Red-Light Running Violation Detection of Vehicles in Video Using Deep Learning Methods

Nam Nguyen Van[1,2(✉)], Hanh Le Thi[1], Minh Phan Nhat[3], and Long Lai Ngoc Thang[4]

[1] Department of Data Governance, Viettel Group, 7 Alley, TonThatThuyet Street, CauGiay, Hanoi, Vietnam
{namnv78,hanhlt87}@viettel.com.vn
[2] Thuyloi University, 175 Tayson, DongDa, Hanoi, Vietnam
nvnam@tlu.edu.vn
[3] Fontbonne University, 6800 Wydown Blvd., St. Louis, MO 63105, USA
phanm@fontbonne.edu
[4] Hanoi University of Science and Technology, 1 DaiCoViet, HaiBaTrung, Hanoi, Vietnam
long.lnt183581@sis.hust.edu.vn

Abstract. Recently, Traffic Monitoring Systems (TMS) based on camera are widespread used in many large cities thanks to advances in artificial intelligence especially in deep learning and computer vision. Detection of traffic violation of vehicles is a critical problem for law enforcement in such TMS due to complicated trajectories of different vehicle types in road. Existing methods based on computer vision techniques for detecting, tracking vehicles and then applying violation rules on the perceived path of every vehicles. In this paper, we present a novel approach which is based on the flexible LSTM recurrent neural networks in addition to the traditional fixed rules to detect red-light running violation of vehicles. We also present our improvements on the existing DeepSort tracking algorithm for faster and more accurate ID matching. We evaluate our deep LSTM with attention mechanism on a dataset (Dataset and code are available here: https://github.com/namnv78/RunningRedlight) of 108 traffic videos captured from three road intersections in Vietnam including 628 red-light running violated vehicles. Our method achieved a precision, recall and F1-score of more than 99% which is 3% higher than the traditional rule-based method.

Keywords: Traffic violation detection · Vehicle detection and tracking · Recurrent neural networks

1 Introduction

Robust detection and identification of law-violating vehicles from monitoring camera in cross-roads are critical for ensuring road safety and enforcing the laws.

N.-S. Vo et al. (Eds.): INISCOM 2022, LNICST 444, pp. 214–227, 2022.
https://doi.org/10.1007/978-3-031-08878-0_15

The task is highly challenging under a full range of traffic, lighting and weather conditions. Solving this problem necessitates solving and integrating solutions of multiple difficult sub-tasks, including (a) detection, embedding and tracking of vehicles, and (b) automatic violation detection. Although effective techniques for sub-tasks exist, there is a large gap on building an integrated solution that works in real time and filling this gap demands novel adaptation and systematic thinking.

We based our method on the efficient YOLOv4 [2] and the light-weight SelectSLS [8] for vehicle detection and embedding. Once the vehicles are detected and embedded, tracking can be carried out using several methods such as DeepSORT [16], JDE [15] and DeepMOT [17]. Typically these methods work by extracting features from bounding boxes of the vehicles and associating them with the existing trajectories. This strategy works well for a small number of vehicles, but the computation cost quickly grows prohibitive in crowded roads and with tracking duration.

Our main contributions are as follows. We considerably reduce the computation burden as well as ID missing ratio compared to the original DeepSort tracking algorithm by matching the new detection and tracks by group, involving and approximating only the Mahalanobis distance of the centroid coordinates but not the size of the bounding boxes. We then, proposed Deep LSTM and Dense LSTM to detect running red-light violations of the vehicles from the their trajectories. We also found that the Deep LSTM with attention mechanism produces the best accuracy in our dataset including 108 traffic videos in Vietnam. This is because the model can learn and keep the most important weights corresponding to the essential evidences for the violations.

The rest of the paper is as follows. Section 2 reviews related works. Section 3 describes our tracking algorithm named WAYS. Section 4 briefly summarize the rule-based method for violation detection. Section 5 presents in detail our recurrent-based methods. In Sect. 6, we demonstrate our experimental results on real dataset. Finally, the last section concludes our works.

2 Related Works

Here we briefly review recent methods that solve the three main sub-tasks of identification of law violation vehicles: vehicles detection, tracking and traffic law violation recognition. Although they differ in specific architectures, they are all trained deep neural networks.

