An Embedded Deep Learning-based Package for Traffic Law Enforcement

Abbas Omidi[1], Amirhossein Heydarian[1], Aida Mohammadshahi[1],
Behnam Asghari Beirami[2], Farzan Haddadi[1]
[1]Iran University of Science and Technology, [2]K. N. Toosi University of Technology
Tehran, Iran
abbas_omidi@elec.iust.ac.ir, heydarian_a@elec.iust.ac.ir, aida_mohammadshahi@elec.iust.ac.ir
behnam.asghari1370@gmail.com, farzanhaddadi@iust.ac.ir

Abstract

Crossing Heavy Good Vehicles (HGVs) from the overtaking lane in highways is not only a traffic violation but may also cause severe casualties in case of an accident happening in such velocities. Currently, the only way to prevent this violation is to identify the violating vehicles by the highway police, so in this paper, a violation detection system using an embedded camera is introduced using algorithms based on deep learning and image processing techniques. The embedded system benefits of a multi-stage deep system based on the YOLO network, which consists of four stages of cascaded detection, including overtaking lane detection, HGV detection, license plate detection, and character recognition. In this research, the developed deep learning models, after some initial training, are fine-tuned on a local Persian dataset collected with distributed cameras. The accuracy obtained on the test dataset of each of the four separate stages was above 85% and the results show the efficiency of the proposed smart system with 70% accuracy in the union of all stages. All data including local datasets, implementations, codes, and results are available on the project's GitHub (https://github.com/NEFTeam/Traffic-Law-Enforcement).

1. Introduction

As the number of automobiles on the road has risen in recent years, traffic violations have become more common. This has resulted in several issues, including an increase in traffic accidents and injuries or deaths [1]. As a result, detecting violations plays a significant role in reducing these casualties. For police officers, however, it is extremely difficult. Due to high speed or distance, it is possible that officers will not see violations or that cars will not stop after a violation has been detected. It is also possible that officers will not be able to recognize the license plate number with the naked eye. Furthermore, there are no adequate sites for policemen to monitor traffic in other regions, such as suburban roadways. As a result, several violations occur in regions where there is no surveillance. Automatic systems could be a great substitute in these cases, detecting many sorts of traffic violations with high accuracy, no missed violations, and at a cheaper cost. As a result, intelligent traffic monitoring could be highly beneficial in detecting violations and recognizing license plate numbers of violating vehicles [2, 3, 4].

Passing of Heavy Goods Vehicles (HGV) through the highway-overtaking lane is the violation addressed in this study. Because of the large weight of these cars compared to passenger cars, accidents involving this type of car at high speeds in the overtaking lane generally result in fatalities and irreversible dangers, and they usually generate significant traffic congestion on highways than other car incidents. Indeed, the probability of a fatality when an HGV is involved in an accident is multiplied by 2.6 [5]. This violation can currently only be prevented by policemen, and it will not be identified automatically. However, a series of steps are introduced in this article to prevent HGVs from crossing the overtaking lane and better control them, including identifying the overtaking lane in a traffic camera image, identifying HGVs in the lane, detecting the HGV's license plates, and finally, recognizing the characters of the license plates and registering them to impose fines.

The first step in using a traffic monitoring system is to determine the road lanes. There are numerous line detection algorithms available, including both traditional image processing algorithms and deep learning algorithms. Commonly, traditional methods locate road lines based on edges, which are highly dependent on the hyperparameters and the environment under consideration [6, 7, 8, 9]. Deep learning approaches such as Point Instance Network (PINet) [10] and LaneATT [11] have also been proposed in recent years, and several artificial neural networks have been built for this purpose [12, 13, 14, 15]. Despite the high efficiency of the

previously proposed methods, most of them are designed for a specific situation and a particular application.

The detection of objects (such as HGVs) in a picture is one of the most fundamental tasks in computer vision, and it has been revolutionized by deep learning. Earlier object detection methods such as region-based convolutional neural networks [16] work in stages, including finding proposed regions, classifying them, and then performing post-processing to remove duplicates and refine bounding boxes. The complexity of optimization due to the requirement of training multiple networks, as well as the long prediction time, were also issues with this method. YOLO (You Only Look Once) [16] proposed a CNN-based, one-stage, and real-time method that, in addition to addressing previous problems, had significant improvements in object detection accuracy.

