Image and Vision Computing 87 (2019) 47–56

Contents lists available at ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

Automatic License Plate Recognition via sliding-window darknet-YOLO deep learning[☆]

Hendry[a, b], Rung-Ching Chen[a, *]

[a]Department of Information Management, Chaoyang University of Technology, Taichung, Taiwan
[b]Faculty of Information Technology, Satya Wacana Christian University, Central Java, Indonesia

ARTICLE INFO

Article history:
Received 28 March 2019
Accepted 18 April 2019
Available online 7 May 2019

Keywords:
Automatic License Plate Recognition
Deep learning
YOLO network

ABSTRACT

Automatic License Plate Recognition (ALPR) is an important research topic in the intelligent transportation system and image recognition fields. In this work, we address the problem of car license plate detection using a You Only Look Once (YOLO)-darknet deep learning framework. In this paper, we use YOLO's 7 convolutional layers to detect a single class. The detection method is a sliding-window process. The object is to recognize Taiwan's car license plates. We use an AOLP dataset which contained 6 digit car license plates. The sliding window detects each digit of the license plate, and each window is then detected by a single YOLO framework. The system achieves approximately 98.22% accuracy on license plate detection and 78% accuracy on license plate recognition. The system executes a single detection recognition phase, which needs around 800 ms to 1 s for each input image. The system is also tested with different condition complexities, such as rainy background, darkness and dimness, and different hues and saturation of images.

1. Introduction

With recent improvement in intelligent transportation systems and Graphical Processing Units (GPU) to support deep learning computation, Automatic car License Plate Detection and Recognition (ALPR) has become an interesting research field. ALPR has many potential application domains in transportation. Advanced computer vision technologies and artificial intelligence algorithms have been proposed to identify vehicle licenses. ALPR usually consists of four steps: image collection, object detection, segmentation, and optical character recognition [1,2].

The first step of ALPR is image collection. Images may be extracted from a video, collections of images and cameras. Usually, in the research area, image collections are provided from an open dataset. The second step is detecting the license plate in the images. This phase usually takes place in the object detection step. Edge detection is commonly used for plate detection. In addition to edge detection, many algorithms have been proposed to solve plate detection. After the plate is detected, the segmentation phase is conducted to divide the region into locations for detecting digits and letters. The last step is recognition of each segmented region into digits and letters to read the license plate. Recent research has largely divided the ALPR phase into two main phases, the license plate detection phase and the object recognition phase [1,3–5]. These algorithms separate the detection and recognition phase which are time-consuming and computationally complex. Character segmentation during object recognition is also a challenging task [3,6].

With the development of GPUs that support high computational complexity, Convolutional Neural Network (CNN) has become increasingly common in object detection applications. CNN achieves impressive results in image classification [7-11], digit recognition [12-14], object detection [15-19] and many information retrieval domains. Region-based detection algorithms combined with CNN (R-CNN) [20] have demonstrated state-of-the-art performance in object detection. However, computational complexity makes R-CNN and their improvement algorithms [21] unsuitable for real-time detection.

The You Only Look Once (YOLO) architecture consists of 27 CNN layers, with 24 convolutional layers, followed by two fully connected convolutional layers as shown in Fig. 1. Since YOLO has issues with detecting small objects [23], detecting characters in the confined spaces of license plates reduces YOLO performance drastically. In this paper, we proposed a modified YOLO using a sliding-window and single class tiny YOLO detector to detect small objects such as characters in a license plate. We also consider the environmental conditions of the license plate, such as illumination, weather conditions, indoor, outdoor and various other conditions. The main contributions of this work are as follows:

☆ This paper has been recommended for acceptance by Sinisa Todorovic.
* Corresponding author.
E-mail addresses: hendry@uksw.edu (. Hendry), crching@cyut.edu.tw (R.-C. Chen).

https://doi.org/10.1016/j.imavis.2019.04.007
0262-8856/© 2019 Elsevier B.V. All rights reserved.

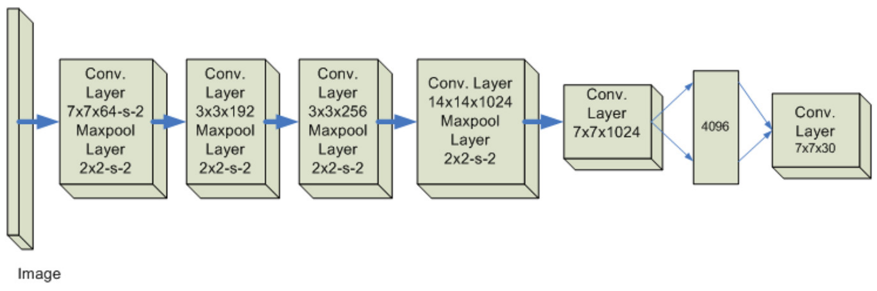


Image
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x256 Maxpool Layer 2x2-s-2
Conv. Layer 14x14x1024 Maxpool Layer 2x2-s-2
Conv. Layer 7x7x1024
4096
Conv. Layer 7x7x30

Fig. 1. YOLO's system architecture [22].

- We propose a sliding-window single class detector via tiny YOLO CNN classifiers. We reduce the original YOLO architecture with its 27 CNN layers and 24 convolutional layers to a modified tiny YOLO which consists of only 13 CNN layers, with 7 convolutional layers. We use a single class detector for each class. We use 36 tiny YOLO models to detect 36 classes (10 digits, an alphabet consisting of 25 letters, and 1 plate). Taiwan license plates do not contain the letter "O" (zero and "O" are the same). The detection and recognition phases run at the same time within a given image. With this framework, our system can detect license plates with high precision. It also recognizes the characters and digits without using character segmentation as YOLO directly detects and recognizes them in one phase.
- We develop an algorithm to detect and recognize characters and digits in the same phase. To the best of our knowledge, this is the first work that uses only one phase to read and detect the car license plate. YOLO detects and reads all the objects inside the image. The next step is the automatic localization process, in which each character inside the plate area is grouped. YOLO then ranks the confidence of its segmentation of the location for each character as YOLO may detect 1 real object as several predicted objects. This approach takes advantage of both CNN for feature learning and automatic localization to avoid the character segmentation process.
- We also propose a modified YOLO to handle YOLO issues when detecting small objects in confined spaces. YOLO already encounters face problems when detecting small objects [23,24]. Since car license plates consist of characters in small spaces, our modified YOLO offers high precision in detecting and recognizing license plates.

The remainder of this paper is organized as follows. Section 2 describes previous work on car license plate detection and recognition. Section 3 briefly describes our proposed methodology. Section 4 describes the dataset and the training environment of the system. Section 5 provides the training and testing results of the system. Section 6 presents the conclusions of this paper.

2. Related work

2.1. Car plate detection and recognition

Much research has been conducted in the field of ALPR [4,1,5,25,26]. License plate detection generates a bounding box for the car license number in the image. Existing algorithms use morphological categorization methodologies that categorize the image in four ways: edge-based [5], color-based [27], texture-based [28], and character-based [3,29]. Edge detection is one of the most commonly used features in plate detection [5]. Since a license plate consists of a dense black dot matrix forming the character on white background, edge detection is useful in detecting the license plate. Edge-based approaches detect the edge density of the black characters against the white background. Although computation is rapid, the edge detection algorithm is sensitive to unwanted edges. This can happen when many colors appear in the image, or when the differentiation between black and white interferes with edge detection.

