

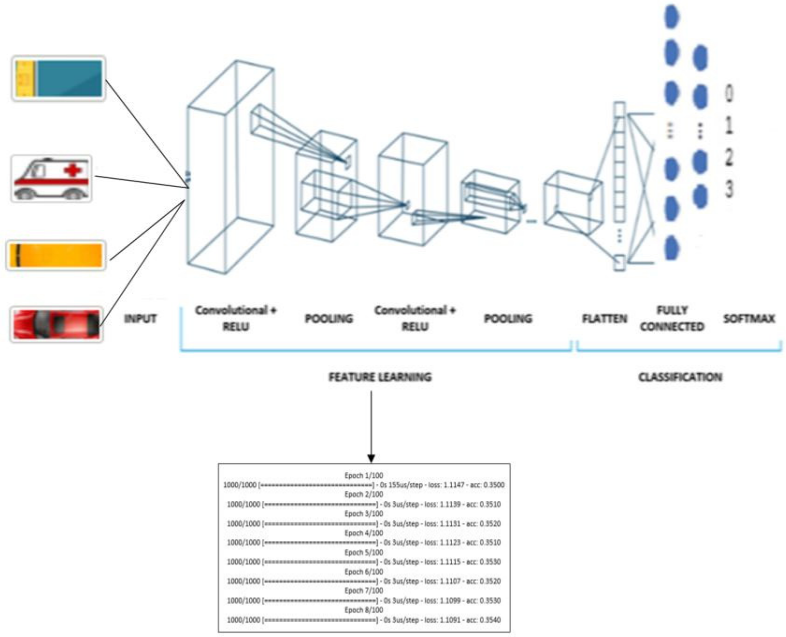

Figure 5. Categorization of vehicles using a 2D-CNN.

Images are captured from Pygame with the fixed size and saved as ".png"; then, the images captured from Pygame are used to test the 2D-CNN model for real-time classification. The program uses the output from the trained CNN model (0,1,2, and 3) as a condition to either externalize the greenlight if the output is (3—ambulance) for emergency vehicle preemption or change it to red if the output is (0—car, 1—bus, and 2—truck). Below is the algorithm used for vehicle classification using the CNN and Keras:

- Import various python libraries (TensorFlow, Numpy, cv2);
- Make a folder containing the labels for the training and testing datasets;
- Divide the features and labels into sets for testing, validation, and training;
- Prepare the images by applying effects such as grayscale conversion, image augmentation, and dataset normalization;
- Make an image generator so that there are enough images available for use;
- A convolution and a max pooling layer should be added, along with several hidden layers. Depending on the number of features required, flatten and dropout;
- For process optimization, use the ReLU activation function and Adam optimizer;
- The accuracy score of the evaluation metric should be used to assess the neural network's performance;
- The accuracy score will indicate how well the CNN model operates on the test dataset.

In Figure 6, the flow conversation used to train the 2D-CNN model is displayed. The evaluation score can be derived using Equation (2) [29].

$$\text{Score} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \tag{2}$$

where:

- True positive is denoted by TP and false positive is represented by FP;
- The number of vehicles that were accurately detected is indicated by false negative (FN);

The ReLU is utilized for activation. Assuming x as the input to the neurons, the activation function in deep neural networks calculates the output, as depicted in Equation (3).