Traffic violation registration requires two separate stages of license plates detection and character recognition. Several classical approaches have been proposed in the literature for detecting license plates, including edge-based and color-based approaches [17, 18]. Recently, deep learning-based object detection algorithms are widely used to detect license plates. The great speed, accuracy, and ease of Yolo-based methods for real-time applications are what make them an excellent option [19, 20, 21]. For the task of the plate's character recognition, due to the poor performance and the difficulties in classical approaches, deep learning-based models are the most common alternatives. For example, recurrent neural network (RNN) and bidirectional long short-term memory (LSTM) networks are used in the different studies for character recognition [22, 23, 24, 25]. however, despite its advantages, the method is less implemented in real-time systems due to model complexity.

In this study, a real-time integrated automated system is developed that identifies and records violations of HGVs crossing the overtaking lane. To the best of our knowledge, there are just a few works for this particular violation. To begin with, a suitable local Persian dataset is collected for test and train of our models. The system is divided into two parts: software and hardware. The software part of the system is based on the use of the YOLO object detection method successively for lane detection, HGV detection, license plates detection, and finally, plate character recognition. For the hardware section, an embedded camera is employed for data collection and real-time road monitoring in this system. The proposed embedded system benefits of NVIDIA Jetson Nano which is a compact and powerful computer that allows many neural networks to run in parallel for tasks like image classification, object identification, segmentation, and audio processing.

This is how the paper is structured: Section 2 provides background information. Section 3 introduces the methodology. Section 4 proposes our dataset and hardware system configuration, as well as experimental results and discussions. Finally, in Section 5, the conclusion is given.

2. Background

This section reviews the YOLO object detection, Hough transform, Canny edge detection, and perspective correction.

2.1. YOLO

The YOLO network structure consists of three main parts. The first part is called the backbone and is responsible for extracting image features in different scales. The second part is the neck that combines the extracted features to prepare for the prediction. Finally, the network head uses the neck features to predict box and class of each object in the image [16]. Fig. 1 shows the structure of the YOLO network.

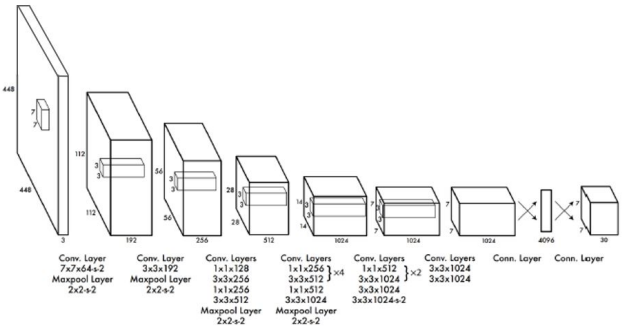

Figure 1. The backbone, neck and head form the three main parts of the YOLO network. [16].

YOLO models the detection problem as a regression problem by dividing the input image into an $S \times S$ grid. Each grid cell predicts B bounding boxes holding x, y, w, h and an "objectness" score $P(Object)$ which shows whether the grid cell contains an object or not. Each grid cell also predicts a conditional probability $P(Class|Object)$ for the class of the object associated with that cell. Thus, for each grid cell, YOLO predicts $B \times 5$ parameters and C class probabilities. These predictions are encoded as an $S \times S \times (B \times 5 + C)$ tensor. Eventually, Non-Maximum Suppression (NMS) and thresholding are applied to produce final object detection predictions [16].

Training is accomplished by decreasing the loss function based on sum-squared error, all in one stage. Since there are no objects in many grid cells, the gradients from cells that contain the objects increase the weights. YOLO handles this problem by raising bounding box weights for objects (λ_{coord}) and reducing them for non-objects (λ_{noobj}). The total loss function is composed of three main components. Eq.(1) shows the GIoU part which is responsible for

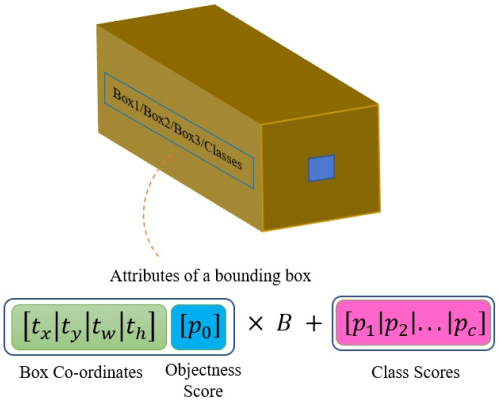

Figure 2. Final YOLO output consisting of Box Co-ordinates, objectness Scores and Class Scores.

bounding box prediction. It computes the sum squared error between the network output and labels [16].

$$\text{GIoU} = \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \quad (1)$$

Objectness part is indicated as follows and is responsible for predicting whether a grid contains an object or not [16]:

$$\text{Objectness} = \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{noobj} (C_i - \hat{C}_i)^2 \quad (2)$$

