License Plate Recognition

Le Quang Minh; Ton Nu Quyen Mi

Advisor: Truong Hoang Vinh

November 12, 2021

Abstract: In latest years, the license plate recognition system has turned out to be an essential position withinside the improvement of society for vehicle management, investigation of stolen vehicles, and traffic tracking and control. License plate recognition has 3 stages: license plate localization, character segmentation, and character recognition. Although there have been many applications for surveillance systems, there are still many problems with license plate recognition, such as traffic jams due to a large number of license plates, unclear signs, tilt-able pictures, bad weather, and night shots. In this report, we are going to update an existing CNN model with a YOLOv4 detector according to two main steps: LP detection and Optical Character Recognition (OCR). It first retrieves the LP using YOLO and then passes the result to OCRNetwork for final character recognition. After many improvements, this model achieved an overall accuracy of 83.7%.

Keywords: license plate recognition (LPR), convolution neural networks (CNN), YOLOv4 detector, vehicle plate recognition

1 Introduction

Vehicles have been the favored mode of transportation for individuals going out as the industrialization process has progressed. To maintain order and safety in public metropolitan areas, license plate monitoring is becoming increasingly necessary (traffic, crossroads, crowded events, train stations, airports, etc.). It is easier to track down crooks in our culture. As a result, the duty of recognizing license plates has higher requirements as well.

First of all, we will have the definition of license plate recognition. License plate recognition (LPR) is a technology that allows law enforcement and crime prevention agencies to identify and recover stolen automobiles based on their license plates or vehicle numbers. Image processing, pattern recognition, and optical character recognition are among the technologies used in LPR systems. Fixed cameras, such as those located on highway overpasses or toll plazas, are used to collect photographs of license plates, while certain mobile devices may be able to record images of license plates as they pass by.

The first step of license plate identification is to precisely find the license plate in the image, and the second part is to recognize the characters on the positioned license plate. To improve the accuracy of license plate recognition technology, a lot of research has been done utilizing various methodologies such as color-based [1], edge detection-based [2], and template-based matching [3], vehicle tracking [4]. However, when the image background is too detailed or the image is blurred, identifying characters incorrectly, or impossible to distinguish at challenging image angles, all of the aforementioned methods have certain drawbacks. As a result, many research articles on license plate recognition using deep learning methods have been published. In this, the implementation of convolution neural networks to improve the accuracy of license plate identification technology has recently been the focus of research on vehicle number recognition [5].

The strategies described above are intended to improve the model's recognition ability. In this research project, The YOLO - YOLOv4 - detector is utilized as the main approach to train the license plate

identification model, as well as some rudimentary CNN methods in character recognition. The model is based on data collected from a variety of sources, including shots taken from various angles (frontal, inclined, oblique, from above, etc.) and in varied lighting conditions in various locations (indoor, outdoor light, dark, basement,...). The result will be calculated precisely with each accurate number in its exact spot once it has been detected. The goal of the project is to improve the overall training model's accuracy as well as the accuracy of correct 100 percent recognition.

2 Related Works

Several approaches of license plate detection were mentioned in Refs. Actually, color-based license plate recognition has become obsolete in recent research papers, but in [1] the deep learning method was used, the color recognition technique was improved by YOLOv2 and resulted in identification results. license plate form is CRNN based on license plate coding rules. The article [2] describes a method for recognizing vehicle license plates based on contour detectors. We employed geometric information about the contours as well as a mathematical picture model based on a two-dimensional discrete-valued Markov series with two states to detect VLPs. The new technique has the potential to greatly lower the computing cost of detecting and, hence, identifying license plate characters. And [3], after images are processed to improve image quality, they are converted to black and white binary images. Each pixel is represented by a binary number (bit) consisting of 0s and 1s. The Infrared sensor is then used to identify the object and pattern matching for license plate recognition. The Vehicle Tracking approach stated in paper [4] is a relatively recent deep learning-based solution. This study describes an end-to-end LPR approach that is based on result integration and makes effective use of timing-sequence information. The single-shot multibox detector (SSD) [6] is chosen as the method to locate license plates in the manuscript.

Above are some related articles on the issue of license plate recognition with many different approaches. But the most popular and being studied further are the methods that use CNN and focus on developing deep learning [7]. In [5] offers an efficient hierarchical methodology for license plate recognition systems that first identify vehicles using deep learning algorithms and then extracts license plates from identified vehicles to minimize plate detection false positives. Finally, in the last stage, we propose an LPR convolution neural network (LPRCNN) to enhance character identification in blurred and obscure pictures. This is also one of the well-researched methods developed for our paper.