$$f(x) = y = \max(0,\ x) \tag{3}$$

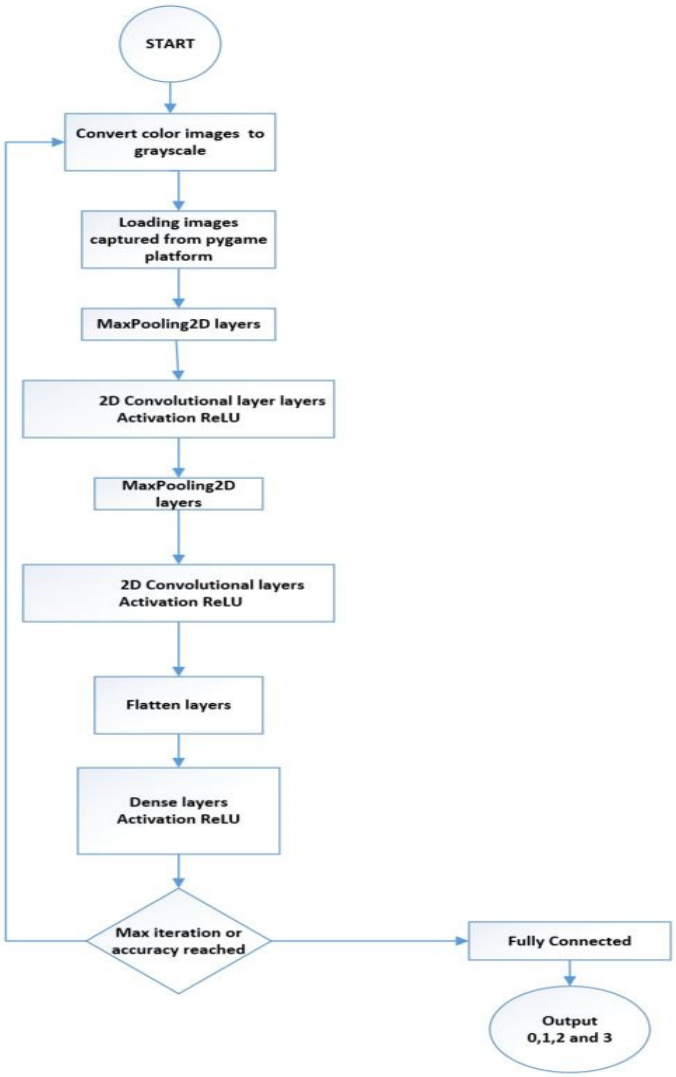

Figure 6. A 2D-CNN model's training flowchart.

A 2D-CNN model minimizes the mean absolute error (MAE) to forecast traffic between the estimated value $\hat{y}_n$ and the actual value y_n. The loss function can be obtained using the formula given in Equation (4).

$$MAE = \frac{1}{N}\sum_{n=1}^{N} |y_n - \hat{y}_n| \tag{4}$$

A preemption timer is initiated as the green light is extended to track how long it remains on. The system switches back to its standard green light cycle after the preemption period is up, allowing other directions to pass through the intersection. After the ordinary traffic light cycle, the system resumes regular functioning until the next emergency vehicle is detected. Throughout this whole process, the primary thread is monitoring how long the current green signal will last. The algorithm determines the next signal, which turns green for the predetermined period when the green timer on the current signal hits zero. Based on the percentage of each class of vehicles that were present at a signal and considering the typical beginning speeds, ambulance detection, and acceleration times of cars, the right green signal time was chosen. The following step is to use Equation (5) to calculate the time until the traffic light turns green.

$$\text{TGST} = \frac{\sum \text{AutomobileGroup}(\text{NoOfAuotmobile}_{\text{AuotmobileGroup}} * \text{AverageTme}_{\text{AutomobileGroup}})}{(\text{NoOfTrafficLanes} + 1)} \tag{5}$$

The average time it takes each vehicle class to cross a junction is estimated, where:

- TGST is the traffic signal time for the green light;
- The string number of automobile group encodes, as decided by the vehicle detection module, the quantity of each kind of vehicle at the signal;
- Average time measures the typical amount of time that it takes a vehicle in that group to cross an intersection;
- The intersection's lane configuration is the number of lanes.

The typical amount of time needed for each kind of vehicle to cross a junction varies depending on intersection characteristics and vehicle types, like emergency vehicles. The signals are repeatedly swapped between the lanes with the highest and lowest densities. The yellow signals have been taken into consideration, as well as the present signal configuration. The signals are shown in the following order: red, green, yellow, and then red. Figure 7 below demonstrate the algorithm of the program.

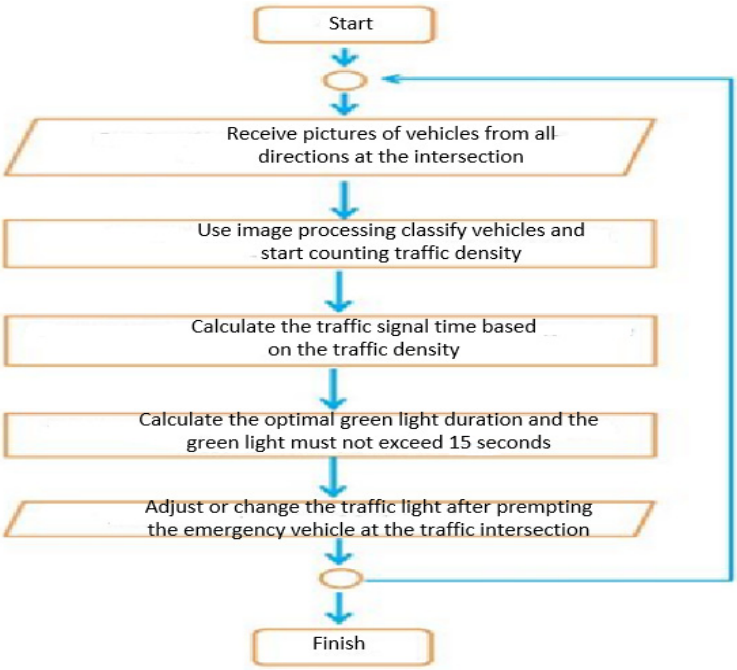

Figure 7. Algorithm of the simulated program.

After the simulation, the following equation can be used to calculate the vehicle time waiting ($V\overline{T}_w$) and vehicle time lost ($V\overline{T}_L$). The time delay (T_D) is the length of time that a traffic signal was green and no vehicles—including cars, lorries, buses, and ambulances—passed through a junction. Equation (6) can be used to calculate the average lost time $\overline{T}_L$ for each type of vehicle at the intersection, including cars, trucks, buses, and ambulances.

$$V\overline{T}_L = \frac{T_D}{N} \tag{6}$$

where:

- Total delay (T_D) is the total of all car delays over a certain period measured in seconds;
- Total number of vehicles (N) is the entire count of automobiles that traversed the junction or segment of road during the same duration.

