# Deep Learning Based Framework for Iranian License Plate Detection and Recognition 

Mojtaba Shahidi Zandi • Roozbeh Rajabi

Received: date / Accepted: date


#### Abstract

License plate recognition systems have a very important role in many applications such as toll management, parking control, and traffic management. In this paper, a framework of deep convolutional neural networks is proposed for Iranian license plate recognition. The first CNN is the YOLOv3 network that detects the Iranian license plate in the input image while the second CNN is a Faster R-CNN that recognizes and classifies the characters in the detected license plate. A dataset of Iranian license plates consisting of ill-conditioned images also developed in this paper. The YOLOv3 network achieved $99.6 \% \mathrm{mAP}, 98.26 \%$ recall, $98.08 \%$ accuracy, and average detection speed is only 23 ms . Also, the Faster R-CNN network trained and tested on the developed dataset and achieved $98.97 \%$ recall, $99.9 \%$ precision, and $98.8 \%$ accuracy. The proposed system can recognize the license plate in challenging situations like unwanted data on the license plate. Comparing this system with other Iranian license plate recognition systems shows that it is Faster, more accurate and also this system can work in an open environment.


Keywords Iranian license plate • detection • recognition • convolutional neural network • YOLOv3 • Faster R-CNN

## 1 Introduction

Automatic License Plate Recognition (ALPR) systems identify vehicle license plates automatically and reduce human resources; so in the recent years has become more important. There are more than 1.2 billion vehicles in the world and over 19 million of these numbers are belong to Iran and it is enough reason for importance of developing more reliable ALPR systems in this country.

## R. Rajabi

Communications and Electronics Department, Faculty of Electrical and Computer Engineering, Qom University of Technology, Qom, Iran E-mail: rajabi@qut.ac.ir

ALPR systems play important role in smart cities such as parking lots, traffic surveillance, access control, toll payments, and so on 20 .

Traditional systems employ features that has been regarded by humans to represent the features of the images. Traditional systems require humandesigned models for recognition responses from raw input pixels. But systems based on deep learning techniques automatically select the features themselves and do not need humans to design features [38]. These systems learn low-level representations of the underlying data by modifying filters [21]. Deep learning has several methods and most common method in image processing field is Convolutional Neural Networks (CNNs) 40. CNNs are based on sharing weights and sparse connections that giving high learning potential 37. Two biggest challenges in CNNs there are that first of them is demand for large amounts of training samples and second one high computational cost [27. In recent years, CNNs have become immensely more powerful as have been discovered clever solutions to reduce the amount of training samples and computational cost [11]. Since the CNNs in many task within the field of visual recognition are state-of-the-art it is believed that also are state-of-the-art within ALPR [23].

The ALPR systems have a complex task in changing condition and with many variations [1. The camera maybe have different distance and angle with vehicle, the resolution of cameras are different, uneven lighting, reflection and refraction of light, big background interference, large angle incline, dust and moisture are other complexities. A good ALPR system in addition to working good in mentioned conditions, should also detect located plates in different parts of the image, and if there are more than one plate in the image, the system should locate all of them. Many of the systems claim accuracy more than $90 \%$ [5. However, in most cases claimed accuracy is incomparable to other systems because in each one systems different data sets are used and there is not an accepted universal data set. The complexity of the environment in recognizing license plates in different test sets will significantly impact the accuracy and makes direct comparisons of the accuracy without considering these complexities meaningless [16. Traditional algorithms often work well under certain circumstances but in open environment like uneven lighting, reflection and refraction of light, big background interference, large angle incline, dust and moisture fail to achieve satisfactory accuracy 39 .

ALPR systems often have tree main steps: license plate detection, license plate segmentation and license plate recognition [19. The license plate detection step get the input image then locate plates in the image. For this step five main approaches are proposed: 1- Edge-based approaches: these approaches detect license plate by knowing that the license plate is rectangular with a known aspect ratio 12,28. 2- Color-based approaches: these approaches detect license plates by exploiting the fact that license plates often has a different color compared to the background 4. 3- Texture-based approaches: these approaches detect license plates by use the unconventional pixel density distribution inside the license plate [2]. 4- Character-based approaches: theses approaches detect plate as strings of characters [22]. 5- Learning-based approaches: these
approaches use machine learning techniques to recognition characters based on features and now this approaches usually use CNNs [17,34, 25]. The license plate segmentation step segment the character of the license plate that located with previous step. For this step two main approaches are proposed: 1- Projection-based approaches: these approaches segmentation characters by knowing that characters and the background of the license plate have different colors that giving opposite values after binarization of the image 33]. 2-Pixel-connectivity-based approaches: these approaches segmentation characters by labeling all connected pixels in the binary image. The pixels having the same properties as the characters are considered part of the character [4]. The license plate recognition step recognize characters and for this step two main approaches are proposed: 1- Template matching approaches: these approaches recognition carachters by comparing the similarity of the character and a template [33,18]. 2- Learning-based approaches: these approaches use machine learning techniques to recognition characters based on features and now this approaches usually use CNNs [17, 34, 25, 6].