Vehicles Detection: There is a rich set of powerful deep learning techniques for detecting objects in traffic scene like cars, bikes and drivers. One of the most used methods is Faster-RCNN [11] which consists of two two-stages. The first stage employs a Regional Proposal Network (RPN) to regress the rectangular bounding box of the objects. The second stage is to classify objects in the these bounding box using a convolutional neural network (CNN). This two-stage strategy is accurate but at the cost of running time due to a high number of proposal regions. Faster-RCNN can only predict at 5 fps using GPU while achieving an

accuracy of 73.4% and 70.4% mAP on PASCAL VOC 2007 and 2012, respectively with 300 proposals per image.

One-shot techniques avoid the cost by regressing the bounding boxes and predicting objects at the same time, e.g., those proposed in SSD [7] and YOLO [10]. Here input image is divided into a grid of cells, each of which is mapped to a point of the feature map resulting from the image via a multi-channel CNN such as Resnet [3] or VGGNet [12]. Several anchor boxes are proposed to regress a bounding box as well as to predict the probabilities of objects. With fewer number of proposals, one-shot models are much faster. Over the years, the accuracy of one-shot methods has improved. The latest version of YOLO [2] utilizes data augmentation and achieves an impressive accuracy of 43.5% AP (65.7% AP50) on MS COCO dataset and a frame rate of 65 fps on GPU Tesla V100.

Vehicles Tracking: Tracking moving vehicles boils down to matching features extracted from the recognized object bounding boxes to existing trajectories. At present, there are several powerful online tracking methods including DeepSORT [16], JDE [15] and DeepMOT [17]. However, feature matching using the default Hungarian algorithm is expensive for a large number of objects, which is often the case at busy intersection of the roads.

Recognition of Traffic Law Violation

Very few works have been proposed to automatically reason the traffic violation of vehicles given that they can be detected and tracked in videos. In [14], the authors proposed a flowchart for identifying the violation of over speed, of running over red light, of crossing the line and of retrograde after detecting the stop lines, the lanes and the traffic lights. They tracked the center of the vehicles and detected rules violations using certain thresholds. For instance, if the red light is on and the y-coordinate of the center is greater than a threshold then the vehicle runs the red light. However, at a high density of traffic, there is a long queue of vehicles waiting at the stop lines with various values of y-coordinate but none breaks the rule. Similarly, their flowchart for detecting other violations is not reasonable. The work in [1] aims to detect the violation of crosswalk at signal, and of no helmet, but this is also very limited without any explanation to cover all possible situations in reality.

Due to the complicated traffic situation in Vietnam with majority of motorbikes, we propose an comprehensive approach for this issue including vehicle detection, embedding, tracking and law violation recognition using deep learning methods.

3 Vehicle Tracking Algorithm

We improved the original DeepSORT tracking algorithm to adapt to the city traffic as shown in Fig. 1. Our algorithm firstly detects the vehicles and embeds them. Then, we calculate the Mahalanobis distance as well as the cosine similarity between the newly detected bounding boxes and the ones from the existing tracks. Finally, our group-based matching algorithm divides the bounding boxes

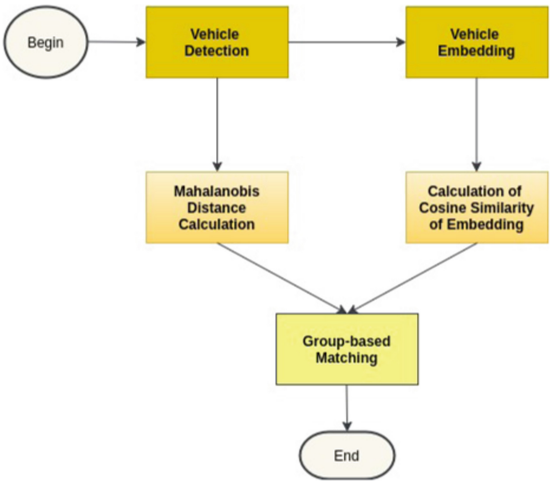

Fig. 1. Improved vehicle tracking algorithm

into two groups according their labels. The first group includes cars, trucks and buses and the second includes motorbikes and bicycles. Since the time complexity of Hungarian algorithm is $O(n^3)$ in the worst case for n objects, the matching within each group will considerably decrease the processing time. For example, if there are two equal groups, the complexity will be reduced by 4 times.

