Novel and Real-Time GPU Accelerated Speed Violation Detection and License Plate Identification System

Udaya Dampage (✉ udayadampage@ieee.org)

General Sir John Kotelawala Defence University

KKN Hasantha

General Sir John Kotelawala Defence University

HADS Gimhana

General Sir John Kotelawala Defence University

HMTYN Bandara

General Sir John Kotelawala Defence University

SV Maddumage

General Sir John Kotelawala Defence University

Research Article

Keywords: Convolution Neural Networks(CNN), Graphical Processing Unit(GPU), Deep SORT(Simple Online Real-Time Tracking), MOT(Multi-Object Tracking), Object Detection, YOLO(You Look Only Once), Transportation 4.0

Posted Date: November 1st, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-1014673/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License. [Read Full License](#)

Abstract

Commuters lose a lot of time and effort due to the inefficiency of traffic management. Although nowadays most of the processes are automated, it seems a speed violation detection is the least focused area apart from using speed guns which the RADAR may make mistakes and yet, doing such will benefit people by saving their time and let them escape from the troublesome situations. To address this issue, a real-time solution by fully automating the process of detecting the speed violation and the license plates of the offenders is proposed in this paper. A vehicle approaching a specific area will be automatically identified and tracked from a reference starting point. Within the covered range of the camera according to the traffic density present at that instance, the maximum speed for a vehicle is estimated and the vehicles that exceed the stipulated limit are identified as a violation. The core part of the proposed system is license plate recognition. To properly extract the license plate with the best view to proceed with the identification process is another problem that needs to be focused on. We utilized deep neural networks in a novel way for the aforesaid purpose. As these neural networks consist of numerous parameters, we utilized GPU for processing to gain smoothness in real-time. Using our novel segmentation free license plate identification method which utilizes object detection principle to fully capture the speed violation along with its offender. Numerous field trials proved that the proposed methodology provides far superior performance levels compared to the conventional systems and the other existing methodologies, which will certainly cater to the demanding requirements of Transportation 4.0.

1. Introduction

The increasing growth of traffic density in urban areas all over the world possess a greater challenge to traffic management. Many times, traffic density increases because of speed violations. Detecting these speed limit violations by humans in a place such as a junction given the traffic density is quite challenging. Considering the global scenario modifying the latest computer vision techniques to reduce computational time at a high traffic volume junction and increasing the accuracy for speed violation detection is an enhancement to the existing system rather than using speed guns to track the violation.

Considering the approaches for a speed violation detection project computer vision algorithms are appropriate due to the giant steps which took in the last decade. The computer vision field has progressed from CNN (Convolution Neural Network Architectures) such as Alex net to VGG. These architectures gave us regular size design neural network architectures. Based the architectures we use enable us to select hardware and software specifically for the research. Individually there are multiple license plate detection systems [1] and multiple object tracking systems developed for various applications namely for medical imaging[2], security, and autonomous driving[3]. With the increasing growth of traffic density in urban areas, there is a greater challenge posed globally to traffic management. This is where we identified our research problem which is to detect unique vehicle registration numbers and multiple objects tracking systems in computer vision to identify traffic violations so that administrative authorities can access the information.

Performance even with the distorted images must be taken into consideration. The speed is also a vital factor because the number plate should be recognized in real-time. Efficiency in the process is also important.

When initializing the system, automatically imaginary lines will be drawn to illustrate the region it detects the speed and license plate localization system as it is not necessary to run license plate character recognition, since the tracking system and speed violation detection system at the begins identifies the violation. However, the license plate localization system is always in search of a number plate and it will trigger the license plate character recognition system only when there is a violation.

Once the vehicle enters the range of monitoring, we start tracking the vehicle to its destination (the point where the license plate is identified). The pre-trained deep neural network-based method called tiny-YOLO was used for vehicle identification to initialize tracking, precisely to initialize correlation tracker in the Deep SORT for tracking purposes. The vehicle is tracked over multiple cameras.

Whenever a vehicle is identified as an offender, a front image of the vehicle is taken from the camera mounted. This image is first fed to a number plate localization neural network where only the number plate is extracted from the original image. Then extracted number plate is fed to a character segmentation neural network where individual characters are segmented. These segmented characters are then fed to a character recognition neural network at which individual characters are identified. This process happens in real-time. The calculations at the neural networks are done on a GPU to achieve optimum performance.

