Romanian Journal of Petroleum & Gas Technology
Vol. V (LXXVI) – No. 1/2024

INTEGRATED CONVOLUTIONAL NEURAL NETWORKS APPLICATION FOR ROAD TRAFFIC EFFICIENCY

Ștefan Liviu-Andrei [1]

Cărbureanu Mădălina [2]

[1] Romania
[2] Petroleum-Gas University of Ploiesti, Romania
email: mcarbureanu@upg-ploiesti.ro

DOI: 10.51865/JPGT.2024.01.12

ABSTRACT

Road traffic is a problem that every country has to face for the benefit of its population. Everything related to the transportation of people, goods, and services can be delayed due to a deplorable infrastructure. From minor tasks like individual shopping to important services like ambulances and fire trucks, all vehicles in Romania will eventually pass through a traffic light-controlled intersection that is not managed in real-time and that is also highly inefficient. At such a location, it is critical for traffic flow to be as smooth as possible to prevent time losses, and for traffic information to be stored into a cloud for process optimization. The purpose of the developed application is to reduce the real-time traffic jams to allow the optimal vehicles movement in safe conditions for all traffic participants. Using deep learning (DL) and artificial intelligence (AI) techniques, the application is based on two convolutional neural networks (CNN's), respectively one for real-time detection of cars and people, and another one for real-time detection of cars license plates. The cars detections with CNN's is important for road traffic optimization, for the efficient pedestrian detection necessary for crosswalks, and the car licence plates detection necessary for preventing traffic violations. The developed application contains several modules that operate based on deep learning and devices that communicate with each other, ensuring efficient data saving in cloud, data visualization, real-time traffic light control, and the information processing from traffic cameras.

Keywords: Deep learning, convolutional neural networks, road traffic, real-time detection, optimization.

INTRODUCTION

In Bucharest, was observed that a driver loses between ninety one and one hundred and thirty-four hours per year due to congestion, globally being in eighth place in terms of traffic. One of the contributing factors is the low level of modernization of the road infrastructure, respectively from the total of four hundred and fifty intersections, only two hundred and sixty are intelligent [15, 16, 17, 20]. From these statistics, it can be conclude that the traffic in Bucharest and in many other cities can be improved by investing in efficient infrastructure. There are already traffic management systems that are detecting the urban traffic and that are providing solutions for public transport and special vehicles

[4, 11]. For instance, intelligent traffic lights are reducing the necessary transportation time by approximately twenty-five percent, are reducing the time spent at traffic lights by forty percent, and emissions by twenty percent [19]. In Bucharest, the use of intelligent traffic lights has reduced the traffic by values between fifteen and eighty percent in one hundred seventy six intersections [21].

As existing solutions on market, UTI has created a system (whose interface is presented in Figure 1) with the following benefits: the traffic improvement by fifteen and eighty percent, the public transport services improvement by departure and arrival schedules monitoring, the real-time warning of the faulty equipment's from intersections and their immediate replacement or repair, the improving of the intervention time for emergency vehicles (police, ambulance and firefighters), the traffic incidents reduction and finally, the pollution reduction [21].

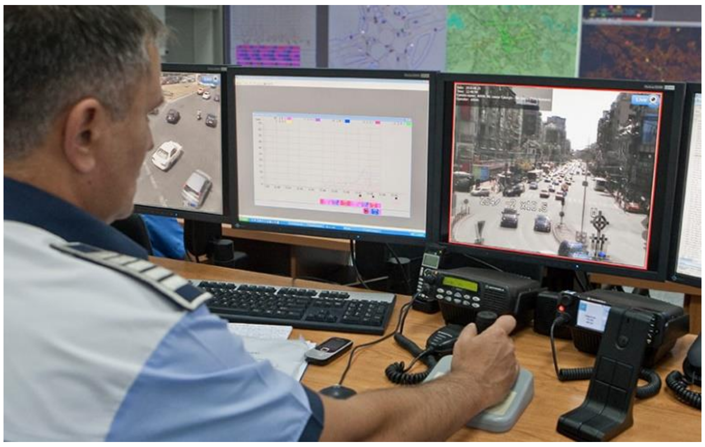


Figure 1. *The system created by UTI* [21]

Another system created for traffic management is the Dutch system presented in Figure 2, which dynamically changes the traffic light colour, giving each participant four seconds for the green light. It manages to increase the intersection efficiency from ten to fifteen percent, the average waiting time for a traffic participant being less than twenty seconds, while the green light duration is around four seconds [13].