Vehicle queue length (VQ_L) refers to the period that cars, trucks, buses, and ambulances take at the crossroad while awaiting the green light. The typical wait time $V\overline{T}_w$ is

derived by dividing the vehicles queue length as stated in Equation (7) by the flow rate of the cars, trucks, buses, and ambulances minus the departure rate.

$$\overline{VT}_w = \frac{VQ_L}{N} \tag{7}$$

where:

- Vehicle queue length (VQ_L) is the number of cars in the line at any moment;
- N is the total number of vehicles in the entire simulation.

Every car logs its location, speed, amount of waiting time, and starting time. The simulation loop updates all vehicles, renders the current state, verifies simulation end conditions, and extends simulation time. It computes and adds the lost time and queue length for each vehicle when it arrives at a destination or meets certain criteria. The average lost time and average queue length are determined by dividing the total lost time and total queue length by the total number of vehicles that have completed their journey. The pseudo-code for calculating average lost time and vehicle queue length within the simulated program is shown by Algorithm 1 below.

Algorithm 1 Vehicle delay

```
Total vehicles = 0
Total loss time = 0
Total queue length = 0
class Automobile: location
waiting_time, speed, and waiting_start_time
function update():
        If position = waiting_location, then
                waiting_start_time = current_time if waiting_start_time is null.
          otherwise:
          if waiting_start_time is not null:
wait
vehicle movement based on speed
start up the cars
Set up the simulation's parameters.
As the simulation is running, for every vehicle in the vehicles:
vehicle.update()
display the state of the simulation
the check_simulation_end_condition function
            If some vehicle has finished, then:
                    total vehicles += 1
                        total lost time += calculate
lost time for completed vehicle()
total queue length += calculate queue length for completed vehicle()
advance time for simulation
Total lost time/total vehicles equals average lost time.
Total queue length/total vehicles equals average queue length.
```

Vehicle count and lane priority traffic are regulated using a traffic light control system signal and gauge density and give priority to lanes with more vehicles by detecting and counting the vehicles as they pass through the intersection. In Pygame, vehicles are generated randomly at predetermined intervals of time.

4. Simulation, Experimental Data, and Results Analyses

Traffic simulation's constituent parts are their infrastructure and the traffic that uses it. While traffic is made up of vehicles, such as trucks, buses, pedestrians, bicycles, passenger cars, trains, and trams, infrastructure components include streets, traffic signals, railroad lines, induction loop detectors, etc. Many metrics, including queue length, sys-

tem throughput, travel time, delay, etc., are commonly used to assess the effectiveness of traffic systems.

4.1. Simulation Tools

All the simulations are executed using the Anaconda Navigator tool called Spyder. To thoroughly compare the outcomes of the two systems, ten simulations of both adaptive systems, each lasting five minutes, were run side by side, as shown in Figure 8.