3 Data Preparation

Over 13,000 photos were used in the study, separated into two groups: training (over 7,000 images) and testing (5,000 images).

On kaggle.com, the data from the training set is gathered. After cleaning the dataset (removing fully blurred, highly cropped, and duplicated photos), the images were aggregated into a dataset, labeled with the class "license plate," and then exported to yolo. The following is a list of the dataset's components:

- The majority of the images come from the parking management business Green Parking's vehicle surveillance camera in the parking basement (about over 3,000 photos).
- The Morrocan Vehicle Registration Plate dataset was used to generate 1,596 pictures.
- 433 photos from the dataset Car License Plate Detection
- In addition to supplementing the missing data such as the special angle of the license plate image (angle of inclination, diagonal, proximity of the screen, ...), image light (dark, blackened, blurred, ...), 500 images were photo in real-time with the phone
- The rest of the data is used with the data augmentation method to create and add to the dataset.

Figure 1: Some images in dataset

From the Green Parking company's data set, an independent test set of 5,000 photos is extracted. The dataset is also cleaned, and the XML file is exported with the label of each vehicle's license plate number.

4 Methods

The proposed approach is composed by two main steps: LP detection and Optical Character Recognition (OCR). Given an input image, the first module detects vehicles in the scene, You Only Look One network (YOLO) [8] searches for LPs. These positive and rectified detections are fed to an OCR Network for final character recognition.

4.1 LP detection

4.1.1 YOLO architecture

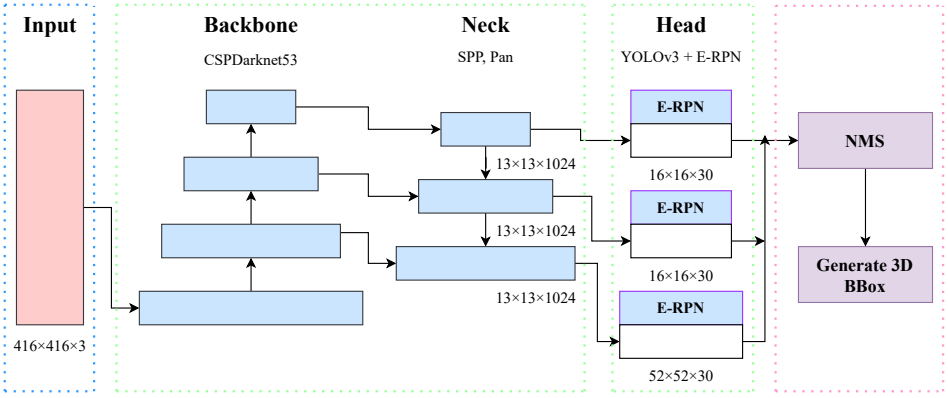


Figure 2: Normal-YOLOv4 network architecture

YOLOv4 Backbone Network - Feature Formation: In YOLOv4 [9] network implements CSP-Darknet53 [10] for the backbone network. This network consists of 53 consecutively connected convolutional layers, each of which is followed by batch normalization and a Leaky Relu activation. To reduce the size of the output after each convolution layer, the author downsamples with filters of size 2. This trick has the effect of minimizing the number of parameters for the model.

	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			

Figure 3: Layers in Darknet53

YOLOv4 Neck - Feature Aggregation: YOLOv4 [9] chooses PANet [11] for the feature aggregation of the network. They don't write much on the rationale for this decision, and since NAS-FPN [12] and BiFPN [13] were written concurrently, this is presumably an area of future research. Additionally, YOLOv4 [9] adds a SPP [14] block after CSPDarknet53 [10] to increase the receptive field and separate out the most important features from the backbone.