The last part is the classification error and is responsible for object class prediction [16]:

$$\text{Classification} = \sum_{i=0}^{s^2} \sum_{c \in classes} 1_{ij}^{obj} (p_i(c) - \hat{p}_i(c))^2 \quad (3)$$

The final loss function is composed of the summation of Eqs. (1), (2) and (3). YOLO was initially created in Darknet [16]. It is a low-level language-based research framework with a lot of flexibility for deep learning. The Original YOLO was the first object detection network to integrate object localization and classification problems into a single end-to-end neural network. BatchNorm, higher resolution, and anchor boxes were added in YOLO-v2 [26]. YOLOv3 improved performance on smaller objects by adding connections to the backbone network layers and making predictions at three different scales [27]. This paper utilizes YOLO-v5 which is implemented in PyTorch instead of Darknet and comes with several training techniques and structural improvements such as CSPNet [28] as backbone, new feature aggregation method, and mosaic data augmentation. These innovations were initially known as YOLO-v4. However, since YOLO-v4 was released in DarkNet framework, it was changed to YOLO-v5 to prevent version collisions [29].

2.2. Hough Transform

The Hough algorithm is a popular technique for detecting the borderlines of any type of geometric shape, even if the shape is curved or has broken lines [30]. In general, a line can be represented by equation $y = mx + c$, but there is another conventional form for representing the equation of a line, which is polar coordinate $\rho = x\cos(\theta) + y\sin(\theta)$. In this equation, ρ represents the vertical distance of the line from the origin and θ is the angle of the line ρ with the horizontal axis of the coordinates. Therefore, a line in a polar representation can be identified with two parameters ρ and θ (Fig. 3).

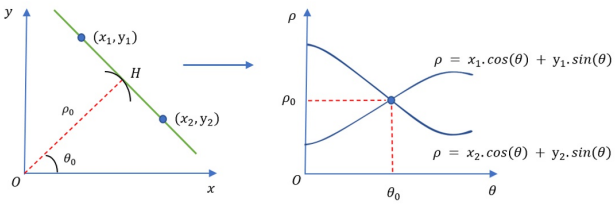

Figure 3. Representation of the polar coordinate in Cartesian coordinate and its mapping in Hough space.

To distinguish a line in Cartesian coordinates, any point on it with (x, y) coordinates can be converted to a curve in polar coordinates using $\rho = x\cos(\theta) + y\sin(\theta)$ formula. After mapping all of an image's edge points into Hough space, a large number of curves are formed, and for every two edge points that one line passes through them, their corresponding curves intersect each other on a specific (ρ, θ) pair. Finally, the Hough Transform algorithm discovers edge lines by looking for (ρ, θ) pairs with more than a threshold number of intersections.

2.3. Edge Detection

The Canny edge detection algorithm consists of 5 steps as follows [31]: 1. Noise reduction; 2. Gradient calculation; 3. Non-maximum suppression; 4. Double threshold; 5. Edge tracking by Hysteresis. In the first step, a Gaussian filter is used to reduce the noise on the image and smooth it. for this purpose, convolution is applied using a Gaussian

kernel with odd width. It is obvious that the bigger kernel results in a more visible blur. The equation of a Gaussian filter with kernel size $(2k+1) \times (2k+1)$ is [31]:

$$H_{ij} = \frac{1}{2\Pi\sigma^2} \exp\left(-\frac{(i-(k+1))^2+(j-(k+1))^2}{2\sigma^2}\right); 1 \leq i,j \leq (2k+1) \quad (4)$$

In the second step, calculating the gradient of the image helps to detect the edge intensity and direction. Edges are related to pixels' intensity changes. It can be done by Sobel filtering which is able to detect the intensity changes for both horizontal (G_x) and vertical (G_y) directions by defining Sobel kernels as shown in the Eq. (7). In addition, magnitude (G) and slope (θ) of gradient are calculated as below:

$$|G| = \sqrt{G_x^2 + G_y^2} \quad (5)$$

$$\text{angle}\,(\theta) = \arctan\frac{G_y}{G_x} \quad (6)$$

$$K_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, K_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad (7)$$

This step results in edges with different thicknesses. The next step, non-maximum suppression is performed to overcome this problem and get thin edges in the image. Every pixel is checked in this step to find the pixels with the maximum intensity in their neighborhoods in the direction of the gradient. In the double threshold step, two threshold values are selected, minVal and maxVal, and the pixels are categorized into three classes. Edges with higher intensity than maxVal are considered to be edges for sure. Edges with lower intensity than minVal are removed and are not considered as edges. Pixels with intensity values between maxVal and minVal are the third class. The Edge tracking by Hysteresis as the last step specifies which pixels in the third class are considered as a part of an edge. One is considered an edge if there is at least one edge pixel around it.