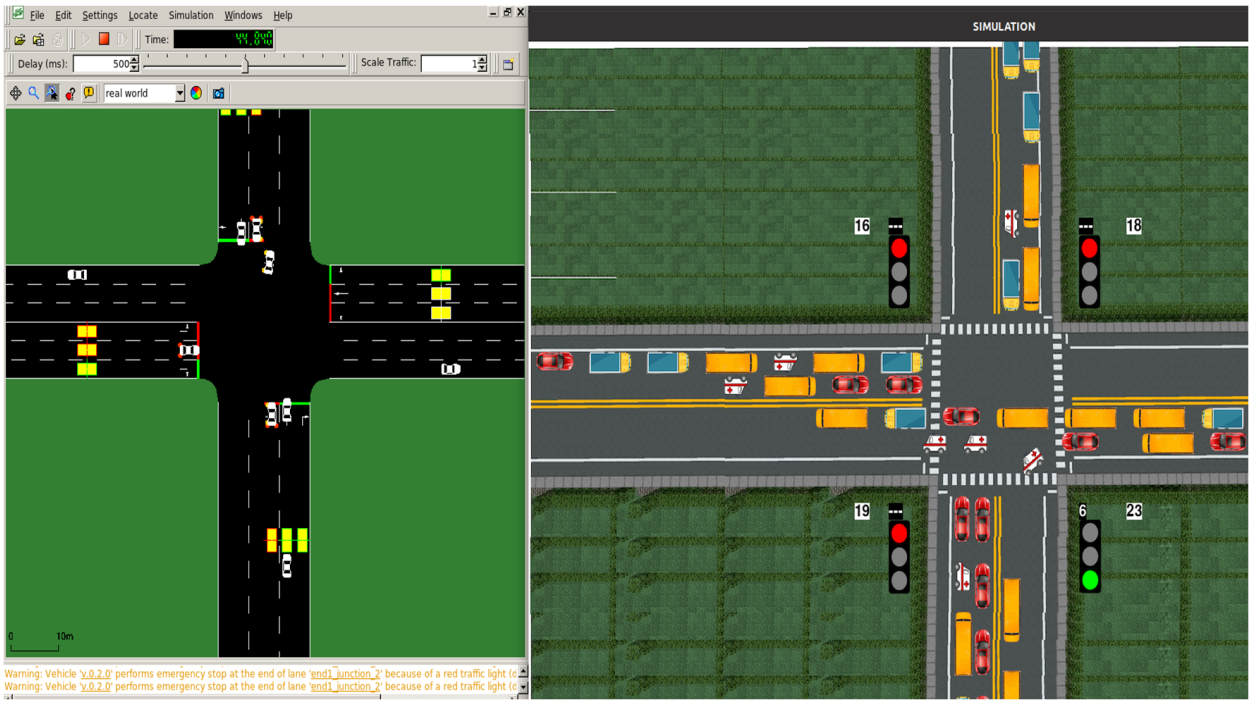

Figure 8. SUMO and Pygame simulation platforms.

A simulation of actual traffic was created using Pygame. It facilitates system visualization and comparison with an existing static system. Pygame is a collection of cross-platform Python utilities for game development. Pygame is incredibly portable, running on nearly every platform and operating system. It has an LGPL license and is free to use [30]. The traffic simulation of the proposed model using Pygame was simulated on the Ubuntu platform. The suggested method aims to utilize the resources for time allocation and traffic management as effectively as possible. No matter how many vehicles are present, every vehicle must pass over the crossing [30]. Consequently, navigating through traffic will always require the least amount of time. In SUMO, vehicles are randomly generated and loaded into the network at a random speed. In order to dramatically reduce individual waiting times, the method seeks to schedule cars at a junction by calculating the number of vehicles on each road and spreading this traffic using a variety of criteria [30]. Vehicle data for departure, duration, etc., are recorded in the output.xml file during simulation.

4.2. Experimental Data

The vehicle test images from the Pygame simulation were used to train the model, and it achieved a 96% accuracy rate in predicting the type of vehicle. Table 3 below shows all the parameters used by the 2D-CNN during training to achieve a high accuracy rate.

Table 3. Architecture details of the 2D Convolutional Neural Network.

Number	Architecture Details
1	Input image size (640, 480)
2	Total number of layers: 2 Total MaxPooling2D layers used: 2 Total fully connected layers used: 3 Activation layers: 4 Dropout layer: 2
3	Kernal size at each Conv2D layer: 3 × 3
4	Pool size at each maxPooling2D layer: (2,2)
5	Output class labels: 3

Table 4 shows the output of the 2D-CNN module during training for vehicle classification, accuracy, and time loss.

Table 4. The 2D-CCN accuracy and loss with epoch.

Epoch	Time Taken(s)	Accuracy	Loss
1	11	0.478	1.998
2	9	0.800	0.678
3	10	0.877	0.423
4	11	0.906	0.312
5	11	0.921	0.273
6	10	0.923	0.255
7	11	0.935	0.232
8	12	0.936	0.228
9	11	0.938	0.233
10	9	0.939	0.214
11	11	0.944	0.211
12	11	0.947	0.206
13	9	0.945	0.216
14	11	0.951	0.201
15	10	0.953	0.186

The accuracy and time loss during the training of the labeled dataset is shown in Figure 9.

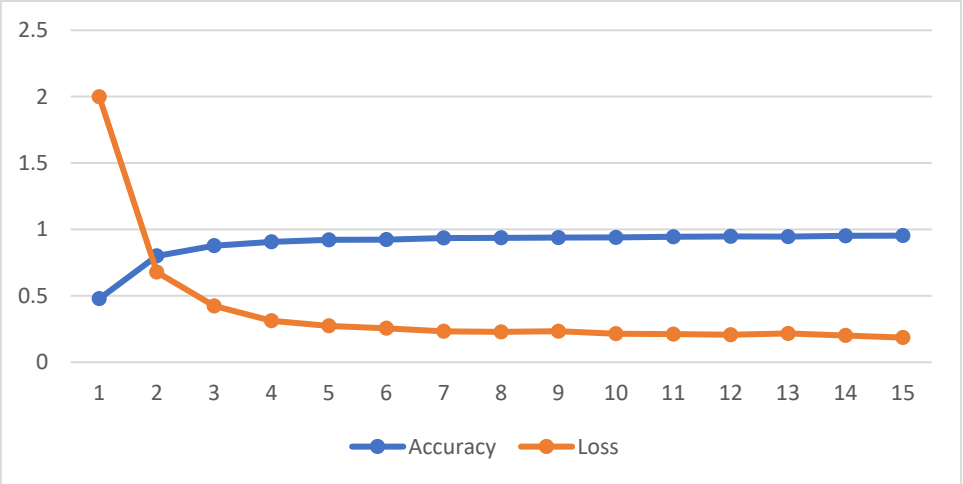

Figure 9. Comparison of the time loss and accuracy for the trained dataset.