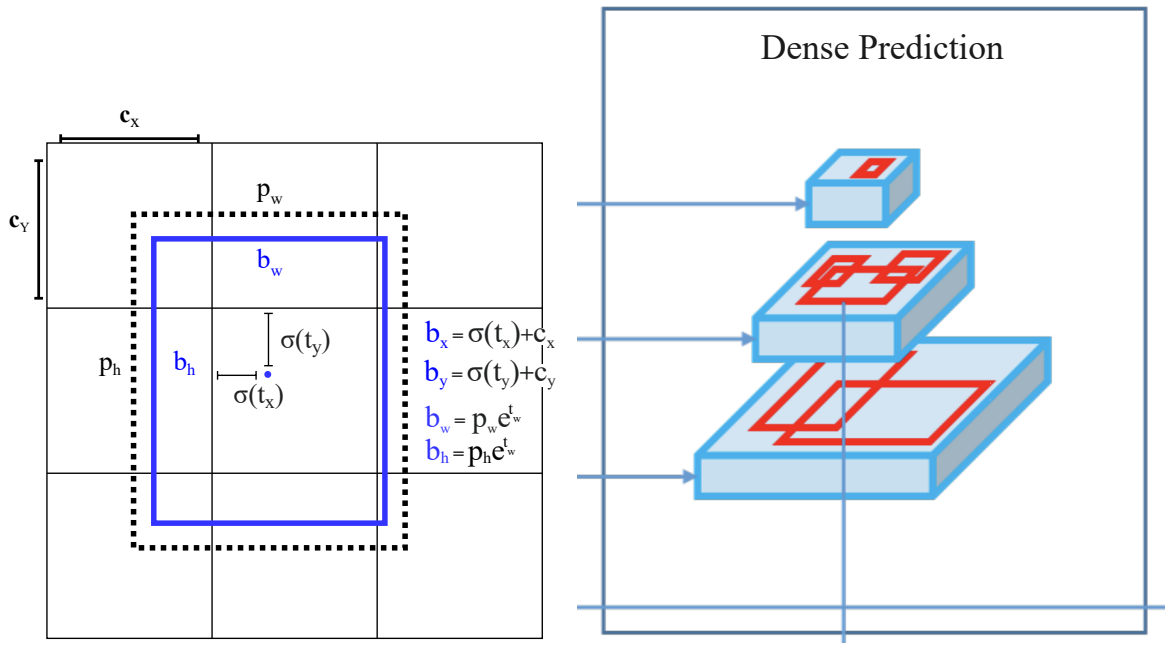
YOLOv4 Head - The Detection Step: YOLOv4 [9] deploys the same YOLO head as YOLOv3 [15] for detection with the anchor based detection steps, and three levels of detection granularity.

4.1.2 Output of YOLO

$$y^T = \left[p_0, \langle \underbrace{t_x, t_y, t_w, t_h}_{\text{bounding box}} \rangle, \langle \underbrace{p_1, p_2, \ldots, p_c}_{\text{scores of } c \text{ classes}} \rangle \right]. \tag{1}$$

In the equation (1):

- p_0 is the predicted probability that the object will appear in the bounding box.



(a) Formula to estimate bounding box from anchor box

(b) Layers in Dense Prediction

Figure 4: (a) is show what is output of YOLOv4 model, (b) The dense prediction is the final prediction which is composed of a vector containing the coordinates of the predicted bounding box (center, height, width), the confidence score of the prediction and the label.

- $\underbrace{\langle t_x, t_y, t_w, t_h \rangle}_{\text{bounding box}}$ help define bounding box . In there t_x, t_y are the coordinates of the center and t_w, t_h is the width and length of the bounding box.
- $\langle \underbrace{p_1, p_2, \ldots, p_c}_{\text{scores of } c \text{ classes}} \rangle$ is the vector of the predictive probability distribution of classes.

Understanding the output of YOLO is quite important for us to configure the correct parameters when training the model through open sources like Darknet. Thus the output will be determined by the number of classes according to the formula $(n\_class + 5)$. For example, if you train 80 classes, you will get 85 output. In case you apply 3 anchors/cell, the number of output parameters will be follow equation (2) and the filters is $85 \times 3 = 255$:

$$\text{filters} = (n\_\text{class} \ + 5) \times 3 \tag{2}$$

In the equation (2):

- filters is the number of output parameter.
- n_class is the number of classes in the model that you want to detect.