Color-based approaches are based on differentiating the colors of the car body and the license plate. Deb and Jo [30] proposed a HSI color model to detect candidate license plate regions, which is later verified by histogram when the license plate is detected. Similar to edge detectors proposed by Refs.[25,26,31,32], to reduce noise from the license plate, Mathur et al. [27] proposed a gray-scale image as an input. Although color-based detection is easily implemented and can detect inclined and deformed license plates, this algorithm failed to distinguish license plates when the car body color is similar to the license plate color. Further, this algorithm is very sensitive to illumination changes in outdoor locations.

Different from the edge detector and color approaches, Hsu et al. [4] proposed clustering methods that extract cluster dense sets and rectangular shapes which are similar to a license plate shape. Gaussian cluster EM algorithm [28] is then used to calculate the dense cluster sets of the license plate. This approach is categorized as texture-based. Texture-based approaches use more discriminating characteristics than edges and color but have higher computational complexity. This approach is not suitable for real time license plate detection because it is time-consuming.

Character-based approaches detect the objects using the string of characters by learning character features directly from the image. CNN is can learn features directly from the image. Character-based methods are a reliable and fast detector for object detection problems. YOLO [22] can detect many objects with high precision and rapid computation speed. Other state-of-the-art object detection algorithms before the emergence of CNN include the deformable part model (DPM) [33] which uses handcrafted features such as a histogram [34].

Our work takes a character-based approach since we use CNN to learn characters and digits in the license plate directly by extracting its features. CNN's strong capabilities to perform classification guarantee the character detection performance.

2.2. You Only Look Once (YOLO)

YOLO is a convolutional neural network object detection system targeted for real time processing. Fig. 2 shows that YOLO works by dividing the input images into an $N*N$ grid cell. Each grid cell predicts only one object. Each grid cell predicts a fixed number of boundary boxes. The blue dot is the center of the detected object surrounded by gray bounding boxes. However, the one-object rule limits how close the detected object can be. For that reason, YOLO has limitations on how close objects can be. Fig. 2 illustrates how YOLO rules limit the close detection of objects in the image. These limits are shown by the red rectangles between objects detected. When one object has been

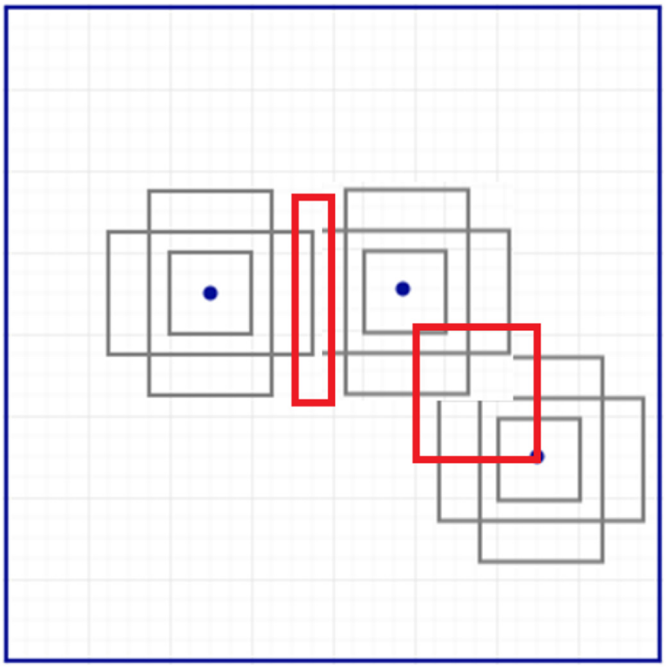


Fig. 2. YOLO boundary boxes limitation.

detected, it will prevent the other bounding boxes from detecting the object when there is an intersection between bounding boxes.

YOLO architecture consists of 27 CNN layers, with 24 convolutional layers, two fully connected layers, and a final detection layer [22] (see also Fig. 1). The output YOLO network will be a vector of $N*N*(5B+C)$, where B is each grid cells predicting bounding boxes and each box has one box confidence score, and C is one set of class scores for all bounding boxes in the region. Each bounding box contains five elements: (x, y, δ, w, h). The confidence score reflects how each boundary box contains an object and how accurate the bounding box is. The major concept of YOLO is to build a CNN network to predict the (7, 7, 30) tensor. YOLO performs a linear regression using two fully connected layers, and makes the prediction using a final layer to retain boundary boxes with a high confidence score.

Since YOLO predicts multiple bounding boxes for each grid cell, to predict the object needing detection, one responsible loss function must be calculated. YOLO uses three loss calculation functions: classification loss, localization loss, and confidence loss. To select one that can be used as a loss function, the Intersection over Union (IoU) is calculated and then the IoU with the highest value is selected. YOLO uses sum-squared error to calculate error loss between the predictions and the real objects. Classification loss is the squared error of the class probability. Localization loss measures the errors in the boundary box locations and sizes. Confidence loss is the error in the measurement of the likelihood of objects in the box. The final loss of YOLO loss is the sum of all the loss functions (loss = classification loss + localization loss + confidence loss)

Our method modifies the original YOLO into a sliding-window single class tiny YOLO detector. We broaden the YOLO network model to 36 parallel execution models with smaller networks. We thus enhance YOLO capability to detect small objects in tight spaces while considering speed and execution time.

3. Proposed methodology

In this section, we describe our proposed methodology to detect and recognize car license plates. Instead of using many class predictors, we use Sliding-Window Single Class Detection (SWSCD). Our model thus has greater computational complexity than the original YOLO. To handle this trade-off, we decrease the number of layers in our YOLO model.

Fig. 3 shows the system architecture of the proposed method. The computation of sliding windows consists of 36 labels which are 25 letters (O and zero are the same on Taiwan license plates), ten numerals and one plate label. After the detection of all objects in the image, the system will perform localization and group the numerals and letters within plate regions. While the original YOLO consists of