4.3. Simulation Results and Analysis

The suggested adaptive traffic light control system built with Pygame and the adaptive traffic signal system built with SUMO were both evaluated in a real-time setting with the same amount of time (300 s). In this research, average vehicle throughput was compared with a system built using SUMO. Ten simulations of both adaptive systems, each lasting 5 min, were run side by side to comprehensively compare the results of the two systems. A performance indicator was the number of cars that could cross the intersection in a certain period. This has an impact on both the length of other signal waits, as well as vehicle wait times. Multiple vehicles passing in opposite directions and the total number of passing vehicles were used to tabulate the data.

(a) Simulation using SUMO

Figure 10 shows random vehicles generated using SUMO after 300 s for lane priority using the adaptive traffic light control module.

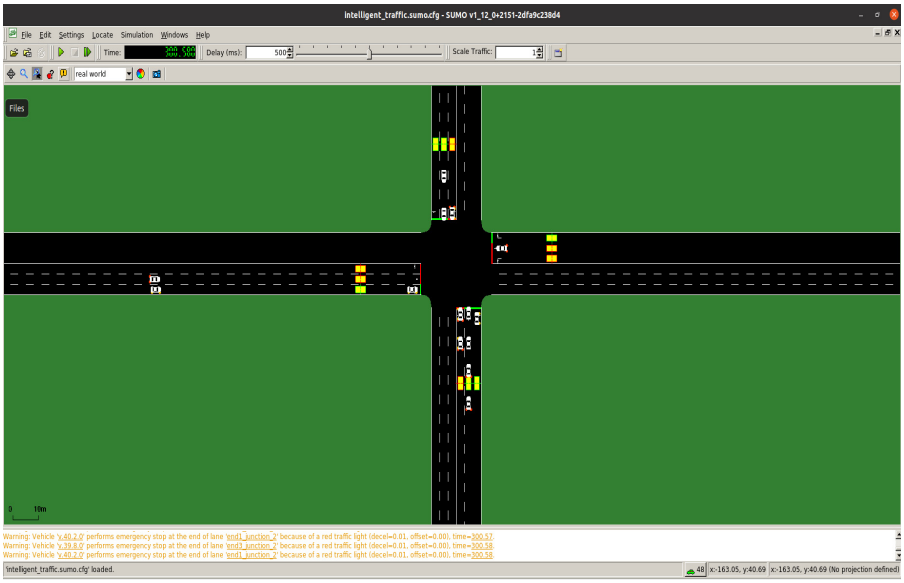

Figure 10. Simulation of an adaptive traffic intersection using SUMO.

After 300 s, the printed output shows both the total number of vehicles loaded in the simulation passing through the junction and the vehicle speeds for each vehicle class. Since traffic density varies depending on the time of day (morning, during the day, in the afternoons or evenings), the data above for an adaptive traffic signal can reflect a simulation of real-world traffic. Another example of how an adaptive traffic signal might increase vehicle throughput at a traffic intersection and save both drivers' and passengers' time is shown in Table 5.

Table 5. Results for the adaptive traffic light using SUMO.

Number	Direction 1	Direction 2	Direction 3	Direction 4	Total
1	48	46	43	57	194
2	43	43	51	45	182
3	44	51	45	50	190
4	52	50	55	37	194
5	43	47	51	44	185
6	50	54	38	50	192
7	54	61	38	35	188
8	42	48	55	51	196
9	48	41	51	43	183
10	53	49	43	41	186

(b) Simulation using Pygame

Various test simulations were used to test a vehicle detection module, including ones where the number of randomly generated vehicles passing the intersection varied. The algorithm can detect and count vehicles crossing the intersection. Both simulations were run, and the total number of vehicles going through the intersection in all directions, as well as the number of vehicles traveling in each direction, were counted. Figure 11 shows a lane priority adaptive traffic light that was captured after 300 s.

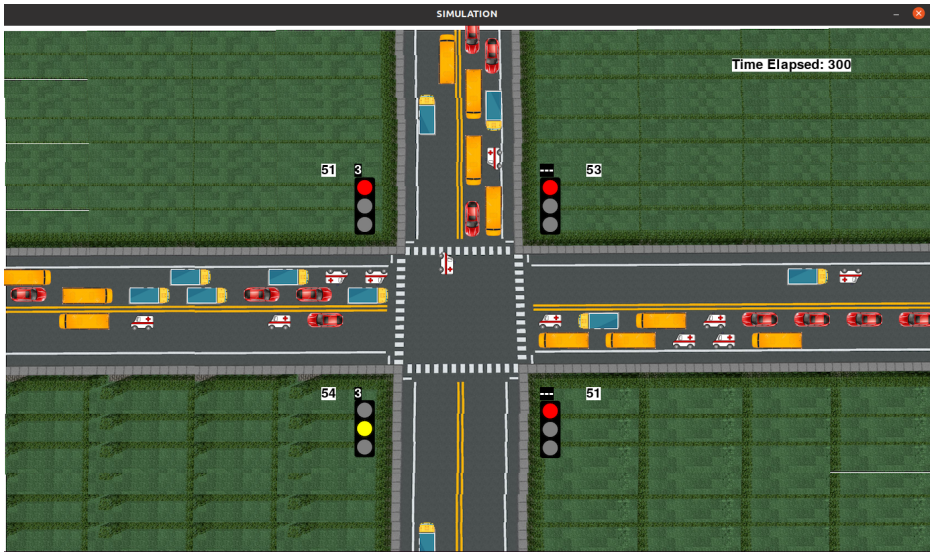

Figure 11. The simulation of the suggested system.

