EURASIP Journal on Advances in Signal Processing

RESEARCH

Open Access

Improved vehicle detection systems with double-layer LSTM modules

Wei-Jong Yang[1], Wan-Ju Liow[1], Shao-Fu Chen[1], Jar-Ferr Yang[1*], Pau-Choo Chung[1] and Songan Mao[2]

*Correspondence:
jefyang@mail.ncku.edu.tw
[1] Department of Electrical Engineering, Institute of Computer and Communication Engineering, National Cheng Kung University, Tainan, Taiwan
Full list of author information is available at the end of the article

Abstract

The vision-based smart driving technologies for road safety are the popular research topics in computer vision. The precise moving object detection with continuously tracking capability is one of the most important vision-based technologies nowadays. In this paper, we propose an improved object detection system, which combines a typical object detector and long short-term memory (LSTM) modules, to further improve the detection performance for smart driving. First, starting from a selected object detector, we combine all vehicle classes and bypassing low-level features to improve its detection performance. After the spatial association of the detected objects, the outputs of the improved object detector are then fed into the proposed double-layer LSTM (dLSTM) modules to successfully improve the detection performance of the vehicles in various conditions, including the newly-appeared, the detected and the gradually-disappearing vehicles. With stage-by-stage evaluations, the experimental results show that the proposed vehicle detection system with dLSTM modules can precisely detect the vehicles without increasing computations.

Keywords: Vehicle detection, LSTM-based object refiner, Spatial priority order, Adaptive miss-time threshold, Adaptive confidence threshold

1 Introduction

Accurate object detection is a challenging problem in computer vision for many years. The object detection is treated as an essential kernel for many important vision-based technologies, such as human skeleton estimation [1, 2], face recognition [3, 4] and object segmentation [5–7]. The traditional feature extractions, such as histogram of oriented gradient (HOG) [8], shift invariant feature transform (SIFT) [9], or deformable parts model (DPM) [10, 11] can help to achieve an acceptable detection performance by using either AdaBoost [12] or support vector machine (SVM) [13] classifier. Deformable models with multi-scale deformable features can detect a large number of object classes to win the triumph for years on PASCAL visual object classes (VOC) challenge [15]. After 2012, the CNN-based algorithms in various structures [15–19] won the contests in ImageNet large scale visual recognition challenge (ILSVRT) [20]. Thus, the convolutional neural network (CNN) systems quickly dominated the object detection area. The CNN-based object feature extractors are also adopted for many other computer vision tasks successfully.

For real-time object detection, we not only need to perform object classifications and draw their bounding boxes. In earlier approaches, the detection algorithms use multiple-size sliding windows, which slide across the whole image to classify the presence of the object within the regions of interest with considerably large computation cost. To achieve real-time detection, object detection has achieved significant advances with the CNN frameworks. We can simply divide the state-of-the-art CNN-based object detectors into two categories: two-stage approaches [21–23] and one-stage approaches [24–28]. The two-stage approaches include region convolutional neural network (R-CNN) [21], fast R-CNN [22] and faster R-CNN [23] models to classify the objects and find their bounding boxes. The R-CNN approaches use a selective search algorithm to generate regions of interest to reduce the computation from the sliding window approaches. The candidate region proposals are warped into a square and go through the CNN model to extract feature vectors for the classifiers to detect the objects and show their bounding boxes. The fast R-CNN receives each region of interest (ROI) and uses ROI pooling to resize all regions to a fixed size for selective search. Thus, the fast R-CNN, which need not perform convolutions for all region proposals, only performs once to reduce the computation effectively. The faster R-CNN replaces the selective search by the region proposal network (RPN) [23], which will produce the regions of interest from the feature retrieved by the CNN backbone network. With the different sizes of the anchor boxes, the whole network detection can detect the objects much more accurately. The single shot detection (SSD) method [24] first trained the model with end-to-end fashions that can perform independently on target detection from multiple feature maps to detect all large and small objects. Recently, the famous one-stage approaches including you only look once (YOLO) [25].

Generally, for most vision-based networks, we could further improve their performances by including temporal data information. However, the batch data array including T frames will heavily increase the computation burden of the networks. Instead of batch data arrays, the enhanced detection and tracking networks try to retrieve from temporal feature maps to raise the importance of 2D features of the networks [26–28]. By including temporal information, it is noted that the object detection networks should also consider the practical issue of implementation for real-time applications. In this paper, we suggest that the object detection system can combine the long short-term memory (LSTM) modules [29] to improve its detection performance. In Sect. 2, we first review the basics of the LSTM modules. Then, we suggest the combinations of the YOLO network and LSTM modules to become an effective object detection and tracking system in Sects. 3 and 4. In Sect. 5, we then exhibit the performances of the combined YOLO and LSTM modules by simulations. Finally, we conclude the main contributions and future work of this paper in Sect. 6.

2 Related work

The YOLO approaches have higher speed performance than the SSD method. Recently the improved accuracy and speed versions of YOLO methods, namely YOLOv2, YOLOv3 and YOLOv4 have been proposed in [30–32]. The LSTM technique, which is evolved from the recurrent neural network (RNN) for speech and language modeling [33, 34], can solve the vanishing gradient problem and model the self-learned context

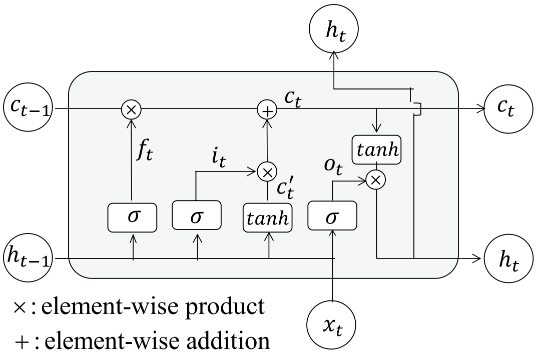


Fig. 1 Long short term memory (LSTM) module

information. By using the temporal information, the LSTM module has been effectively used for precision 3D pose estimation [35] and single object tracking [36, 37].

The long short-term memory (LSTM) extends the RNN by adding forget, input and output gates. To catch the details of the LSTM module, as shown in Fig. 1, the cell state vector c_t and output hidden state vector h_t can be respectively computed by:

$$c_t = f_t * c_{t-1} + i_t * c_t', \tag{1}$$

and

$$h_t = \tanh(c_t) * o_t, \tag{2}$$

where f_t, i_t and o_t are the activations of the tth forget, the tth input and the tth output gates, which are respectively given as

$$f_t = \sigma(w^f[h_{t-1}, x_t] + b_f), \tag{3}$$

$$i_t = \sigma(w^i[h_{t-1}, x_t] + b_i), \tag{4}$$

$$o_t = \sigma(w^o[h_{t-1}, x_t] + b_o), \tag{5}$$

and the update cell state vector is obtained by

$$c_t' = \tanh(w^g[h_{t-1}, x_t] + b_g). \tag{6}$$

In (3)–(6), the weights, w^ρ and the offsets, b_ρ for $\rho = f$, i, o and g need to be trained for the best connections of the previous hidden vector, h_{t-1} and current input vector,x_t. As stated in (3)–(4), the gating activations are squashed into a range between 0 and 1 by a sigmoid function. The gating activation describes the ratio of the multiplied vector components being passed through the gate. If it is "0", the gate will control "nothing passes", while "1" means "all pass". As exhibited in (4), the updated cell state vector, c_t' is regulated by tanh function. As shown in (5), the current cell state vector, c_t is a combination of the previous cell state vector controlled by the forget gate and the updated cell state vector controlled by the input gate. Finally, as stated in (2) and (6), the vector component is regulated by tanh function in a range between − 1 and + 1.

To include temporal information, the LSTM modules can be added in any layer of detection networks. In order to track the multiple vehicles better, we adopt a modified version of long short-term memory (LSTM) module with a dual-layer and multi-stage structure. Instead of inner layers, which have larger feature maps, we suggested that the LSTM modules to directly track the object detection outputs to simplify the computation. It is noted that the proposed LSTM modules can be applied to any of the latest detection methods. Without loss of generality, we choose the YOLOv2 as the initial object detector, which will be improved step-by-step to accomplish the performance improvements in this paper.