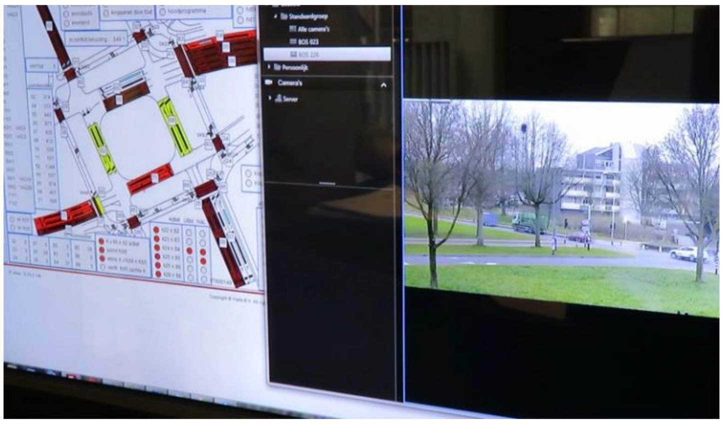


Figure 2. *The Dutch system interface* [13]

The system aims to offer an alternative for areas with road traffic primarily related to arterial roads intersecting with secondary streets and pedestrians. It includes several components that are communicating with each other to gather traffic information and then to process it, displays critical information, and adjust the traffic lights for an efficient traffic management using CNN's. Respectively, CNN's are used for cars and pedestrians detection and also for cars license plates detection. CNN's are the best artificial intelligence technique for this case, as the other algorithms cannot be used to solve this problem [7, 8].

CNNs, is a type of deep learning neural network, consist of layers of interconnected nodes with weights and thresholds. Signals are transmitted if the aggregated values exceed the threshold. Key layers in CNNs include convolutional, pooling, and fully-connected layers [1, 5, 6, 14, 18]. Convolutional layers apply filters to images to detect specific patterns, with outputs which varies depending on the filter, stride (movement over pixels), and padding (handling filter edges). Pooling layers reduce image size, complexity, and increase efficiency through max pooling (taking the maximum pixel value) or average pooling (taking the average pixel value) [14, 18]. Fully-connected or dense layers are the layers that connect with all other nodes in the previous layers. These are used for classification, utilizing functions such as ReLU (Rectified Linear Unit), Sigmoid, and Softmax [14, 18].

THE ROAD TRAFFIC EFFICIENCY APPLICATION DEVELOPMENT

The application uses cloud data storage via a server, a device communication that uses sockets and a serial port, a portable router for local communication, CCTV (Closed Circuit Television) cameras for data reception, and a locally created website for processed data visualisation. It employs two CNN's attached to the processing device. Python (for handling the system) and C (for controlling the Arduino traffic lights) as main programming languages. Were used technologies as Ultralytics', CNN's calibrated for the developed application, Proxmox Virtual Environment OS for creating a virtual machine to access cloud data with Network Attached Storage (NAS), Synology technology for storage space, a Linux Ubuntu container as a connector for Twingate technology to access the local server from any location, which supports both the NAS and web developed application. Also, it uses different types of classes and objects in its components [9, 10].

The transfer of data between the proposed application components is illustrated in Figure 3. CCTV cameras monitor are recording the traffic, sending information to a laptop that processes it based on the traffic light colour, the number of visible cars, and pedestrians wishing to cross. The detection device sends the number of cars or a pedestrian code to the Raspberry Pi, which decides the green light duration based on the traffic light's current and last colours from the Arduino. Arduino executes the commands and only sends traffic light colour changes back to the Raspberry Pi. Periodically, the detection device sends images, recorded traffic footage, and a CSV file containing details such as license plate numbers of cars that ran red lights, the time, location, and confidence level to the NAS. This information is used by the developed web application to display a graph of detections per minute and the time of incidents, with options to view images by time or to search license plates based on the entered time. In Figure 3 is presented the block diagram for the developed application.

