Iranian License Plate Recognition Using a Reliable Deep Learning Approach

Soheila Hatami[a], Farideh Sadat Jamali[b], and Majid Sadedel[c,*]

a. *Faculty of Mechanical Engineering, Tarbiat Modares University, Tehran, Iran.* (E-mail address: soheila.hatami@modares.ac.ir)

b. *Faculty of Mechanical Engineering, Tarbiat Modares University, Tehran, Iran.* (E-mail address: f.jamali@modares.ac.ir)

c. *Faculty of Mechanical Engineering, Tarbiat Modares University, Tehran, Iran.* (Tel.: +98(21)82884987, Fax: +98(21)82884987, E-mail address: majid.sadedel@modares.ac.ir) – Corresponding author.

Abstract. The issue of Automatic License Plate Recognition (ALPR) has been a challenging one in recent years because of weather conditions, camera angle, lighting and different license plate characters. Due to advances in deep neural networks, it is now possible to recognize Iranian license plates using specific neural networks. The proposed method recognizes license plates in two steps. First, license plates are detected through the YOLOv4-tiny model, which is based on Convolutional Neural Network (CNN). Secondly, Convolutional Recurrent Neural Network (CRNN) and Connectionist Temporal Classification (CTC) are applied to recognize the license plate characters. For labels, one string of numbers and letters is enough without segmenting and labeling each separately. The proposed method boasts an average response time of 0.0074 seconds per image and 141 frames per second (fps) in the Darknet framework and 0.128 seconds per image in the TensorFlow framework for the License Plate Detection (LPD) part. This method has been proven to provide a highly accurate model with minimal storage space requirements, using less than 2MB for the Character Recognition (CR) model. There was an average accuracy of 75.14% and a response time of 0.435 seconds for the end-to-end process. The released code is available through GitHub.

KEYWORDS: YOLO, CTC, CRNN, TensorFlow, Darknet, Object Detection, Automatic License Plate Recognition

1. Introduction

Automatic License Plate Recognition (ALPR) has revolutionized the way we manage urban traffic, record violations, and recover stolen vehicles. Its wide range of applications has contributed to building smarter cities, and we continue to explore new possibilities. Despite the challenges, ALPR systems have come a long way, and today, we have access to advanced deep learning methods and neural networks that enable us to design innovative ALPR systems. These new approaches are more adaptive and less dependent on traditional segmentation methods, giving us more control over our system [1]. We look forward to continuing the development of ALPR systems and the positive impact they have on our lives

In the field of license plate recognition, challenges related to datasets can often arise. These challenges may include issues such as incorrect positioning or identification of license plates, poor image quality, and obstacles hindering the detection process. Adverse weather conditions can also hurt the accuracy of the image. However, with determination and a comprehensive dataset, these challenges can be overcome. By using deep neural networks and considering all steps of the ALPR process, we can create an approach that delivers a better accuracy/real-time performance trade-off. The proposed method presented in this paper serves as a testament to the power of innovation and the endless possibilities that exist when we push the boundaries of what is possible. The main contributions of the proposed method in this paper are as follows:

- This paper aims to provide suitable datasets for ALPR. There is a lack of existing datasets with Persian letters and numbers written on Iranian license plates as string labels, making it challenging to train models. These datasets could be publicly available for future works.
- Current license plate recognition techniques have limitations due to limited datasets and the inability to recognize certain types of license plates. To overcome this, we need to improve and generalize these techniques by incorporating diverse types of license plates and conditions in the datasets.
- This new approach to ALPR eliminates the need for the Character Segmentation (CS) step. Using a Convolutional Recurrent Neural Network (CRNN) and Connectionist Temporal Classification (CTC), a more cost-effective model is constructed that requires less storage space, making it ideal for mobile applications.

- The final model detects license plates in 0.435 seconds. In Darknet, this value is 0.0074 seconds. The TensorFlow framework recognizes the characters in 0.127 seconds.
- Achieving 141 frames per second, the License Plate Detection (LPD) in the Darknet framework sets new standards and opens up exciting possibilities for real-time applications.