The printed output after 300 s displays the overall number of vehicles going through the junction, as well as the number of vehicles traveling through the junction in each direction, as shown in Figure 12.

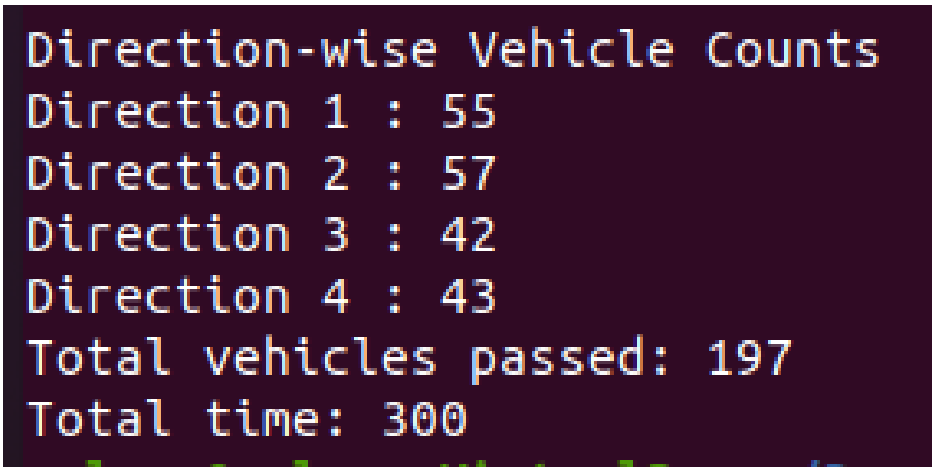

Figure 12. Vehicle throughput and direction for the proposed adaptive traffic light.

The above data for an adaptive traffic light can reflect real-life traffic simulation since traffic density varies depending on the time of the day (morning, during the day, and in the afternoons or evenings). Table 6 also reflects how an adaptive traffic light can provide more throughput of vehicles at the traffic intersection and save time for the drivers and passengers.

Table 6. Results of the proposed adaptive system simulation.

Number	Direction 1	Direction 2	Direction 3	Direction 4	Total
1	55	57	42	43	197
2	43	43	57	45	188
3	49	51	45	55	200
4	55	53	56	47	211
5	43	47	51	48	189
6	46	54	53	54	207
7	44	61	40	33	178
8	42	46	55	51	194
9	48	41	51	44	184
10	58	50	43	41	192

(c) Results Analysis

All 10 simulations with different traffic density, when comparing the results on the traffic throughput, shows that the suggested technique is more effective than the adaptive traffic light using SUMO. Emergency vehicles also spend less time at traffic intersections thanks to emergency vehicle detection in real time using the 2D-CNN model and time adjustment algorithm used in this system. The graph shows the two systems' performance at each simulation using the data in Tables 5 and 6, which were generated during the simulation of traffic lights using Pygame and SUMO, as shown in Figure 13.

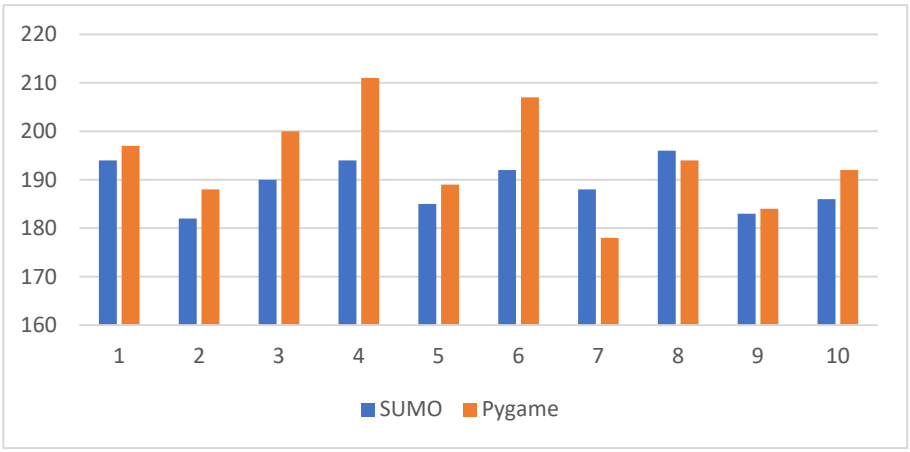

Figure 13. Correlation of adaptive traffic systems.