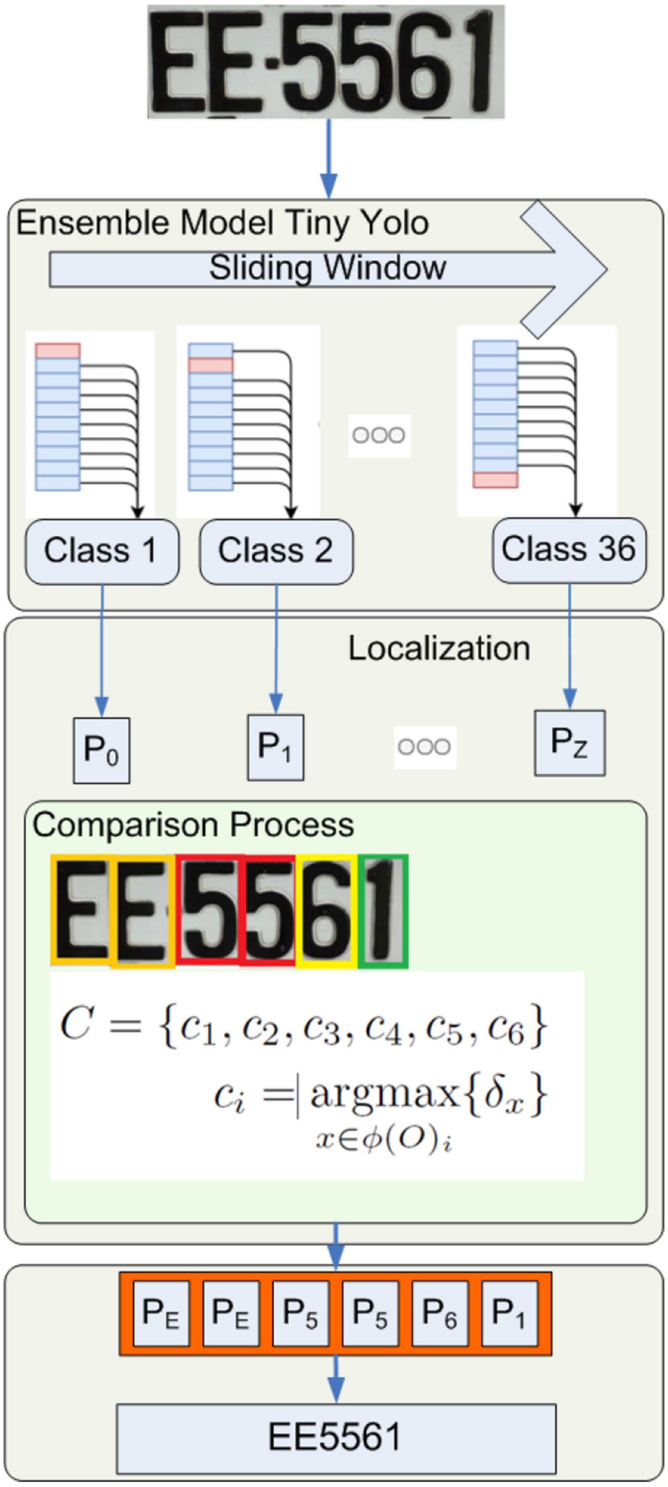
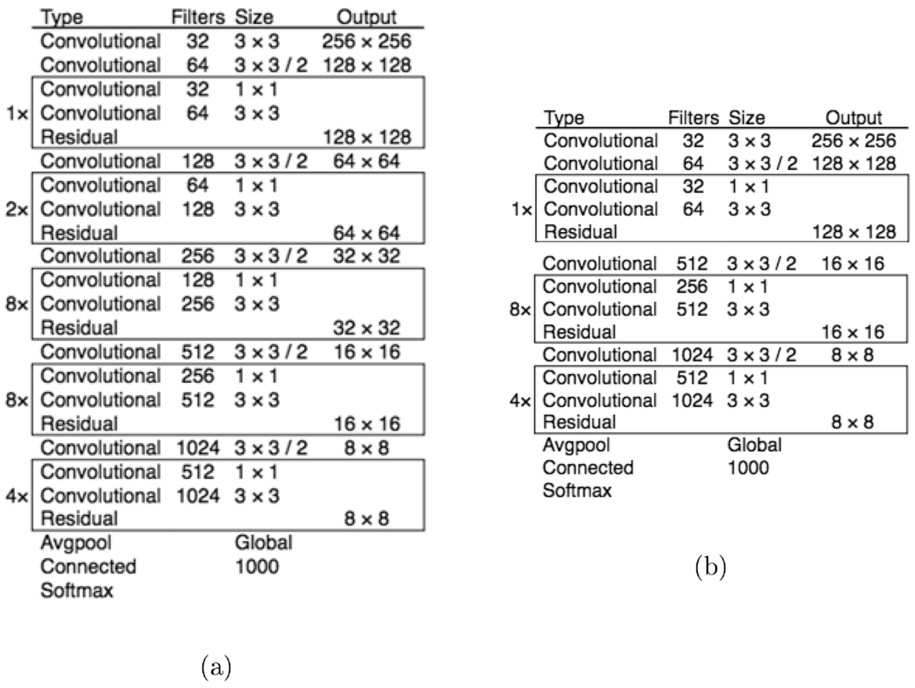

EE-5561
Ensemble Model Tiny Yolo
Sliding Window
Class 1
Class 2
Class 36
Localization
P_0
P_1
P_Z
Comparison Process
$C = \{c_1, c_2, c_3, c_4, c_5, c_6\}$
$c_i = \operatorname{argmax}_{x \in \phi(O)_i}\{\delta_x\}$
P_E P_E P_5 P_5 P_6 P_1
EE5561

Fig. 3. The system architecture.



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

(a)

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

(b)

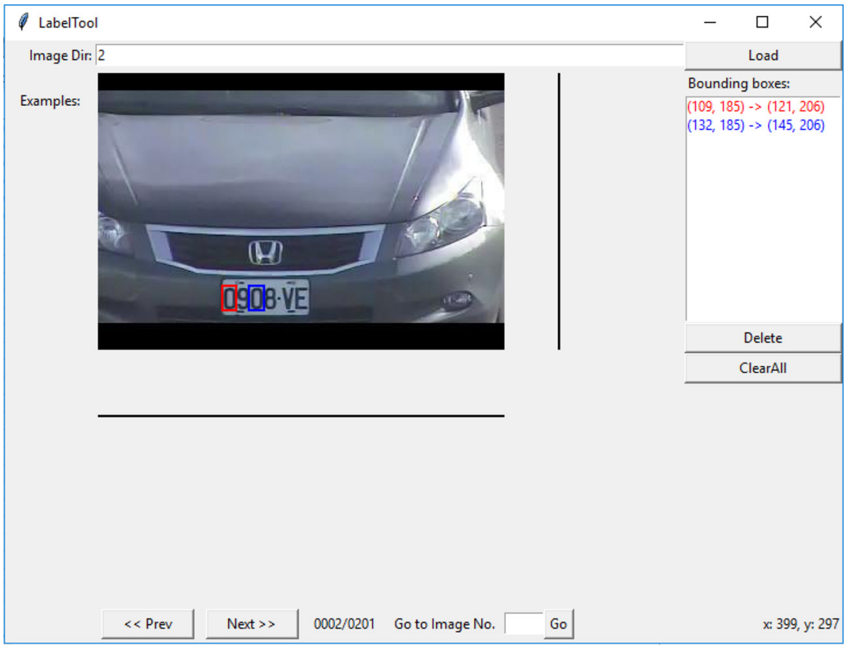
Fig. 4. The structure comparison. (a). Original YOLO structure, (b) modified YOLO structure.

27 CNN layers, with 24 convolutional layers, followed by two fully connected convolutional layers, we reduced this to 10 CNN layers, with 13 convolutional layers, to detect only a single class for each tiny YOLO. The original YOLO structure is shown in Fig. 4a and our modified YOLO is shown in Fig. 4b.