2.4. Perspective Correction

The purpose of perspective correction is to neutralize the effect of the camera angle and achieve a front-view image. To do this, it is needed to get a mapping matrix that multiplies individual pixels of the current image to move the pixel to its new location and get an image of the corrected license plate. The goal is to find the center point in the angled image and select a trapezoidal area around the plate that naturally looks like a regular square or rectangle from front [32]. This trapezoid is optionally selected and there is no restriction on the length of the horizontal sides or the angles of the lateral sides. The construction of the transformation matrix by this algorithm is such that the algorithm looks for mapping to move a point with coordinates (x_1, y_1) to a new point (x_2, y_2). In terms of matrix relations, these new coordinates are obtained as:

$$\begin{bmatrix} x_2 \\ x_2 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \quad (8)$$

In the above equation, H is a homography matrix, which is a 3×3 matrix, which handles transferring and mapping the points of input to the output [33].

$$H = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \quad (9)$$

Thus transfer of a point (x_1, y_1) to (x_2, y_2) by the homography matrix will be:

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \quad (10)$$

In the above equations, the homography matrix (H) is obtained by coordinate of four corners of the bounding box of the previous and final frame. By transferring the previous points to these points, the license plate with a corrected perspective is obtained.

3. Methodology

As mentioned earlier, our model includes four steps of lane detection, HGV detection, license plates detection, and finally, plate character recognition. The flowchart of the proposed model is shown in Fig. 4 and detailed information is given in the next subsections.

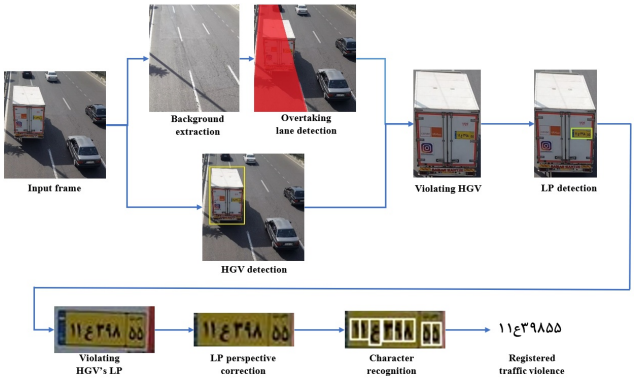

Figure 4. Functional block diagram of the proposed embedded system.

3.1. Overtaking Lane

In this subsection, a hybrid method based on both classical and deep learning algorithms has been proposed to detect overtaking lane. A vehicle-free background image is

obtained by averaging the input camera frames. The edges of the background image are then extracted using Canny edge detector. The next step is to use the YOLO network to locate areas where road lines are most likely to exist. The importance of YOLO is to eliminate the effect of undesired edges through line detection procedure. YOLO's detected regions are combined to form a YOLO binary mask. Finally, desired edges are extracted by applying the road line mask to the Canny edge output.

With the Hough transform, lines of the remaining edges are extracted and the line mask is obtained. Several steps of dilation and erosion are applied to the line mask to merge close lines and prevent them from being rediscovered. The final lines' equations are calculated, and the area between them is referred to as the road lane. Since the overtaking lane is in the leftmost place, the mask of the left lane is output as the overtaking lane. In (11) OV represents overtaking lane, (x_i, y_i) are pixel coordinates and m,b are leftmost lines' slopes and intercepts.

$$OV = \{(x_i, y_i) \mid (m_1 x_i + b_1 < y_i) \wedge (m_2 x_i + b_2 > y_i)\} \tag{11}$$

Fig. 5 shows the block diagram of overtaking lane detection.

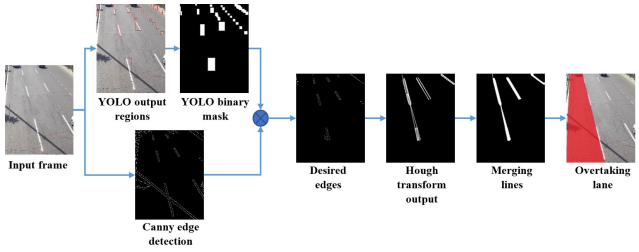

Figure 5. Overtaking lane detection block diagram.

3.2. HGV Detection

The next stage in detecting a violation is to locate HGVs. At this stage, a YOLO network detects all the heavy vehicles in the frame. At the next step after detecting the HGVs, the algorithm compares the vehicle's bounding box center to the overtaking lane mask obtained from the previous stage, and if it matches, the vehicle is marked as violating.