3 The proposed methods

Figure 2 exhibits the conceptual diagram of the proposed object detection and tracking system, which includes two subsystems, improved YOLO (iYOLO) object detector and double-layer LSTM (dLSTM) object refiner. After the iYOLO detector, the dLSTM refiner takes T consecutive outputs of the iYOLO to refine the final prediction. Before the dLSTM refiner, however, we need to spatially order the iYOLO outputs, which are the bounding boxes and confidences of the detected objects, to correctly characterize their spatial associations. After the spatial association, the dLSTM object refiner then performs the final refinement of the iYOLO outputs. As shown in the top part of Fig. 2, the detailed descriptions of the iYOLO object detector, the multi-object spatial association, and the dLSTM refiner are addressed in the following subsections.

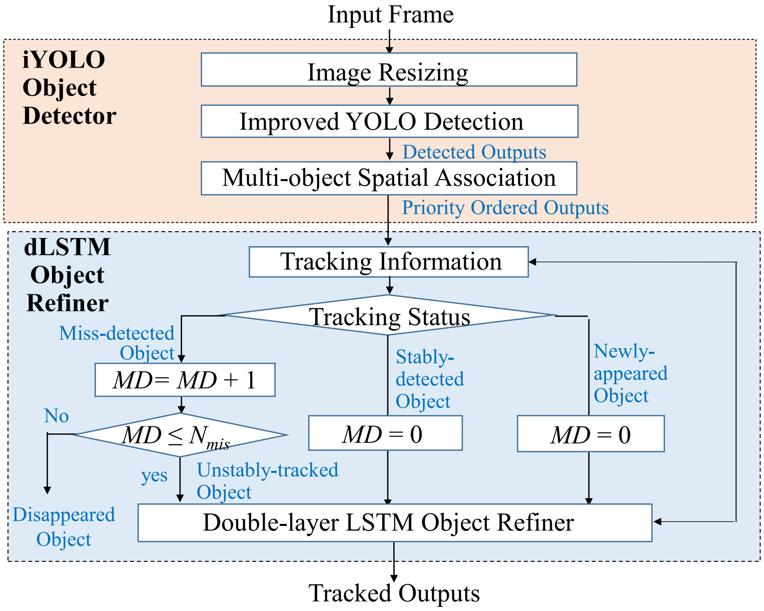


Fig. 2 Flow chart of the proposed object detection system

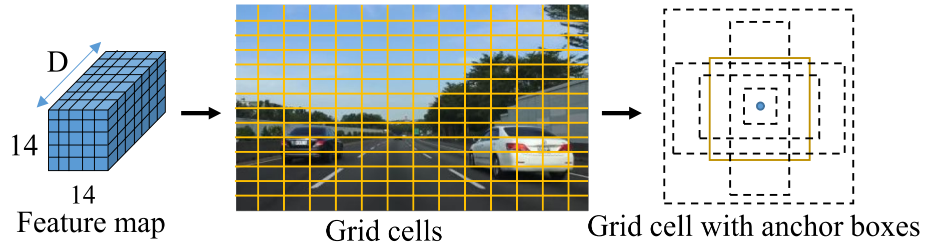


Fig. 3 Anchor boxes of iYOLO object detector

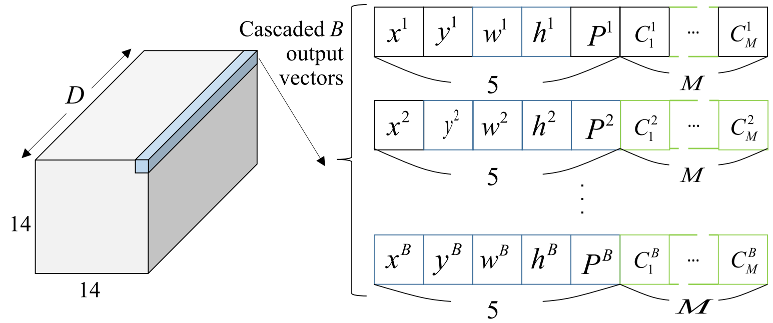

Cascaded B output vectors
D
14
14
x^1 y^1 w^1 h^1 P^1 C_1^1 ··· C_M^1
5
M
x^2 y^2 w^2 h^2 P^2 C_1^2 ··· C_M^2
5
M
x^B y^B w^B h^B P^B C_1^B ··· C_M^B
5
M

Fig. 4 Outputs of the proposed iYOLO object detector

3.1 The iYOLO object detector

The proposed object detection system, as shown in the top part of Fig. 2, first resizes the images to 416×416 or 448×448 size as the inputs of the improved YOLO (iYOLO) object detection network. For performance improvement and computation reduction, the proposed iYOLO object detector is designed to classify 30 on-road moving objects with one combined-vehicle class, including car, bus, and truck classes together. The iYOLO also combines low and high level features to detect the objects. The details of data representation, network structure, and loss functions of the iYOLO are stated as follows.

For moving object detection, we not only need to predict the locations and box sizes of the detected objects but also need to detect their classes. Therefore, in the iYOLO, the output data is a three-dimension array, which is with the size of $14 \times 14 \times D$, where D denotes the channel number of the information for representation of detections and classifications. The 14×14 array is considered as the grid cells of the image. Thus, there are 196 grid cells in total. Each grid cell, as shown in Fig. 3, contains five bounding boxes, which are called as "anchor boxes".

Each grid cell contains D elements, which carry the positions, confidences and class information, where D usually is given by:

$$D = B \times (5 + M), \tag{7}$$

where M is the number of classes and B denotes the number of the anchor boxes in a single grid cell for detecting. As shown in Fig. 4, each grid cell contains B bounding boxes while each bounding box comprises $(5 + M)$ parameters. For the bth box, we have

5 parameters including its bounding box, $\{x^b, y^b, w^b, h^b\}$ and the occurrence probability, $P^b = P(o^b)$. The bounding box is defined by the center with coordinates x^b and y^b and the width and the height the box with w^b and h^b, respectively. In Fig. 4, C_i^b denotes the conditional probability of the bth box that contains ith class as:

$$C_i^b = P(\,i\text{th}\ \ \text{class}|o^b), \tag{8}$$

for $1 \leq i \leq M$. Therefore, we can find the probability of the ith object in the bth box is given by:

$$P(i\text{th class, } o^b) = C_i^b P^b, \tag{9}$$

where P^b denotes the occurrence probability of the bth box. If $C_i^b P^b$ passes a pre-defined threshold, we will consider the ith class object being existed in the bth bounding box and detected by the proposed iYOLO detector.

3.2 Network structure of iYOLO detector

The proposed iYOLO network structure as shown in Fig. 5 is composed of several stages of convolutional layers and max pooling layers. The convolutional layers [38] with batch normalization [39] are mostly with 3×3 or 1×1 convolutions. The pooling layers perform with stride 2 of direct down sampling. In this paper, we include car, truck and bus classes into vehicle class. Thus, the number of output classes of the iYOLO are reduced to 30. As shown in Fig. 5, we eliminate three sets of two repeated $3 \times 3 \times 1024$ convolution layers compared to the YOLOv2. The reason for decreasing high-level layers is that we can reduce the computations since we use the vehicle class to represent all type of cars. The more high-level layers we reduce; the less complex the model becomes.

In order to enhance the performance, the proposed iYOLO further includes two low-level features to help the final detection. As marked by the thick red lines and green-box functions in Fig. 5, we concatenate the outputs of 12th (after the green-boxed max-pooling) and 17th features (after the green-boxed $3 \times 3 \times 512$ convolution) layers with the final feature. To keep the size of low-level features the same as that of the high layer feature, we introduce two convolution layers with $3 \times 3 \times 128$, $1 \times 1 \times 64$ for first low-level

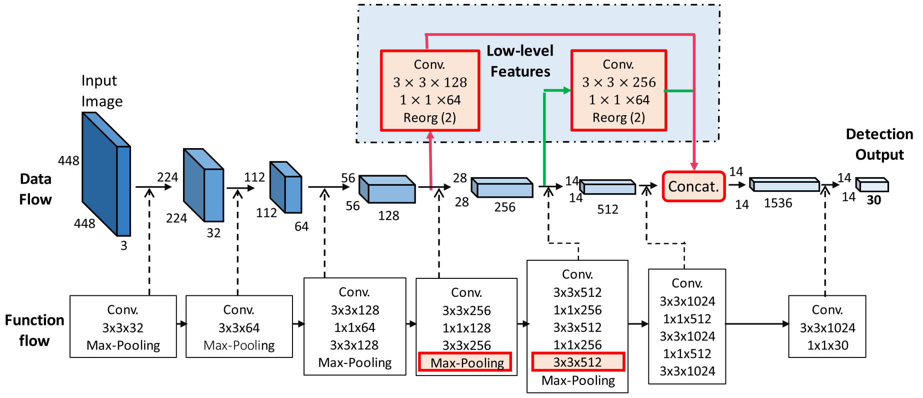


Fig. 5 The network structure of the proposed iYOLO detection

feature and $3 \times 3 \times 256$, $1 \times 1 \times 64$ convolution and reorganize their features into the half of the original resolution in marked functions before the concatenation.