This paper is structured as follows: In section 2, we will provide a concise and informative overview of the current datasets and deep learning methods, along with some related work. Section 3 will provide a comprehensive and detailed description of the proposed dataset and method. In section 4, we will present the evaluations and final results with complete confidence. Lastly, in section 5, we will summarize our conclusions with absolute certainty.

2. Background and Related Works

2.1. Background

Datasets

Creating a suitable dataset that covers all aspects of computer vision work is a major challenge. Bias in datasets, especially in ALPR datasets, has become a popular and controversial topic [2]. ALPR is a remarkable technology that depends on LPD as its first step. LPD is a complex process that requires a rich dataset to train the network. Successful object detection networks are trained on rich datasets such as ImageNet [3], COCO [4], and PASCAL VOC [5]. In some cases, small available datasets can generate good results even if they are not general and this may due to the nature of the deep learning task like in [6] work. Fortunately, there are several publicly available datasets of license plates from different countries, including CCPD [7] and SSIG-SegPlate [8]. These datasets are making it possible to develop ALPR systems that can help make our roads safer and more secure.

To recognize license plates accurately, we need datasets specific to each country. However, finding datasets for some countries, like Iran, can be challenging as there are few available. Despite this, we can still make progress by persevering and seeking out the resources we need. In the paper [9], a dataset of approximately 15,000 diverse images from three regions was used. The dataset lacks the important requirement of being publicly available and focuses on one license plate per image, potentially weakening the trained network's ability to detect multiple plates per image. However, the dataset can still be useful in testing recognition systems, with room for improvement by increasing image diversity and accessibility. The dataset used in [10] has limitations due to the use of a fixed-angle camera, resulting in a less consistent dataset. However, it can still be valuable for training models to recognize license plates. Future studies may benefit from capturing images from different viewpoints and angles to create a more comprehensive dataset for wider applications. The only publicly available dataset for ALPR is Iranis [11]. However, this dataset is only suitable for the last step of ALPR, i.e. Character Recognition (CR). It is worth noting that this dataset treats license plate characters as separate objects and labels them individually, which makes creating a dataset a tedious task. IR-LPR [12] is a public dataset with 20967 car images and 27745 license plates available for CR tasks. Annotations are available for each character, not the entire license plate.

Deep Learning Approaches

Convolutional Neural Networks (CNNs) have made significant progress in the field of computer vision in recent years, making them one of the most widely used networks. Their well-constructed algorithm has enabled several architectures of CNN to be proposed, each performing better than others for specific problems. The depth of the network is a key factor in the development of various architectures [13], and CNNs have proven to be extremely effective in this regard. The best agents of CNNs, such as Regions-based Convolutional Neural Network (R-CNN) [14], Fast R-CNN [15], [16], Faster R-CNN [16], You Only Look Once (YOLO) [17], [18], [19], [20], and Single Shot MultiBox Detector (SSD) [21], inspire us all to push the boundaries of what is possible in computer vision.

YOLO [17] is an incredible object detector that is known for its accuracy and efficiency. It is widely used for real-time tasks and is based on state-of-the-art algorithms. The process of YOLO involves predicting several regulated bounding boxes and class probabilities for each section of an image, making it one of the most efficient object detectors out there. YOLO has several versions, each with improvements in accuracy and performance compared to previous versions. YOLOv2 [18] can detect over 9000 object categories that are implemented through the Darknet-

19 framework. Its architecture consists of 11 layers as the trained network on ImageNet (classification) and 19 layers on COCO (detection). The next version, YOLOv3 [19], uses the Darknet-53 framework. Its architecture consists of two sets of 53 layers. Compared to its previous version, it uses a residual block instead of anchor boxes and detects small images more accurately in terms of performance. YOLOv4 [20], the latest version of the YOLO detector, is better than its predecessors in terms of accuracy and speed. With a 10% improvement in mean Average Precision (mAP) and a 12% improvement in frames per second (fps), it sets a new standard for object detection. Meanwhile, SSD [21] is another impressive object detector that can detect multiple objects in one shot, making it faster than R-CNN in terms of processing. These object detectors are inspiring examples of what can be achieved with cutting-edge technology and innovation.