3.1. Car license plate and objects detection

The car license plate and character detection is handled by the SWSCD-YOLO framework with the detection and recognition performed in one phase. We start by generating a bounding box for each character for training purposes using the BBox label tool [35]. Fig. 5 shows the labeling process using Bbox label tools to create bounding boxes for each class. This process is performed for 36 class labels: 25 letters, 10 digits, and 1 plate label.

Fig. 6 shows a portion of the bounding box process for the 36 classes. One image can host more than one bounding box. We use a single class detector model under which one class label belongs to one training model. The return values of the bounding box labeling tool are object coordinates in the form (x_1, y_1, x_2, y_2). The YOLO input value is not in the form of object coordinates. Instead, the YOLO



(109, 185) -> (121, 206)
(132, 185) -> (145, 206)

Fig. 5. BBox label tool user interface for bounding boxes.

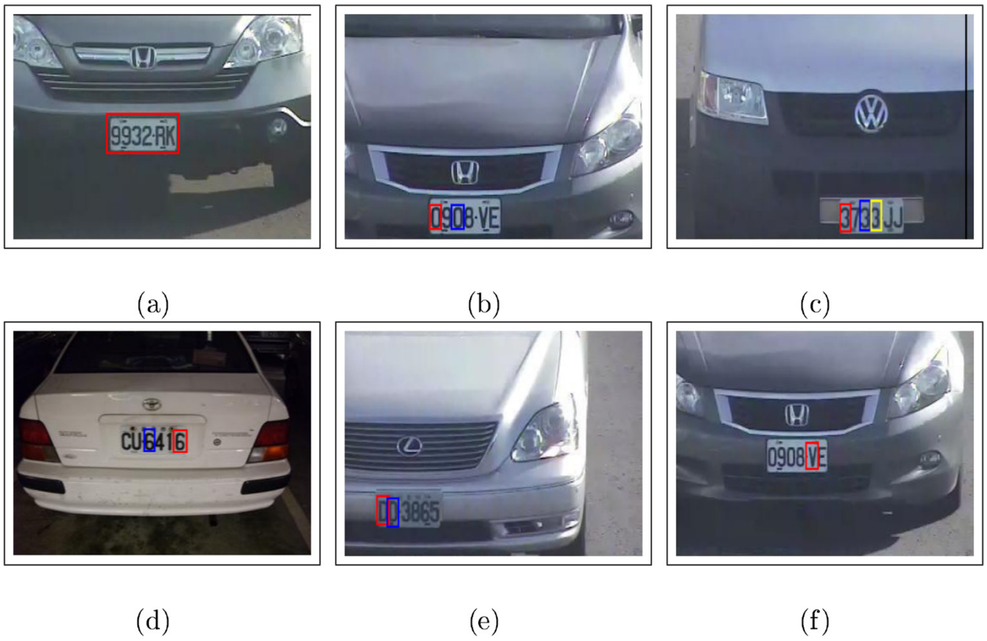


Fig. 6. Bounding box labeling process for classes: (a). Class "Plate", (b). Class digit "0", (c). Class digit "3", (d). Class digit "6", (e). Class letter "D", (f). Class letter "V".

input value is the center point of the object and its width and height (x, y, w, h). Hence, we need to convert the bounding box coordinate into the YOLO input model. The conversion process is shown in Eqs. (1)–(4).

$$dw = \frac{1}{W}$$
$$x = \frac{(x_1 + x_2)}{2} * dw$$
$$dh = \frac{1}{H} \quad (1)$$

$$y = \frac{(y_1 + y_2)}{2} * dh \quad (2)$$

$$w = (x_2 - x_1) * dw \quad (3)$$

$$h = (y_2 - y_1) * dh \quad (4)$$

where W is the width of the image and H is the height of the image.

In the object detection phase, we put the given image into the SWSCD-YOLO via sliding-window processing. This means that each class of object, plate, digit, and letter, will be detected in a sliding-window fashion across the image. Each model returns the detection value in the (l, δ, x, y, w, h) configuration where l is the label of the object, δ is the confidence of the detected object, x, y is the location of the detected object, and w, h is the size of the detected object. Fig. 4b is our SWSCD-YOLO configuration file.

Fig. 7 presents the SWSCD-YOLO object detection results. Fig. 7a shows that SWSCD-YOLO returns many objects detected in the same location. In Fig. 7b, SWSCD-YOLO returns objects detected outside of the license plate region (red rectangle). The localization process is needed to check whether characters from SWSCD-YOLO detector belong to the license plate.

3.2. Localization

After the system detects all the objects in the image, the system begins to identify the localization of the plate. Each character will be

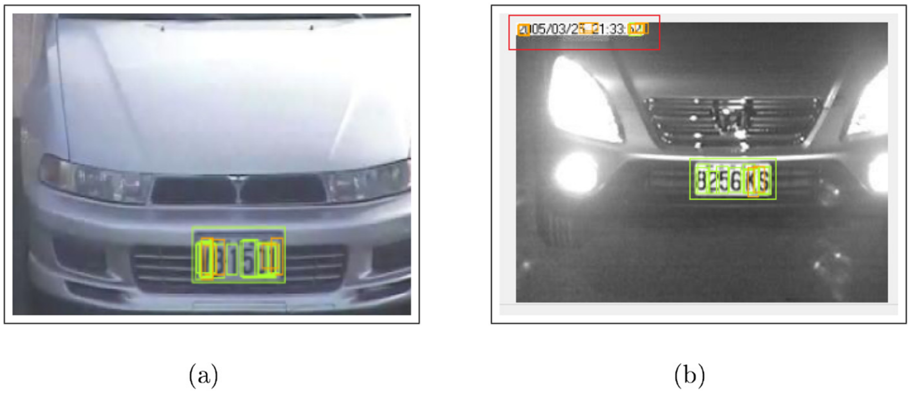


Fig. 7. YOLO object detection: (a). License detection in normal condition, (b). License detection in nighttime condition.

grouped to the local plate. The localization formula is given as Eqs. (5)–(8).

$$P(x) = \begin{cases} 1, \text{if } (x_i > x_p \wedge x_i + w_i < x_p + w_p) \\ 0, \textit{otherwise} \end{cases} \tag{5}$$

$$P(y) = \begin{cases} 1, \text{if } (y_i > y_p \wedge y_i + h_i < y_p + h_p) \\ 0, \textit{otherwise} \end{cases} \tag{6}$$

$$o_j = \begin{cases} 1, \text{if } P(x)_i * P(y)_i = 1, \\ 0, \textit{otherwise} \end{cases} \tag{7}$$

$$O = \forall(o_j = 1) \tag{8}$$

where $P(x)$ is the probability of x_i in x_p, x_i is the axis position of i object, and x_p is the axis position of this plate. $P(y)$ is the probability of y_i in y_p, y_i is the ordinate position of i object, and y_p is the ordinate position of this plate. O is the set of o, where o is the object in the (l, δ, x, y, w, h) configuration.