Since the proposed iYOLO will output the information of bounding box location, classification and confidence results simultaneously. Therefore, the prediction of the module is composed of three loss functions: (1) location loss of the bounding box, $g=\{x, y, w, h\}$, where (x, y) denotes the center position while w and h respectively represent width and height of the bounding box; (2) classification loss defined by the conditional probability for specific class,$p_s(i)$; and (3) confidence loss related to probability $P_s(o)$ that states an object existing in the sth grid cell. The total loss function is given by

$$L(f, c, g, \overline{g}) = L_{\text{loc}}(f, g, \overline{g}) + L_{\text{con}}(f, g) + L_{\text{cls}}(f, g, c), \tag{10}$$

where f is the input image data, c is the class confidence, g and $\overline{g}$ denote the predicted and the ground truth boxes, respectively. As stated in (10), the total loss is composed of the location loss, confidence loss, and class loss functions balanced by weighting factors λ_{loc}, λ_{obj}, λ_{noobj}, and λ_{cls} separately. These four loss functions are described as follows.

The location loss, L_{loc} is given as:

$$L_{\text{loc}} = \lambda_{\text{loc}} \sum_{s=1}^{S^2} \sum_{b=1}^{B} \alpha_{s,b}^{o} [\left\| g_s - \overline{g}_s \right\|^2], \tag{11}$$

where $g_s= \{x_s, y_s, w_s, h_s\}$ and $\overline{g}_s= \{\overline{x}_s, \overline{y}_s, \overline{w}_s, \overline{h}_s\}$ are the predicted and ground truth bounding boxes, S and B denote the numbers of grid cells and anchor boxes, respectively. In (11), λ_{loc} represents the location weighting factor and $\alpha_{s,b}^{o}$ means the responsibility for the detection of the sth grid with the bth box. If the bounding box passes the intersection over union (IoU) threshold 0.6, then the box specific index, $\alpha_{s,b}^{o}$ will be 1, otherwise $\alpha_{s,b}^{o}$ goes 0.

The confidence loss, L_{con} is expressed as:

$$L_{\text{con}} = \lambda_{\text{obj}} \sum_{s=0}^{S^2} \sum_{b=0}^{B} \alpha_{s,b}^{o} [(P_s(o) - P_s(\overline{o}))]^2 + \lambda_{\text{noobj}} \sum_{s=0}^{S^2} \sum_{b=0}^{B} (1 - \alpha_{s,b}^{o}) [(P_s(o) - P_s(\overline{o}))]^2, \tag{12}$$

where the first term exhibits the bounding box confidence loss of the objects while the second term denotes the confidence loss without the objects. In (12), λ_{o} and λ_{no} express the confidence weighting factors with object and no-object cases, respectively. The loss values are only valid for responsible bounding boxes, $\alpha_{s,b}^{o} = 1$, since that the non-responsible bounding boxes don't have truth label.

The classification loss, L_{cls} is given as:

$$L_{\text{cls}} = \lambda_{\text{cls}} \sum_{s=1}^{S^2} \sum_{b=1}^{B} \alpha_{s,b}^{o} \sum_{i \in \text{classes}} [(p_s(i) - \overline{p}_s(i))]^2, \tag{13}$$

where $p_s(i)$ denotes the i-class confidence in specific grid cell of the anchor box and λ_{cls} denotes the classification weighting factor. If the bounding box does not contain any object, the predicted probability should be decreased to close to 0. On the contrary, if the box contains an object, the predicted probability should be push to near 1.

Before discussing the proposed dLSTM object refiner, we should properly associate the outputs of each detected object of the iYOLO according to its spatial position. The spatial association of the temporal information of multiple objects is designed to collect all the outputs of the same physical object in a spatial-priority order to become the time series inputs of the dLSTM object tracking modules.

3.3 Multi-object association

To make a good association of a series of outputs for each detected object, we need to design a proper association rule to construct a detected data array as the input of the dLSTM object refiner. Usually, the iYOLO shows the detection results according to the confidence priority, which is not a robust index for object association since the confidences vary from time to time. Therefore, we utilize the close-to-the-car distance as the object association index since any on-road moving objects will physically travel arround their nearby areas smoothly. For the i^{th} detected object with its bounding box, $g_i = \{x_i, y_i, w_i, h_i\}$, the association should be based on the spatial positions in the image frame. If we set the frame left-upper corner of as the origin at (0, 0), the right-lower corner at $(W-1, H-1)$, where W and H are the width and height of the frame, respectively. The position at $(W/2, H)$, which is used to measure the closeness of the detected vehicle to the driving car, is set as the reference point. If the detected object is closer to the reference point, it will be more dangeous to the car. Thus, the bounding box of a detected object is closer to the reference point, it should be more important for the object and we should give it a higher priority. The priority of the detected object is spatially ordered by a priority-regulated distance to the critical point as

$$d_i = \left((\Delta x_i^d)^2 + \rho(\Delta y_i^d)^2\right)^{1/2}, \tag{14}$$

where the horizontal distance between the ith bounding box and the reference point is given as:

$$\Delta x_i^d = \min_{x \in R}(x - W/2), \tag{15}$$

with $R = \{x | (x_i - w_i/2) \leq x \leq (x_i + w_i/2)\}$ and the vertical distance is expressed by

$$\Delta y_i^d = H - y_i - (h_i/2). \tag{16}$$

The horizontal distance, Δx_i^d defined in (14) finds the minimum displacement of any point in the bounding box to the reference point horizontally. If Δx_i^d is smaller, the bounding box will be closer to the reference point. If Δy_i^d is smaller, the bottom of the bounding box of the object is close to the bottom of the frame.

After the computation of all priority-regulared distances of the detected objects, the object indices are determined by the priority-regulated distances in decenting order. The smaller priority-regulated distances will be given a higer order, i.e., a small prioity index to the detected object. If $\rho = 0.5$, Fig. 6 shows the order of the object confidences and the priority orders of the prority-regulated distances between the reference point and the

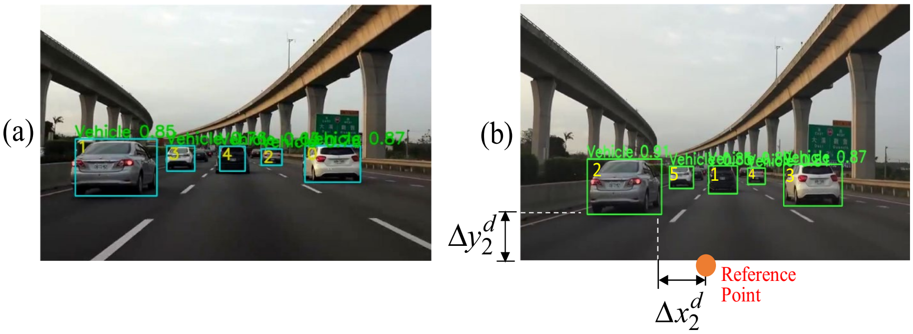


Fig. 6 Priority orders of detected objects: **a** ordered in the confidences; **b** ordered in the regulated distance to the reference point with $\rho = 0.5$

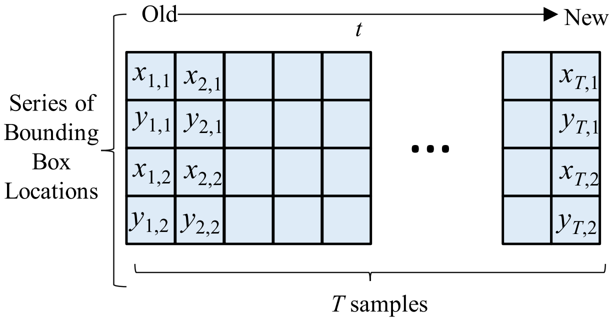


Fig. 7 Location 2D time series data array

detected objects. The order of the objects with the regulated distances is spatially stable since the spatial positions of the real objects will not change too quickly. Even if the object is moving horizontally to occlude some objects, the tracked objects will be still reasonable and stable since we don't care about the ones which are occluded with one combined-vehicle class. We can then focus on the objects, which are geometrically close to the driving car and give them higher priorities for tracking.