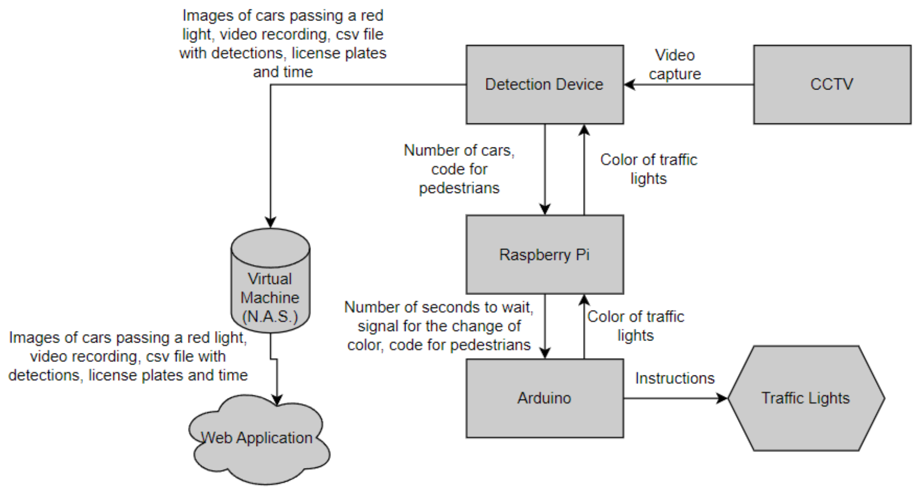

Images of cars passing a red light, video recording, csv file with detections, license plates and time
Detection Device
Video capture
CCTV
Number of cars, code for pedestrians
Color of traffic lights
Raspberry Pi
Virtual Machine (N.A.S.)
Images of cars passing a red light, video recording, csv file with detections, license plates and time
Number of seconds to wait, signal for the change of color, code for pedestrians
Color of traffic lights
Arduino
Instructions
Traffic Lights
Web Application

***Figure 3**. Block diagram for the developed road traffic efficiency application*

The detection device is the main component of the entire system being implemented in Python. It uses a convolutional neural network created by the Ultralytics company called YOLOv8, or You Only Look Once version 8. YOLOv8 was created and refined using the PyTorch library for object detection in an image and was optimized for this task by the Ultralytics team.

The convolutional neural network YOLOv8 model's architecture has the following modules:

- The Conv module includes a Conv2D layer for convolving a 2D image with the the following parameters: *k* (kernel size), *s* (stride size), *p* (padding type), and *c* (the number of color channels); it also includes a BatchNorm2D layer for normalizing a batch of images and uses the SiLU (Sigmoid weighted Linear Unit) activation function to enhance training efficiency and model generalization.
- The SPPF module comprises a Conv module and three MaxPooling2D layers for image size linear reduction; the obtained results are concatenated and passed through another Conv module.
- The Bottleneck module consists of two Conv modules, which may or may not have a shortcut to the unprocessed information from the Conv modules.
- The Detect module is split into Bounding box detection and Classification detection; each uses two Conv modules followed by a Conv2D layer; the difference between the Conv2D layers in Bounding box detection and Classification detection lies in the number of color channels, each having separate loss functions: BCE (Binary Cross Entropy) for classification and a combination of CIoU (Combined Intersection over Union) and DFL (Dual Focal Loss) for bounding box detection.
- The C2f module, the most complex one, uses a Conv module followed by a split function to divide the channels and *n* bottleneck modules, all concatenated, with the result passing through another Conv module.

This convolutional neural network developed by Ultralytics is subsequently retrained into two new networks, one for people and vehicles detection using the COCO (Common Objects in Context) dataset, and another one for license plate detection using a dataset obtained from Roboflow. The DL network for the COCO dataset was trained in twenty five epochs, while the network for the Roboflow dataset having fewer elements was trained in fifty epochs.

The vehicle detection module was considered a submodule of the Detection Device module and it continuously monitors the real-time data from CCTV cameras, scanning each frame and saving the system time into a variable when it was detected a car which turns red light. It uses a neural network to check a predefined area for correct car movement direction, ignoring cars from the opposite direction. If detection accuracy exceeds fifty percent threshold, it records the vehicle in a variable only if the secondary traffic light who controls that section is red. For cars in another region of interest, surpassing the traffic light separation line, the vehicle's coordinates and detection accuracy are saved in a separate list, and an image highlighting the vehicle in the restricted area during the red light is saved. License plate detection is performed for all vehicles that ran the red light, assigning each car a license plate based on location to ensure the coordinates match. EasyOCR is used for Optical Character Recognition (OCR) to verify the text of the license plate image in binary format for accurate reading. Finally, all detection data is written to a CSV file, and a frame highlighting vehicles that ran the red light is written to a new video. After two hundred and fifty frames, the video and CSV writing is off, data is sent to the information processing module along with generated images, and a separate file is uploaded to the cloud before starting the process again.