The matching algorithm is based on a matrix of costs relating to both the feature and the location of the boxes. The location metric is the squared Mahalanobis distance and calculated as

$$d_{maha}[i,j] = (l_j - m_i)^T C_i^{-1} (l_j - m_i)$$

where $l_j = (x, y, a, h)$ corresponding to coordinates of the center, aspect ratio and height of the newly j^{th} detected box, respectively; m_i and C_i are the mean and covariance of the previous measurements for the i_{th} track estimated by Kalman filter. Since we estimate only the future position of the vehicles, d_{maha} can be approximated as follows [9]:

$$d_{maha}[i,j] \approx \frac{v_1^2}{c_{11}} + \frac{v_2^2}{c_{22}} \tag{1}$$

where $[c_{11}, c_{22}]$ is the diagonal of matrix C and $v = [v_1, v_2] = l_j - m_i$. As shown in [9], this approximation can decrease the processing time without loss of accuracy.

Given the embedding e_j^* of the j^{th} detected bounding box with detected label L_j^* of the group L, and the embedding of the i^{th} track at the age of k with the label L_j^k, the cosine distance between them is:

$$d_{embed}[i,j] = \begin{cases} 1 - \frac{e_i^k . e_j^*}{\|e_i^k\| \|e_j^*\|}, & \text{if } L_j^k \in L \\ \infty, & \text{if } L_j^k \notin L \end{cases} \tag{2}$$

The new detection is admissible to the track if the following condition holds $b = (d_{maha} \leq t_{maha})\&(d_{embed} \leq t_{embed})$ for pre-defined thresholds t_{maha} and t_{embed}.

Input :
A group of labels: L ;
Tracking IDs: Track_IDs={1,..,N} ;
Detected Bounding Box IDs including their labels: Box_IDs={1,..,M} ;
Age of tracks A: ={1,..,K} ;
Output:
Matched = {(i,j)}, $i \in \overline{1,N}, j \in \overline{1,M}$ and
Unmatched= {j, $(j \in Box\_IDs)\&(j \notin Matched)$ }

1 $D^L_{maha} \leftarrow \{d_{maha}[i,j]\}^{NxM}$ Eq. 1 on L ;
2 $D^L_{embed} \leftarrow \{d_{embed}[i,j]\}^{NxM}$ Eq. 2 on L ;
3 $Matched \leftarrow \emptyset$;
4 $Box\_IDs\_L \leftarrow \{j \in Box\_IDs | (Label(j) \in L)\}$;
5 $Unmatched \leftarrow Box\_IDs\_L$;
6 **for** $k = 1;\ k < K;\ k++$ **do**
7 $Track\_IDs\_k\_L \leftarrow \{i \in Track\_IDs | (A(i) = k)\&(Label(i) \in L)\}$;
8 **for** $i \in Track\_IDs\_k\_L$ **do**
9 $d_{embed}[i, j*] = 0$;
10 **for** $j \in Unmatched$ **do**
11 **if** $(d_{embed}[i, j*] < D^L_{embed}[i,j])$ **then**
12 $d_{embed}[i, j*] = D^L_{embed}[i,j]$;
13 $j^* = j$;
14 **end**
15 **end**
16 **if** $(d_{embed}[i, j*] \leq t_{embed})\&(D^L_{maha}[i, j*] \leq t_{maha})$ **then**
17 $Matched \leftarrow Matched \cup \{(i, j*)\}$;
18 $Unmatched \leftarrow Unmatched \setminus \{j*\}$;
19 **end**
20 **end**
21 **end**

Algorithm 1: Group-based Tracking Algorithm

The group-based matching algorithm works as follows (see Algorithm 1 for pseudo-code). The input consists of the group of labels L, a list of track indices $Track\_IDs$ including their corresponding ages A and labels, a list of detected bounding boxes with their labels. The output of the algorithm will be a set of matching (i, j) between the i_{th} track and the j_{th} detected bounding box and a set of unmatched detected boxes. From line 1 to line 5, we compute the matrices of cosine similarity and of Mahalanobis distance between the detected bounding boxes and the tracks of all ages of which their labels belong to L. From line 6 to 21, we iterate over the tracks from the youngest to the oldest ones. For a given track i, we find a j among M detected bounding boxes in the same label group so that the cosine distance between i and j is minimal. If the cosine distance and

the corresponding Mahalanobis distance between them are small enough then the matching is admissible. Otherwise, it is rejected.