3.4 LSTM refiner

After determining the priority order of the detected objects, we collect all bounding boxes as a 2D data array with the same priority order. For example, the data array for the first priority object with $g_t^{(1)} = \{x_t^{(1)}, y_t^{(1)}, w_t^{(1)}, h_t^{(1)}\}$, for $t = 1, 2, \ldots, T$. For simplicity, we ignore the index of the priority order. For each detected object at instant T, we then collect an array of bounding boxes as

$$L = \{X_1, X_2, X_3 \ldots, X_T\}, \tag{17}$$

where $X_t = [x_{t,1}, y_t, x_{t,2}, y_{t,2}]^{\mathrm{T}}$ denotes positions of left-top corner $(x_{t,1}, y_{t,1})$ and bottom-right corner $(x_{t,2}, y_{t,2})$ of the bounding box at the tth instant. After collection of X_t for T consecutive samples, Fig. 7 shows the 2D time-series data array of bounding boxes for each detected object.

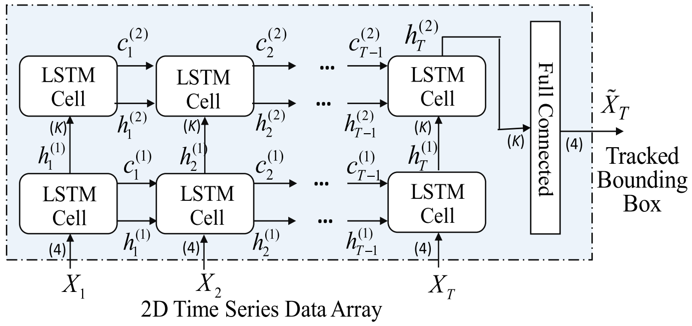


Fig. 8 The detailed structure of dLSTM object refiner

With the 2D data array for each object, the double-layer LSTM (dLSTM) is designed to reduce unstable bounding boxes. In order to achieve better performance, we might use a longer LSTM module, however, it would increase some unnecessary delay of tracking. As shown in Fig. 8, the double-layer LSTM (dLSTM) refiner contains K-element hidden state vectors with T time instants. The fully connected layer take $h_T^{(2)}$, the Tth hidden state vector of the second LSTM layer and output 4-point prediction position of bounding box, $\tilde{X}$. As stated in (17), the dLSTM network inputs a series of bounding box data X_t for for $t=1, 2, \ldots, T$. In Fig. 8, $c_t^{(l)}$ and $h_t^{(l)}$ denote the cell and the hidden states of the lth layer at the tth time step of the dLSTM model, respectively. To make the model deeper for more accurate earnings [30, 31], we stack two LSTM layers to achieve better time series prediction and avoid long delay simultaneously. As stated in (6), the first LSTM layer inputs the location data array L in chronological order. It generates the K-dimension hidden state $h_t^{(1)}$ and the K-dimension cell state $c_t^{(1)}$. Then, we will output hidden state features of the first LSTM layer as the inputs of the second LSTM layer. In the dLSTM refiner, the hidden state $h_t^{(l)}$ is treated as the decision output and the cell state $c_t^{(l)}$ gets the updates from the output of the previous step $h_{t-1}^{(l)}$. The second LSTM only returns the last step of its output sequences for dropping the temporal dimensions. Finally, the second LSTM layer followed by the fully connected layer interprets the K feature vector to the predicting location $\tilde{X}$ learned by the dLSTM module.

To train dLSTM module, the IoU-location loss, which combines intersection over union (IoU) and position mean square error (MSE) losses as

$$L_{\text{dLSTM}} = -\alpha \cdot \frac{1}{n}\sum_{i=1}^{n} \log(\text{IoU}(X_i, \overline{X}_i)) + \beta \cdot \frac{1}{n}\sum_{i=1}^{n} \left\|X_i - \overline{X}_i\right\|^2, \tag{18}$$

where X_i and $\overline{X}_i$ denote the locations of the predicted and ground truth bounding boxes, respectively. In (18), the IoU, which represents the ratio of intersection and union of the predicted and groundtruth bounding boxes, gives $0 < \text{IoU} < 1$. Thus, we use $-\log(IoU)$ as the first loss function. In addition, the mean square error (MSE) of X_i and $\overline{X}_i$ is used for the coordination loss function. To balance IoU and MSE loss functions, we need to select α and β for a better combination. After the training process, it is noted that the dLSTM refiners with the same weights are used for all detected objects in this paper.

3.5 Object tracking status

For each detected object, as shown in the bottom part of Fig. 2, we need to constantly monitor the object tracking condition, which could be a newly-appeared, tracked, or disappeared object. The tracking status will help to control the dLSTM refiner correctly. Not only the bounding boxes, we also need adopt the occurrence and conditional probabilities, as shown in the bottom part of Fig. 2, for effective object tracking. We assume that we have already initiated P dLSTM refiners to track P priority objects. For each detected object at instant T, from the iYOLO, we have collected the tracking information:

1. Priority Index: p,
2. Input Data Array: $\{X_1, X_2, X_3 \ldots, X_T\}$,
3. Output Predict Data: $\tilde{X}_T$,
4. T sets of top five confidences given by the iYOLO: $\{p(i*), i^*\}$, for $i^*=1, 2, \ldots, 5$,

where i^* carries the index of the top i class and $p(i*)=C_{i*}^{b}p^{b}$, which records the confidences of top five classes. As shown in Fig. 9, there are four possible conditions of confidence plots in practical applications, where Th_p denotes the detection threshold of confidence. With all outputs of the detected object collected from the iYOLO, the status of the dLSTM refiner can be determined as the follows:

At $T+1$ instant for a detected object, whose the confidence estimated by the iYOLO is higher than the threshold, we need to first distinguish that the object is a newly-appearing (red solid line) or stably-tracked object (red-dash line) as shown in Fig. 9. With the same priority index, we first check the IoU of bounding boxes of the object obtained at T and $T+1$. If the IoU of two consecutive bounding boxes is greater than a threshold and the object is with the same class, we then determine this object as the stably-tracked one. In this case, we need to update the collected information by left-shifting data array as,

$$X_t = X_{t+1}, \text{ for } \quad t = 1, 2, \ldots, T, \tag{19}$$

and top five confidences.

If the IoU of the current and previous bounding boxes is lower than the threshold or the class index is different, we then treat it as a newly-appeared object. In this case, we need initialize a new dLSTM refiner and a new set of data array with the same X_{T+1} as

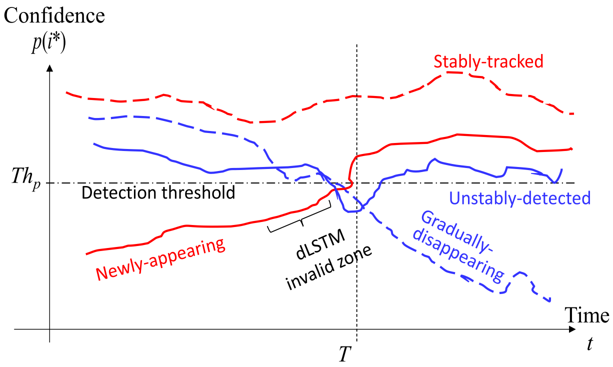


Fig. 9 Confidence plots of gradually-appearing, stably-tracked, unstably-detected and gradually-disappearing objects

$$X_t = X_{T+1}, \quad \text{for} \quad t = 1, 2, \ldots, T, \tag{20}$$

We store the new set of top five confidences with the same $\{p(i*), i^*\}$, for $i=1, 2, \ldots, 5$. The newly-appeared object becomes the tracked object in the next iterations. For both tracked and newly-appeared cases, we entrust the detection ability of the iYOLO. Once the iYOLO decides it as the positive confidence, which is greater than the threshold, it must be an active object. It is noted that the dLSTM refiner will not change the detection performance for the stably-tracked (red-dash line) and newly-appearing (red-solid line) object. For these two cases, the miss-detected (MD) counter will be reset to zero shown in the bottom part of Fig. 2 and the dLSTM refiners are actually used to refine the tracked bounding boxes.