The pedestrian detection module is also a submodule of the Detection Device module and it continuously monitors real-time data from CCTV cameras, scanning each frame to check if a pedestrian is trying to cross the street. It verifies over two frames if the person occupies at least one-quarter of the registration area (indicating proximity to the camera) and if there are minimal differences in movement (small changes in detection coordinates). If both conditions are met, a frame highlighting the pedestrian is written to a new video. After two hundred and fifty frames, the video stops and then starts again from the beginning.

A selection of the code developed to show the decision structure behind the pedestrian traffic light colour change, where the threshold is the minimum confidence level of the network, is presented next. In this case, 0.5 or 50%, x_1, y_1, x_2, y_2 are the current element's coordinates, *past_*x_1, *past_*y_1, *past_*x_2, *past_*y_2 are the previous element's coordinates, *class_id* is the object class type (a number representing the detection class), *score* represents how confident the detected object is, while *W* and *H* are representing width and height of the capture device or CCTV.

```
if score > threshold and int(class_id) in object:
    if $abs(x_1 - x_2) > W/4$ and $abs(y_1 - y_2) > H/4$ and counter > 0:
        if $abs(past\_x_1 - x_1) < 30$ and $abs(past\_x_2 - x_2) < 30$ and $abs(past\_y_1 - y_1) < 30$ and
            $abs(past\_y_2 - y_2) < 30$:
            number_vehicles = -1
```

The information processing module is another submodule of the Detection Device module and it updates the CSV file to include a header with parameters such as: date and time, car video frame number, car ID, car bounding box (car BB), license plate bounding box (licence plate BB), licence plate bounding score (licence plate BB score), license plate number, and license plate number score (Table 1). It also fills in values for missing frames to ensure consistency for data visualization. A new video is created with smoother transitions and smaller fluctuations, making it easier to visualize the vehicles running red lights. Once the video is complete, the data is uploaded to the cloud using a separate thread.

The data upload to the cloud module is another submodule of the Detection Device module and it uses an SSH (Secure Shell) connection to transfer the information using SCP (Secure Copy Protocol). To connect to SSH, the device's IP address, username, and user password are required. Once the connection is established, SCP is used to upload the video file and the image folder along with the CSV file to the NAS address, after which the connection is closed. The Raspberry Pi module (Figure 3), is responsible with the communication between the Detection Device and Arduino modules. It uses threads to enable communication with the Detection Device module on two separate threads, one for receiving data and one for transmitting data [12]. The main thread deals with receiving messages from Arduino, and the final thread deals with setting the input pin values on Arduino to activate the desired traffic light. The time allocated for each car is determined using the following formula (1).

$$time = number\_of\_cars \times estimated\_passing\_time + buffer\_seconds \quad (1)$$

The number of cars value is received from the detection device module, the estimated passing time being seven, while the buffer value is five. The values for the estimated passing time and buffer are handpicked.

The secondary component is the Arduino module (Figure 3), responsible for controlling the traffic lights. It receives the number of cars from the Raspberry Pi module through a serial connection and it determines through input pins which traffic light should be green in real time. This component is made using Arduino and is programmed in C language. Depending on the signal received, another traffic light is activated using relays that allows the power transfer [2]. The code reflecting this traffic light control and the number of cars transmitted through the serial connection is presented next.

```
void loop() {
 if(digitalRead(main_light_signal) == HIGH) {main_light_change();}
  else if(digitalRead(secondary_light_signal) == HIGH) {
     if (Serial.available() > 0) {number_of_cars = Serial.parseInt();}
     secondary_light_change();}
  else if(digitalRead(pedestrian_light_signal) == HIGH) {pedestrian_light_change();}
  else {main_light_change();}
 delay(1000);}
```

The time required for a pedestrian to cross the street is based on formula (2):

$$time = estimated\_crossing\_time \times linear\_meters\_of\_crosswalk \qquad (2)$$

The value of the estimated crossing time is seven, while the linear meters of the crosswalk refer to the length of the pedestrian crossing in linear meters.

The developed application support component is NAS (Figure 3), responsible for the entire system database. It was created using the Proxmox hypervisor, configured as a virtual machine (VM). The VM consumes fewer resources than a conventional device because Proxmox can control the level of resources used. Thus, in Figure 4, are presented the VM resource consumption for processor, RAM memory, internet traffic and storage.