One of the major concerns is the processing power requirement. To speed up the process Amazon EC2 G4 GPU instance was used to implement a parallel processing version of the deep neural networks. We have run two deep neural networks on GPU and one deep neural network on CPU. If we had enough processing power, we could have implemented a deep neural network for violations detection as well. To implement a real-time system, we have to consider the processing power constraint as well.

2. Related Work

This research work mainly focuses on observations abstracted from the research done for the main functionalities which are vehicle detection, vehicle tracking, license plate recognition, and speed violations mainly related to the area of computer vision.

There are many methods for license plate recognition which is the core part of this research. From them, several alternatives for license plate recognition were found. Alternatives are analyzed by accuracy and implementation.

A. License Plate Identification

Mainly in the OpenCV-based approach has been used as a method of number plate extraction and number plate recognition[4]. In that paper, the first image has captured the image of the vehicle is captured and then resizing the image to the preferred size and bringing the image to grayscale is performed. The threshold is applied to localize the license plate. The license plate is then scanned and cropped to segment the license plate before character recognition.

The template matching[5] approach differs from the OpenCV method at the character recognition stage. In this alternative, a method called template matching is used to identify individual characters. In template matching, a huge database of every possible character is maintained. Whenever a segmented character is outputted from the previous stage, that character is compared with all the characters in the database. The best match is selected as the output. The template matching method would be a bit time-consuming as each character is compared with every character in the database.

Neural Network[6] based method differs from the previous method at pre-processing and character recognition stages. Noise filtering and shadow removing must be done. In addition to this, a down-sampling of the input image may be required depending on the images used to train the neural network. Character recognition is done using a trained neural network. There is a different kind of neural network architectures such as Convolution neural networks[7], Artificial Neural Networks, and Linear Vector Quantization.

To improve the existing license plate identifications identifying the most suited one is a must. A review paper[8] that compares those methods claims the neural network-based method has higher accuracy. On the other hand, the neural

network-based method works fine even with distorted input images. This is because the neural network is trained with a huge dataset. This data set consists of quality images as well as distorted images. Hence, whenever a distorted image is given as the input to the system, there is a high probability that the vehicle number is recognized correctly.

Multi-column deep neural network (MCDNN) is proposed for Chinese character recognition[9]. It was tested on a regular computer with a CUDA graphics card. Because there are 3755 or more classes of Chinese characters, recognition is significantly more difficult. They used photos at a resolution of 40X40 pixels. The system's deep neural network was a feed-forward connectionist system made up of a series of convolutional, max-pooling, and fully connected layers. They have built nine MCDNNs from previously trained 8 nets. Four of them were basic DNNs with one column. The best MCDNN has a 4.215% error, much lower than the best DNN error, 5.528%. The human error rate was calculated to be 3.87 percent. This MCDNN also has a new record-breaking error rate of 0.291 percent when considering the top 10 predictions, which will be significant for more complicated context-driven systems incorporating language models. However, the GPU/CPU code has not been properly optimized. They said at the end that this may be improved more.

The use of character-level convolutional networks (ConvNets)[10] for text classification was investigated. They created a few large-scale datasets to demonstrate that character-level convolutional networks can reach state-of-the-art or competitive outcomes. They investigated using temporal (one-dimensional) ConvNets to handle text as a form of the raw signal at the character level. The gradients are obtained through back-propagation to do 7 optimizations, and the character level ConvNet architecture is modular. One large ConvNet and one tiny ConvNet were present. Both have nine layers, six of which are convolutional and three of which are fully linked. Their analysis shows that character-level ConvNet is an effective method but on the other hand, how well the model performs in comparisons depends on many factors, such as dataset size, whether the texts are curated, and choice of the alphabet.

Our proposed system differs from the mentioned literature because this system is implemented for a real traffic environment and tested for actual traffic data. But all the schemes of the above-mentioned pieces of literature were tested under constraints. Most of them have vehicles coming at specified speed ranges and specified alignments for the license plates. But our proposed system is capable of handling the license plate identification by mitigating mentioned constraints to a certain degree with good lighting conditions.