After SWSCD-YOLO returns objects which are detected in the image, the system sorts the position and rank based on the object's confidence. Objects detected are compared to each other in order to determine which object will be chosen if they occupy a similar position.

$$\begin{aligned} C &= \{c_1, c_2, c_3, c_4, c_5, c_6\} \\ c_i &= \arg \max_{x \in \phi(O)_i} \{\delta_x\} \end{aligned} \tag{9}$$

The Taiwan license plates in the AOLP dataset consist of 6 objects. Furthermore, C is a set which consists of 6 elements {c1,c2,...,c6}, meaning that each object on the license plate can be mapped into 6 positions using function $\phi(O)_i$. For each position, we choose the highest confidence as our prediction objects. Fig. 8 shows the overall process of our proposed method. Fig. 8a shows that a rectangle was weakly detected and the object has a confidence δ lower than 0.4, while a strong confidence object has a confidence higher than 0.4. Fig. 8b shows the localization process that ensures all objects are inside the license plate area and Fig. 8c shows the license plate recognition.

Table 1
Variables and variation scope for three categories of AOLP dataset [4] .

	AC	LE	RP
Pan	$-30^\circ \sim 30^\circ$	$-40^\circ \sim 40^\circ$	$-60^\circ \sim 60^\circ$
Tilt	$0^\circ \sim 60^\circ$	$20^\circ \sim 70^\circ$	$0^\circ \sim 50^\circ$
Size (in width ratio)	0.20 ~ 0.25	0.10 ~ 0.20	0.10 ~ 0.40
Ave. Illum. intensity	60 ~ 130	40 ~ 150	40 ~ 150
Distance	<5m	<15m	<15m
Proj. orientation	$<10^\circ$	$<15^\circ$	$<30^\circ$
Number of samples	681	757	611

4. Experimental setup

4.1. The datasets

We use AOLP dataset [4] to train our models. The dataset consists of 2049 images of Taiwan car license plates. The samples are collected at different locations, times, traffic densities, and weather conditions. The dataset consists of 3 categories which are Access Control (AC), Traffic Law Enforcement (LE), and Road Patrol (RP). Those categories have unique characteristics as shown in Table 1.

In this paper, we used the AOLP dataset for the experimental model. In AC scenarios, the camera is often placed less than the 5 m distance from the plate, at $\pm 30^\circ$ in the pan and $0^\circ \sim 60^\circ$ in the tilt. 0° tilt is parallel to the ground. In the image, the width of a plate is between the ratio of 0.20 and 0.25 of the image, and orientation is less than 10°. The total number of images is 681, collected under various conditions as described earlier. The training set consists of 581 images, while the test set consists of 100 images. In the LE scenarios, the images are collected from cases in which a vehicle violates traffic laws and is captured by a roadside camera. For this application, 757 images were collected. In the RP scenarios, the images are collected from the camera which is installed in a patrolling vehicle. The images are collected from various arbitrary viewpoints and distances. For this application, 2049 images were collected. The preview of the dataset is shown in Fig. 9.

4.2. Training

For the training, we reproduced the images and categorized the characters we wanted to recognize. One image could be reproduced several times depending on the objects on the license plate. For this preprocessing step, we manually detected and recognized the license plates. We use a bounding box labeling tool (BBox Label Tool [35]) to give a coordinate location to the objects we wanted to detect and recognize. The labeling tool will return four points of the position coordinate, along with the class label. The training model environment is an Nvidia GTX970 GPU accelerator 4 GB memory, i7 Central Processing Unit (CPU), and 16 GB DDR2 memory.

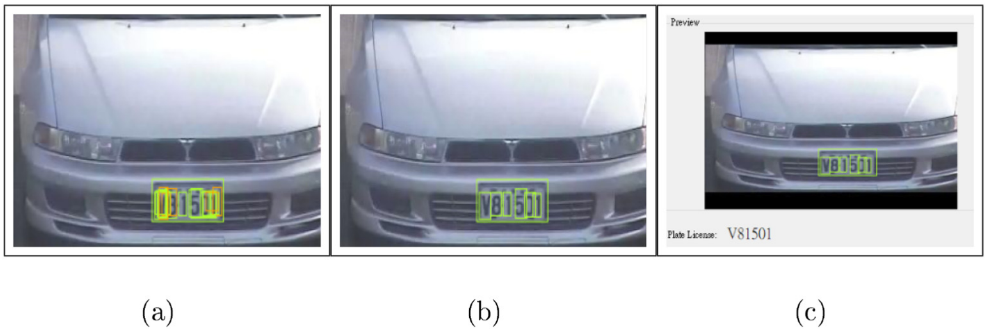


Fig. 8. License plate detection in overall process: (a). YOLO detection, (b). Localization process, (c). License plate recognition.

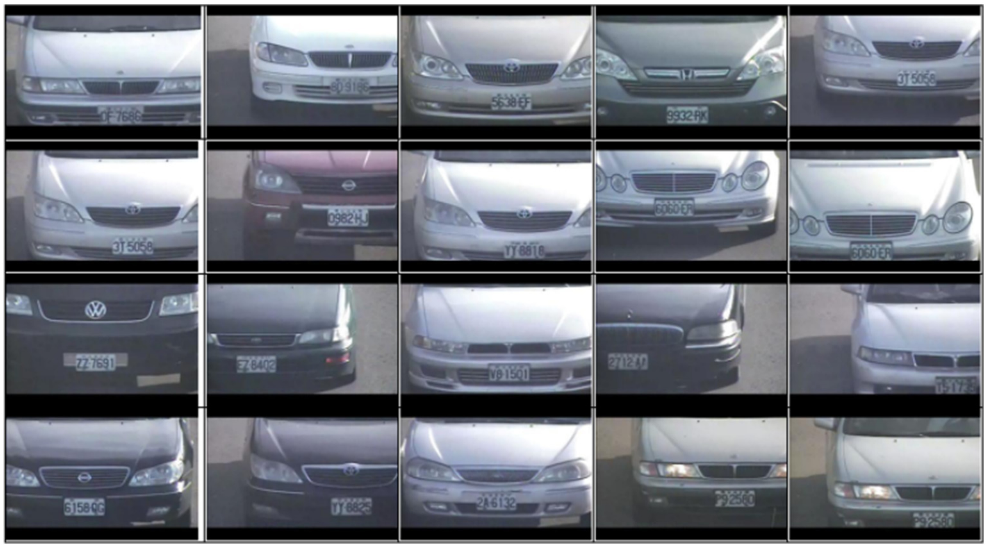


Fig. 9. Examples of datasets.

5. Results

Our YOLO model sets the parameters of the second to last layer to reduce it to a single class. We develop a sliding-window single class detector for the 36 classes: 10 numerical digits, 25 letters, and one plate class. The test images from the AOLP dataset's AC, LE, and RP categories evaluate the system performance. Table 2 shows the training validation loss for all classes after 2000 training epochs. The average of the training validation loss for digits, letters and the plate is around 0.04. Thus, our training model detected the objects with high accuracy. Fig. 10 shows our training validation loss for digits, letters, and the plate. The training model converges after 700 iterations and is stable for the remainder of the training phase. The average validation loss for digits is 0.0459, for letters, 0.0219, and for the plate, 0.0246.