For a dLSTM refiner, we hope not only to improve the accuracy of bounding boxes but also to raise the detection performance for gradually-disappearing (blue-dash line) and unstably-tracked (blue-solid line) conditions as shown in Fig. 9 while the dLSTM refiner obtains the miss-detected information from iYOLO. Based on the fact of no-sudden-disappeared objects, we will not turn off the dLSTM refiner at once if the miss-detection case happens. To improve the detection performance, we further design a miss-detected (MD) counter, which is reset to zero if the iYOLO actively detects the object, which possesses sufficient confidence with a bounding box. Once the detected object does not have large enough confidence at some instants, we will increase MD counter of the tracked object by one until MD is larger than the miss-detection threshold, N_{mis}. When MD $\leq N_{\text{mis}}$, the tracking system will still treat the object as a tracked object but in the "unstably-detected" condition. For any unstably-detected object, the system will give the output of the dLSTM refiner as the compensation, i.e., $\hat{X}_{T+1} = \tilde{X}_T$. Once MD $> N_{\text{mis}}$, the tracking system will delete the object and stop the dLSTM refiner hereafter. In general, we can improve the detection performance for unstably-detected and gradually disappeared conditions as shown in Fig. 9.

If the detected object is with a larger bounding box, it should not disappear in a shorter time. On the contrary, the object is with a smaller bounding box, it could be closer to the disappearing case. As shown in Fig. 2, we suggest the adaptive missed-detection counting (AMC) threshold as:

$$N_{\text{mis}} = \begin{cases} 10, & \text{for} \quad A_T \geq A_{\max}, \\ 5, & \text{for} \quad A_{\max} > A_T \geq A_{\min}, \\ 2, & \text{for} \quad A_{\min} > A_T. \end{cases} \tag{20}$$

where $A_T = w_T h_T$ denotes the area of the latest bounding box of the detected object by the iYOLO before T time instant. To terminate the dLSTM refiner, the AMC threshold will set to $N_{\text{mis}} = 10$, 5, and 2 for large, middle, and small detected bounding boxes, respectively. With the AMC threshold, we could help to raise the detection rate and properly avoid the false positive rate. Since the dLSTM refiner will take 10 sets of bounding boxes from iYOLO, i.e., $T=10$, we choose the AMC threshold, N_{mis} to be 10, 5, and 2 for large, middle, and small detected bounding boxes empirically. With the above adaptive confidence threshold, we could recover the miss-detected object, which could be frequently disturbed by various environmental changes. Since the dLSTM module needs the data vectors of consecutive bounding boxes of the detected object, for the miss-detected object, we will replace the output of the iYOLO with $\hat{X}_{T+1} = \tilde{X}_T$.

4 Results and discussion

With 1280×720 videos, the proposed object detection and tracking system and the other systems are implemented in Python 3.5 and Keras 2.1.5 with Tensorflow 1.4.0 backend as the deep learning function library. We used a personal computer with Intel Core i5-8400 CPU 2.8 GHz, 16 GB 2400 GHz RAM for the hardware system. The iYOLO and dLSTM networks utilize NVIDIA Geforce GTX 1080 8G to accelerate the testing and training speeds.

To evaluate detection performances, we started from COCO pre-trained weights and then combined the KITTI dataset [36] and a self-build dataset collected in the Taiwanese highway to train the vehicle detectors. Since we focus on the vehicle detection and tracking on Taiwanese highway traffic roads, we labeled cars, buses and trucks as a single vehicle class. In the KITTI dataset, we filtered out the images that don't contain any vehicles to obtain 7480 images. In the self-build dataset, the testing videos, we collected 2635 images in day and night with various conditions of tunnel, shading and weather conditions from the Taiwanese highway. The original resolutions of the KITTI and self-build dataset are 1224×370 and 1280×720, respectively.

To balance the loss functions stated in (10), we set $\lambda_{\text{loc}}=5$, $\lambda_{\text{obj}}=1$, $\lambda_{\text{noobj}}=0.5$ and $\lambda_{\text{cls}}=1$ in training process. Table 1 shows the details of all the comparing detection network models with 5 model indices. With model indices #1 and #2, the "SSD COCO" and "YOLOv2 COCO" models are pre-trained with the COCO database, respectively. With model index #3, the "YOLOv2 Re-trained" model denotes the VOLOv2 is re-trained by combining original automobile-related classes to one vehicle class and includes the "self-build dataset" collected in Taiwanese roads in training. With model index #4, the "YOLOv2_Reduced" model denotes that VOLOv2 has been eliminated three sets of two repeated $3 \times 3 \times 1024$ convolution layers. With model index #5, the "Proposed iYOLO" model shown in Fig. 5, the final feature of the "YOLOv2_Reduced" model concatenates with two additional low-level features for object detection.

For the dLSTM refiner depicted in Fig. 8, we capture the trajectory of the moving objects from the video sequences for training. The trajectory is labeled by a sequence of (x_1, y_1, x_2, y_2), where (x_1, y_1) and (x_2, y_2) are top-left and right-bottom corners of the bounding boxes of the detected object, respectively. The moving object is characterized by the vehicle's trajectory. There are 27 video sequences which contain 8100 location sequences.

Table 1 Details of comparing detection network models

Model index	Network model	Self-build dataset	Drop weights	Low-level features
1	SSD-COCO	–	–	–
2	YOLOv2-COCO	–	–	–
3	YOLOv2_Re-trained	✓	–	–
4	YOLOv2_Reduced	✓	✓	–
5	Proposed iYOLO	✓	✓	✓

4.1 Vehicle detector by iYOLO

For practical applications in Taiwan, the experimental results for the vehicle detections in the Taiwanese dataset are demonstrated. Generally, the YOLOv2 achieves an excellent performance in object detection and has good speed in computation. However, it is not robust for detecting vehicles for some cases. In the testing phase, there are 10 videos with 3150 frames which contain 10,707 vehicles. In detection, the true positive rate (TPR) and false positive rate (FPR) are respectively given as:

$$\text{TPR} = \frac{\text{Detected Objects in Total Frames}}{\text{Number of Objects in Total Frames}}, \tag{22}$$

$$\text{FPR} = \frac{\text{Number of False Positives}}{\text{Number of Total Frames}}. \tag{23}$$

The F1-score is defined as

$$\text{F1 - score} = (2 \cdot \text{TPR})/(2 \cdot \text{TPR} + \text{FNR} + \text{FPR}), \tag{24}$$

where FNR denotes false negative rate. Table 2 shows the performances achieved by the proposed iYOLO and the other detection models enlisted in Table 1. In Table 2, the YOLOv2 with model index #2 trained on COCO dataset has poor detection performances in Taiwanese highway dataset since the COCO dataset does not match up with the true scenarios in Taiwan. Since the "YOLOv2_Re-trained" model with model index #3 deals with one combined-vehicle class and is retrained by the self-build dataset, it can improve the detection rate and false positive rate. The "YOLOv2_Reduced" with model index #4 after network reduction achieves better detection rate but performs worse in false positive rate. Finally, we found that the proposed iYOLO with model index #5 concatenated with two low-level features achieves the best performance. If we slightly increase the input size to 448 × 448, we can further improve the detection and false positive rates simultaneously. Two detected examples for YOLOv2 with COCO, YOLOv2_re-trained, YOLOv2_reduced and the proposed iYOLO are visually exhibited in Fig. 10. The results also demonstrated that the proposed iYOLO performs better than the others.