3.3. Plate Detection

Plate detection is referred to the determination of the area encompassing the license plate. The proposed plate detection method consists of two parts. In the first part, the method uses YOLO for plate detection, and in the second part, it uses perspective correction to provide the front view of the plate. Based on the experiments, perspective correction has a huge impact on the accuracy of the next stage, character recognition, as the Persian characters are similar in some angles. Fig. 6 shows the effect of perspective correction on detection accuracy, where number 7 is misclassified as number 1 before the perspective correction.

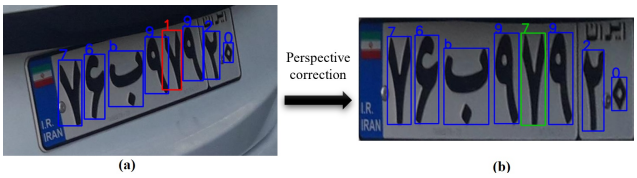

Figure 6. The effect of perspective correction (a) without perspective correction 7 is misclassified as 1 (red box), (b) correct recognition of 7 after perspective correction.

3.4. Character Recognition

Precise recognition of license plate characters is critical because it completes all of the preceding processes. In fact, by recognizing the license plate characters, the identity of the violating HGV is determined. Inappropriate lighting, blurred license plate images, and various environmental conditions make character recognition challenging. The YOLO network has been used to overcome these problems and take all conditions into account. Several datasets of Persian license plates have been acquired in various conditions to train the network. Each Persian number and letter is labeled as a rectangular box and the corresponding number or letter is assigned to them. The network is trained with various training data in different conditions, and when the violating vehicle is detected, after the license plate detection, the license plate characters are determined by the trained network.

Order of the identified characters is crucial in identifying the vehicle. For this purpose, coordinates of bounding boxes are identified and the characters are arranged according to x coordinates of their bounding boxes. The character with the smallest x of the bounding box sits on the left, and as the length increases, the location of the characters is determined, and finally, the license plate and the identity of the car are determined. The mathematical model of the characters' placement is as follows in which a represents a character and $x(a)$ represents the length of the bounding box of the character.

$$\text{license plate characters} = \{(a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8) \mid x(a_i) < x(a_{i+1})\} \tag{12}$$

The pseudocode of the proposed system is shown in the Alg. 1.

Algorithm 1 The pseudocode of the proposed system

Begin
Input traffic video
Background image generation based on averaging of the frames:
$Background\ image = \frac{\sum_{i=1}^{n} frame_i}{n}$
Determine overtaking lane from background image:
$OV = \{(x_i, y_i) \mid (m_1 x_i + b_1 < y_i) \wedge (m_2 x_i + b_2 > y_i)\}$
for frame **in** traffic video **do**
  Detect HGVs
  $HGV_i = Boundingbox_i$
  Check existence of HGV in over taking lane:
  $inLane = \{HGV_i \mid center(HGV_i)\ in\ OV\}$
  for each $car \in inLane$ **do**
    Detect licensce plate:
    $licensce\ plate_i = Bounding\ box_i$
    Correct prespective of licensce plate
    Recognize characters :
    $\{(a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8) \mid x(a_i) < x(a_{i+1})\}$
  end for
end for

4. Experiments

4.1. Datasets

To train the model, various datasets have been used in this research. Some of these datasets have already been collected in previous studies which will be mentioned further, and some of these datasets have been collected by the authors. Of course, all the functions presented in this study are tested on the local dataset.

4.1.1 Overtaking Lane Dataset

Road lines are similar in different countries. so, in order to have a large dataset for training line recognition, London traffic videos have been used. These videos are recorded by fixed traffic cameras and are freely available. The dataset consists of 150 images with of 352×288 pixels, each of which belongs to a specific camera, obtained by averaging video frames. In this way, moving cars and objects are removed, and each photo shows us the empty road, which facilitates the work of labeling the lines for YOLO. Also, 295 local images of Iran taken by the embedded camera have been added to the dataset. Sample images are shown in Fig. 7.

4.1.2 HGV Dataset

For identification of HGVs, the default dataset used for training of YOLO, the Coco dataset [34], has been used alongside the local dataset. The Coco dataset contains 330,000 images from 80 different classes that include all of the classes we are considering, including trucks and buses. But in this research, the trained model on this dataset has been fine-tuned on the local dataset collected by the authors. Sample images of data set are shown in Fig. 8.