The ALPR system takes a significant step forward with the CRNN with CTC [22] method, which has been proposed for recognizing image-based sequences. This innovative approach combines Deep Convolutional Neural Network (DCNN) and Recurrent Neural Network (RNN) with CTC [23] to create an effective structure for image-based sequence recognition. By extracting feature sequences from input images, the CRNN structure can predict each frame of the feature sequence, and translate its predictions into a label sequence. What's truly remarkable about this method is that it does not require character-level annotations for the learning process. Ground truth texts for the input images are enough to produce sequence labels. With no limit to the dimension of the input sequences, and the ability to produce outputs of different lengths, the CRNN with CTC method is a breakthrough in the world of image-based sequence recognition.

2.2. Related Works

The ALPR process is a fascinating field of study, consisting of three main steps that work together to achieve remarkable results. Through published studies, different methods have been suggested on how to implement this process, and researchers have reviewed a variety of previous studies in the field to identify approaches that work best. These studies have led to groundbreaking advancements in traditional image processing methods, as well as the development of modern techniques like machine learning and deep learning algorithms. By building on these innovative approaches, we can continue to push the boundaries of what is possible and unlock new potential in the field of ALPR.

The edge density approach has been widely used for LPD with great success. Massoud *et al.* [24] and note [25] utilize this method in their LPD systems. Massoud et al. achieved a 91% accuracy rate through the use of Sobel edge detector filter and prior knowledge of Egyptian license plate characters. [25] used Canny edge detection and Probabilistic Neural Network (PNN) for character recognition. Overall, the edge-based method is a reliable and effective approach for LPD.

LPD has become a popular research area, with several recent works utilizing different neural network architectures. CNNs have gained significant attention due to their high performance in various applications. Researchers have developed different models to achieve high accuracy and real-time detection. For instance, Singh and Bhushan [26] have implemented Faster R-CNN, which uses RPN for object recognition. Their model has an impressive accuracy rate of 99% in detecting Indian license plates. Similarly, Jamtsho *et al.* [27] have achieved an overall mAP of 98.6% with 0.0231 training loss using YOLOv2 to extract Bhutanese license plates. Lin and Wu [28] have proposed a one-step model that uses a modified version of YOLOv2-tiny to extract license plates and recognize license plate characters. Their model has an overall recall rate of 84.5% and a lower computational load than YOLOv2-tiny. Tourani *et al.* [9] have relied on the real-time feature of YOLOv3 to recognize Iranian license plates. They have modified the last convolutional layer of YOLOv3 to fit the superior network on the Iranian license plate. Their proposed approach has an end-to-end recognition accuracy of 95.05% with an average time of 119.73 milliseconds to specify a sequence of characters in real-time. Lastly, Silva and Jung [29] have used a hierarchical CNN to detect vehicles and license plates using the same network. Their model is inspired by YOLO architecture and processed faster than typical YOLO with lighter computational cost. By making some modifications to the network architecture, including reducing the number of max-pooling layers and using transfer learning for the first eleven layers, they achieved excellent results.