4.2 Vehicle detection with dLSTM refiner

To evaluate the tracking performances achieved by the dLSTM refiner after the proposed iYOLO network, we first setup the experiments and determine the parameters, {α,

Table 2 Performance comparisons of various detection networks

Model index	Input Size	Detection rate (%)	False positive (%)	F1-score (%)
1	300 × 300	61.83	15.49	71.13
2	416 × 416	54.40	2.52	70.94
3	416 × 416	65.28	1.09	79.97
4	416 × 416	73.77	6.95	82.34
5	416 × 416	78.30	5.19	86.09
5	**448 x 448**	**87.08**	**0.75**	**93.62**

Bold indicates better results

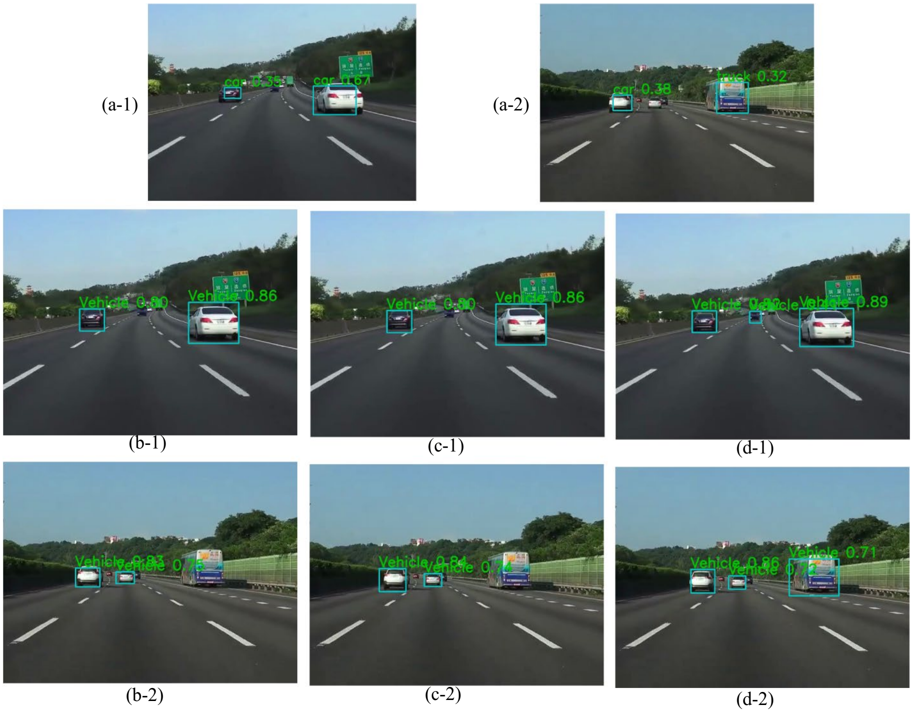


Fig. 10 Detection examples achieved by: **a** YOLOv2 with COCO; **b** YOLOv2_Re-trained; **c** YOLOv2_Reduced and **d** the proposed iYOLO

β}, and {$A_{\max}$, $A_{\min}$} stated in (18) and (21), respectively. The training dataset of dLSTM refiner is collected from self-build traffic trajectory dataset with sequential data arrays of (x_1, y_1, x_2, y_2). With random error deviations of groundtruth, the dLSTM model is trained for 15,000 epoches using ADAM optimizer with a learning rate 0.0001. Unlike traditional classification and detection training, which involves images, the trajectory datasets in sequential data arrays will not take too much time. During the training, the many-to-one LSTM strategy is used where the time steps set to $T = 10$. In other words, for every 10 time steps of training data, there will generate a corresponding ground truth location. In the testing phase, we inherit the iYOLO testing videos which have 10 videos with 3150 frames. There are total 10,828 vehicles in these frames.

Since the loss function plays an important role in training the dLSTM refiner, we should first determine the weighting parameters defined in (18). We should find a better loss function which could effectively use both MSE location loss and IoU loss. In the experiments, we adopted the direct object status determination with fixed with fixed $N_{\text{mis}} = 30$. Table 3 shows the detection performances with different sets of α and β. The results show that $\alpha = 1$ and $\beta = 0.5$ could achieve the best performance.

With $\alpha = 1$ and $\beta = 0.5$, we need to design a suitable adaptive missed-detection counting (AMC) threshold to help the dLSTM refiner better. As stated in (21), we need to determine a better pair of {$A_{\max}$, $A_{\min}$}, which can help to achieve the best performance with a higher detection rate and a lower false positive rate. The performances of AMC strategy with different pairs of {$A_{\max}$, $A_{\min}$} are shown in Table 4. For image size of 448×448, we found that the AMC strategy with $A_{\max} = 5000$ and $A_{\min} = 1000$ can help

Table 3 The comparison of different α and β weighted in training the dLSTM Tracker based on the iYOLO results

Test case	α	β	Detection rate (%)	False positive rate (%)	F1-score (%)
(a)	0	1	72.67	29.23	71.98
(b)	0.25	1	91.62	6.22	92.62
(c)	0.5	1	**91.86**	**5.81**	**92.94**
(d)	0.75	1	91.48	6.36	92.48
(e)	1	0	91.53	6.31	92.53
(f)	1	0.25	91.79	6.05	92.79
(g)	1	0.50	91.71	6.13	92.71
(h)	1	0.75	91.76	6.09	92.76
(i)	1	1	91.83	6.01	92.83

Bold indicates better results

Table 4 The comparison of AMC threshold with different sets of $\{A_{\max}, A_{\min}\}$ for the dLSTM refiner

AMT strategy	$A_{\max}$	$A_{\min}$	Detection rate (%)	False positive rate (%)	F1-score (%)
Direct	–	–	91.86	5.81	92.94
(a)	20,000	2000	91.86	5.81	92.94
(b)	10,000	2000	91.40	3.11	93.47
(c)	5000	1000	**92.19**	**2.57**	**93.63**

Bold indicates better results

Table 5 Experimental results with dLSTM refiners with different threshold settings

Network models	Detection rate (%)	False positive rate (%)	F1-score
iYOLO-only	87.08	**0.75**	93.62
iYOLO + dLSTM with fixed $N_{mis} = 14$	91.86	5.81	92.94
iYOLO + dLSTM with AMC Threshold[a]	**92.19**	2.57	**93.63**
YOLO-v4	**96.22**	7.13	**96.96**

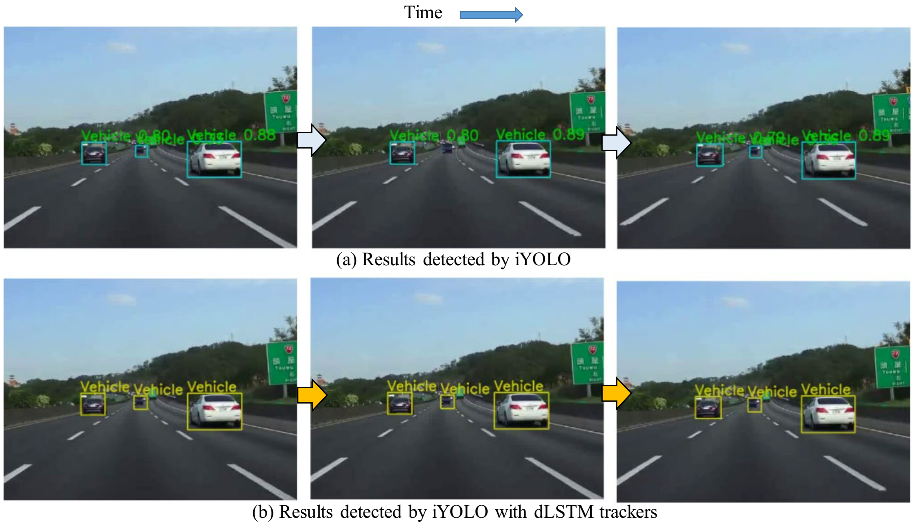
Bold indicates better results

[a] $A_{\max} = 5000$ and $A_{\min} = 1000$

the proposed dLSTM refiner to achieve the best performance. With the AMC strategy, we successfully reduce the false positive rate to avoid the unreasonable time extension for those small missing objects. With $\alpha = 1$ and $\beta = 0.5$ and the AMC threshold with $A_{\max} = 5000$ and $A_{\min} = 1000$, the proposed adaptive confidence threshold can help the proposed dLSTM refiner to achieve the best performance.