B. Vehicle Detection

A paper has presented vehicle detection and helmet detection system computer vision. It detects traffic density on-road and according to the density it controls traffic. It counts the number of vehicles and according to the density a dynamic timing is set. YOLO (You Only Look Once)[11] is used to detect objects. YOLO uses deep learning and convolution neural networks for this purpose since it detects objects in real-time given the level of certainty with localizing the object with bounding boxes. Artificial neural networks (ANN) are used in this research to detect vehicles. There are two parts to this research. In the first part moving vehicles are successfully detected. The second part is dynamic traffic control. The number of cars in a specific region is calculated by the difference between entering and leaving vehicles. Then the density at any intersection point can be found by dividing the number of cars in that lane by length and the width of the lane. In this research, logic was set up for controlling the traffic signals by detecting some rules. Detecting vehicles at night-time was not accurate.

C. Vehicle Tracking

Tracking several objects for various purposes has been a trending topic in the scientific world for the past decade. Multiple object tracking was also possible with more processing power. The majority of the proposed systems, on the other hand, have used background removal approaches such as Particle Filters[12] or Kalman Filters[13].

However, most background subtraction approaches failed for this task because they treat all moving objects as items of interest. To put it another way, the tracking system should only track automobiles. The inability to distinguish two overlapping objects is another issue with the background subtraction method. Most research introduces a different method to initialize and maintain tracking just for items with past track records, which are computationally inefficient.

Max Margin Object Detection [14] is an alternative for background subtraction. There, must initialize tracking by giving the initial coordinates of the object to be tracked. Then Max Margin object detection will calculate a score for a sliding window for a non-overlapping window and select the one with the highest score as the next coordinate set. Even though we choose this method as this one is the most computationally efficient, this is a greedy method which means it does not give the optimal solution.

A paper presents an improved background-updating algorithm by using wavelet transform on the dynamic background and then tracking the vehicles by a feature-based tracking method[15]. The approaches presented as feasible to detect moving vehicles in video streams are the background difference method, inter-frame difference method, edge-detection method, optical-flow method, and block matching method. The model designed through the paper employs a background detection and a feature-tracking method to identify moving vehicles. It is described in four steps: Background difference is performed with the filtered image while updating the background using a dynamical background updating algorithm. This algorithm considers the weighting factor to adjust the speed of the background update. This parameter must be adjusted such that it is neither too low nor too high. The complications observed through this weighting factor have been resolved using the wavelet transform. The second step is to perform edge detection on a selected area to detect the contours of the vehicle. The color histogram of the target is plotted in the third step and the center of gravity of the moving target is calculated as the fourth step. Three modules are present within the system presented: video loading, detection of violating vehicles, and violation evidence storage.

A tracking system that uses tracking by detection paradigm[16]. YOLOv3 is applied to achieve a faster detection. By improving the ideas of deep SORT presents a better algorithm for the challenges present in occlusion handling, fast movements, and changes of shape. The generic method consists of three steps. First, object identification is done using the YOLOv3 scheme. The second stage utilizes the state of the track as an eight-quantity vector to model the state. The final step is, given the predicted state using Kalman filters using the previous information and the newly detected box in the current frame association is made. The main limitation of this tracking algorithm is it only enables short-range tracking because Kalman filters only accurately predict the short term. This algorithm requires high computation power in GPU and CPU. As the bounding box size increases the accuracy of this algorithm decreases.

D. Speed Violation Detection

A violation detection using infrared sensors along with license plate identification is proposed as a solution for the traffic management problem[17]. These two sensors detect the time difference when the car triggers the sensors. Sensors take continuous reading and output analog voltage. Speed is calculated based on the basic principle of distance divided by the time difference. This paper proves that the suggested method outputs the speed which equals the speed indicated in the speed meter of the vehicle. But the results show that the license plate identification method suggested is depending highly on lighting conditions and angle of capture. This license plate identification needs to be improved to get real-time capturing of the violations.

3. Proposed Method

The proposed methodology can be represented as a flow presented in Fig. 1.