The above tracking algorithm can produces accurate vehicle' states including their identity, label, four coordinates of their bounding box as well as the frame's sequential number during a time interval with the same traffic light (i.e. red, green or yellow). These are then used for red-light running violations of the vehicles.

4 Rule-Based Method

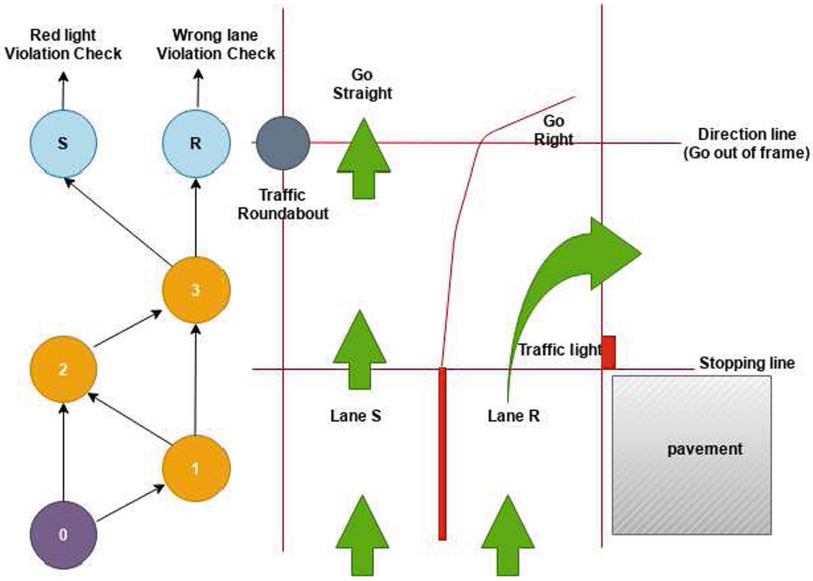

Fig. 2. The violation detection's diagram. Six states of vehicles are: 0: initial state, 1: under red line, 2: cross line, 3: over red line, S: straight, R: right. (Color figure online)

Based on a simple diagram, we can apply the rule of red-light running violation at an typical intersection: if a vehicle runs over the stop line while the red-light is on then it is violated. The violation detection's state transition diagram is therefore shown in Fig. 2. At a typical cross-road, a camera is configured to monitor one traffic flow as well as the corresponding red light signals. There is an additional green arrow light permitting vehicles to turn right even if the red light is on. The road thus has two or more lanes: lane S for going straight forward and lane R for turning right. A stopping line and a direction line are drawn to determine the states of the vehicles. Six possible states of a moving vehicle on the intersection are defined as follows:

- 0: the initial state when the vehicle is detected and tracked.
- 1: the state denotes whether the vehicle enters the region under the red line in lane S or R.
- 2: the state indicates whether the vehicle crosses the stopping line in lane S or R.
- 3: the state lets us know that the vehicle has entered the violation region if the red light is on.
- S: the state notifies that the vehicle is going straight, and
- R: this state indicates that the vehicle is turning right.

Based on the final states (S, R) and the passing ones (0, 1, 2, 3), we can decide whether the vehicle violates the traffic laws. If a vehicle is at the final state S while the red light is on then a running a red light violation is detected. In case the vehicle is currently at R and it has been detected passing the lane S then a wrong lane violation is recognized.

The rule-based method can work fast in real time deployment. However, due to the limited and fixed states, violations can be missed in cases of occlusion or truncation. We, therefore, propose recurrent-based methods which use recurrent neural networks to learn the violation from all the perceived time-series states of the vehicles.

5 Recurrent-Based Methods

Using recurrent neural networks, our recurrent-based methods can take as input a sequential data of every vehicles' states. The networks are mainly based on the Long-Short-Term-Memory (LSTM) thanks to their ability of learning long sequence.

5.1 Long Short Term Memory (LSTM)

LSTM [4] are improved recurrent neural networks (RNN) which can mitigate exploding and vanishing gradients with three interactive gates instead of one as in Fig. 3. These are the forget gate (f), the input modulated gate (i) and the output gate (o). As in original RNN, the x input is a sequence and h and c are the hidden and cell states, respectively. The forget gate decides whether the states are memorized or not aiming to save memory for important information.