4.1.3 Loss function

YOLO's loss function is divided into 2 parts: $\mathcal{L}_{\text{loc}}$ (localization loss) measures the error of the bounding box and $\mathcal{L}_{\text{cls}}$ (confidence loss) measures the error of the probability distribution of classes.

$$\mathcal{L}_{\text{loc}} = \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + \left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \tag{3}$$

$$\mathcal{L}_{\text{cls}} = \underbrace{\sum_{i=0}^{S^2}\sum_{j=0}^{B}\left(\mathbb{1}_{ij}^{\text{obj}} + \lambda_{\text{noobj}}\left(1 - \mathbb{1}_{ij}^{\text{obj}}\right)\right)\left(C_{ij} - \hat{C}_{ij}\right)^2}_{\text{cell contain object}} + \underbrace{\sum_{i=0}^{S^2}\sum_{c\in\mathcal{C}}\mathbb{1}_{i}^{\text{obj}}\left(p_i(c) - \hat{p}_i(c)\right)^2}_{\text{probability distribution classes}} \tag{4}$$

$$\mathcal{L} = \mathcal{L}_{\text{loc}} + \mathcal{L}_{\text{cls}} \tag{5}$$

- $\mathbb{1}_{i}^{\text{obj}}$. The indicator function has a value of 0,1 to determine if cell i contains an object. Equals 1 if it contains the object and 0 if it doesn't.
- $\mathbb{1}_{ij}^{\text{obj}}$. Indicate whether the bounding box j of cell i is the bounding box of the predicted object or not?
- C_{ij}. Cell's confidence score i, P (contain object) * IoU (predict bbox, ground truth bbox).
- $\hat{C}_{ij}$ Predicted confidence score.
- $\mathcal{C}$: Gather all the layers.
- $p_i(c)$: Conditional probability, whether or not cell i contains an object of class $c \in \mathcal{C}$. $\hat{p}_i(c)$: Conditional probability prediction.

The above formula can be quite confusing for beginners at first. Let's simplify their purpose:

- $\mathcal{L}_{\text{loc}}$ is the loss function of the predicted bounding box relative to the actual.
- $\mathcal{L}_{\text{cls}}$ is the loss function of the probability distribution. Where the first sum is the loss of predicting whether there is an object in the cell or not? And the second sum is the loss of the probability distribution if there is an object in the cell.

In addition, to adjust the loss function penalty in case of wrong prediction of the bounding box, we using the adjustment coefficient λ_{coord} and we want to reduce the loss function in case the cell contains no objects. by the adjustment factor λ_{noobj}

4.1.4 Detection

License plates are intrinsically rectangular and planar objects, which are attached to vehicles for identification purposes. We decided to use YOLOv4-tiny for detect LP because of lack of hardware resources. Here we train YOLOv4 tiny on Google Colab, we train dataset with over 7,000 images and we split to 700 images for validation. Then, We will use the file yolov4-tiny.cfg to configure the training model. You download the file in open-source darknet to your computer and create file yolo-tinyv4-obj.cfg with the same content as in yolov4-tiny.cfg (or copy yolov4-tiny.cfg to yolo-tinyv4-obj.cfg) and change these line:

- Change line batch to batch equal to 64
- Change line subdivisions to subdivisions equal to 16
- Change line max_batches to (classes*2000, but not less than number of training images and not less than 6,000), f.e. max_batches equal to 6000 if you train for 3 classes. Here we change to 8000.
- Change line steps to 80% and 90% of max_batches, f.e. steps=6400,7200
- Change line classes equal to 80 to your number of objects here we change classes = 1
- Change filter number equal to 255 to filter equal to 18. This is the last layer of the base network. Thus, we have the output shape that varies with the quantity type according to Equation (2): $(n\_classes + 5) \times 3 = (1 + 5) \times 3 = 18$

That is the most important step when training YOLO model another step your can follow here. After training we the best weight have mAP is 97% (Our code).

- **IoU** (intersect over union) - average intersect over union of objects and detections for a certain threshold = 0.24
- **mAP** (mean average precision) - mean value of average precisions for each class, where average precision is average value of 11 points on PR-curve for each possible threshold (each probability of detection) for the same class (Precision-Recall in terms of PascalVOC, where Precision=TP/(TP+FP) and Recall=TP/(TP+FN))(Read more in page 11)