ALPR is a challenging field where researchers have proposed various approaches to address environmental challenges when localizing license plates in an image. In this regard, several studies have shown promising results by using state-of-the-art techniques. Samadzadeh *et al.* [30] developed a robust ALPR system by using an SSD and Residual Network (ResNet18) for the CR step. For the Character Segmentation (CS) step, they employed an open-source text detector as text detection and two activation maps to correct the perspective transformation of the license plate and segmentation boundaries. Wang *et al.* [31] proposed a method that adapted the Multi-Task Convolutional Neural Network (MTCNN) for LPD by creating a series of reinforcements. The LPD step is followed by the recognition of license plate characters using a CRNN with CTC. Silva and Jang [32] introduced the Warped Planar Object Detection Network (WPOD-NET) for LPD using YOLO, SSD, and Spatial Transformer Networks (STN). In contrast to most research in this area, Al-Batat *et al.* [33] placed vehicle detection as the first step of a YOLO-based pipeline for ALPR. For LPD and CR, they have employed YOLOv4-tiny. Their approach was evaluated using five famous public datasets with an average accuracy of about 90.3%. Al-Batat et al. also added the capability of classifying vehicle types (emergency vehicles and trucks) using ResNet50. Overall, these studies have shown promising results in improving the accuracy and speed of ALPR systems. Further research in this area can help address several challenges and improve the performance of ALPR systems, making them more reliable and efficient.

3. The Proposed Method

In this section, we will delve into the proposed method in detail. With the need for a dataset as the first step to detect license plates and recognize their characters, we have prepared datasets for each of the steps (LPD and CR). To detect license plate rectangles in the input image or video frames, we use a model called YOLOv4-tiny, which is a lighter version of YOLOv4 with fewer parameters. Although this model sacrifices some accuracy, it enables real-time application. Despite the challenge of connecting the LPD and CR steps, we have succeeded in connecting them by converting the LPD training weights obtained from the Darknet framework into a model within the TensorFlow framework. By attaching the CR step trained in the TensorFlow framework to this model, we can now perform the entire process in the TensorFlow framework. As shown in Figure 1, the entire process is performed in a flowchart, starting with the detection of license plates, then cropping them from the input image, and finally, recognizing the characters written on the license plates. This method serves as an inspiration for achieving complex tasks through innovation and perseverance.

Figure 1.

3.1. Dataset for License Plate Detection

The dataset used in this section comprises 3065 images, which were obtained under various conditions to make it as diverse as possible. Despite security concerns, these images are available to researchers who require them for their research work. It's important to note that these images should not be used for any commercial purposes. The significant diversity in the dataset as in Figure 2, ranging from weather and lighting conditions to camera distance from the car, is a testament to our commitment to learning and familiarizing the proposed network with different image scenarios. Obtaining images from different sources and locations adds to the dataset's richness, making it an ideal resource for researchers worldwide. The manual labeling of the images in the dataset by the labelImg toolbox shows our dedication to excellence and attention to detail. With most of the images taken using mobile phone cameras with different resolutions and formats, we saved them all in .jpg format for convenience. Finally, we converted all the labels of these images to .txt file format, which could later be used by models such as YOLO. We hope that this dataset will inspire researchers worldwide to pursue new avenues of study and discovery.

Figure 2.

3.2. Dataset for Character Recognition

The ability to recognize Persian characters written on license plates has been made possible thanks to a carefully curated dataset of labeled images. This dataset contains a wide variety of data, including single-digit, single-letter, and combinations of letters and numbers. The dedication to ensuring accuracy and reliability is evident in the inclusion of blank-labeled images, which prevent non-character objects from being recognized as characters. With a dataset of 5455 images, careful preparation has led to the successful conversion of Persian characters into their English equivalents, resulting in a powerful and effective string of English letters and numbers for each image. The result is a training dataset of 3364 images that can be used to train a desired network. The labels are considered image names in string format, eliminating the need for separate label files next to the images. This inspiring achievement demonstrates the potential of technology to overcome challenges and improve our lives. The format of all images is the same as .jpg. In this dataset, all the characters of Iranian license plates are covered as follows:
{alef, be, pe, te, se, jim, dal, ze, sin, shin, sad, ta, eyn, fa, qhaf, kaf, gaf, lam, mim, non, he, vav, ye, tashrifat, malol, D, S}
In addition, it includes all numbers from 0 to 9.
Persian letters can be transformed into English letters and numbers, and then converted back to their original form to be used as image labels. The dataset showcases the characters used, with similar counts for most numbers, except for 1, 8, and 9, which are commonly found on license plates. While some letters like “sin” or “be” are uncommon, special plates like “tashrifat” are rare on the roads. Despite these variations, the transformation from one form to another allows us to create something new and inspiring as shown in Figure 3.