4.3 Vehicle detection performances with dLSTM

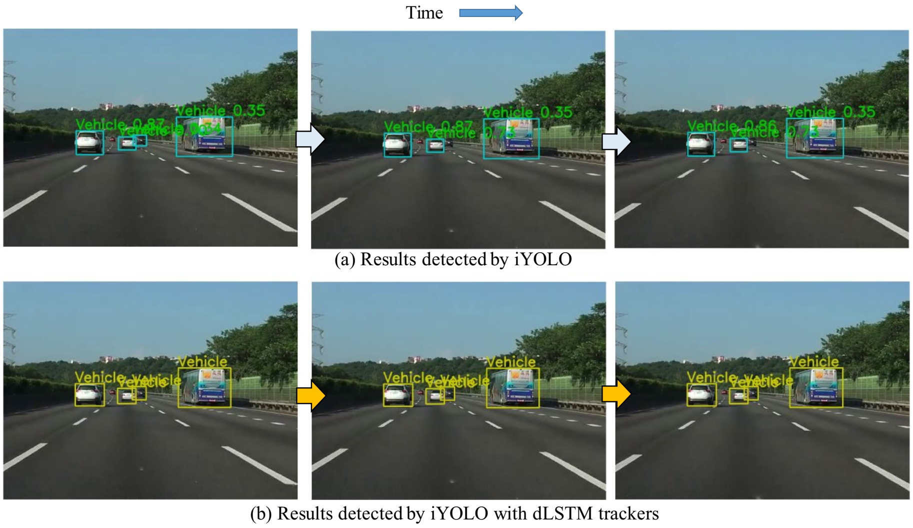
To evaluate the vehicle tracking performance improved by the dLSTM, Table 5 shows the detection rates, false positive rates and F1-scores achieved by the iYOLO and the iYOLO and dLSTM with/without controlling by AMC threshold. The simulation results show that the dLSTM refiners significantly improve the detection rate.



(a) Results detected by iYOLO

(b) Results detected by iYOLO with dLSTM trackers

Fig. 11 Visual results of video set #1 achieved by: **a** iYOLO detector only; **b** the iYOLO with dLSTM refiners



(a) Results detected by iYOLO

(b) Results detected by iYOLO with dLSTM trackers

Fig. 12 Visual results of video set #2 achieved by: **a** iYOLO detector only; **b** the iYOLO with dLSTM refiners

The dLSTM refiner physically tries to infer the existence of the vehicle, which has been tracked. If the tracked vehicle becomes disappeared, the dLSTM refiner will not release it at once. That is why the proposed tracking method could have a larger false positive rate (FPR). With the help of the AMC threshold, the proposed method can reduce the FPR more. It is noted that all the false positive cases are the fast-disappeared vehicles, which are with fast speed. For those cases, the proposed method actually will not cause any safety problem of driving since those vehicles in a distance are fast away from the target driver. It is interesting that YOLO-v4 has highest accurate detection rate, however, it also has highest FPR at the same time. The blinking false cases in YOLO-v4 are suddenly-appeared vehicles mostly. The FPR of YOLO-v4 could be reduced if it conducts the same improving procedures and cooperates with the dLSTM refiners.

Figures 11 and 12 show two visual simulation results for video set#1 and set#2, respectively. The iYOLO cannot detected small vehicles occasionally, however, the iYOLO with dLSTM can successfully tracked the results. In summary, the proposed dLSTM refiner can help to compensate the frames that the vehicles are not detected properly. The dLSTM refiner solves the blinking issues caused by the miss-detecting bounding boxes and successfully achieves the multiple objects tracking with the support of the decision system stated in Fig. 2. With better performances, the proposed vehicle detection and tracking system can achieve 21 frames per second, which reaches the nearly real-time requirements.

5 Conclusion

In this paper, we proposed a robust object detection system by introduction of the double-layer long short-term memory (dLSTM) refiner. We started from the YOLO-v2, the improved YOLO (iYOLO) vehicle detector is achieved by parameter reduction, combined-vehicle class and concatenating two low-level features. With the training data in the combination of the KITTI and self-build Taiwanese highway traffic datasets, we can raise the vehicle detection rate much higher. Finally, we further proposed the dLSTM refiner, which needs the multi-object association rule and the adaptive missed-detection counting (AMC) threshold method to improve its performances. The multi-object association rule can help to successfully collect the time series detection results of the objects while the AMC threshold can help to reduce the false positive rate for tracking vehicles. Simulation results show that the proposed system can successfully track the temporal locations to compensate miss-detected bounding boxes for unstably-detected and gradually-disappearing objects. Since the dLSTM refiner acts as a temporally predictor in uses of past detected bounding boxes, we can obtain more precise detection results than those achieved by the original object detector. However, we cannot gain any benefits for gradually-appearing objects, which have lack of the prior information. The improvements of the object detector and the designs of the dLSTM refiner can be applied to any latest object detectors.

Abbreviations

AMC: Adaptive missed-detection count; CNN: Convolutional neural network; dLSTM: Double-layer long short term memory; DPM: Deformable parts model; FNR: False negative rate; FPR: False positive rate; HOG: Histogram of oriented gradient; ILSVRC: ImageNet large scale visual recognition challenge; IoU: Intersection over union; iYOLO: Improved you only look once; LSTM: Long short term memory; MD: Miss-detected; MSE: Mean square error; R-CNN: Region convolutional neural network; RNN: Recurrent neural network; ROI: Region of interest; RPN: Region proposal network; SIFT: Shift invariant feature transform; SSD: Single shot detection; SVM: Support vector machine; TPR: True positive rate; VOC: Visual object classes; YOLO: You only look once.

Authors' contributions

W.-J.Y. carried out fundamental studies, participated in the design of adaptive schemes of the proposed system, assembled formulations and drafted the manuscript. W.-J.L. and S.-F.C. carried out software simulations and data analyses. P.-C.C. and J.-F.Y. conceived of the study, and participated in its design and coordination and helped to draft the manuscript. S.M. controlled progress and direction of the research and helped to correct final writings. All authors read and approved the final manuscript.

Authors' information

Wei-Jong Yang received the B.S. degree in computer science from Tunghai University, Taiwan, in 2012 and the M.S. degree in computer science and information engineering from National University of Tainan, Taiwan in 2015. He received the Ph.D. degree from the Graduate Institute of Computer and Communication Engineering in National Cheng Kung

University, Taiwan in 2020. Currently, he is a postdoctoral researcher in the Department of Electrical Engineering, National Cheng Kung University. His current research interests include pattern recognition, machine learning and deep learning for designs of smart systems. Wan-Ju Liow received the B.S. degree in communication engineering from National Central University in 2019. Currently, she is a Master student in computer and communication engineering from the National Cheng Kung University, Tainan, Taiwan. Her current research interests include pattern recognition and street object tracking with deep learning systems. Shao-Fu Chen received the B.S. degree in communication engineering from National Central University and M.S. degree in computer and communication engineering from the National Cheng Kung University, Tainan, Taiwan in 2017 and 2019, respectively. Her current research interests include image processing, deep learning systems, machine learning and recurrent neural networks. Pau-Choo Chung received the Ph.D. degree in electrical engineering from Texas Tech University, Lubbock, TX, USA, in 1991. She was with the Department of Electrical Engineering, National Cheng Kung University (NCKU), Tainan, Taiwan, in 1991 and became a Full Professor in 1996. She applies most of her research results to healthcare and medical applications. Dr. Chung is a member of the Phi Tau Phi Honor Society, was a member of the Board of Governors of CAS Society from 2007 to 2009 and from 2010 to 2012, and is currently an ADCOM Member of the IEEE CIS and the Chair of CIS Distinguished Lecturer Program. She also is an Associate Editor of IEEE Transaction on Neural Networks and the Editor of Journal of Information Science and Engineering, the Guest Editor of Journal of High Speed Network, the Guest Editor of IEEE Transaction on Circuits and Systems-I, and the Secretary General of Biomedical Engineering Society of China. She is one of the Co-Founders of Medical Image Standard Association (MISA) in Taiwan and is currently on the Board of Directors of MISA. Her research interests include image/video analysis and pattern recognition, bio signal analysis, computer vision, and computational intelligence. She is IEEE fellow. Jar-Ferr Yang received the Ph.D. degree in electrical engineering from University of Minnesota, Minneapolis, MN, USA in 1988. He joined National Cheng Kung University (NCKU), Taiwan, in 1988 and was promoted to Distinguished Professor in 2004. Dr. Yang was the Distinguished Lecturer Program by the IEEE Circuits and Systems Society (CAS) from 2004 to 2005. He was the Chair of the IEEE CAS Multimedia Systems and Applications Technical Committee from 2008 to 2009. He was an Associate Editor of IEEE Transaction on Circuits and Systems for Video Technology and EURASIP Journal of Advances in Signal Processing. He is IEEE Fellow. Currently, he is an Associate Editor of IET Signal Processing. He was a recipient of the NSC Excellent Research Award in Taiwan in 2008. He has published over 135 journal and 216 conference papers. Currently, his research interests include multimedia processing, coding, and recognition. Songan Mao received his M.S. degree in soft engineering from University of Electronic Science and Technology of China, Chengdu, People's Republic of China and Ph.D. in electrical and computer engineering from Purdue University, USA. Currently, he is a senior engineer in QTL Technology and Product Management Team, San Diego, USA, His current research interests include image processing, computer vision, machine learning, and object detecting.