Using YOLO, only the vehicle type was detected. But with the modifications to the Deep SORT tracking algorithm vehicles were successfully tracked and able to distinguish each vehicle of the same type. It was carried out with the use of transfer learning. Then generic algorithms with matrixes were used to detect the speed violation. The bounding box has been created such that there is minimum noise, only the vehicle was cropped. This was carried out using a CNN.

A. Speed Violation Detection

First, marking two lines across the road or rather mark two points was carried out. The deep SORT algorithms were executed to separate each vehicle and obtain the trajectory of the vehicle. Since the vehicle is represented by the bounding box and it was programmed in such a way that when the bounding box intersects the pixel region of the reference line marked in the road, the timer was made to start. Because of the presence of the larger vehicle density large matrices are used. In these matrices, as the first few entries are being filled the last rows must be erased without affecting the top entries. In the time matrix, the starting and the end time is marked, and by subtraction, the time was calculated. For a curved road, the distance in each path is different but in a linear road, all the paths have the same distance. The camera is positioned in a be such that the paths or the distance between the two points are almost linear in every case. Therefore, the error caused by the curvature is almost negligible. If we are detecting speed in a highway then we want the instantaneous velocity results. As the vehicle reaches the endpoint/line then the time is stored into the corresponding position in another matrix. These two can be called the starting time matrix and the stopping time matrix. If a depth camera had been used, then it can accurately detect the distance between the two points. However here depth cameras are not used since the distance is known, then subtract the two matrix values to find the time, and then the speed can be calculated.

To apply these values to the matrix, accelerated algorithms are used. Usually, to generate these matrices packages NumPy and pandas were used. In NumPy, there is a version called CUDA-enabled NumPy. There is a separate matrix for velocity. As a new vehicle approaches, its time matrices are inserted into the first row and the previous gets shifted to the row below. If this continues there will be an infinite matrix that cannot be processed by the GPU entries are filled, then the bottom one is erased using counters. However, if the vehicle is between the two lines and its start time matrix gets erased then there will be a problem as there is no value to subtract from which will cause the program to crash then and there, hence such values are stored in a database. This database also gets updated in real-time. A frame-by-frame capturing is carried out if a violation was identified. Therefore, the actual tracking must start from the second line. If the label indicates that the speed of the vehicle at the second line is above the allocated speed, then the algorithm starts capturing images of the vehicle from the second line no matter the route was taken by the vehicle. It is important to extract the frame which our algorithms can perform the license plate detection task at best. In the rare case that another vehicle is in front of the detected vehicle and the number plate cannot be seen and it continues to obscure the number plate then capture the image until the car leaves the feed of the camera should be done. The number of captured images changes with each car as each speed is different. Also, another possibility is that sunlight or rain can obscure the image. The best frame captured can be extracted using a neural network or an SVM (Support Vector Machine). A traditional CNN can be used for the task, then this image is given to another neural network to extract the license plate and in turn, is forwarded to the license plate detection algorithm that localizes each letter and provides the output.

B. License plate identification

According to the traditional method, first histogram equalization has been carried out onto the image, then binarization followed by the gamma correction, and finally white background with the letters in black has been detected. There is a function called control lines in OpenCV that helps to draw boxes around pixel clusters and break them down as boxes with letters. This is given to either pi tesseract or something similar. However, the time consumption is enormous because of the manual decision of the gamma value and the binarization value between 0 and 255. If the plate has

some alignment, then the top line of the cropped image can be aligned by using the rectification method. However practical reasons like the line not being detected are present, the aforesaid method can be utilized under controlled lighting conditions however, the detection is not achieved in real-time.

The license plate localizer created using YOLO was found to be very effective and it cuts out the license plate accurately irrespective of the position of the vehicle. The advantage of using YOLO instead of the OpenCV method is that the algorithm (NN) knows what are the values that can be used for image binarization and gamma values. This algorithm needed at least 2000 images with different colors, lighting conditions, orientations, shadows. YOLO custom neural network with 128 layers and branches that can do this with only 500 images were created that included convolution, dense, and localization layers as depicted in Fig. 2. In vertical number plates, there are a lot of tasks that need to be performed with word segmentation, line segmentation.