Figure 7. Samples of overtaking lane dataset, (a) and (b) examples of London dataset and (c) example of the local dataset

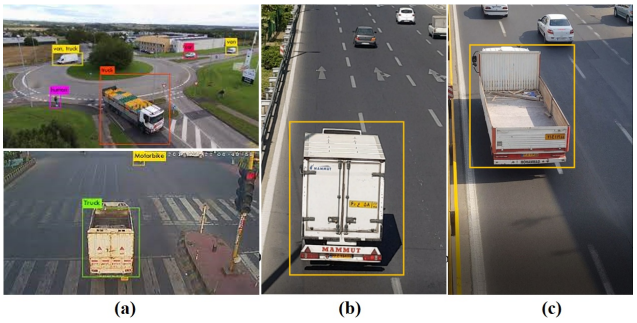

Figure 8. Samples of HGV datasets. (a) shows HGVs of Coco dataset and (b) and (c) are examples of the local dataset.

4.1.3 License Plate Dataset

The dataset used in this section is a local dataset containing 295 photos of various cars in Iran with Persian license plates, taken with the embedded cameras. Sample images of dataset are shown in Fig. 9.

Figure 9. Samples of local dataset which consists of HGVs with Persian license plate.

4.1.4 Plate Character Dataset

A dataset of Persian license plates was collected using an embedded camera to perform character recognition. Various data augmentation approaches, such as adding noise, blur, color shift, and contrast have been used to simulate different environmental conditions. Finally, all of the character

bounding boxes in each license plate are labeled, and a total number of 10105 Persian license plates, along with their labels, has been prepared. Sample images of the dataset are shown in Fig. 10.

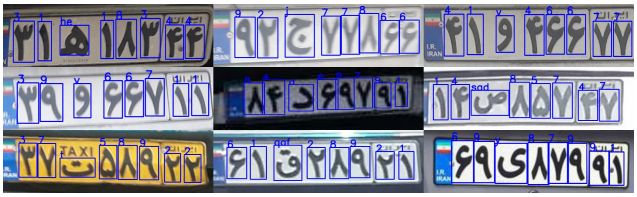

Figure 10. Samples of the persian license plate characters dataset captured from different angles.

The embedded camera was used to collect a dataset of urban streets and out-of-town roads containing 314 different samples of HGVs to train the system and evaluate its performance in real-world situations. These videos vary in location and time and also include light and heavy traffic places to cover various roads, heavy vehicles, and license plates. The integrated model was trained on 250 images of this data set in every stage, such as overtaking, lane detection, HGV detection, license plate detection, and character detection. Finally, the model was evaluated on 45 images of the data set including different types of heavy vehicles. The evaluation dataset contains 20 violating and 25 non-violating vehicles.

Figure 11. Samples of the local dataset in different locations and times of day, including images with violating HGVs and non violating HGVs

4.2. Hardware Configuration

The hardware used in this research consists of Jetson Nano ™ designed by NVIDIA® and Waveshare 12.3MP camera. The NVIDIA® Jetson Nano B01 ™ Developer Kit with GPU processor is a small, powerful computer that lets multiple neural networks run in parallel for applications such as image classification, object recognition, segmentation, and speech processing [35]. The image provided by Waveshare 12.3MP camera is of size 4056 × 3040 pixels. The camera is based on the Sony IMX477-160 sensor and can capture up to 90 frames per second. Also, the operating system used in this hardware is based on Ubuntu 18.04. Fig. 12 shows the embedded system.

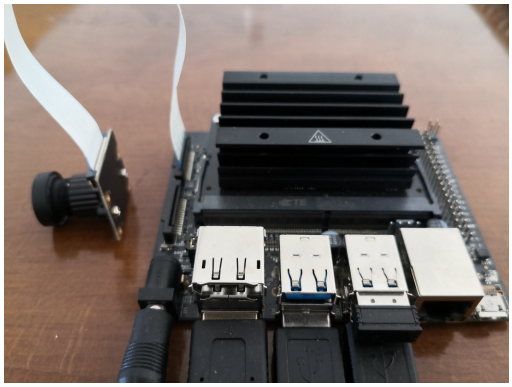

Figure 12. Waveshare 12.3MP camera connected to NVIDIA Jetson Nano™.

4.3. Experimental Results

The evaluation has been done separately for each of the stages of overtaking lane detection, HGV detection, license plate detection, and character recognition in the following tables.

The Intersection over Union (IoU) criterion was used to evaluate the performance of the algorithm in the overtaking lane extraction. Local dataset backgrounds with their ground truth masks are prepared for evaluation. The overtaking lane mask is obtained using our algorithm for all of the images. The overlap between the predicted and ground truth mask is measured based on IoU. Accuracy is determined based on a 50% threshold So that correct prediction indicates an IoU value above 0.5.