Funding

This work was partially supported by the Ministry of Science and Technology under Grant MOST 109-2218-E-006-032, Taiwan and Qualcomm Taiwan University Research Project SOW#NAT-435536, USA.

Availability of data and materials

In this research, we started from COCO pre-trained weights and then combined the KITTI dataset [36] and a self-build dataset collected in the Taiwanese highway to train the vehicle detectors. The self-build dataset and all related material will be available for any readers if they have requested.

Declarations

Ethics approval and consent to participate

In this research, we declare that the studies do not involve any human participants, human data, human tissue, and animals.

Consent for publication

In this research, we declare that the manuscript does not contain any individual person's data in any form (including individual details, images or videos).

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Department of Electrical Engineering, Institute of Computer and Communication Engineering, National Cheng Kung University, Tainan, Taiwan. [2]Qualcomm Incorporated, San Diego, USA.

Received: 5 May 2021 Accepted: 18 January 2022
Published online: 02 February 2022

References

1. Z. Cao, G. Hidalgo, T. Sion, S.-E. Wei, and Y. Sheikh, OpenPose: realtime multi-person 2D pose estimation using part affinity fields (2018). arXiv:1812.08008.
2. L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler, and B. Schiele, Deepcut: Joint subset partition and labeling for multi person pose estimation, in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4929–4937 (2016).
3. K. Zhang, Z. Zhang, Z. Li, Y. Qiao, Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Process. Lett. **23**(10), 1499–1503 (2016)

4. Y. Taigman, M. Yang, M. A. Ranzato, and L. Wolf, Deepface: closing the gap to human-level performance in face verification, in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1701–1708 (2014).
5. J. Long, E. Shelhamer, and T. Darrell, Fully convolutional networks for semantic segmentation, in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440 (2015).
6. K. He, G. Gkioxari, P. Dollár, and R. Girshick, Mask r-cnn, in *Proceedings of IEEE Conference on Computer Vision (ICCV)*, pp. 2961–2969 (2017).
7. V. Badrinarayanan, A. Kendall, R. Cipolla, Segnet: a deep convolutional encoder–decoder architecture for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **39**(12), 2481–2495 (2017)
8. N. Dalal, and B. Triggs, Histograms of oriented gradients for human detection, in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 886–893 (2015).
9. D. G. Lowe, Object recognition from local scale-invariant features, in *Proceedings of IEEE Conference on Computer Vision (ICCV)*, pp. 1150–1157 (1999).
10. P.F. Felzenszwalb, R.B. Girshick, D. McAllester, D. Ramanan, Object detection with discriminatively trained part-based models. IEEE Trans. Pattern Anal. Mach. Intell. **32**(9), 1627–1645 (2009)
11. P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features. Proc. IEEE Conf. Comput. Vis. Pattern Recognit. **1**, 511–518 (2001)
12. Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. **55**(1), 119–139 (1997)
13. C.-W. Hsu, C.-C. Chang, and C.-J. Lin, A practical guide to support vector classification. Technical Guide, Department of Computer Science, National Taiwan University, Taipei 106, Taiwan. http://www.csie.ntu.edu.tw/~cjlin2010.
14. M. Everingham, L. Van Gool, C.K. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge. Int. J. Comput. Vis. **88**(2), 303–338 (2010)
15. A. Krizhevsky, I. Sutskever, and G. E. Hinton, Imagenet classification with deep convolutional neural networks, in *Proceedings of Neural Information Processing Systems Conf. (NIPS)*, pp. 1097–1105 (2012).
16. M. D. Zeiler, and R. Fergus, Visualizing and understanding convolutional networks, in *Proceedings of European Conference on Computer Vision (ECCV)*, pp. 818–833 (2014).
17. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, Going deeper with convolutions, in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9 (2015).
18. K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *Proceeedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778 (2016).
19. J. Hu, L. Shen, and G. Sun, Squeeze-and-excitation networks, in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141 (2018).
20. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, Imagenet: a large-scale hierarchical image database, in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* pp. 248–255 (2009).
21. R. Girshick, J. Donahue, T. Darrell, and J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587 (2014).
22. R. Girshick, Fast r-cnn, in *Proceedings of IEEE Conference on Computer Vision (ICCV)*, pp. 1440–1448 (2015).
23. S. Ren, K. He, R. Girshick, and J. Sun, Faster r-cnn: towards real-time object detection with region proposal networks, in *Procedings of Neural Information Processing Systems Conference (NIPS)*, pp. 91–99 (2015).
24. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, SSD: single shot multibox detector, in *Proceedings of European Conference on Computer Vision*, pp. 21–37 (2016).
25. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, You only look once: unified, real-time object detection, in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788 (2016).
26. Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu and N. Yu. Online multi-object tracking using CNN-based single object tracker with spatial-temporal attention mechanism, in *Proceedings of the IEEE International Conference on Computer Vision,* pp. 4836–4845, (2017).
27. B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, High performance visual tracking with siamese region proposal network, in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8971–8980 (2018).
28. X. Wang, R. Girshick, A. Gupta, and K. He, Non-local neural networks, in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7794–7803 (2018).
29. S. Hochreiter, J. Schmidhuber, Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
30. J. Redmon, and A. Farhadi, YOLO9000: better, faster, stronger, in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7263–7271 (2017).
31. J. Redmon and A. Farhadi, YOLOv3: An incremental improvement, in Proceedings of Computer Vision and Pattern Recognition (2018). arXiv:1804.02767.
32. A. Bochkovskiy, C.-Y. Wang, H.-Y. M. Liao, YOLOv4: optimal speed and accuracy of object detection, in Proceedings of Computer Vision and Pattern Recognition (2020). arXiv:2004.10934.
33. A. Graves, A.-R. Mohamed, and G. Hinton, Speech recognition with deep recurrent neural networks, in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645–6649 (2013).
34. M. Hermans, and B. Schrauwen, Training and analysing deep recurrent neural networks, in *Proceedings of Neural Information Processing Systems Conference*, pp. 190–198 (2013).
35. K. Lee, I. Lee and S. Lee. Propagating LSTM: 3D pose estimation based on joint interdependency, in *Proceedings of European Conference on Computer Vision*, pp. 119–135 (2018).
36. S. Yun and S Kim, Recurrent YOLO and LSTM-based IR single pedestrian tracking, in *Proceedings of 19th International Conference on Control, Automation and Systems, Jeju, Korea* (2019).
37. G. Ning, Z. Zhang, C. Huang, Z. He, X. Ren and H. Wang, Spatially supervised recurrent convolutional neural networks for visual object tracking, in *Proceedings of IEEE International Symposium on Circuits and Systems, Baltimore, MD*, pp. 1–4 (2017). https://doi.org/10.1109/ISCAS.2017.8050867.
38. V. Nair, and G. E. Hinton, Rectified linear units improve restricted Boltzmann machines, in *Proceedings of the 27th International Conference on Machine Learning*, pp. 807–814 (2010).

39. S. Ioffe, and C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift (2015). arXiv:1502.03167.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.