The next task is to direct the program to read from left to right. What happens is that the letters usually are displayed according to the speed at which they are detected. How this was resolved by directing the program to read the coordinate on the far left and then move to the right and first select the lowest y-coordinate. This was done by using a custom function. However, in our model, there were bounding boxes that sometimes cut out the letters hence we used the line division method getting the mid-point of each bounding box from top and bottom and then right and left to represent by a dot, and this dot's x and y coordinate and were then saved to a text file. Each vehicle has three text files one for start time then stop time and the license plate. The next step is to upload to the cloud using the text file and the API and write a program using python. The architecture of the license plate character recognition model can be demonstrated as follows. When building the character recognizer accuracy should be the optimizing factor and processing speed should be the satisfactory factor. Therefore to process in real-time tiny-YOLO-based approach was taken because it has the capability of processing at 45 FPS.

C. Deep Learning Hardware and Software

In terms of hardware GPUs and CPUs are the important components to perform the CNNs and other generic algorithms. When it comes to the CPU fewer cores, but each core is much faster and much more capable at sequential tasks. But the GPUs have more cores, but each core is much slower (low clock speed) and "dumber", and it is more suited for parallel computing since it has a large number of computing elements. A comparison of CPUs and GPUs is demonstrated in the following Fig. 3.

The cores mentioned here are tensor cores which are special hardware used to perform deep learning computations. Tensor cores use mixed precision (In deep learning we do arithmetic using 32-bit Floating point) such as FP16(16-bit floating point) used for multiplication, and FP32(32-bit floating point) is used for the addition. GPU performs matrix multiplication can be accelerated since all output elements are independent and can be trivially parallelized. To program GPUs CUDA was used. CUDA is like C/C++ that directly run-on GPU. NVIDIA provides optimized API which made programming much easier.

When performing deep learning, the deep learning framework the following features were considered for the selection,

- Allow rapid prototyping
- Automatically computing gradients using computational graphs.
- Runs efficiently on GPUs.

Following alternatives for implementation of the neural network inside the selected method were considered.

- TensorFlow GPU version
- Caffe with CUDA and CUDDN enabled
- Torch- In PyTorch there are three levels of abstractions that enable performing neural networks. The lowest level of abstraction is a tensor which is a NumPy array that can be run on GPU. Auto grad level of abstraction is the package for building computational graphs and module-level of abstraction represents the neural network layer. Dynamic computational graphs let us use regular python control flow during the forward pass.

Among the above-mentioned frameworks from factors such as coding support, testing methods this work used was the TensorFlow.

4. Performance Analysis And Results

The performance of the speed violation and license plate identification was analyzed separately. This implemented system used a neural network for number plate localization. The reason for not implementing that neural network for violation detections was to minimize the processing power to implement more deep neural networks.

A. Speed Violation Detection

In this, a threshold was assigned as a speed limit. The speed limit is and a variable that depends on the specific junction, vehicle density, and positioning of the camera since the distance is pre-known, and as soon as the bounding box of the vehicle intercepts the pixel region of the second reference line speed was captured as presented in Fig. 4.

In Fig. 4, the vehicle is represented by a 100-pixel square of which the central point is the same as the central point of the YOLO detected bounding box to avoid error due to the bounding box. By using this method and the results it was verified that the speed is approximately similar to the speed meter reading in the vehicle. After that, if a vehicle exceeds the speed limit the vehicle was captured frame by frame by the IP camera as indicated in Fig.5.

B. License Plate Identification

In most works of literature, the license plate has been detected using preprocessing methods such as gamma corrections, otsu thresholding, resizing, and finally the identification process.

The work which this paper presents focuses on the complete redesign of tiny YOLO architecture. The following Fig. 6 and Fig. 7 represent the output of a few selected license plates.

Even though the algorithm successfully detects the characters, characters are not in order. Our work also presents how to order the characters in an ordered way without using the word or line segmentation as depicted in Fig. 8 and Fig. 9. Using the middle point of each character is identified and plotted in the X, Y plane, by using their definition of SVM clustering is done. Then 3D gaussian was introduced to perform this methodically.

The training was carried out for the deep neural network for a set of vehicle number plates. The system was validated and tested on vehicle number plates that have a similar distribution to real data and were tested for 154 frames. Initially, the system was trained for the vehicle number plates extracted from videos. Training results can be demonstrated by following Fig. 10.