In real world applications, the data acquisition conditions are changing and traditional machine learning methods may fail in recognizing the license plates accurately. So in this paper we proposed a framework based on deep learning schemes to overcome these limitations. In this paper, Iranian License Plate Recognition System based on deep convolutional neural networks are proposed. This system consists of two main part. First part is license plate detection based on YOLOv3 31 network. In this part the image of Iranian car feed into the system and the YOLOv3 detect plate or plates from the image and crop them as output. Second part is license plate recognition based on the Faster R-CNN [32,36]. In this part the license plate that have been cropped from YOLOv3 are coming to Faster R-CNN and all numbers or alphabets in the plate are detected and recognition. The architecture of proposed system are shown in Fig. 1

The following steps have been taken to prepare the proposed system: 1preparing and labeling dataset for YOLOv3 (building YOLOv3 dataset) 2training YOLOv3 network 3- testing YOLOv3 network 4- preparing and labeling images for Faster R-CNN (building Faster R-CNN dataset) 5- training Faster R-CNN network 6- testing Faster R-CNN network

Since there is not any Iranian license plate dataset, we developed a dataset for Iranian license plate recognition in this paper. The images crawled on the web and the dataset consists of car images in different ill-conditioned situations. Total number of images in the dataset is 2316 .

The rest of the paper is organized as follows: section 2 explains the architecture of the proposed YOLOv3 for detection of license plates. Section 3 explains the architecture of the proposed Faster R-CNN for recognition of alphabets and numbers. Section 4 explains the gathered dataset for testing and training of these two networks. In section 5 experiments and results are explained and the proposed system has been compared with other systems and section 6 concludes the paper.


Fig. 1 Architecture of the proposed license plate Recognition system

## 2 License plate detection using YOLOv3

YOLO (2015) 29 was a CNN with one-shot architecture. The one-shot architectures processes the image with a single CNN without the need of region and object proposals so this architectures is simpler than R-CNN. YOLO mAP score on VOC 2007 dataset was 63.4 with a processing time of 20 ms per image. YOLOv2 (2016) [30] is an improved version of the YOLO network. YOLOv2 training with a multi-scale method, so it can run in different sizes and offer a trade-off between speed and accuracy [24]. The version that runs at 67 FPS achieves a mAP on VOC 2007, while a slower version that runs at 40 FPS achieves a 78.6 mAP . YOLOv3 (2017) 31] is the third version of YOLO [7] network. YOLOv3 uses a few tricks to improve training and increase performance like multi-scale predictions (prediction of small, medium amd large objects in image) and use a better backbone classifier (YOLOv3 uses Darknet-53 while YOLOv2 uses Darknet-19). YOLOv3-320 gives a MAP of 51.5 with 45 FPS on COCO dataset for IOU 0.5 and YOLOv3-608 gives a MAP of 57.9 with 20 FPS on COCO dataset for IOU 0.5 .

### 2.1 Darknet-53

YOLOv3 uses Darknet-53 for performing feature extraction 31 and 53 means this network has 53 convolutional layers. This new network is much more powerful than Darknet-19 (used in YOLOv2). Darknet-53 has similar performance to ResNet-152 but it is faster. Darknet-53 architecture is shown in Table 1

Table 1 Darknet-53

| Repeat | Type | Filters | Size | Output |
| :---: | :---: | :---: | :---: | :---: |
|  | Convolutional | 32 | $3 \times 3$ | $256 \times 256$ |
|  | Convolutional | 64 | $3 \times 3 / 2$ | $128 \times 128$ |
| $1 \times$ | Convolutional | 32 | $1 \times 1$ |  |
|  | Convolutional | 64 | $3 \times 3$ |  |
|  | Residual |  |  | $128 \times 128$ |
|  | Convolutional | 128 | $3 \times 3 / 2$ | $64 \times 64$ |
| $2 \times$ | Convolutional | 64 | $1 \times 1$ |  |
|  | Convolutional | 128 | $3 \times 3$ |  |
|  | Residual |  |  | $64 \times 64$ |
|  | Convolutional | 256 | $3 \times 3 / 2$ | $32 \times 32$ |
| $8 \times$ | Convolutional | 128 | $1 \times 1$ |  |
|  | Convolutional | 256 | $3 \times 3$ |  |
|  | Residual |  |  | $32 \times 32$ |
|  | Convolutional | 512 | $3 \times 3 / 2$ | $16 \times 16$ |
| $8 \times$ | Convolutional | 256 | $1 \times 1$ |  |
|  | Convolutional | 512 | $3 \times 3$ |  |
|  | Residual |  |  | $16 \times 16$ |
|  | Convolutional | 1024 | $3 \times 3 / 2$ | $8 \times 8$ |
| $4 \times$ | Convolutional | 512 | $1 \times 1$ | $8 \times 8$ |
|  | Convolutional | 1024 | $3 \times 3$ |  |
|  | Residual |  |  |  |
|  | Avgpool |  | Global |  |
|  | Connected |  | 1000 |  |
|  | Softmax |  |  |  |

### 2.2 YOLOv3 architecture

YOLOv3 31 is the third version of YOLO (You Only Lock Once) network. YOLOv3 use Darknet-53 for performing feature extraction. The Darknet-53 has 53 convolutional layers and in detection stage the YOLOv3 has 106 layers. The architecture of YOLOv3 have been show in Fig. 2.