Figure 13. Overtaking lane detection results. The top images represent the predicted overtaking lane by the system in red and the bottom images show the ground truth in green.

Table. 1 shows the performance results of overtaking lane detection algorithm on this task.

Table 1. Overtaking lane results.

	IoU	Accuracy
Overtaking lane	0.881	93.1%

Precision, recall, F1, and accuracy are used to evaluate the HGV detection, license plate detection, and character recognition sections. The existing characters which are not detected are referred to as False Negative, while the number of characters that do not exist and are misidentified is referred to as False Positive. The results show that all sections are above 85%. Also, the recall of the HGV detection section is above 95%. Therefore, each step individually has an appropriate performance.

Table 2. Stage results.

Stage	Precision	Recall	F1	Accuracy
HGV	93.7%	95.7%	94.7%	90.0%
License plate	93.8%	91.5%	92.6%	86.2%
Character	93.5%	94.0%	93.7%	88.2%

General evaluation of the implemented system is investigated as follows. The system works properly when all the steps perform flawlessly. In other words, the HGV passing through the overtaking line must be detected, the license plate must be obtained and all the characters must be recognized correctly. As a result, precision, recall, F1, and the overall accuracy which is defined as the number of correct license plates divided by the number of HGVs are reported in Table 3.

Table 3. Overall perfomance of the system.

	Precision	Recall	F1	Accuracy
Proposed method	96.9%	72.1%	82.7%	70.5%

Figure 14. Sample results of the proposed system performance including 4 stages of overtaking lane detection, HGV detection, License plate detection and Character recognition

4.4. Discussion

In this section, two types of problems in detecting violations are mentioned. The first type includes solvable problems using computer vision algorithms. For example, violating vehicles have a high speed when crossing the overtaking lane. This high-speed mobility can blur license plates and affect their readability. The solution for these problems will be discussed in future articles. However, some challenges are outside the scope of computer vision algorithms including occlusion and distortion (Fig. 15.a), filth and mud (Fig. 15.b), and absence of license plate (Fig. 15.c).

Figure 15. Challenging situations captured by embedded camera.

5. Conclusion

In this paper, a multi-stage system was introduced to monitor the violation of crossing heavy goods vehicles (HGVs) through the overtaking lane, which is an intelligent system based on deep learning and was implemented on embedded cameras. The system consists of four cascaded deep stages based on YOLO. After fine-tuning the model on the local data set and evaluating it on the test data, the model has achieved accuracy above 85% in various stages in separate tests. Finally, a total accuracy of 70.5% was achieved based on correct prediction at all stages.

Acknowledgement

We appreciate the people who contributed to the development of www.tfljamcams.net and the provision of London traffic camera images. Thanks also to the collectors of the COCO dataset and the developers of the YOLOv5 network who shared their models on github.com/ultralytics/yolov5.

Authors' Contribution

The first four authors have equal contributions in this article at all stages, and the fifth author is the supervisor of the project.

References

[1] Lele Xie, Tasweer Ahmad, Lianwen Jin, Yuliang Liu, and Sheng Zhang. A new cnn-based method for multi-directional car license plate detection. *IEEE Transactions on Intelligent Transportation Systems*, 19(2):507–517, 2018.

[2] Nourdine Aliane, Javier Fernandez, Mario Mata, and Sergio Bemposta. A system for traffic violation detection. *Sensors*, 14(11):22113–22127, 2014.

[3] Julien A Vijverberg, Nick AHM de Koning, Jungong Han, Peter HN de With, and Dion Cornelissen. High-level traffic-violation detection for embedded traffic analysis. In *2007 IEEE International Conference on Acoustics, Speech*

and Signal Processing-ICASSP'07, volume 2, pages II–793. IEEE, 2007.

[4] Ala Mhalla, Thierry Chateau, Sami Gazzah, and Najoua Essoukri Ben Amara. An embedded computer-vision system for multi-object detection in traffic surveillance. *IEEE Transactions on Intelligent Transportation Systems*, 20(11):4006–4018, 2018.

[5] Michel Gothié, Véronique Cerezo, and Florence Conche. Relationship between road infrastructure characteristics and hgv accidents. In *The 10th International Symposium on Heavy Vehicle Transport Technology, Paris*, 2008.

[6] Fatemeh Mazrouei Sebdani and Hossein Pourghassem. A robust and real-time road line extraction algorithm using hough transform in intelligent transportation system application. In *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, volume 3, pages 256–260. IEEE, 2012.

[7] Weifeng Liu, Zhenqing Zhang, Shuying Li, and Dapeng Tao. Road detection by using a generalized hough transform. *Remote Sensing*, 9(6):590, 2017.