The average mean average precision (mAP) is the integral over the precision $p(o)$.

$$mAP = \int_0^1 p(o)do \tag{10}$$

where $p(o)$ is the precision of the license plate detection. Precision is represented by:

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{N} \tag{11}$$

where TP represents True positives, FP false Positives, and N is the total number of objects retrieved ($TP + FP$). Table 3 shows the calculated average precision for digits, letters, and the plate. Table 4 shows

Table 2
Training validation loss for all classes.

Class	Loss	Class	Loss	Class	Loss
0	0.0459	C	0.0412	P	0.0211
1	0.0336	D	0.0215	Q	0.0245
2	0.0604	E	0.0421	R	0.0281
3	0.0775	F	0.0254	S	0.0214
4	0.0322	G	0.0273	T	0.0612
5	0.0821	H	0.0216	U	0.0123
6	0.0601	I	0.0253	V	0.0273
7	0.0421	J	0.0512	W	0.0484
8	0.0644	K	0.0436	X	0.0451
9	0.1259	L	0.0412	Y	0.0342
A	0.0219	M	0.0476	Z	0.512
B	0.0325	N	0.0123	Plate	0.0246

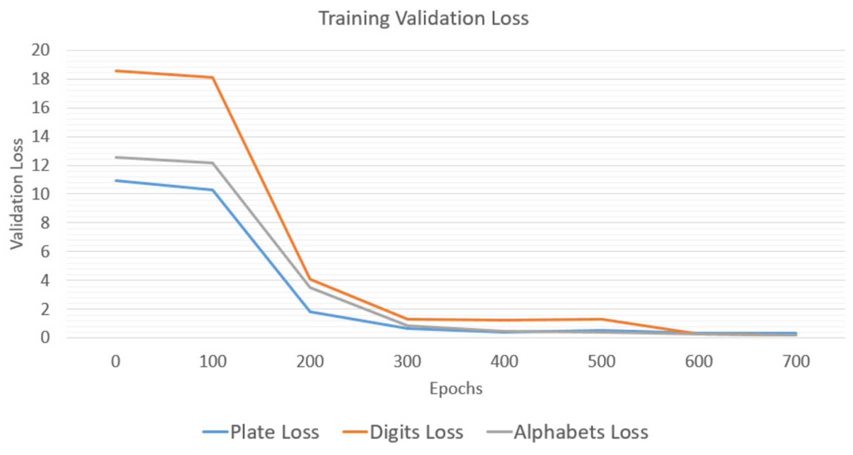


Fig. 10. Training validation loss.

Table 3
mAP and AP performance.

Average class	Value
Digits	0.729882
Letters	0.775544
Plate	0.9900999
mAP	0.83184

Table 4
AP performance for AC, LE and RP dataset.

Confidence	AC AP value	LE AP value	RP AP value
0.1	0.781667	76.45	75.423
0.2	0.76	73.43	72.12
0.3	0.75	72.11	70.64
0.4	0.6833	64.73	61.12
0.5	0.67667	52.15	50.26
0.6	0.675	52.15	50.26

the different confidence levels used in the SWSCD-YOLO detection phase. A lower confidence returns a better AP value, which means that more combination results returned by the detector are better. The drawback with more combination results being returned by the detector is that computational complexity rises when executing the localization process. However, this can be tolerated because at most the total combination will be equal to 216 (6 sequences * 36 classes). The computation time is thus approximately less than 50 ms.

Fig. 11a–b gives the test results from our proposed methods. The system can detect and recognize the license plate across many different conditions. Fig. 11a shows the test results in the daytime with normal weather conditions. Fig. 11b shows the test results for night. In Fig. 11b, our system was unable to detect the digit 3 before digit 7 because of noise below the 3 which changed its shape. Our proposed system can detect and recognize license plates in each image in 825.81 ms on average.

5.1. License plate detection

The detection performance of our proposed method SWSCD-YOLO is shown in Table 5. We compare this with results from previous work to show the robustness of our proposed method. Hsu et al. [4] use clustering methods that extract cluster dense sets and rectangular shapes similar to the shape of a license plate. Li et al. [3] use two-way learning to detect and remove false positives. CNN-1 and CNN-2 are proposed to learn the features using CNN-1 and a text-salience map to generate a candidate bounding box. CNN-2 is then used to remove false positives and to classify objects into plate or non-plate classes.

Table 5
Comparison of plate detection results by different methods on the AOLP dataset.

Method	AC		LE		RP	
	P[a]	R[b]	P[a]	R[b]	P[a]	R[b]
Hsu et al. [4]	91	96	91	95	91	94
Li et al. [3]	98.53	98.38	97.75	97.62	95.28	95.58
Ours	100	99.53	99.01	98.62	95.67	95.71

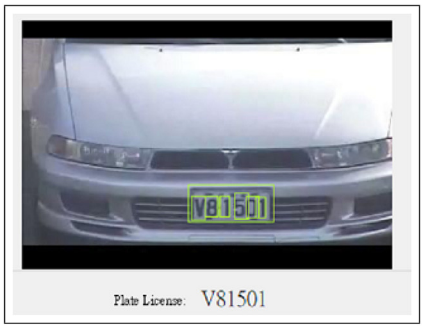
[a] P: Precision (%),
[b] R: Recall (%),

Based on Table 5, our approach outperforms all previous works in precision and recall. SWSCD-YOLO achieves 100% precision and 99.01% recall in the AC category, the best figures from the two methods. The precision value 1.47% higher and recall value 0.63% higher than its closest competitor. In the LE category, our precision and recall values also achieve the best results, 99.01% and 98.62%, respectively, with the precision value 1.26% higher and recall value 1% higher than the second best approach. In the RP category, our precision and recall are slightly higher than the second best, 1% higher. Overall our proposed methods achieve higher precision and recall value. Further, for each image in the AOLP dataset, our processing time is only 800–1000 ms, which outperforms Li et al. [3] (2–3 s).

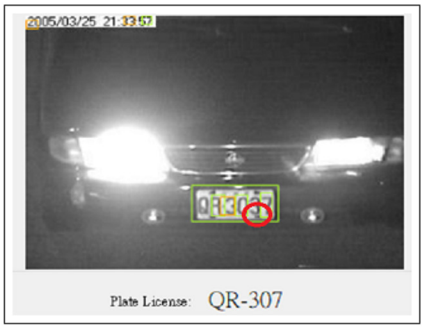
5.2. License plate recognition

In this section, we describe our license plate recognition results. We use only the AC category to train the system, and later use the AC, LE, and RP categories to test the recognition results. The RP category has a great degree of rotation and projective orientation, making it in appropriate for features extraction using a sliding window. Inspired by Li et al. [3], we apply a Hough transformation [36] to first correct the rotations. SWSCD-YOLO achieve approximately 78% accuracy with the AOLP dataset. To the best of our knowledge, this work achieves the best accuracy with a YOLO framework. Our system is quite fast, with an execution time for each image of approximately 800 ms–1 s. Some methods achieved higher accuracy because they applied two phase detection, character segmentation, and noise reduction, which our system does not use. Because of the additional steps, the computational complexity of these systems is significantly higher and processing takes longer. Li et al. stated that their system requires at least 2–3 s merely for license plate detection, with several more seconds for the character recognition process.