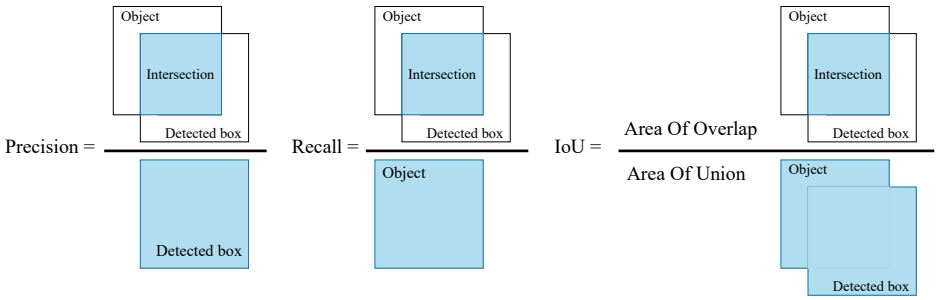


Figure 5: Fomula to calculate IoU

For each image, we manually annotated the 4 corners of the LP in the picture (sometimes more than one LP detect). A few annotated are show in figure 6 6.

Figure 6: Examples of the annotated LPs in the training dataset.

Given the reduced number of annotated images in the training dataset, the use of data augmentation is crucial. The following augmentation transforms are used:

- Reduce the sea size by adding a random size margin to the original image, then resize the image to the original image size.

- Change the brightness of the photo
- Rotation: a 3D rotation with randomly chosen angles is performed, to account for a wide range of camera setups

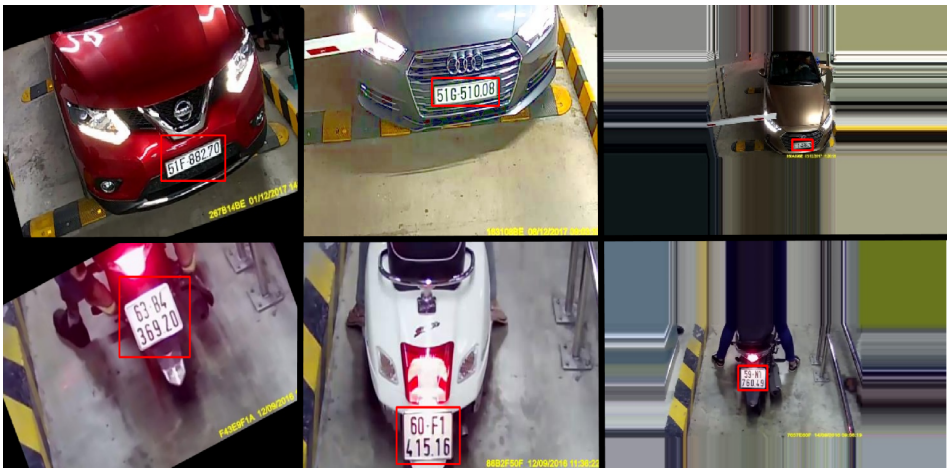


Figure 7: Different augmentations for the same sample. The red quadrilateral represents the transformed LP annotation.

4.2 Optical Character Recognition

4.2.1 Image segmentation

After detect LP, we crop the image with LP region:

Figure 8: Crop LP

We will convert the color image from BGR to HSV, Then we use **adaptive threshold** to highlight the parts we want to get (black), you can see in Fig 9.

Figure 9: LP after apply adaptive threshold

In Fig.10 to separate each character from the license plate, we use the algorithm **Connected components analysis(CCA)**

Figure 10: LP after using CCA algorithm

Next we use opencv to find all the rectangular shaped contours on the image and sort them left to right.

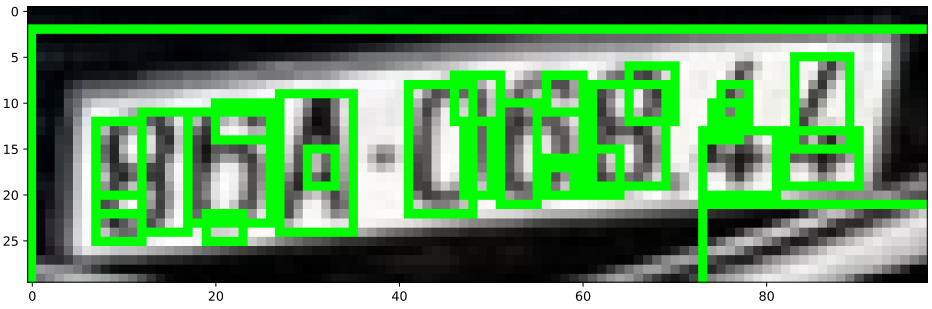


Figure 11: LP after draw contours

As you can see in Fig.11 this causes many contours to be found other than just the contours of each character within the license plate number. In order to filter out the unwanted regions we apply a couple parameters to be met in order to accept a contour. These parameters are just height and width ratios.