Figure 3.

Here, according to the available dataset and the selected network model, 80% of the images, i.e., 2692 images for the training, 10%, i.e., 336 images for the validation, and the last 10% part for the test set are considered. After preparing the images for the dataset, the license plates need to be detected and then recognized in two steps, respectively.

3.3. License Plate Detection Network

In this section, the powerful YOLOv4-tiny model is utilized to detect license plates in the input image, providing the highest quality outputs. With the Darknet framework, we are able to achieve the best results in both training and testing. However, we understand that being flexible is important, so we convert the LPD model into a form that can be used in other frameworks such as TensorFlow and PyTorch. This allows us to effortlessly connect the model to other parts of our project. By using a 416 x 416 dimension for all input images, we ensure that our training network is optimized for the best performance. The YOLOv4-tiny model is truly remarkable, and we look forward to seeing the incredible results it can produce. Table 1 provides a list of the changes made to the model configuration, as well as the fine-tuned hyperparameter values for training the model. To optimize the processing of data while considering the limited memory of the GPU used, the batch size and subdivisions have been adjusted accordingly in this structure. The use of 32 batches means that data is processed in batches of 32 images, with larger batches requiring more memory. In addition, the images are split into four blocks and processed in parallel for each subdivision, which enables more efficient processing. During training, the momentum value has been set at 0.9, which helps to stabilize the gradient and improve the accuracy of the model. Similarly, the default weight decay value of 0.005 has been used to prevent imbalances in the dataset and make the weights weaker for normal features. The learning rate value has been set at 0.00261, which is optimal for this structure, and its low value helps to prevent the network from diverging due to unstable gradients. It is important to note that due to having only one class, the number of classes has been set to 1. This has resulted in a change in the filters to (number of classes + 5) × 3, which gives a result of 18 filters. It is crucial to maintain the anchor boxes used in the Darknet framework to train the weights and transform them into a model in the TensorFlow framework. To improve the training data, pre-trained weights on the COCO dataset have been used. These pre-trained weights have helped to facilitate faster and better network

learning, and transfer learning through them. After training the data in the Darknet framework, the weights obtained are transformed into a model in the TensorFlow framework. Finally, it is worth mentioning that the weights obtained in the Darknet framework have a size of 23.5MB, while the model in the TensorFlow framework has a size of 26.3MB. There isn't a significant difference in storage space between the two, and both require only a small amount of space for storage. Overall, these optimizations and adjustments have led to a more efficient and accurate model.

Table 1.

3.4. Character Recognition Network

To recognize the license plate characters in the proposed method, it is no longer necessary to segment the characters individually, and as mentioned before, it is only necessary to detect the license plate and obtain its rectangle, as the input for the recognition step.

The proposed method of recognizing license plate characters is efficient and eliminates the need to segment individual characters. Detecting the license plate and obtaining its rectangle is sufficient for the recognition step. It is worth noting that the model compensates for images with fewer than eight characters by implementing the "X" character. This ensures that the model recognizes the license plate characters, even if some parts of them are missing due to the absence of characters in the license plates. In case the trained model receives eight (or fewer) characters and is unable to recognize some of them, the term "Not Known" refers to the non-recognition of those characters. This approach ensures that the model provides accurate results in most cases.