The advantage of SWSCD-YOLO is that the system executes the detection and recognition in the same phase, followed by auto localization to group the characters into the plate area. A confidence rank is then applied to remove the unwanted digits and characters from



(a)



(b)

Fig. 11. Results for SWSCD-YOLO object detection.

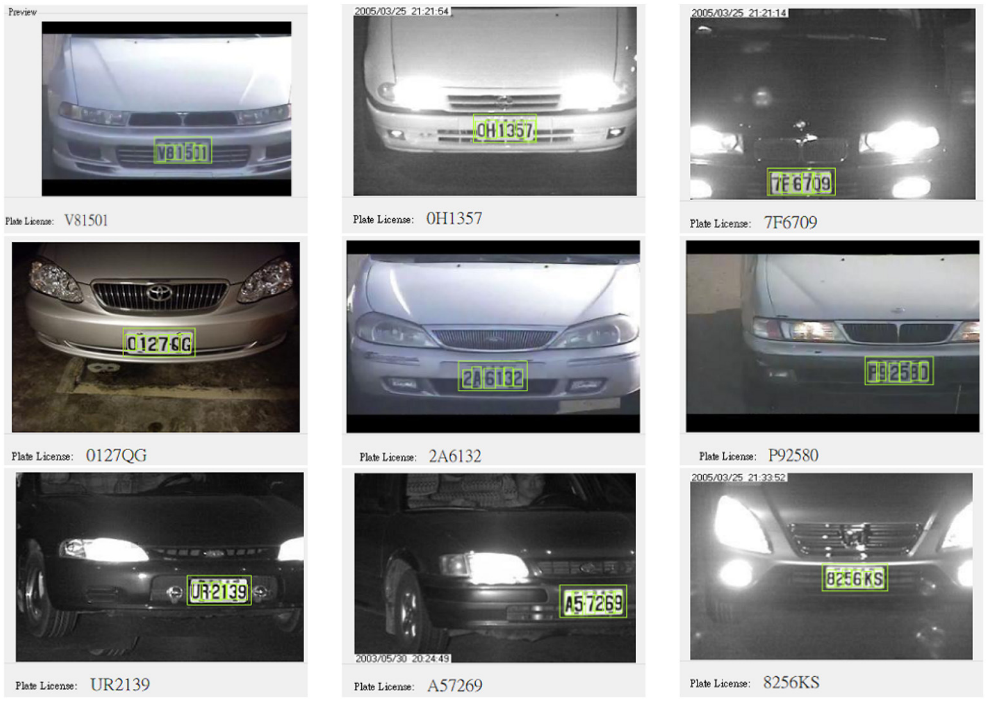


Fig. 12. License plate recognition results, SWSCD-YOLO could detect and recognize license plate under many conditions, i.e., daylight, nighttime, illuminations, orientation, etc.

the license plate area. Because noise reduction, character segmentation, and 1 phase detection are not used in our system, it does not require more time or computations. Fig. 12 shows some experimental results for license plate detection and recognition using the AOLP dataset. The ensemble model enhances the capability of SWSCD-YOLO detection and recognition by specific training for the digits and letters of the license plate using small resolution images. The ensemble model consists of many base classifier models, which require more time for processing. This paper proposes training the model using a sliding window, which causes the base classifier to be executed serially. The computation time can be shortened, if the base classifiers can be executed in parallel

6. Conclusion

In this paper, we present an automatic license plate recognition system with the highest possible recognition speed with little sacrifice of accuracy. We modified the original YOLO to create a tiny YOLO single class detector with 36 different models. The detector works by the sliding window for all classes in each image to prevent small object detection issues. The experiments demonstrate that our average model loss is 0.40, with an average detection and recognition speed of 825.81 ms. The system does not apply either character segmentation or noise reduction and consists of a single phase detection and recognition process. The system achieves 98.22% accuracy in license plate detection and 78% accuracy in license plate recognition.

In future work, we will explore raising license plate recognition accuracy using an applied noise reduction algorithm without significantly raising the computation time. The drawback of an ensemble model is that using a single class classifier will greatly affect computation time. To address this issue, we are considering two solutions. First, we can adopt a proposal-based approach, such as Fast R-CNN, to reduce the computation time in the base classifier. Second, we can use parallel computing to simultaneously compute the base classifier instead of using a sliding window.

Acknowledgments

This research is supported by the Ministry of Science and Technology, Taiwan, with the project numbers MOST-107-2221-E-324-018-MY2 and MOST-106-2221-E-324-025.

References

[1] A. Safaei, H.L. Tang, S. Sanei, Real-time search-free multiple license plate recognition via likelihood estimation of saliency, Comput. Electr. Eng. 56 (2016) 15–29. http://www.sciencedirect.com/science/article/pii/S0045790616302531. https://doi.org/10.1016/j.compeleceng.2016.09.010.
[2] T. Kumar, S. Gupta, D.S. Kushwaha, An efficient approach for automatic number plate recognition for low resolution images, Proceedings of the Fifth International Conference on Network, Communication and Computing, ICNCC '16, ACM, New York, NY, USA, 2016, pp. 53–57. https://doi.org/10.1145/3033288.3033332.
[3] H. Li, P. Wang, M. You, C. Shen, Reading car license plates using deep neural networks, Image and Vision Computing 72 (2018) 14–23. http://www.sciencedirect.com/science/article/pii/S0262885618300155. https://doi.org/10.1016/j.imavis.2018.02.002.
[4] G. Hsu, J. Chen, Y. Chung, Application-oriented license plate recognition, IEEE Trans. Veh. Technol. 62 (2) (2013) 552–561. https://doi.org/10.1109/TVT.2012.2226218.
[5] C.E. Anagnostopoulos, I.E. Anagnostopoulos, I.D. Psoroulas, V. Loumos, E. Kayafas, License plate recognition from still images and video sequences: a survey, IEEE Trans. Intell. Transp. Syst. 9 (3) (2008) 377–391. https://doi.org/10.1109/TITS.2008.922938.
[6] H. Li, P. Wang, C. Shen, Toward end-to-end car license plate detection and recognition with deep neural networks, IEEE Trans. Intell. Transp. Syst. 20 (3) (2019) 1126–1136. https://doi.org/10.1109/TITS.2018.2847291.
[7] M. Paoletti, J. Haut, J. Plaza, A. Plaza, A new deep convolutional neural network for fast hyperspectral image classification, ISPRS J. Photogramm. Remote Sens. 145 (2018) 120–147. deep Learning RS Data. http://www.sciencedirect.com/science/article/pii/S0924271617303660. https://doi.org/10.1016/j.isprsjprs.2017.11.021.
[8] Y. Seo, K. shik Shin, Hierarchical convolutional neural networks for fashion image classification, Expert Syst. Appl. 116 (2019) 328–339. http://www.sciencedirect.com/science/article/pii/S0957417418305992. https://doi.org/10.1016/j.eswa.2018.09.022.
[9] S. Wan, Y. Liang, Y. Zhang, Deep convolutional neural networks for diabetic retinopathy detection by image classification, Comput. Electr. Eng. 72 (2018) 274–282. http://www.sciencedirect.com/science/article/pii/S0045790618302556. https://doi.org/10.1016/j.compeleceng.2018.07.042.