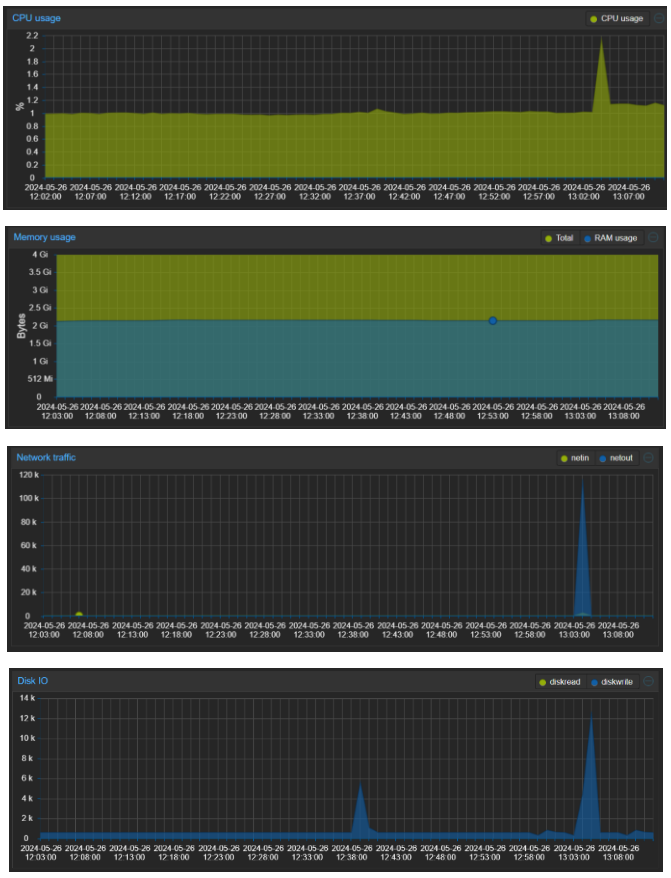
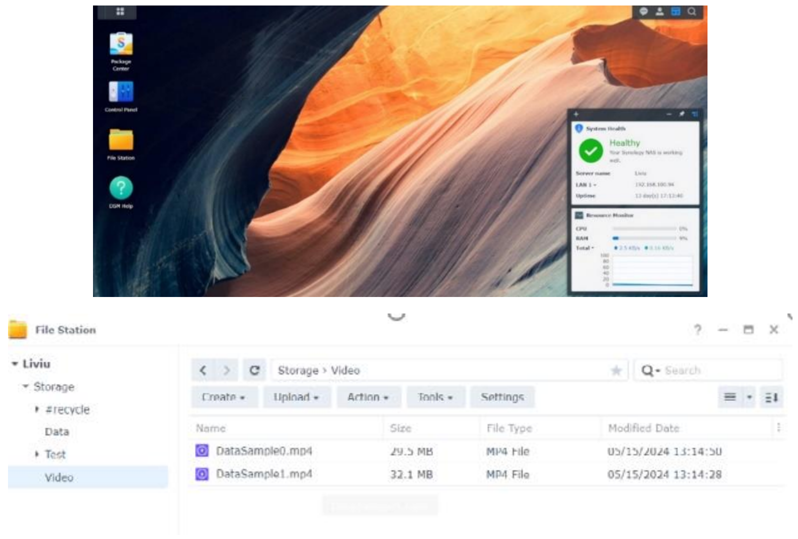


Figure 4. *The VM usage of resources*

The VM allows the direct viewing of data directly from its local IP address and the information uploading and downloading. In Figure 5, the graphical representation of the VM can be observed, and the contents of the virtual machine's files are depicted.

The interface component is a Web Application (Figure 3) developed using Python with the Flask module and HTML language for the web pages' structure [3]. This component downloads data from NAS and puts it into a source file, from where it is used to generate the web application. The web application consists of three pages: the main page, where a chart displaying all detections within a time interval is visible, a secondary page where a search can be made for a specific date and time to obtain the detections from a certain interval, and another secondary page that displays the image resulting from the date and time search.



Name	Size	File Type	Modified Date
DataSample0.mp4	29.5 MB	MP4 File	05/15/2024 13:14:50
DataSample1.mp4	32.1 MB	MP4 File	05/15/2024 13:14:28

Figure 5. *Displaying the interface and data from VM*

The hardware application components and all of its modules are presented in Figure 6, where the Detection Device is a laptop with the source code running, the Arduino module and Raspberry Pi module activated, two CCTV's for vehicle and pedestrian detection, the traffic lights for each individual area, and a router for internet access for both CCTV's that needs an internet connection to be able to redirect their streams toward the Detection Device.