The above Fig. 10 shows training accuracy achieved as 84%. Following table 1, represents the results for the test set locations 1 and 2, which are presented in Fig. 11 and Fig. 12 respectively.

Table 1- Character recognition results of the license plate when the camera is placed near the toad(Place 1)

	Lane 1	Lane 2
All Correct	76.74% (33/43)	63.41% (26/41)
One character incorrect	11.63% (5/43)	21.95% (9/41)
Two characters incorrect	9.3% (4/43)	2.44% (1/41)
Three characters incorrect	0% (0/43)	9.76% (4/41)
Four characters incorrect	0% (0/43)	2.44% (1/41)
More than four characters incorrect	2.33% (1/43)	0% (0/41)

The results show when the license plate is larger in view to get better results.

Table 2-Character recognition results of the license plate when the camera is placed near the road(Place 2)

	Towards camera	Away from the camera
All Correct	63.33.%(19/30)	65% (26/40)
One character incorrect	20% (6/30)	20% (8/40)
Two characters incorrect	3.33% (2/30)	10% (4/40)
Three characters incorrect	63.33% (1/30)	2.5% (4/40)
Four characters incorrect	0% (0/30)	0% (1/41)
More than four characters incorrect	6.67% (2/30)	2.5% (0/41)

The values presented in table 2, demonstrates relatively high accuracy for the license plates in a real traffic environment despite the footage obtained by a low-quality camera for different alignments.

5. Conclusion

In a summary, the model created was successful since the modified tiny YOLO architecture provides a 98% accuracy. The results proved that the violations were detected as planned. Both CPU and GPU have been employed in the model. The authors at the present plan the real-life implementation of the project. Our research introduced several novel processes. The first is in the form of the application of the algorithms. Numerous customizations were adopted into the model. Also, simpler adaptations of methods were achieved as in the case of using 500 images instead of the required 2000 images for license plate localization. The parameters have been optimized according to the priorities of the model. One of the important factors that are required to be highlighted is that the resources have been effectively used in the proposed method. Instead of using a depth-sensing camera, a novel method was adopted in the proposed scheme only utilizing a basic varifocal lens camera. Numerous field trials proved that the proposed methodology provides far superior performance levels compared to the conventional systems and the other existing methodologies, which will certainly cater to the demanding requirements of Transportation 4.0.

6. Future Work

Future research is in progress to be carried out by using the Mask-RCNN object detection architecture which requires higher computational power. The aforesaid improves the speed violation method proposed since the inception of the

pixel region of the reference signal can be determined much accurately which results in a much accurate and precise violation detection system. The works in progress to scale up the proposed method for multiple junctions.

Declarations

CONFLICT OF INTEREST

There is no conflict of interest.

AUTHOR'S CONTRIBUTIONS

Conceptualization: Udaya Dampage; Methodology: Udaya Dampage and K.K.N. Hasanthan; Formal analysis and investigation: K.K.N. Hasanthan, H.A.D.S.Gimhana, H.M.T.Y.N.Bandara, S.V.Maddumage; Writing - original draft preparation: Udaya Dampage, H.A.D.S.Gimhana; Writing - review and editing: Udaya Dampage; Resources: H.M.T.Y.N.Bandara, S.V.Maddumage; Supervision: Udaya Dampage.