The most salient feature of YOLOv3 compared with previous YOLO is that it makes detection at three different scales which down sampling the input image by 32,16 and 8 respectively [31. In YOLOv3 its eventual output is generated by applying a $1 \times 1$ kernel on feature map of three different size at three different places in network. The shape of detection kernels come from Eq. 1

$$
\begin{equation*}
1 \times 1 \times(B \times(5+C)) \tag{1}
\end{equation*}
$$

where $B$ is the number of bounding box that a cell on feature map can predict and C is the number of classes. In the proposed YOLOv3 just one class named


Fig. 2 Architecture of proposed YOLOv3 network. The input image has $320 \times 320$ pixels and YOLOv3 network detect license plate on input image with 3 scales in stride $32,16,8$ for large, medium and small license plate respectively.

License Plate is considered and $\mathrm{B}=3$ so the detection kernel size become $1 \times 1 \times 18[1 \times 1 \times(3 \times(5+1))]$. A grid cell architecture in feature map is shown in Fig. 3. Since in proposed LPDS the input image is $320 \times 320$, the detection give feature maps of $10 \times 10 \times 18,20 \times 20 \times 18$ and $40 \times 40 \times 18$. The $10 \times 10$ layers responsible of detection large objects, whereas $20 \times 20$ layers detect medium objects and $40 \times 40$ layers detect small objects. This LPDS use 9 anchors; three biggest anchors for large scale, next three for medium scale and the last three for the small scale. YOLOv3 at each scale can predict 3 boxes using 3 anchors, so this LPDS predict 6300 boxes in total.

## 3 License Plate Recognition with Faster R-CNN

After cropping the license plate from the input image, numbers and alphabets in the plate should be recognize separately. This is accomplished using Faster R-CNN 32. In the R-CNN family; after R-CNN and Fast R-CNN, the Faster R-CNN have been introduced. Region-based CNN (2014) [14] achieved a mAP score of 66 with a processing time of 20 second per image in VOC 2007 dataset.


Fig. 3 Architecture of a grid cell in feature map. $t_{x}, t_{y}, t_{w}, t_{h}$ are box coordinates that are center x , center y , box width and box height respectively. $P_{0}$ show probability that box are background and $P_{\text {License-plate }}$ show probability that box are consist of a license plate.

Fast R-CNN (2015) [13] is an improved and simplified version of the R-CNN model that the speed and accuracy have been increased. Fast R-CNN achieved a mAP score of 70.0 on the VOC 2007 with a processing time of 2 seconds per image, 10 times faster than R-CNN. Faster R-CNN (2016) [32] is improvement version of Fast R-CNN by changing the region proposer completely that results speed and accuracy improved. Faster R-CNN achieved a mAP score of 73.2 with a processing time of 140 ms per image. This is over 10 times faster than Fast R-CNN and over 100 times faster than R-CNN. In fainal by changing the underlying architecture from VGGnet to the ResNet mAP score of 76.4 is achieved.

The Faster R-CNN architecture consist of RPN as a region proposal algorithm and the Fast R-CNN as a detector network. In proposed Faster R-CNN a ResNet101 [15] are used as Feature extractor. Faster R-CNN architecture has been shown in Fig. 5

### 3.1 ResNet

ResNet [15] won the ILSVRC 2015 and represents a new revolutionizing way of building CNNs, called residual CNNs. This was achieved by using skip con-
nections. A skip connection is a connection used by the input signal to bypass a number of layers. With skip connection technique, CNNs with over 1000 layers are trainable. In the field of general object classification residual CNNs remain the state-of-the-art (June 2017) as in the ILSVRC 2016 no revolutionary architectures were submitted. ResNet family architecture is shown in Fig. 4 15.