[10] N. Sharma, V. Jain, A. Mishra, An analysis of convolutional neural networks for image classification, Procedia Comput. Sci. 132 (2018) 377–384. international Conference on Computational Intelligence and Data Science. http://www.sciencedirect.com/science/article/pii/S1877050918309335. https://doi.org/10.1016/j.procs.2018.05.198.

[11] X. Wang, R.C. Chen, F. Yan, Z. qiang Zeng, C. Hong, Semi-supervised adaptive feature analysis and its application for multimedia understanding, Multimed. Tools Appl. 77 (2017) 3083–3104.

[12] A. Baldominos, Y. Saez, P. Isasi, Evolutionary convolutional neural networks: an application to handwriting recognition, Neurocomputing 283 (2018) 38–52. http://www.sciencedirect.com/science/article/pii/S0925231217319112. https://doi.org/10.1016/j.neucom.2017.12.049.

[13] W. Tao, M.C. Leu, Z. Yin, American Sign Language alphabet recognition using Convolutional Neural Networks with multiview augmentation and inference fusion, Eng. Appl. Artif. Intell. 76 (2018) 202–213. http://www.sciencedirect.com/science/article/pii/S0952197618301921. https://doi.org/10.1016/j.engappai.2018.09.006.

[14] S.R. Kulkarni, B. Rajendran, Spiking neural networks for handwritten digit recognition—supervised learning and network optimization, Neural Networks 103 (2018) 118–127. http://www.sciencedirect.com/science/article/pii/S0893608018301126. https://doi.org/10.1016/j.neunet.2018.03.019.

[15] Z. Deng, L. Lei, H. Sun, H. Zou, S. Zhou, J. Zhao, An enhanced deep convolutional neural network for densely packed objects detection in remote sensing images, 2017 International Workshop on Remote Sensing with Intelligent Processing (RSIP), 2017. pp. 1–4. https://doi.org/10.1109/RSIP.2017.7958800.

[16] V.V. Khryashchev, A.A. Ostrovskaya, V.A. Pavlov, A.S. Semenov, Optimization of convolutional neural network for object recognition on satellite images, 2018Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO), 2018. pp. 1–5. https://doi.org/10.1109/SYNCHROINFO.2018.8457056.

[17] D. Jung, J. Son, S. Kim, Shot category detection based on object detection using convolutional neural networks, 2018 20th International Conference on Advanced Communication Technology (ICACT), 2018. pp. 36–39. https://doi.org/10.23919/ICACT.2018.8323638.

[18] L. Yang, L. Wang, S. Wu, Real-time object recognition algorithm based on deep convolutional neural network, 2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), 2018. pp. 331–335. https://doi.org/10.1109/ICCCBDA.2018.8386537.

[19] I.S. Ahmad, B. Boufama, P. Habashi, W. Anderson, T. Elamsy, Automatic license plate recognition: a comparative study, 2015 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), 2015. pp. 635–640. https://doi.org/10.1109/ISSPIT.2015.7394415.

[20] M. Zhang, W. Li, Q. Du, Diverse region-based CNN for hyperspectral image classification, IEEE Trans. Image Process. 27 (6) (2018) 2623–2634. https://doi.org/10.1109/TIP.2018.2809606.

[21] S. Ren, K. He, R.B. Girshick, J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, CoRR abs/1506.01497. (2015) http://arxiv.org/abs/1506.01497.

[22] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You Only Look Once: unified, real-time object detection, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. pp. 779–788. https://doi.org/10.1109/CVPR.2016.91.

[23] E. Dong, Y. Zhu, Y. Ji, S. Du, An improved convolution neural network for object detection using YOLOv2, 2018 IEEE International Conference on Mechatronics and Automation (ICMA), 2018. pp. 1184–1188. https://doi.org/10.1109/ICMA.2018.8484733.

[24] M.H. Putra, Z.M. Yussof, K.C. Lim, S.I. Salim, Convolutional neural network for person and car detection using YOLO framework, Telecommun. Electr. Comput. Eng. 10 (2018) 67–713. http://journal.utem.edu.my/index.php/jtec/article/view/3599/2491.

[25] H. Sheng, C. Li, Q. Wen, Z. Xiong, Real-time anti-interference location of vehicle license plates using high-definition video, IEEE Intell. Transp. Syst. Mag. 1 (4) (2009) 17–23. https://doi.org/10.1109/MITS.2010.935911.

[26] J. Jiao, Q. Ye, Q. Huang, A configurable method for multi-style license plate recognition, Pattern Recogn. 42 (3) (2009) 358–369. http://www.sciencedirect.com/science/article/pii/S0031320308003336. https://doi.org/10.1016/j.patcog.2008.08.016.

[27] N. Mathur, S. Mathur, D. Mathur, A novel approach to improve Sobel edge detector, Procedia Comput. Sci. 93 (2016) 431–438. proceedings of the 6th International Conference on Advances in Computing and Communications. http://www.sciencedirect.com/science/article/pii/S1877050916314727. https://doi.org/10.1016/j.procs.2016.07.230.

[28] M.R. Gupta, Y. Chen, Theory and use of the EM algorithm, Found. Trends Signal Process. 4 (3) (2011) 223–296. https://doi.org/10.1561/2000000034.

[29] K. Lin, H. Tang, T.S. Huang, Robust license plate detection using image saliency, 2010 IEEE International Conference on Image Processing, 2010. pp. 3945–3948. https://doi.org/10.1109/ICIP.2010.5649878.

[30] K. Deb, K. Jo, HSI color based vehicle license plate detection, 2008 International Conference on Control, Automation and Systems, 2008. pp. 687–691. https://doi.org/10.1109/ICCAS.2008.4694589.

[31] D. Zheng, Y. Zhao, J. Wang, An efficient method of license plate location, Pattern Recognition Letters 26 (15) (2005) 2431–2438. http://www.sciencedirect.com/science/article/pii/S0167865505001406. https://doi.org/10.1016/j.patrec.2005.04.014.

[32] Y. Qiu, M. Sun, W. Zhou, License plate extraction based on vertical edge detection and mathematical morphology, 2009 International Conference on Computational Intelligence and Software Engineering, 2009. pp. 1–5. https://doi.org/10.1109/CISE.2009.5364222.

[33] 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) (2010) 1627–1645. https://doi.org/10.1109/TPAMI.2009.167.

[34] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1, 2005. pp. 886–893. vol. 1. https://doi.org/10.1109/CVPR.2005.177.

[35] BBox label tool, accessed: 2018-06-12. https://github.com/puzzledqs/BBox-Label-Tool.

[36] S. Rasheed, A. Naeem, O. Ishaq, Automated number plate recognition using hough lines and template matching, vol. 1, 2012. pp. 199–203.