A couple other parameters on area of region etc are also placed. If using deep learning classification method, it is time consuming because the amount of noise is very large. We just need to tweak these parameters according to the data we can quickly remove most of the noise in a simple way, get an image of the characters on the desired license plate. Then we return to the form (28, 28, 1) which is the input input size of the CNN network in the next step.

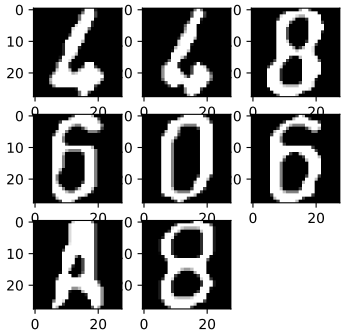


Figure 12: After extract contours of image

4.2.2 Characters Recognition

In Vietnam, the license plate accepts 31 characters including both letters and numbers (0-9). Due to the small number, in this step, we add a background class to use the CNN model to classify what is the character and what is the noise. So the total of classes to be classified is 32 (dataset we use in here get from [c])

input shape	$28 \times 28 \times 1$
conv1	(32, (3×3), relu)
conv2	(32, (3×3), relu)
pool1	(2×2) max pooling
drop1	0.25
conv3	(64, (3×3), relu)
conv4	(64, (3×3), relu)
pool2	(2×2) max pooling
drop2	0.25
conv5	(64, (3×3), relu)
conv6	(64, (3×3), relu)
pool3	(2×2) max pooling
drop3	0.25
FC1	(512, relu)
FC2	(32, softmax)

Table 1: CNN Network

In table 1, you can see that we built a simple CNN model to perform feature extraction and classification. This model is designed with increasing number of filters [32, 64, 64] so as to get as close to the

output as the feature space decreases but the number of learning ways (= number of filters) increases. Depending on the complexity of the data we can change. Finally we use a flatten layer and a softmax activation for classification.(Our code)

5 Results and Discussion

5.1 Evaluation Metrics

The dataset we use to evaluate is getting from here. We use 5,091 images and label to calculate accuracy. We follow the evaluation metric that has been widely used in LPR research [16]. Therefore, if only one of the consecutive characters is misclassified or not detected, it is treated as a failure case. We denote this metric as a recognition accuracy. We get 53% for this metric.

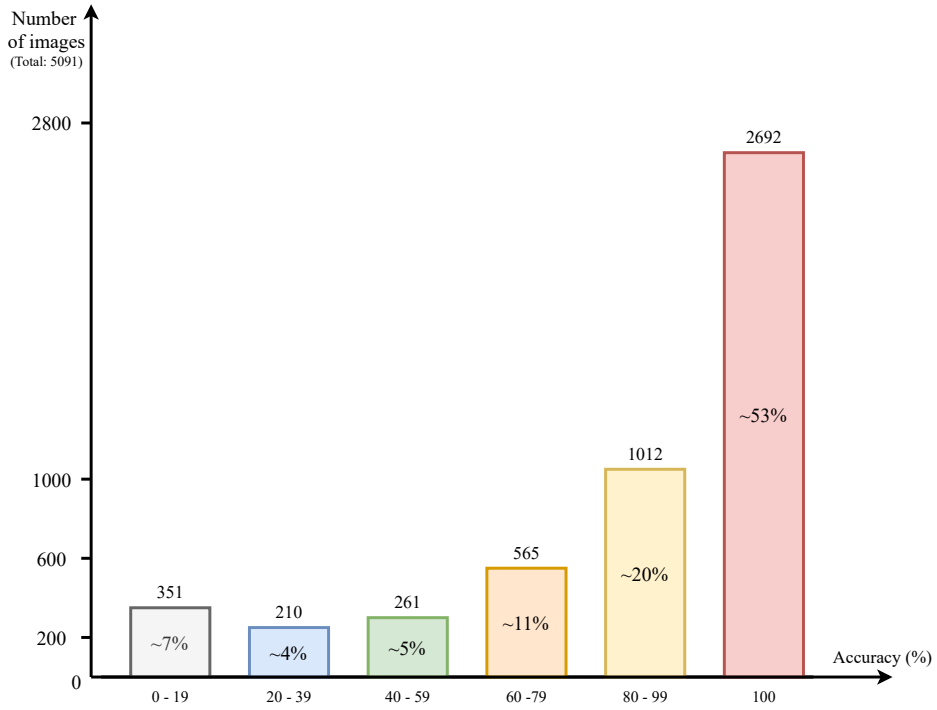


Figure 13: Bar chart of the accuracy of the results

We also calculate the average metric, that we compare predicted LP with true LP to find accuracy for each image but if only one of the consecutive characters is misclassified or not detected we also add them to sum then and divide it for all the image. The final result is 83.7%.