| layer name | output size | 18-layer | 34-layer | 50-layer | 101-layer | 152-layer |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| conv1 | $112 \times 112$ | $7 \times 7,64$, stride 2 |  |  |  |  |
|  |  | $3 \times 3$ max pool, stride 2 |  |  |  |  |
| conv2.x | $56 \times 56$ | $\left[\begin{array}{l}3 \times 3,64 \\ 3 \times 3,64\end{array}\right] \times 2$ | $\left[\begin{array}{l}3 \times 3,64 \\ 3 \times 3,64\end{array}\right] \times 3$ | $\left[\begin{array}{c}1 \times 1,64 \\ 3 \times 3,64 \\ 1 \times 1,256\end{array}\right] \times 3$ | $\left[\begin{array}{c}1 \times 1,64 \\ 3 \times 3,64 \\ 1 \times 1,256\end{array}\right] \times 3$ | $\left[\begin{array}{c}1 \times 1,64 \\ 3 \times 3,64 \\ 1 \times 1,256\end{array}\right] \times 3$ |
| conv3_x | $28 \times 28$ | $\left[\begin{array}{l}3 \times 3,128 \\ 3 \times 3,128\end{array}\right] \times 2$ | $\left[\begin{array}{l}3 \times 3,128 \\ 3 \times 3,128\end{array}\right] \times 4$ | $\left[\begin{array}{l}1 \times 1,128 \\ 3 \times 3,128 \\ 1 \times 1,512\end{array}\right] \times 4$ | $\left[\begin{array}{l}1 \times 1,128 \\ 3 \times 3,128 \\ 1 \times 1,512\end{array}\right] \times 4$ | $\left[\begin{array}{l}1 \times 1,128 \\ 3 \times 3,128 \\ 1 \times 1,512\end{array}\right] \times 8$ |
| conv4.x | $14 \times 14$ | $\left[\begin{array}{l}3 \times 3,256 \\ 3 \times 3,256\end{array}\right] \times 2$ | $\left[\begin{array}{l}3 \times 3,256 \\ 3 \times 3,256\end{array}\right] \times 6$ | $\left[\begin{array}{c}1 \times 1,256 \\ 3 \times 3,256 \\ 1 \times 1,1024\end{array}\right] \times 6$ | $\left[\begin{array}{c}1 \times 1,256 \\ 3 \times 3,256 \\ 1 \times 1,1024\end{array}\right] \times 23$ | $\left[\begin{array}{c}1 \times 1,256 \\ 3 \times 3,256 \\ 1 \times 1,1024\end{array}\right] \times 36$ |
| conv5-x | $7 \times 7$ | $\left[\begin{array}{l}3 \times 3,512 \\ 3 \times 3,512\end{array}\right] \times 2$ | $\left[\begin{array}{l}3 \times 3,512 \\ 3 \times 3,512\end{array}\right] \times 3$ | $\left[\begin{array}{c}1 \times 1,512 \\ 3 \times 3,512 \\ 1 \times 1,2048\end{array}\right] \times 3$ | $\left[\begin{array}{c}1 \times 1,512 \\ 3 \times 3,512 \\ 1 \times 1,2048\end{array}\right] \times 3$ | $\left[\begin{array}{c}1 \times 1,512 \\ 3 \times 3,512 \\ 1 \times 1,2048\end{array}\right] \times 3$ |
|  | $1 \times 1$ | average pool, $1000-\mathrm{d} \mathrm{fc}$, |  |  |  |  |
| FLOPs |  | $1.8 \times 10^{9}$ | $3.6 \times 10^{9}$ | $3.8 \times 10^{9}$ | $7.6 \times 10^{9}$ | $11.3 \times 10^{9}$ |

Fig. 4 ResNet family architecture [15. In proposed system ResNet-101 that is shown in column 3 used.

### 3.2 RPN (Region Proposal Network)

Just before feeding into RPN, image is resized to size of 128 by 640 pixels. First in the RPN, input image being fed into ResNet-101 (backbone convolutional neural network). As the stride in proposed system are 16 the output feature of backbone network is a feature map with $8 \times 40 \times 512$ size (this means that two consecutive pixels in the backbone (ResNet-101) output features correspond to 16 pixels in the input image). The Network, for every point in the feature map should learn whether an object is present in the input image at its corresponding location and estimate its size. This is done by placing a set of anchors on input image for each location on the output feature map for the backbone network. This anchors indicate possible objects in various size and aspects ratios at this location. In the proposed system the anchors uses 4 scale of box area and 3 aspect ratio so for every location in feature map 12 $(4 \times 3)$ location in input image is being correspond. So the output feature map corresponding $3840(8 \times 40 \times 12)$ anchors in total.


Fig. 5 Faster R-CNN architecture. The Faster R-CNN architecture consists of RPN as a region proposal algorithm and the Fast R-CNN as a detector network. In the proposed Faster R-CNN a ResNet101 are used as a feature extractor.

In RPN and after backbone network, there is a convolutional layer with $3 \times 3$ filters, 1 padding and 512 output channels. The output channels is connected to two $1 \times 1$ convolutional layers for classification and box-regression (this classification just determine that the box is object or not) [32].

### 3.3 Fast R-CNN (as a detector for Faster R-CNN)

The Fast R-CNN detector also consist a CNN backbone (ResNet101), an ROI pooling layer and fully connected layers that followed by two branches for classification and bounding box regression. In this section first the input image passed through backbone CNN (ResNet101) for feature map extraction of size $8 \times 40 \times 512$ (this is same as extracting feature map in RPN and just weight sharing between RPN backbone and Fast R-CNN backbone). In the next stage, the bonding box proposals from the RPN are used to pool features from the backbone feature map. This is done by the region of interest (ROI) pooling layer. The ROI pooling layer works by: a) Taking the region corresponding to a proposal from the backbone feature map. b) Dividing the region into a fixed number of sub-windows. c) Performing max-pooling to over this sub-windows to give fixed size output. The output of ROI pooling layer is $\mathrm{N} \times 7 \times 7 \times 512$ where N is the number of proposals from the region proposal algorithm. In the
final step features pass through two fully connected layers, and then fed into sibling classification and regression branches. The classification layer has C unit for each of the classes (in this system C is equal to 25 ). The features pass through a softmax layer to get the classification scores. And the regression layer coefficient are used to improve the predicted bounding box. That is, all the classes have individual regressors with 4 parameters each corresponding to $\mathrm{C} \times 4$ output units in the regression layers [32].