[8] Hui Kong, Jean-Yves Audibert, and Jean Ponce. General road detection from a single image. *IEEE Transactions on Image Processing*, 19(8):2211–2220, 2010.

[9] Hui Kong, Jean-Yves Audibert, and Jean Ponce. Vanishing point detection for road detection. In *2009 ieee conference on computer vision and pattern recognition*, pages 96–103. IEEE, 2009.

[10] Yeongmin Ko, Younkwan Lee, Shoaib Azam, Farzeen Munir, Moongu Jeon, and Witold Pedrycz. Key points estimation and point instance segmentation approach for lane detection. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[11] Lucas Tabelini, Rodrigo Berriel, Thiago M Paixao, Claudine Badue, Alberto F De Souza, and Thiago Oliveira-Santos. Keep your eyes on the lane: Real-time attention-guided lane detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 294–302, 2021.

[12] Seokju Lee, Junsik Kim, Jae Shin Yoon, Seunghak Shin, Oleksandr Bailo, Namil Kim, Tae-Hee Lee, Hyun Seok Hong, Seung-Hoon Han, and In So Kweon. Vpgnet: Vanishing point guided network for lane and road marking detection and recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 1947–1955, 2017.

[13] Zhan Qu, Huan Jin, Yang Zhou, Zhen Yang, and Wei Zhang. Focus on local: Detecting lane marker from bottom up via key point. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14122–14130, 2021.

[14] Lizhe Liu, Xiaohao Chen, Siyu Zhu, and Ping Tan. Condlanenet: a top-to-down lane detection framework based on conditional convolution. *arXiv preprint arXiv:2105.05003*, 2021.

[15] Hala Abualsaud, Sean Liu, David Lu, Kenny Situ, Akshay Rangesh, and Mohan M Trivedi. Laneaf: Robust multi-lane detection with affinity fields. *arXiv preprint arXiv:2103.12040*, 2021.

[16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[17] Vahid Abolghasemi and Alireza Ahmadyfard. An edge-based color-aided method for license plate detection. *Image and Vision Computing*, 27(8):1134–1142, 2009.

[18] Yule Yuan, Wenbin Zou, Yong Zhao, Xinan Wang, Xuefeng Hu, and Nikos Komodakis. A robust and efficient approach to license plate detection. *IEEE Transactions on Image Processing*, 26(3):1102–1114, 2016.

[19] Sérgio Montazzolli and Claudio Jung. Real-time brazilian license plate detection and recognition using deep convolutional neural networks. In *2017 30th SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)*, pages 55–62. IEEE, 2017.

[20] Gee-Sern Hsu, ArulMurugan Ambikapathi, Sheng-Luen Chung, and Cheng-Po Su. Robust license plate detection in the wild. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2017.

[21] Yousri Kessentini, Mohamed Dhia Besbes, Sourour Ammar, and Achraf Chabbouh. A two-stage deep neural network for multi-norm license plate detection and recognition. *Expert systems with applications*, 136:159–170, 2019.

[22] Hui Li and Chunhua Shen. Reading car license plates using deep convolutional neural networks and lstms. *arXiv preprint arXiv:1601.05610*, 2016.

[23] Teik Koon Cheang, Yong Shean Chong, and Yong Haur Tay. Segmentation-free vehicle license plate recognition using convnet-rnn. *arXiv preprint arXiv:1701.06439*, 2017.

[24] Hui Li, Peng Wang, Chunhua Shen, and Guyu Zhang. Show, attend and read: A simple and strong baseline for irregular text recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8610–8617, 2019.

[25] Yongjie Zou, Yongjun Zhang, Jun Yan, Xiaoxu Jiang, Tengjie Huang, Haisheng Fan, and Zhongwei Cui. A robust license plate recognition model based on bi-lstm. *IEEE Access*, 8:211630–211641, 2020.

[26] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[27] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[28] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 390–391, 2020.

[29] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[30] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[31] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[32] Michael J Magee and Jake K Aggarwal. Determining vanishing points from perspective images. *Computer Vision, Graphics, and Image Processing*, 26(2):256–267, 1984.

[33] Daniel Paulus Sihombing, Hanung Adi Nugroho, and Sunu Wibirama. Perspective rectification in vehicle number plate recognition using 2d-2d transformation of planar homography. In *2015 International Conference on Science in Information Technology (ICSITech)*, pages 237–240. IEEE, 2015.

[34] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[35] Stephen Cass. Nvidia makes it easy to embed ai: The jetson nano packs a lot of machine-learning power into diy projects-[hands on]. *IEEE Spectrum*, 57(7):14–16, 2020.