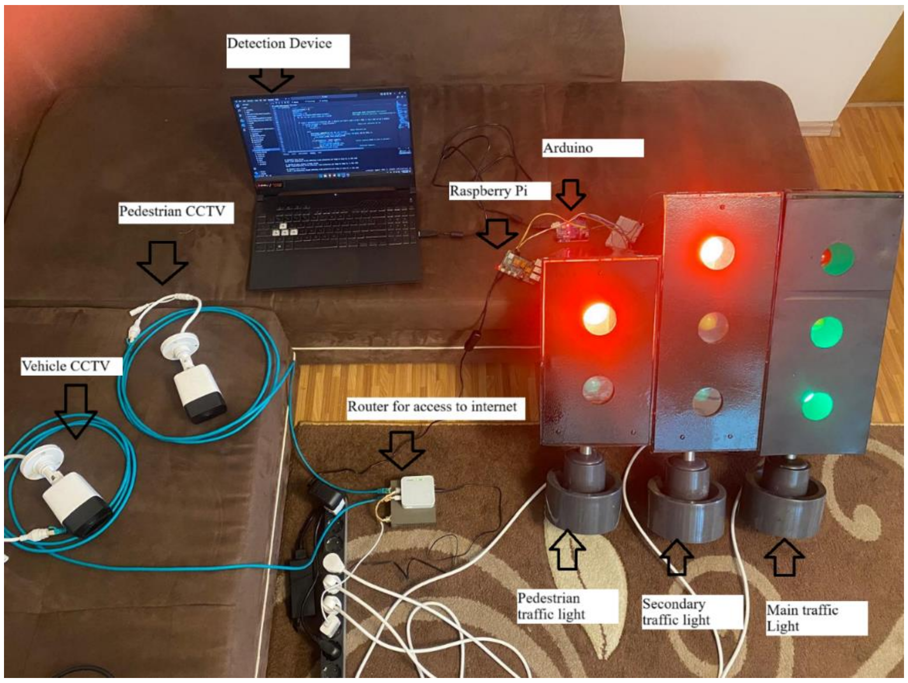


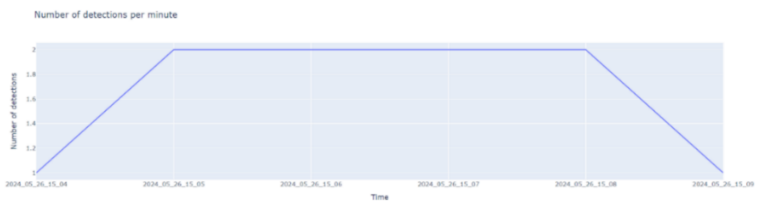
Figure 6. *The developed application hardware components*

SIMULATION PERFORMED WITH THE DEVELOPED APPLICATION FOR ROAD TRAFFIC EFFICIENCY

In order to demonstrate the correct operating of the developed integrated application that uses CNN’s it was achieved a set of simulations (Table 1). Next, is presented a simulation conducted with a video containing cars, the results for the detection graph being presented in Figure 7. For the search of license plate values and images, it was chosen date 2024/05/26, hour 15:09:09 PM. It can be observed the detection of the car’s license plates and images (Figure 7). In table 1 are presented the detailed results of the simulations made with the developed application.

Table 1. *Simulations results*

Simulation no.	***Date and time***	***Frame no.***	***Car ID***	***Car BB***	***Licence plate BB***	***Licence plate BB score***	***Licence plate no.***	***Licence plate no. score***
1	2024.5.26 15:09:09	229	10	907 1495 1656 2091	1173 1867 1377 1952	0.675242	GX15OGJ	0.737784
	2024.5.26 15:04:53	13	1	720 1402 1423 2047	977 1824 1170 1893	0.3765	NA13NRU	0.710235
	2024.5.26 15:05:06	23	1	697 1432 1401 2080	977 1876 1161 1940	0.251044	MA13NRU	0.286378
2	2024.5.15 13:14:54	12	1	722 1402 1424 2045	977 1825 1171 1894	0.382648	NA13NRL	0.326808
	2024.5.15 13:10:25	90	8	2226 1369 2896 2060	2472 1845 2666 1936	0.616166	CT51VSU	0.142973
	2024.5.15 13:11:39	134	8	2263 1498 2964 2148	2523 2020 2723 2113	0.70308	MZ51VSU	0.159333