Image	Ground Truth	Predicted	Failure
	48A02866	8A02866	Mis-classification number 4
	51G10096	09651G0	Because the piture is too tilted, the recognition is wrong
	59B1-38559	598-38559	Wrong classification

Table 2: Error study on dataset

6 Conclusion

In this work, we presented a complete deep learning License Plates Recognition for Viet Nam Number plates. Our results indicate that the proposed approach gets the best performance for an image that is in good condition.
However, it still has some disadvantages:

- When the input image is angled too slanted, some characters will be mis-classified.
- Sometimes misidentified between 8 and B, 0 and D, 2 and Z
- Poor performance when the photo is too blurry

Appendix A. Project Plan management

Task Name	Priority	Owner	Start date	End day	Status	issues
Find documents	High	Both	06/09/2021	20/09/2021	Finished	
Review related papers	Medium	Quyen Mi	06/09/2021	28/09/2021	Finished	
Review and analyze public dataset	Low	Both	06/09/2021	22/10/2021	Finished	Aggregate from multiple datasets
Collect and label data	High	Quyen Mi	06/09/2021	22/10/2021	Finished	Collect and add many special photo angles
Evaluate potential method	Medium	Both	13/09/2021	1/10/2021	Finished	
Experiment	High	Quang Minh	06/09/2021	27/10/2021	Finished	Change the way to calculate accuracy
Compare results	Medium	Quang Minh	06/09/2021	27/10/2021	Finished	Medium performance efficiency
Writing report	Low	Quyen Mi	06/09/2021	10/11/2021	Finished	

Table 3: Project Plan management table

Appendix B. Source code & Data

References

[1] Sen Zhang, Shuai Chen, Jie Li, and Huichen Zhang. An improved vehicle-license plate recognition based on color clues and coding rules. In *2019 IEEE 4th International Conference on Image, Vision and Computing (ICIVC)*, pages 504–508, 2019.

[2] Elena Medvedeva, Igor Trubin, and Pavel Kasper. Vehicle license plate recognition based on edge detection. In *2020 26th Conference of Open Innovations Association (FRUCT)*, pages 291–296, 2020.

[3] Thidarat Pinthong, Worawut Yimyam, Narumol Chumuang, and Mahasak Ketcham. License plate tracking based on template matching technique. In *2018 18th International Symposium on Communications and Information Technologies (ISCIT)*, pages 299–303, 2018.

[4] Liping Zhu, Shang Wang, Chengyang Li, and Zhongguo Yang. License plate recognition in urban road based on vehicle tracking and result integration. *Journal of Intelligent Systems*, 29(1):1587–1597, 2020.

[5] Cheng-Hung Lin, Yong-Sin Lin, and Wei-Chen Liu. An efficient license plate recognition system using convolution neural networks. In *2018 IEEE International Conference on Applied System Invention (ICASI)*, pages 224–227, 2018.

[6] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing.

[7] Cheng-Hung Lin and Yi-Sin Sie. Two-stage license plate recognition system using deep learning. In *2019 8th International Conference on Innovation, Communication and Engineering (ICICE)*, pages 132–135, 2019.

[8] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[9] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection.

[10] Chien-Yao Wang, Hong-Yuan Mark Liao, I.-Hau Yeh, Yueh-Hua Wu, Ping-Yang Chen, and Jun-Wei Hsieh. CSPNet: A new backbone that can enhance learning capability of CNN.

[11] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation.

[12] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7029–7038, 2019.

[13] Mingxing Tan, Ruoming Pang, and Quoc V. Le. EfficientDet: Scalable and efficient object detection.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. 8691:346–361.

[15] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement.

[16] Gee-Sern Hsu, Jiun-Chang Chen, and Yu-Zu Chung. Application-oriented license plate recognition. *IEEE Transactions on Vehicular Technology*, 62(2):552–561, 2013.