In detail, the gates function as the following formula

$$
\begin{aligned}
f_t &= \sigma(U_f x_t + W_f h_{t-1} + b_f) \\
i_t &= \sigma(U_i x_t + W_i h_{t-1} + b_i) \\
o_t &= \sigma(U_o x_t + W_o h_{t-1} + b_o) \\
\tilde{c}_t &= \tanh(U_c x_t + W_c h_{t-1} + b_c) \\
c_t &= f_t c_{t-1} + i_t \tilde{c}_{t-1} \\
h_t &= o_t \tanh(c_t)
\end{aligned}
$$

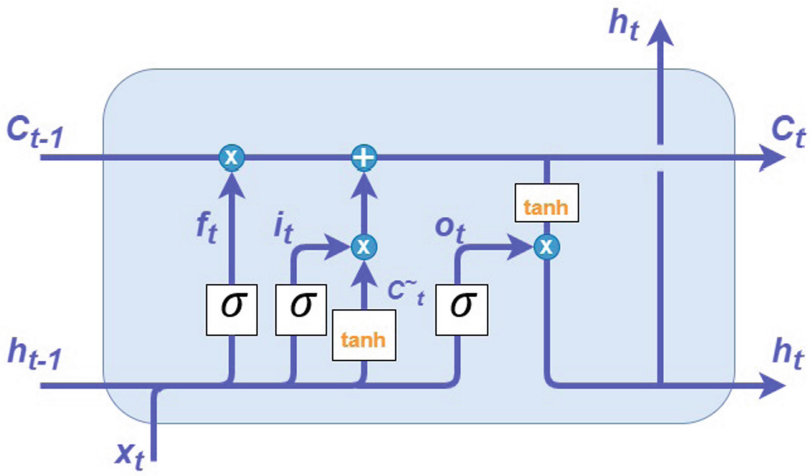

Fig. 3. Long short term memory

where $U_f, U_i, U_o, U_c, W_f, W_i, W_o, W_c$ are the weight matrices and tanh is the activation function. In fact, the early and strong LSTM works similarly to the skip connection used in recent ResNet [3] network to avoid vanishing gradient in deep networks. However, the attention mechanism results in wider networks instead of deep ones.

5.2 Attention Layer

Recently, the attention layer [13] can keep important weights of the network and therefore considerably reduce the size of the network. As shown in Fig. 4, the inputs of this attention layer are the hidden vectors $h_1, h_2, \ldots, h_n$ and the context c from the previous LSTM layer. The layer returns a vector z summarizing all hidden states that are relevant to context c as in the following equations.

$$m_i = \tanh(Wh_i + Uc + b)$$
$$p_i = softmax(<v, m_i>)$$
$$z = \sum_i p_i h_i$$

where W, H are the weight matrices corresponding to the input h_i, c and v is the learnable weight of h_i.

Inspired from LSTM and the attention mechanism, we build two robust networks for classification of input sequences representing the consecutive states of the vehicles while the red-light is on or off.

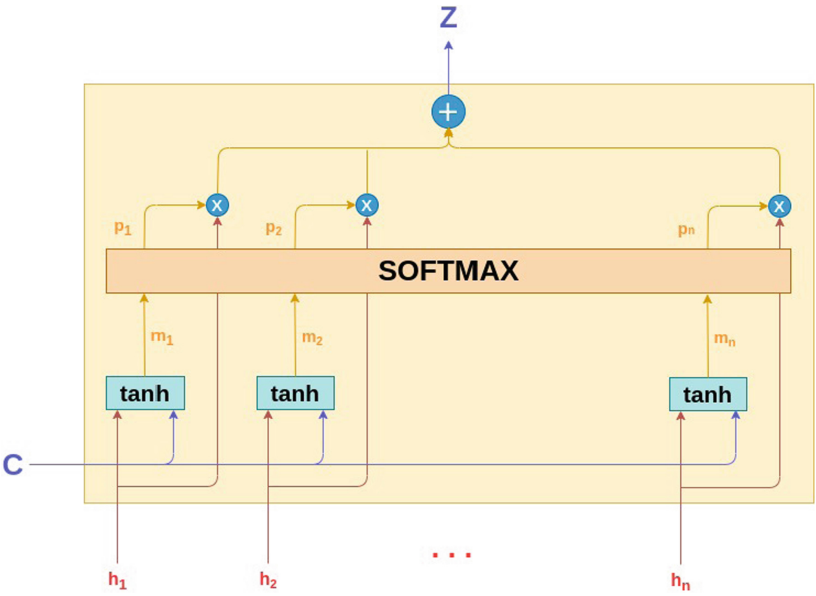

Fig. 4. The attention layer

5.3 Deep LSTM with Attention Mechanism

The Deep LSTM model (Fig. 5) contains three stacked LSTM layers, followed by an attention layer. The use of LSTM cells instead of RNN cells mitigates some of the exploding and vanishing gradient problems. However, LSTM is still struggling with longer sequences in the training data. Combine the output from LSTM layers with the attention mechanism helps the model focus on the more important information in the sequence, thus creating a more powerful and robust model.