Time detection: 2024_5_26_15_9_9 Submit

- GX15OGJ

Time image: 2024_5_26_15_9_9 Submit

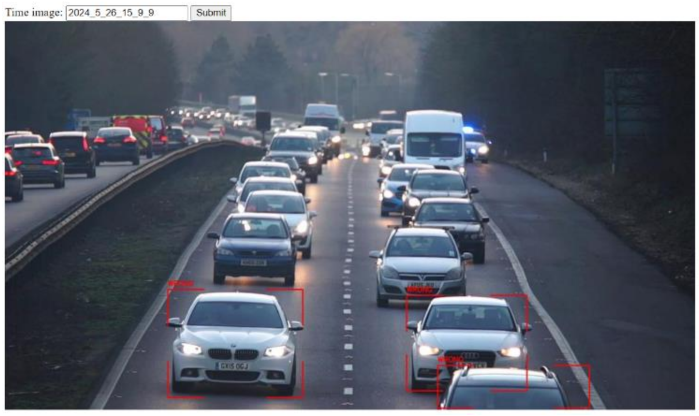
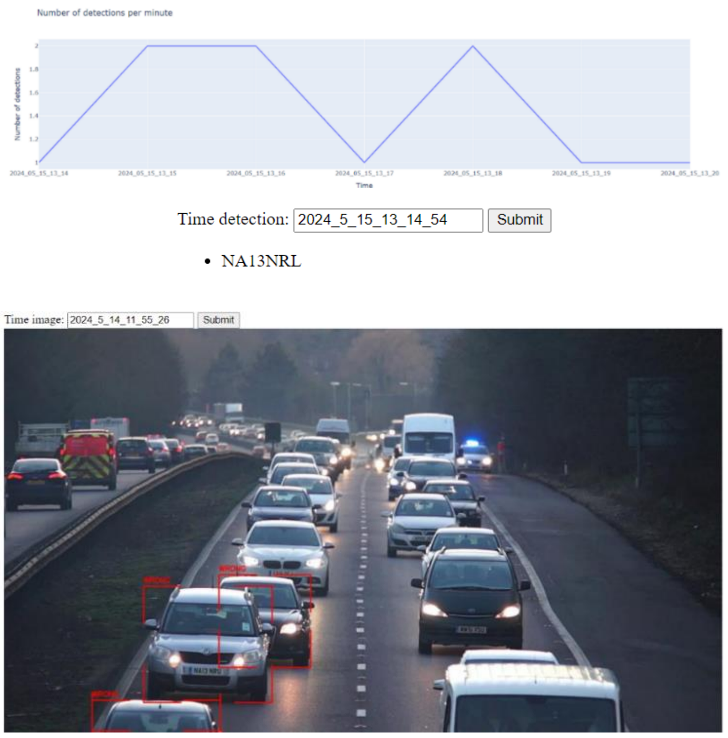


Figure 7. *Display of the obtained results for simulation no.1*

Another simulation was achieved, but with a different date and time. For the search of license plate values and images, it was chosen date 2024/05/15, hour 13:14:54 PM, and the results of the Web Application module can be observed in Figure 8.



Time detection: 2024_5_15_13_14_54

- NA13NRL

Time image: 2024_5_14_11_55_26

Figure 8. *Display of the obtained results for simulation no. 2*

After the simulations, the data collected is not completely accurate. The results of simulation no.1 show a high level of accuracy, while the results of simulation no. 2 show that there are some adjustments to be done. The used convolutional neural network reserved for detecting licence plates was not given enough epochs to train, as such the performances for harder frames of the video to extract information being reduced. The actual number given for the licence place is wrong, as the licence plate number that was detected in the original video is "NA13NRU", thus the OCR used for detection has also performed poorly.

CONCLUSIONS

With a limited budget available for urban infrastructure modernization, the use of integrated CNN's applications for traffic vehicle and pedestrian detection is feasible, especially with enhancements through AI techniques.

The developed integrated application operates autonomously, requiring no human intervention due to its numerous modules designed to facilitate system functionality. However, by modularizing the system from a singular application to a large number of modules, the failure of one such module could lead to the incorrect operation of the entire application, necessitating specialist intervention. Nevertheless, repairing a faulty module is simpler than fixing a standalone application since the problem is easier to identify and isolate. Thanks to its modular and multithreaded structure, the application can continue running even when one of its component modules is faulty.

The system's advantages lie in the flexibility to change both hardware and software component modules without major application changes, as well as its future development possibilities. It addresses not only CNN's but also networking, various coexisting operating systems, camera and video processing development, and web application development.