References

- [1] R. Islam, K. F. Sharif, and S. Biswas, "Automatic vehicle number plate recognition using structured elements," *Proc. - 2015 IEEE Conf. Syst. Process Control. ICSPC 2015*, no. December, pp. 44–48, 2016, doi: 10.1109/SPC.2015.7473557.
- [2] O. V. Olesen, R. R. Paulsen, L. Højgaard, B. Roed, and R. Larsen, "Motion Tracking for Medical Imaging: A Nonvisible Structured Light Tracking Approach," no. February, 2012, doi: 10.1109/TMI.2011.2165157.
- [3] A. Ess, K. Schindler, B. Leibe, and L. Van Gool, "Object detection and tracking for autonomous navigation in dynamic environments," *Int. J. Rob. Res.*, vol. 29, no. 14, pp. 1707–1725, 2010, doi: 10.1177/0278364910365417.
- [4] A. Komarudin, A. T. Satria, and W. Atmadja, "Designing License Plate Identification through Digital Images with OpenCV," *Procedia Comput. Sci.*, vol. 59, no. Iccsci, pp. 468–472, 2015, doi: 10.1016/j.procs.2015.07.517.
- [5] I. Kusumadewi, C. A. Sari, D. R. I. Moses Setiadi, and E. H. Rachmawanto, "License Number Plate Recognition using Template Matching and Bounding Box Method," *J. Phys. Conf. Ser.*, vol. 1201, no. 1, 2019, doi: 10.1088/1742-6596/1201/1/012067.
- [6] H. E. Kocer and K. K. Cevik, "Artificial neural networks based vehicle license plate recognition," *Procedia Comput. Sci.*, vol. 3, no. December, pp. 1033–1037, 2011, doi: 10.1016/j.procs.2010.12.169.
- [7] K. M. Hung and C. T. Hsieh, "A real-time mobile vehicle license plate detection and recognition," *Tamkang J. Sci. Eng.*, vol. 13, no. 4, pp. 433–442, 2010, doi: 10.6180/jase.2010.13.4.09.
- [8] A. Goyal and R. Bhatia, "Various Techniques for Number Plate Recognition- A Review," *Int. J. Comput. Appl.*, vol. 143, no. 11, pp. 25–27, 2016, doi: 10.5120/ijca2016910471.
- [9] D. Cireş, T. R. N. Idsiaş, and D. Cireş, "Multi-Column Deep Neural Networks for Offline Handwritten Chinese Character Classification Multi-Column Deep Neural Networks for Offline Handwritten Chinese Character Classification," no. 3755, pp. 1–6, 2013.
- [10] X. Zhang and Y. LeCun, "Text Understanding from Scratch," pp. 1–9, 2015, [Online]. Available: <http://arxiv.org/abs/1502.01710>.

- [11] J. S. D. R. G. A. F. Redmon, "(YOLO) You Only Look Once," *Cvpr*, 2016, doi: 10.1109/CVPR.2016.91.
- [12] M. Jaward, L. Mihaylova, N. Canagarajah, and D. Bull, "Multiple object tracking using particle filters," *IEEE Aerosp. Conf. Proc.*, vol. 2006, 2006, doi: 10.1109/aero.2006.1655926.
- [13] C. Montella, "The Kalman Filter and Related Algorithms A Literature Review," *Res. Gate*, no. May, pp. 1–17, 2011, [Online]. Available: https://www.researchgate.net/publication/236897001_The_Kalman_Filter_and_Related_Algorithms_A_Literature_Review.
- [14] D. E. King, "Max-Margin Object Detection," 2015, [Online]. Available: <http://arxiv.org/abs/1502.00046>.
- [15] M. Jaward, L. Mihaylova, N. Canagarajah, and D. Bull, "Multiple object tracking using particle filters," *IEEE Aerosp. Conf. Proc.*, vol. 2006, 2006.
- [16] O. Access, "We are IntechOpen , the world ' s leading publisher of Open Access books Built by scientists, for scientists TOP 1 %."
- [17] J. Yim, R. A. Cadiente, G. P. Mayuga, and E. R. Magsino, "Integrated Plate Recognition and Speed Detection for Intelligent Transportation Systems," *ISCAIE 2020 - IEEE 10th Symp. Comput. Appl. Ind. Electron.*, pp. 6–11, 2020, doi: 10.1109/ISCAIE47305.2020.9108807.