Regardless of the distribution, the proposed adaptive system outperforms the adaptive traffic light system in SUMO almost every time. The allocation over all four lanes was set to random in these scenarios. This is a common type of traffic distribution in real-world situations. Emergency vehicles have a high throughput at the intersection using the proposed model.

5. Conclusions

The proposed method ensures that the direction with the most traffic receives a longer green signal than the direction with the least traffic by adjusting the length of the signal based on the volume of traffic at the light. As a result, the proposed system design cleared vehicles 90% faster than the adaptive traffic light system using SUMO. This will reduce gasoline consumption and emissions by eliminating undesired delays, congestion, and waiting times. Additionally, the proposed model can recognize emergency cars and modify the signal timers for quicker passage of the emergency vehicles. Using the 2D-CNN module, emergencies can be classified and given priority at the intersection. The hardware design for this project has been saved for future use. The initiative could be expanded in the

future to improve traffic management and reduce congestion. The detection of accidents or malfunctions is a feature that can be added to improve the project. Because a variety of risky situations, such as angle and left-turn crashes, occur frequently at intersections, they are more likely to result in catastrophic accidents. To save lives and damage, as well as reduce traffic and delays, it is crucial to detect accidents accurately and quickly at junctions. This can be performed by filtering out parked cars and identifying cars that are stopped for an extended amount of time in an uncomfortable place, like the middle of the road. Additionally, vehicles that break traffic laws can be found. Setting a violation line and taking a snapshot when it is crossed while the signal is red may allow for the detection of cars that are running red lights in an image or video feed. Techniques for image processing can help with this. Syncing traffic lights at different crossings will benefit commuters since once a car enters the roadway, there may be less pause, and emergency vehicles can be detected at any point in time to be given priority, not only when they approach the traffic intersection. Also, if emergency vehicles arrive from all directions, the system can be improved to handle situations in which an algorithm calculates the arrival time and urgency of each emergency vehicle and allows that particular emergency vehicle to go first.

Author Contributions: M.J.M.: conceptualization, data analysis, investigation, writing, and experiments; C.T.; P.A. and M.N.: supervising, reviewing, and editing. All authors have read and agreed to the published version of the manuscript.

Funding: The Tshwane University of Technology funded this study.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Deepajothi, S.; Rajan, D.P.; Karthikeyan, P.; Velliangiri, S. Intelligent traffic management for emergency vehicles using convolutional neural network. In Proceedings of the 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 19–20 March 2021; pp. 853–857.
2. Cristiani, A.L.; Immich, R.; Akabane, A.T.; Madeira, E.R.M.; Villas, L.A.; Meneguette, R.I. Atrip: Architecture for traffic classification based on image processing. *Vehicles* **2020**, *2*, 303–317. [CrossRef]
3. Mall, P.K.; Narayan, V.; Pramanik, S.; Srivastava, S.; Faiz, M.; Sriramulu, S.; Kumar, M.N. FuzzyNet-Based Modelling Smart Traffic System in Smart Cities Using Deep Learning Models. In *Handbook of Research on Data-Driven Mathematical Modeling in Smart Cities*; IGI Global: Genval, Belgium, 2023; pp. 76–95.
4. Zhou, B.; Cao, J.; Zeng, X.; Wu, H. Adaptive traffic light control in wireless sensor network-based intelligent transportation system. In Proceedings of the 2010 IEEE 72nd Vehicular Technology Conference-Fall, Ottawa, ON, Canada, 6–9 September 2010; pp. 1–5.
5. Cano, M.-D.; Sanchez-Iborra, R.; Freire-Viteri, B.; Garcia-Sanchez, A.-J.; Garcia-Sanchez, F.; Garcia-Haro, J. A self-adaptive approach for traffic lights control in an urban network. In Proceedings of the 2017 19th International Conference on Transparent Optical Networks (ICTON), Girona, Spain, 2–6 July 2017; pp. 1–4.
6. Zavala, B.; Alférez, G.H. Proactive control of traffic in smart cities. In Proceedings of the International Conference on Artificial Intelligence (ICAI), Las Vegas, NV, USA, 27–30 July 2015; p. 604.
7. Guerrero-Ibáñez, J.; Zeadally, S.; Contreras-Castillo, J. Sensor technologies for intelligent transportation systems. *Sensors* **2018**, *18*, 1212. [CrossRef] [PubMed]
8. Ahmed, F.; Hawas, Y. An integrated real-time traffic signal system for transit signal priority, incident detection and congestion management. *Transp. Res. Part C Emerg. Technol.* **2015**, *60*, 52–76. [CrossRef]
9. Kareem, D.M. Coverage in Wireless Sensor Networks. Master's Thesis, Çankaya University, Çankaya, Turkey, 2015.
10. bin Che Mansor, M.A.H.; Kamal, N.A.M.; bin Baharom, M.H.; bin Zainol, M.A. Emergency Vehicle Type Classification using Convolutional Neural Network. In Proceedings of the 2021 IEEE International Conference on Automatic Control & Intelligent Systems (I2CACIS), Shah Alam, Malaysia, 26 June 2021; pp. 126–129.
11. Chen, L.; Li, S.; Bai, Q.; Yang, J.; Jiang, S.; Miao, Y. Review of image classification algorithms based on convolutional neural networks. *Remote Sens.* **2021**, *13*, 4712. [CrossRef]
12. Survarachakan, S.; Pelanis, E.; Khan, Z.A.; Kumar, R.P.; Edwin, B.; Lindseth, F. Effects of enhancement on deep learning based hepatic vessel segmentation. *Electronics* **2021**, *10*, 1165. [CrossRef]
13. Rajasekar, T.; Mohanraj, P.; Abishek, R.; Haries, M. Adaptive Traffic Congestion Control Approach with Emergency Vehicle Protocol. In Proceedings of the 2023 8th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 1–3 June 2023; pp. 613–620.