## 4 Datasets

After choosing and setting system architecture, to adjust the weights and reducing classification error, the systems should be trained [8]. This is done using transfer learning method that retrains the pretrained CNN with the new dataset without changing the architecture or reinitializing the weights [10,26]. As in the proposed system there are two networks, for training this system two datasets have been created; first training dataset for training YOLOv3 network and second one for training Faster R-CNN network. After training these two networks the system are ready and its performance in challenging situation should be tested. This is done by two other dataset ; first testing dataset for testing YOLOv3 network and second testing dataset for testing Faster R-CNN network. All images are gathered from the public access photos of the internet and databases will be publicly available for academic research after publishment of the paper.

### 4.1 YOLOv3 Dataset

### 4.1.1 YOLOv3 Training dataset

This dataset used for training YOLOv3 network and has very important role in overall system accuracy. Choosing its images in open environment, trains the system for work in challenging situation. So prepared dataset consists of 2316 Iranian car images and this images captured from Iranian car in open environment like in reflection and refraction of light, big background interference, large angle incline, dust, moisture, day, night, snowy and rainy weather, two or more cars in the pictures and so on. These images have been labeled and finally used for training the YOLOv3. This dataset is for detection and cropping plate from Iranian car image with YOLOv3 and just have one output class named License Plate. In Fig. 6 some of the images used for training YOLOv3 with their labels have been shown. Each row of label, point to a license plate in a picture. The first number points to the output class that zero means license plate. The next four numbers point to normalized center x, center y , width and height respectively.


Fig. 6 Some images of YOLOv3 training dataset and their labels in below of each images, that each row of label, point to a license plate in a picture. The first number points to its class label that zero means license plate. The next four numbers point to normalized center x , center y , width and height respectively.

### 4.1.2 YOLOv3 testing dataset

This dataset is for testing the trained YOLOv3 network and to calculate its performance. Easily with a simple dataset can reach to a very high accuracy but for an exact accuracy this dataset also should consists of images in challenging environment. So a dataset consists of 512 Iranian car images for testing YOLOv3 network and calculating its performance prepared.

### 4.2 Faster R-CNN Dataset

### 4.2.1 Faster $R-C N N$ training dataset

The second dataset has been prepared for training Faster R-CNN network. For working system in challenging situations its images should be in challenging environment conditions. So this dataset consists of 1643 Iranian license plate images that captured from proposed YOLOv3 system. As every Iranian license plate consist of 7 numbers and 1 alphabet in total, for training Faster R-CNN, 13144 objects have been labeled. This dataset consist of 25 classes, 10 classes for numbers and 15 classes for alphabets. In Fig. 7 some plates that have been labeled for training Faster R-CNN network is demonstrated.


Fig. 7 Some images of Faster R-CNN training dataset.

### 4.2.2 Faster $R-C N N$ testing dataset

This dataset is used for testing the Faster R-CNN network and calculating its performance. Also this dataset should be in challenging environment situations. The dataset consists of 517 license plates images and 4136 objects.

## 5 Experiments and results

Now the systems are trained with training datasets and their performance calculated by running system on test datasets. Precision, recall and accuracy are the three main criteria for calculation of network performance. These criteria are calculated with equations 2, 3, and 4 respectively.

$$
\begin{align*}
& \text { Precision }=\frac{T P}{T P+F P}=\frac{T P}{\text { AllDetection }}  \tag{2}\\
& \text { Recall }=\frac{T P}{T P+F N}=\frac{T P}{\text { AllGroundTruths }} \tag{3}
\end{align*}
$$



Fig. 8 Examples of license plate detection and character detection and recognition using the proposed system in challenging conditions.

$$
\begin{equation*}
\text { Accuracy }=\frac{T P+T N}{T P+T N+F P+F N} \tag{4}
\end{equation*}
$$

where TP means true positive or correct detection, FP means false positive or wrong detection and FN means false negative or ground truth not detected and finally the TN means that the result should be negative and is negative. In Fig. 8some examples of license plate detection and character detection and recognition using the final proposed system in challenging situations is shown.

### 5.1 YOLOv3 results

The system that is used for training this network is an ASUS notebook with CPU core i7 HQ, GPU GeForce 840M and RAM 8GB. This network trained in Linux operating system using Python and TensorFlow. The proposed YOLOv3 has been trained with 2316 Iranian car images that have been labeled before. The input images have been resized to $320 \times 320$ pixels at the network entrance. Finally after training, the system have been tasted with 512 Iranian car images and the results have been summarized in table. 2.

Table 2 Proposed YOLOv3 test results.

| System | Average <br> Detection <br> Speed | TP | FP | FN | Recall | Precision | Average <br> IOU | mAP |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| YOLOv3 | 0.23 | 511 | 1 | 9 | $98.26 \%$ | $99.8 \%$ | $81.67 \%$ | $99.6 \%$ |



Fig. 9 RPN Classification and localization loss. Left plot shows localization loss and right plot shows objectness loss. Pale plots show original losses while bold plots show smoothed losses with smoothing factor equals to 0.6.