Figures

Figure 1

Overall System Workflow

```

mini_batch = 1, batch = 16, time_steps = 1, train = 0
layer  filters  size/strd(dil)  input          output
0 conv   32    3 x 3/ 2    416 x 416 x 3 -> 208 x 208 x 32 0.075 BF
1 conv   64    3 x 3/ 2    208 x 208 x 32 -> 104 x 104 x 64 0.399 BF
2 conv   64    3 x 3/ 1    104 x 104 x 64 -> 104 x 104 x 64 0.797 BF
3 route  2          1/2 -> 104 x 104 x 32
4 conv   32    3 x 3/ 1    104 x 104 x 32 -> 104 x 104 x 32 0.199 BF
5 conv   32    3 x 3/ 1    104 x 104 x 32 -> 104 x 104 x 32 0.199 BF
6 route  5 4          -> 104 x 104 x 64
7 conv   64    1 x 1/ 1    104 x 104 x 64 -> 104 x 104 x 64 0.089 BF
8 route  2 7          -> 104 x 104 x 128
9 max          2x 2/ 2    104 x 104 x 128 -> 52 x 52 x 128 0.001 BF
10 conv  128    3 x 3/ 1    52 x 52 x 128 -> 52 x 52 x 128 0.797 BF
11 route 10          1/2 -> 52 x 52 x 64
12 conv   64    3 x 3/ 1    52 x 52 x 64 -> 52 x 52 x 64 0.199 BF
13 conv   64    3 x 3/ 1    52 x 52 x 64 -> 52 x 52 x 64 0.199 BF
14 route 13 12        -> 52 x 52 x 128
15 conv  128    1 x 1/ 1    52 x 52 x 128 -> 52 x 52 x 128 0.089 BF
16 route 10 15        -> 52 x 52 x 256
17 max          2x 2/ 2    52 x 52 x 256 -> 26 x 26 x 256 0.001 BF
18 conv  256    3 x 3/ 1    26 x 26 x 256 -> 26 x 26 x 256 0.797 BF
19 route 18          1/2 -> 26 x 26 x 128
20 conv  128    3 x 3/ 1    26 x 26 x 128 -> 26 x 26 x 128 0.199 BF
21 conv  128    3 x 3/ 1    26 x 26 x 128 -> 26 x 26 x 128 0.199 BF
22 route 21 20        -> 26 x 26 x 256
23 conv  256    1 x 1/ 1    26 x 26 x 256 -> 26 x 26 x 256 0.089 BF
24 route 18 23        -> 26 x 26 x 512
25 max          2x 2/ 2    26 x 26 x 512 -> 13 x 13 x 512 0.000 BF
26 conv  512    3 x 3/ 1    13 x 13 x 512 -> 13 x 13 x 512 0.797 BF
27 conv  256    1 x 1/ 1    13 x 13 x 512 -> 13 x 13 x 256 0.044 BF
28 conv  512    3 x 3/ 1    13 x 13 x 256 -> 13 x 13 x 512 0.399 BF
29 conv  123    1 x 1/ 1    13 x 13 x 512 -> 13 x 13 x 123 0.021 BF
30 yolo
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y:
1.05
nms kind: greedynms (1), beta = 0.600000
31 route 27          -> 13 x 13 x 256
32 conv  128    1 x 1/ 1    13 x 13 x 256 -> 13 x 13 x 128 0.011 BF
33 upsample          2x    13 x 13 x 128 -> 26 x 26 x 128
34 route 33 23        -> 26 x 26 x 384
35 conv  256    3 x 3/ 1    26 x 26 x 384 -> 26 x 26 x 256 1.196 BF
36 conv  123    1 x 1/ 1    26 x 26 x 256 -> 26 x 26 x 123 0.043 BF
37 yolo
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y:
1.05

```

Figure 2

Architecture of the CNN(Character Recognition License Plate)

	Cores	Clock Speed	Memory	Price	Speed
CPU (Intel Core i7-7700k)	4 (8 threads with hyperthreading)	4.2 GHz	System RAM	\$385	~540 GFLOPs FP32
GPU (NVIDIA RTX 2080 Ti)	3584	1.6 GHz	11 GB GDDR6	\$1199	~13.4 TFLOPs FP32

Figure 3

CPU vs GPU (At April 2019)

Figure 4

Capturing of the speed

Figure 5

Frame-by- Frame Capturing of the Violated Vehicle

Figure 6

Detection of the Single raw license plate 1

Figure 7

Detection of the single raw license plate 2

Figure 8

Detection of the double line license plate 1

Figure 9

Detection of the double line license plate 2

Figure 10

Training results of tiny-YOLO architecture for character recognition

Figure 11

Location 1 (Lane 1 is leftmost and Lane 2 is the middle lane of the image)

Figure 12

Location 2 (Number plates were recognized of vehicles coming towards and going away)