However, the system has its drawbacks. Its major disadvantage is its dependence on a dedicated CNN to ensure full control for efficient optimization. While this issue has been addressed, the complexity of the problem and limited time have prevented a complete solution. Moreover, there are currently no emergency solutions in case of malfunction, and the system does not allow easy real-time intervention by a specialist, necessitating system shutdown for repairs. This issue could be addressed in the future.

Regarding future work directions it can be mentioned: the creating of a CNN using DL techniques for detecting desired objects, the solutions developing for application reliability and longevity, the development of an optimal algorithm for efficient traffic management, the optimizing of the convolutional neural network reserved for licence detection and the OCR accuracy improvement.

The author's main contributions are:

- Implementation of a DL technique using retrained CNN's in an object detection application;
- Development of a method for transferring information between a local device and the cloud, and vice versa, in a form of a submodule named "data upload to the cloud" of the Detection Device module;
- Development of a web application for data displaying to a user base;
- Development of communication systems between multiple devices on separate threads;
- Development of the algorithm for traffic light color changes present in the Arduino module.

The application successfully achieves its initial development goals of optimizing traffic by directing vehicles from secondary roads while offering reduced congestion for vehicles on main traffic arteries, all without endangering pedestrians.

REFERENCES

[1] Albawi, S., Mohammed, T. A. and Al-Zawi, S., Understanding of a convolutional neural network, International Conference on Engineering and Technology (ICET), Antalya, Turkey, pp. 1-6, 2017.

[2] Bala, S.R., Electronică digitală, Editura Universității Petrol-Gaze din Ploiești, 2021.

[3] Cărbureanu M., Programare Web: ghid teoretic și practic, Editura Universității Petrol-Gaze din Ploiești, 2020.

[4] Javaid, S., Sufian, A., Pervaiz, S., Tanveer, M., Smart traffic management system using Internet of Things, 20th International Conference on Advanced Communication Technology (ICACT), Chuncheon, South Korea, pp. 393-398, 2018.

[5] Li, Z., Liu, F., Yang, W., Peng, S., Zhou, J., A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects, IEEE Transactions on Neural Networks and Learning Systems, vol. 33, no. 12, pp. 6999-7019, 2022.

[6] Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S., Feature Pyramid Networks for Object Detection, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, USA, pp. 936-944, 2017.

[7] Oprea, M., Inteligență artificială: îndrumar de laborator, Editura Universității Petrol-Gaze din Ploiești, 2009.

[8] Oprea, M., Nicoară, S., Inteligență artificială, Editura Universității Petrol-Gaze din Ploiești, 2005.

[9] Oprea, M., Programare orientată pe obiecte: îndrumar de laborator, Editura Universității Petrol-Gaze din Ploiești, 2018.

[10] Oprea, M, Programare orientată pe obiecte: limbajul C++, Editura Universității Petrol-Gaze din Ploiești, 2017.

[11] Patan, R., Suresh, K., Babu, M. R., Real-time smart traffic management system for smart cities by using Internet of Things and big data, International Conference on Emerging Technological Trends (ICETT), Kollam, India, pp. 1-7, 2016.

[12] Rădulescu, G., Sisteme de operare: noțiuni teoretice și practice pentru specializarea calculatoare, Editura Universității Petrol-Gaze din Ploiești, 2022.

[13] Dutch system, https://bicycledutch.wordpress.com/2016/06/21/traffic-lights-in-s-hertogenbosch-an-interview/.

[14] Stanford.edu, http://deeplearning.stanford.edu/tutorial/supervised/Convolutional NeuralNetwork/.

[15] Traffic situation 1, https://www.euronews.ro/articole/bucurestiul-in-topul-celor-mai-aglomerate-orase-in-2022-un-sofer-bucurestean-a-pi.

[16] Traffic situation 2, https://www.euronews.ro/articole/traficul-din-bucuresti-pe-locul-8-in-lume-capitala-ocupa-locul-4-in-europa-in-top.

[17] Traffic situation 3, https://www.fanatik.ro/cele-mai-aglomerate-orase-din-lume-19836733.

[18] IBM, https://www.ibm.com/topics/convolutional-neural-networks.

[19] Intellias, https://intellias.com/smart-traffic-signals/.

[20] Traffic situation 4, https://www.tomtom.com/traffic-index/bucharest-traffic/.

[21] UTi, https://www.uti.eu.com/business-lines/intelligent-transportation-solutions/urban-traffic-management/portfolio/bucharest-romania-adaptive-traffic-management-system/.

Received: May 2024; Revised: June 2024; Accepted: June 2024; Published: June 2024