### 5.2 Faster R-CNN results

The same ASUS notebook system and same software has been used for training Faster R-CNN network. The proposed Faster R-CNN network trained with 1643 Iranian license plate images that have 13144 objects (numbers and alphabets) that have been labeled before. The input plate images resized to $128 \times 640$ pixels at the network entrance. Finally after training, the system have been tested with 517 plates and 4103 objects (numbers and alphabets). As mentioned before, Faster R-CNN have two networks RPN and Box-Classifier (Fast R-CNN detector), RPN model has two output one is for classification whether it is an object or not and the other one is for bounding box coordinates regression. In Fig. 9 two chart is shown that one is for objectness loss and the other one is for localization loss. As shown in Fig. 9, it learned fast in 1350 epochs, then the learning rate became slower. At final after 6597 epochs the localization loss come down to $0.008567 \%$ and objectness-loss become $0.00077695 \%$.

Also Box-Classifier model has two output. One is for classification that determine each box belongs to which class and the other one is for bounding box coordinates regression. In the Fig. 10, two charts are shown that one is for classification loss and the other one is localization loss. At final after 6597 epoch the localization loss come down to $0.08873 \%$ and classification-loss reaches $0.08292 \%$.


Fig. 10 Faster R-CNN object detector (box-classifier) loss. Left plot shows box classification loss and right plot shows localization loss. Pale plots show original losses while bold plots show smoothed losses with smoothing factor equals to 0.6.

Now in Fig. 11 total loss is shown, total loss is sum of RPN losses and box classifier losses. It can be seen that after the 6600 epochs total loss come down to $0.18 \%$.


Faster R-CNN total loss

Fig. 11 Faster R-CNN total loss. Pale plots show original losses while bold plots show smoothed losses with smoothing factor equals to 0.6.

Again after training; the system have been tasted with 517 plates and 4103 objects(numbers and alphabets) so each Faster R-CNN class results and total Faster R-CNN results has been calculated and show in Table. 3 and Table. 4 .
5.3 Comparing proposed system with other systems

In the subsection 5.15.2 experiments and results about the YOLOv3 and Faster R-CNN separately explained and the results after training and testing this

Table 3 Faster R-CNN classes results

| Number or Alphabets | - | 1 | r | $r$ | ${ }^{+}$ | Q | 9 | v | $\wedge$ | 9 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| TP | 134 | 454 | 476 | 373 | 368 | 322 | 330 | 322 | 409 | 367 |
| FN | 7 | 2 | 2 | 3 | 3 | 1 | 5 | 2 | 4 | 5 |
| FP | 1 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Acuuracy | 94.3\% | 99.1\% | 99.5\% | 99.2\% | 98.9\% | 99.6\% | 98.5\% | 98.3\% | 99.0\% | 98.6\% |
| Number or Alphabet | ب | $\because$ | ج | $\bigcirc$ | س | $ص$ | b | $\varepsilon$ | ق | 」 |
| TP | 22 | 6 | 34 | 33 | 25 | 63 | 24 | 30 | 46 | 58 |
| FN | 1 | 0 | 2 | 0 | 0 | 2 | 0 | 0 | 0 | 0 |
| FP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Acuuracy | 95.6\% | 100\% | 94.4\% | 100\% | 100\% | 96.9\% | 100\% | 100\% | 100\% | 100\% |
| Number or Alphabet | ค | ن | و | * | * | * | * | * | * | * |
| TP | 45 | 38 | 21 | * | * | * | * | * | * | * |
| FN | 0 | 0 | 1 | * | * | * | * | * | * | * |
| FP | 0 | 0 | 0 | * | * | * | * | * | * | * |
| Accuracy | 100\% | 100\% | 95.4\% | * | * | * | * | * | * | * |

Table 4 Faster R-CNN totall results

| System | TP | FP | FN | Recall | Precision | Acuuracy |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Faster R-CNN | 4061 | 4 | 42 | $98.97 \%$ | $99.9 \%$ | $98.86 \%$ |

two networks achieved accuracy of $98.08 \%$ for YOLOv3 network and $98.88 \%$ for Faster R-CNN network. As mentioned this proposed system made with a YOLOv3 network for detection Iranian license plate from the input image and a Faster R-CNN network for detection numbers and alphabets from cropped license plate and recognition each of them separately. For training YOLOv3 a dataset consist of 2316 images of Iranian car images is used and these images captured in open environment and with different resolutions. The system for training YOLOv3 was an ASUS notebook with CPU core i7 HQ, GPU GeForce 840M and RAM 8GB. After training this system was tasted with 512 Iranian car image. In table 5 the detection part of the proposed method is compared with some other license plate detection systems. Results show that the proposed YOLOv3 network perform better in comparison with other systems.

For training Faster R-CNN a dataset consists of 1643 images of Iranian License plate that have been already cropped with YOLOv3 are used. This dataset consists of 13144 objects in total. After training, this system was tested with 517 Iranian license plate images and total loss come down to $0.18 \%$. In Table. 6 the proposed method compared with some Iranian license plate recognition systems. As it can be seen from the results, the proposde Faster R-CNN network outperforms other systems.