The input for the model has size (B, N, D), where batch size $B = 64$, a fixed sequence length $N = 147$ (consecutive frames), and $D = 4$ represents the upper left/lower right coordinates of the bounding box. The Deep-LSTM module contains three stacked LSTM layers with hidden states h and cell states c of size 100 for each layer. The attention mechanism takes in both cell state c and hidden states h from the last LSTM layer and outputs a 128-dimension vector z. z is then passed through a classification layer with two nodes, representing if the vehicle ran red light or not.

5.4 Dense LSTM with CNN1D Method

Our Dense LSTM (Fig. 6) consists of 8 LSTM layers which are densely connected. This is inspired from Densenet [5] which establishes skip connections from any layer to all of its later ones to avoid vanishing gradient. Dense LSTM are therefore fast converged and really robust. We also add a CNN1D and max pooling layers

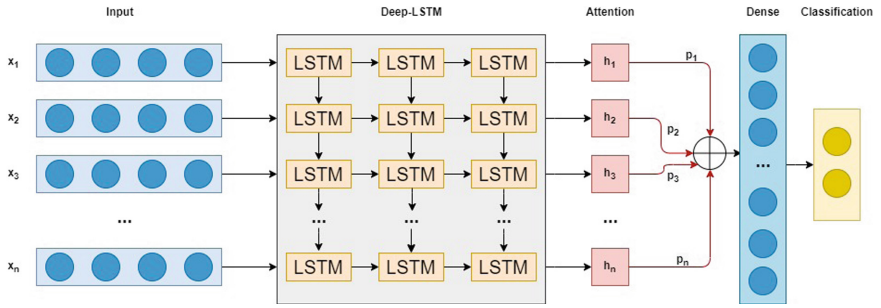

Fig. 5. Deep LSTM with attention mechanism model

to Dense LSTM so that the output can aggregate features from all LSTM layers. Usually, CNN (Convolutional Neural Networks) are used for feature extraction of images. However, one-dimensional CNN (CNN1D) [6] are really efficient for sequential input such as time series, text and speech.

The model has inputs similar to the Dense LSTM model in Sect. 5.3. Both hidden state h and cell state c have the size of 256 in each LSTM cell. Concatenating the hidden states h from 8 LSTM layers results in a tensor of size $(B, 2048, N)$. The convolution filter in the CNN1D layer has the filter size of 3, $padding = 1$ and $stride = 1$. The CNN1D layer takes this tensor in as input and produces an output tensor of size $(B, 128, N)$. The max pooling layer then reduces the feature map size to $(B, 128, 1)$, and fully connects with a 2-neuron classification layer.

6 Experiments

6.1 Dataset

The traffic dataset is captured from three intersections, consists of 105 videos in the training set and 14 videos in the test set. Each video was taken from one of three surveillance cameras in Vietnam, has a resolution of 2560×1980 px, with a length of approximately 15 s, and a sampling rate of 10 frames per second. Each video also has a list of vehicle ids and their bounding boxes location across the frames resulted from the above vehicle tracking algorithm.

After filtering to get vehicles that go forward in the videos, the valid samples are labeled as violating (True) or not violating (False) the traffic light. The dataset statistics is shown in Table 1. The bounding boxes are normalized, and the bounding box list for each sample is zero-padded to have the same sequence length $N = 147$ before being taken as input for the models.