14. Pamula, T. Road traffic conditions classification based on multilevel filtering of image content using convolutional neural networks. *IEEE Intell. Transp. Syst. Mag.* **2018**, *10*, 11–21. [CrossRef]
15. Li, Z.; Liu, F.; Yang, W.; Peng, S.; Zhou, J. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *33*, 6999–7019. [CrossRef] [PubMed]
16. Sun, X.; Lin, K.; Jiao, P.; Lu, H. Signal timing optimization model based on bus priority. *Information* **2020**, *11*, 325. [CrossRef]
17. Jia, X.; Sun, C. Research on Intelligent Monitoring Technology of Traffic Flow Based on Computer Vision. In Proceedings of the 2023 IEEE International Conference on Sensors, Electronics and Computer Engineering (ICSECE), Jinzhou, China, 18–20 August 2023; pp. 557–561.
18. Yogheshwaran, M.; Praveenkumar, D.; Pravin, S.; Manikandan, P.; Saravanan, D.S. IoT based intelligent traffic control system. *Int. J. Eng. Technol. Res. Manag.* **2020**, *4*, 59–63.
19. Aggarwal, S.; Bali, R.S. A vehicular dynamics based technique for efficient traffic management. In Proceedings of the 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Kochi, India, 10–13 August 2015; pp. 2133–2138.
20. Cheng, J.; Wu, W.; Cao, J.; Li, K. Fuzzy group-based intersection control via vehicular networks for smart transportations. *IEEE Trans. Ind. Inform.* **2016**, *13*, 751–758. [CrossRef]
21. Rashid, H.; Ashrafi, M.J.F.; Azizi, M.; Heydarinezhad, M.R. Intelligent traffic light control based on clustering using vehicular ad-hoc networks. In Proceedings of the 2015 7th Conference on Information and Knowledge Technology (IKT), Urmia, Iran, 26–28 May 2015; pp. 1–6.
22. Younes, M.B.; Boukerche, A. Intelligent traffic light controlling algorithms using vehicular networks. *IEEE Trans. Veh. Technol.* **2015**, *65*, 5887–5899. [CrossRef]
23. Guo, Q.; Li, L.; Ban, X.J. Urban traffic signal control with connected and automated vehicles: A survey. *Transp. Res. Part C Emerg. Technol.* **2019**, *101*, 313–334. [CrossRef]
24. Rosayyan, P.; Paul, J.; Subramaniam, S.; Ganesan, S.I. An optimal control strategy for emergency vehicle priority system in smart cities using edge computing and IOT sensors. *Meas. Sens.* **2023**, *26*, 100697. [CrossRef]
25. Nellore, K.; Hancke, G.P. Traffic management for emergency vehicle priority based on visual sensing. *Sensors* **2016**, *16*, 1892. [CrossRef] [PubMed]
26. Kavitha, Y.; Satyanarayana, P.; Mirza, S.S. Sensor based traffic signal pre-emption for emergency vehicles using efficient short-range communication network. *Meas. Sens.* **2023**, *28*, 100830. [CrossRef]
27. Chung, J.; Kim, G.; Sohn, K. Transferability of a Convolutional Neural Network (CNN) to Measure Traffic Density. *Electronics* **2021**, *10*, 1189. [CrossRef]
28. Mandal, V.; Mussah, A.R.; Jin, P.; Adu-Gyamfi, Y. Artificial intelligence-enabled traffic monitoring system. *Sustainability* **2020**, *12*, 9177. [CrossRef]
29. Luo, H.; Yang, Y.; Tong, B.; Wu, F.; Fan, B. Traffic sign recognition using a multi-task convolutional neural network. *IEEE Trans. Intell. Transp. Syst.* **2017**, *19*, 1100–1111. [CrossRef]
30. Li, Q.; Peng, Z.; Feng, L.; Duan, C.; Mo, W.; Zhou, B. ScenarioNet: Open-Source Platform for Large-Scale Traffic Scenario Simulation and Modeling. *arXiv* **2023**, arXiv:2306.12241.