Table 5 Comparing the proposed detection system with other systems.

| Method | Accuracy | Reference |
| :--- | :--- | :--- |
| YOLOv3 | $98.08 \%$ | Proposed |
| Color feature | $96.8 \%$ | $[4$ |
| Horizontal projection | $88 \%$ | $[33$ |
| Horizontal projection | $97.3 \%$ | $[18$ |
| Edge detection with connected compo- <br> nent | $91 \%$ | $[9$ |
| Vertical edge and morphological | $95.2 \%$ | $[3$ |

Table 6 Comparing the proposed recognition system with other systems.

| Method | Accuracy | Reference |
| :--- | :--- | :--- |
| Faster R-CNN | $98.86 \%$ | Proposed |
| Decision tree and SVM | $94.4 \%$ | $[4$ |
| Template Matching | $96.02 \%$ | $[33$ |
| Template Matching | $93 \%$ | $[18$ |
| Hybrid KNN and SVM | $97.03 \%$ | $[35$ |
| CNN | $97 \%$ | 9 |

### 5.4 Performance in presence of noise

In this subsection, we have analyzed the performance of the proposed framework in presence of noise. To this end, Gaussian noise ( $S N R=30 \mathrm{~dB}$ ) is added to the dataset images. Without adding any noise filtering techniques, results show that the proposed framework can still recognize the license plates effectively. Table. 7 shows the comparison results for two cases: noisy and noiseless data.

Table 7 Performance of the proposed framework in presence of Gaussian noise.

| Method | Accuracy | Data |
| :--- | :--- | :--- |
| Faster R-CNN | $97.24 \%$ | Noisy |
| YOLOv3 | $96.37 \%$ | Noisy |
| Faster R-CNN | $98.86 \%$ | Noiseless |
| YOLOv3 | $98.08 \%$ | Noiseless |

## 6 Conclusion

In this paper a license plate recognition system based on deep convolutional neural networks is proposed. In first main part a YOLOv3 network has been used for detection of license plate from input image. YOLOv3 uses Darknet53
as a feature extractor. This network made the detection part real time also this system succeeded to catch an accuracy up to $98 \%$. In the second main part a Faster R-CNN has been used for recognition of the plates that are cropped from the first network. Faster R-CNN uses ResNET101 as a feature extractor. This system also reaches $98.86 \%$ accuracy. Also in this paper two datasets for training the detection and recognition parts of the proposed method has been gathered and labeled. The experimental results show that the proposed system has reached high performance on recognition speed and accuracy in comparison with other Iranian license plate recognition systems, which can fully meet the needs of the practical applications.