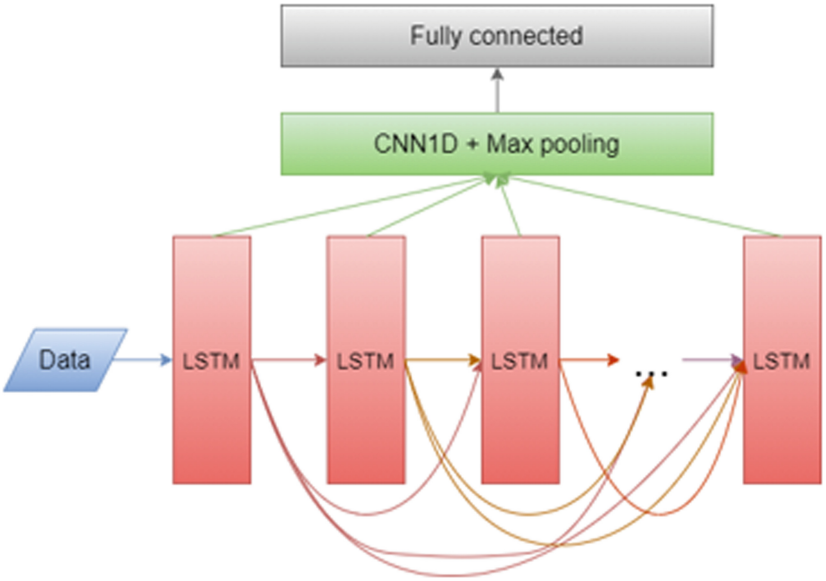

Fig. 6. Dense LSTM with CNN1D model

Table 1. Traffic dataset statistics

	Vehicle counts	Violate	Not violate
Train	1315	628	687
Test	254	44	210

6.2 Evaluation Metrics

The main evaluation metrics for detecting traffic violations is F1-score. F1-score is the harmonic mean of Precision and Recall metrics, where Precision summarizes the exactness of the model and Recall represents the completeness of the predictions.

$$Precision = \frac{TP}{TP + FP}$$
$$Recall = \frac{TP}{TP + FN}$$
$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

where TP = True Positive, FP = False Positive, FN = False Negative.

6.3 Results

After training the Dense LSTM with cross validation of 100 epochs on GPU K8 (8G), we achieved the curves as shown in Fig. 7. The curves show that the model fit very well to the training and validation dataset. We also found similar curves with Deep LSTM. The accuracy of the models is demonstrated in the Table 2. The test precision, recall and f1-score indicate that Dense LSTM with CNN1D exhibits better accuracy than pure LSTM pure LSTM with CNN1D and LSTM with skip connections (Dense LSTM). These are all 98.63%. However, Deep LSTM with Attention Layer expresses the best among all with the precision, recall and f1-score are of 98.85%, 100% and 99.33%, respectively. This is because the model is optimized both in depth and width. We notice that the rule-based method can not detect red-light running violations in which the vehicles do not reach the states S or R. This accounts about 3% in our dataset.

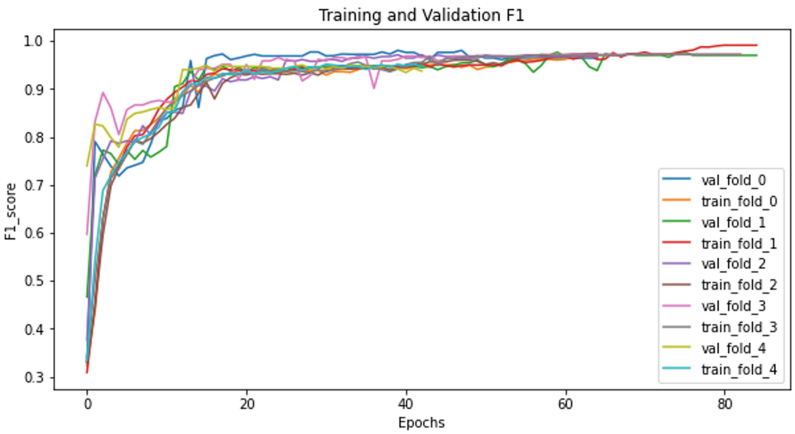

Fig. 7. Training and validation curves of the Dense LSTM with CNN1D

Table 2. Violation detection models results

Model	Test precision	Test recall	Test F1
LSTM	0.8699	0.9439	0.9004
Dense LSTM (LSTM + Skip connections)	0.9420	0.9654	0.9531
LSTM + CNN1D	0.9455	0.9767	0.9602
Dense LSTM + CNN1D	0.9863	0.9863	0.9863
Deep LSTM + Attention	**0.9885**	**1.0000**	**0.9933**

6.4 Performance

In Figs. 8 and 9, we demonstrate the time improved by Mahalanobis distance approximation and cosine distance calculation by group in our tracking algorithm

compared to the original one on a traffic video of ten minute long captured at an intersection of roads. The x-axis denotes the number of calculations and the y-axis is the total processing time for matching (in seconds) of WAYS and the original DeepSORT. We notice that, in this case, DeepSORT and WAYS are all configured with *nn_budget* of 100. From the figures, we find that the time of calculations of Mahalanobis distance and of cosine distance can be saved by nearly 60% and 30%, respectively. Moreover, the time for cosine distance calculation always accounts for nearly 99% of the total matching time. Thus, this shows valuable improvements of the proposed matching algorithm.

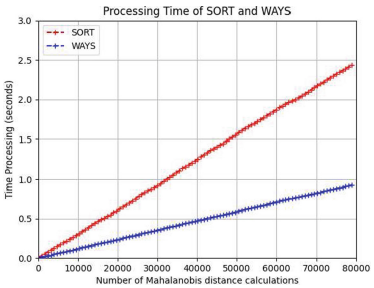

Fig. 8. Matching time due to Mahalanobis distance calculations.

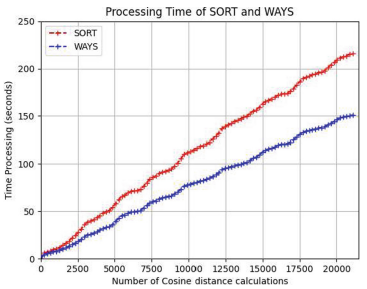

Fig. 9. Matching time due to cosine distance calculations.

7 Conclusion and Perspectives

In this paper, we address the recognition of red light running violations of vehicles in traffic videos using deep learning networks. We improved the traditional DeepSort for 30% faster tracking. Our tracking methods with group-based matching resulted in accurate sequences of states for corresponding vehicles in video. Our classifier of sequences combine LSTM and attention achieved 99.33% of f1-score which higher than the rule-based method thanks to its deep and wide optimization. In the future, we continue apply our classifier for more classes such as reverse direction, stopping and wrong lane violations of the vehicles.

References

1. Bewoor, L.A., Tonge, A., Khiste, R., Chandak, S., Khan, U.: Traffic rules violation detection. Int. J. Adv. Sci. Technol. **29**(4s), 1153–1157 (2020). http://sersc.org/journals/index.php/IJAST/article/view/6667
2. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: YOLOv4: optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934 (2020)
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR abs/1512.03385 (2015). http://arxiv.org/abs/1512.03385
4. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735

5. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017
6. Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., Inman, D.J.: 1D convolutional neural networks and applications: a survey (2019)
7. Liu, W., et al.: SSD: single shot multibox detector. CoRR abs/1512.02325 (2015). http://arxiv.org/abs/1512.02325
8. Mehta, D., et al.: XNect: real-time multi-person 3D human pose estimation with a single RGB camera. CoRR abs/1907.00837 (2019). http://arxiv.org/abs/1907.00837
9. Pinho, R., Tavares, J., Correia, M.: Efficient approximation of the Mahalanobis distance for tracking with the Kalman filter. Int. J. Simul. Model. **6**, 84–92 (2007). https://doi.org/10.2507/IJSIMM06(2)S.03
10. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. CoRR abs/1612.08242 (2016). http://arxiv.org/abs/1612.08242
11. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Proceedings of the 28th International Conference on Neural Information Processing Systems, NIPS 2015, vol. 1, pp. 91–99. MIT Press, Cambridge (2015)
12. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2015)
13. Vaswani, A., et al.: Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS 2017, pp. 6000–6010. Curran Associates Inc., Red Hook (2017)
14. Wang, X., Meng, L.M., Zhang, B., Lu, J., Du, K.L.: A video-based traffic violation detection system, pp. 1191–1194 (2013). https://doi.org/10.1109/MEC.2013.6885246
15. Wang, Z., Zheng, L., Liu, Y., Li, Y., Wang, S.: Towards real-time multi-object tracking. arXiv preprint arXiv:1909.12605 (2019)
16. Wojke, N., Bewley, A., Paulus, D.: Simple online and realtime tracking with a deep association metric. In: 2017 IEEE International Conference on Image Processing (ICIP), pp. 3645–3649. IEEE (2017)
17. Xu, Y., Osep, A., Ban, Y., Horaud, R., Leal-Taixe, L., Alameda-Pineda, X.: How to train your deep multi-object tracker. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020