## References

1. AlyanNezhadi, M.M., Hashemi, S.M.R., Abolghasemi, V.: License plate detection in complex scenes based on fusion of gaussian filtering and bayesian network. In: 2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI), pp. 0022-0026. IEEE (2017)
2. Anagnostopoulos, C.N.E., Anagnostopoulos, I.E., Loumos, V., Kayafas, E.: A license plate-recognition algorithm for intelligent transportation system applications. IEEE Transactions on Intelligent transportation systems 7(3), 377-392 (2006)
3. Ashoori-Lalimi, M., Ghofrani, S.: An efficient method for vehicle license plate detection in complex scenes. Circuits and Systems 2(4), 320-325 (2011)
4. Ashtari, A.H., Nordin, M.J., Fathy, M.: An iranian license plate recognition system based on color features. IEEE transactions on intelligent transportation systems 15(4), 1690-1705 (2014)
5. Asif, M.R., Qi, C., Wang, T., Fareed, M.S., Khan, S.: License plate detection for multinational vehicles - a generalized approach. Multimedia Tools and Applications 78(24), 35585-35606 (2019). DOI 10.1007/s11042-019-08199-4. URL https://doi.org/10. 1007/s11042-019-08199-4
6. Bulan, O., Kozitsky, V., Ramesh, P., Shreve, M.: Segmentation-and annotation-free license plate recognition with deep localization and failure identification. IEEE Transactions on Intelligent Transportation Systems 18(9), 2351-2363 (2017)
7. Chen, R.C., et al.: Automatic license plate recognition via sliding-window darknet-yolo deep learning. Image and Vision Computing 87, 47-56 (2019)
8. Chowdhury, P.N., Shivakumara, P., Pal, U., Lu, T., Blumenstein, M.: A new augmentation-based method for text detection in night and day license plate images. Multimedia Tools and Applications 79(43), 33303-33330 (2020). DOI 10.1007/ s11042-020-09681-0. URL https://doi.org/10.1007/s11042-020-09681-0
9. Dashtban, M.H., Dashtban, Z., Bevrani, H.: A novel approach for vehicle license plate localization and recognition. International Journal of Computer Applications 26(11), 22-30 (2011)
10. Dey, B., Kundu, M.K.: Turning video into traffic data - an application to urban intersection analysis using transfer learning. IET Image Processing 13(4), 673-679 (2019). DOI 10.1049/iet-ipr.2018.5985
11. Estebsari, A., Rajabi, R.: Single residential load forecasting using deep learning and image encoding techniques. Electronics 9(1), 68 (2020)
12. Faradji, F., Rezaie, A.H., Ziaratban, M.: A morphological-based license plate location. In: 2007 IEEE International Conference on Image Processing, vol. 1, pp. I-57. IEEE (2007)
13. Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp. 1440-1448 (2015)
14. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580-587 (2014)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778 (2016)
16. Islam, R., Islam, M.R., Talukder, K.H.: An efficient method for extraction and recognition of bangla characters from vehicle license plates. Multimedia Tools and Applications 79(27), 20107-20132 (2020). DOI 10.1007/s11042-020-08629-8. URL https: //doi.org/10.1007/s11042-020-08629-8
17. Jørgensen, H.: Automatic license plate recognition using deep learning techniques. Master's thesis, NTNU (2017)
18. Kasaei, S.H., Kasaei, S.M., Kasaei, S.A.: New morphology-based method for robustiranian car plate detection and recognition. International Journal of Computer Theory and Engineering 2(2), 264 (2010)
19. Khan, M.A., Sharif, M., Javed, M.Y., Akram, T., Yasmin, M., Saba, T.: License number plate recognition system using entropy-based features selection approach with svm. IET Image Processing 12(2), 200-209 (2018). DOI 10.1049/iet-ipr.2017.0368
20. Khinchi, M., Agarwal, C.: A review on automatic number plate recognition technology and methods. In: 2019 International Conference on Intelligent Sustainable Systems (ICISS), pp. 363-366. IEEE (2019)
21. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature 521(7553), 436-444 (2015)
22. Li, H., Shen, C.: Reading car license plates using deep convolutional neural networks and lstms. arXiv preprint arXiv:1601.05610 (2016)
23. Lu, Q., Liu, Y., Huang, J., Yuan, X., Hu, Q.: License plate detection and recognition using hierarchical feature layers from cnn. Multimedia Tools and Applications 78(11), 15665-15680 (2019). DOI 10.1007/s11042-018-6889-1. URLhttps://doi.org/10.1007/ s11042-018-6889-1
24. Min, W., Li, X., Wang, Q., Zeng, Q., Liao, Y.: New approach to vehicle license plate location based on new model yolo-l and plate pre-identification. IET Image Processing 13(7), 1041-1049 (2019). DOI 10.1049/iet-ipr. 2018.6449
25. Polishetty, R., Roopaei, M., Rad, P.: A next-generation secure cloud-based deep learning license plate recognition for smart cities. In: 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 286-293. IEEE (2016)
26. Pramanik, R., Bag, S.: Segmentation-based recognition system for handwritten bangla and devanagari words using conventional classification and transfer learning. IET Image Processing 14(5), 959-972 (2020). DOI 10.1049/iet-ipr.2019.0208
27. Rajabi, R., Estebsari, A.: Deep learning based forecasting of individual residential loads using recurrence plots. In: 2019 IEEE Milan PowerTech, pp. 1-5. IEEE (2019)
28. Rasheed, S., Naeem, A., Ishaq, O.: Automated number plate recognition using hough lines and template matching. In: Proceedings of the World Congress on Engineering and Computer Science, vol. 1, pp. 24-26 (2012)
29. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, realtime object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788 (2016)
30. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 7263-7271 (2017)
31. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018)
32. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems, pp. 91-99 (2015)
33. Salahshoor, M., Broumandnia, A., Rastgarpour, M.: Application of intelligent systems for iranian license plate recognition. In: 2014 Iranian Conference on Intelligent Systems (ICIS), pp. 1-6. IEEE (2014)
34. Sun, H., Fu, M., Abdussalam, A., Huang, Z., Sun, S., Wang, W.: License plate detection and recognition based on the yolo detector and crnn-12. In: International Conference On Signal And Information Processing, Networking And Computers, pp. 66-74. Springer (2018)
35. Tabrizi, S.S., Cavus, N.: A hybrid knn-svm model for iranian license plate recognition. Procedia Computer Science 102, 588-594 (2016)
36. Tang, L., Gao, C., Chen, X., Zhao, Y.: Pose detection in complex classroom environment based on improved faster r-cnn. IET Image Processing 13(3), 451-457 (2019). DOI 10.1049/iet-ipr.2018.5905
37. Wang, W., Yang, J., Chen, M., Wang, P.: A light cnn for end-to-end car license plates detection and recognition. IEEE Access 7, 173875-173883 (2019)
38. Zang, D., Chai, Z., Zhang, J., Zhang, D., Cheng, J.: Vehicle license plate recognition using visual attention model and deep learning. Journal of Electronic Imaging 24(3), 033001 (2015)
39. Zhang, J., Li, Y., Li, T., Xun, L., Shan, C.: License plate localization in unconstrained scenes using a two-stage cnn-rnn. IEEE Sensors Journal 19(13), 5256-5265 (2019)
40. Zhou, L., Guo, H., Lin, S., Hao, S., Zhao, K.: Combining multi-wavelet and cnn for palmprint recognition against noise and misalignment. IET Image Processing 13(9), 1470-1478 (2019). DOI 10.1049/iet-ipr. 2018.6122