The proposed method presents an efficient approach to CR by utilizing a CRNN, and a CTC within the transcription layer, adopting the overall structure described in [22]. The CNN is employed to extract the features of the input image, which has a width and height of 200 and 50, respectively. The CNN extracts a sequence of feature vectors, which are then passed through an RNN, where a matrix with probabilities is generated, assigning a value to each element of that sequence. To overcome the vanishing gradient problem, Long-Short Term Memory (LSTM) as a type of RNN is chosen. Two LSTMs are combined as bidirectional LSTMs to interpret contexts from both directions. The matrix, which encodes the input, is then used to calculate the loss and train the neural network with Adam optimizer. The information encoded in this matrix can be decoded to obtain the text written in the input image, and both of these tasks are performed by the CTC in the transcription layer. Finally, the CTC recognizes the characters, and there is no need to over-process the recognized characters as position and width are no longer a concern. This method presents a promising and effective solution to CR and can be further refined to improve its performance.

The image recognition process is horizontally performed by the CNN, as illustrated in Figure 4. Each horizontal position is known as a time step and contributes to the feature extraction process. After the matrix output is generated, it is combined with the ground truth label of each input image and passed through the CTC loss function in the transcription layer. This step is particularly interesting since it considers all possible states of the ground truth label related to the text, without requiring knowledge of where each character occurs. The probabilities of each time step are multiplied, and all the states that are intended for a single input are summed to result in the output. The highest value in the resulting set is then selected as the output.

To improve the decoding of duplicate characters, a “blank” character is used, marked with a “-” and subsequently deleted during decoding. This is a useful tool for recognizing duplicate characters during decoding, especially when two characters need to be repeated. These techniques are important when training the network and aiming to minimize loss. Overall, the sum of different input text states can be used to calculate the total loss.

$$Loss\ Function = -\sum_{I_i, l_i \in X} \log p(l_i | y_i)\ which\ X = \{I_i, l_i\}_i \quad [22]\ (1)$$

In this regard, X is the dataset, I_i is the training image, l_i is the ground truth label, and y_i is the sequence created from the input text image by the CNN and RNN. If the network predicts the data that it has not seen before, there is no longer a ground truth label. In this case, it should use the best path decoding. In this type of decoding, the best path is selected in such a way that in the output matrix of the RNN, at each step when the probability of occurrence

of the predicted character is high, the same character is placed in the path and this process continues. Find that until the prediction is done, then the duplicate characters are predicted and finally the “blank” character is removed. For this reason, the greedy search decoder is used in this section, but for more complex texts, the beam search decoder can be used. In the beam search decoder, candidates are first introduced using a tree structure, then points are given to each using the best path, which not only increases the accuracy but also the time for prediction much longer, and this is a drawback for real-time and high-speed methods.

Figure 4.

Table 2 clearly shows the general structure of the network used for the CR step. Using this structure for CR provides a model with less than 2MB, which is very small in size. Also, in the end, the number of parameters of the model used is 434,918, which is a relatively small value and can be used for mobile devices.

Table 2.

The proposed CR algorithm reaches its final stage with the CTC step, where each character is assigned an individual label. Treating the license plate characters as separate entities creates a more efficient and effective approach. ALPR algorithms operate without a CS step, which enhances the processing speed. Even with limited labeled Iranian datasets for Persian characters available, the approach of treating each character as a distinct object unlocks greater convenience. These features combine to make the CR algorithm an innovative and inspiring solution.

All calculations are performed in the Ubuntu 20.04 operating system environment. The hardware used is Intel Core i7 3.50GHz CPU, 32GB RAM, DDR4 for CPU, and Geforce GTX 1050 4GB for GPU, cuDNN, CUDA, TensorFlow 2.3, and Python 3.9 are further used.

4. Results

4.1. License Plate Detection

Since the main framework for implementing the YOLOv4-tiny model is the Darknet framework, therefore, by using this framework, more speed and accuracy can be obtained. But as mentioned before, this framework has less flexibility and as a result, after training the data in this framework, the trained weights need to be converted into a model in the TensorFlow framework and thus that model can be used for further processing.

The training data, i.e., 2452 images, are given to the YOLOv4-tiny network for training, and according to the network framework, the training time for this number of images is less than 90 minutes. Validation data is then used to check the network output on the images, and finally, the weights obtained are used to evaluate the model on images and video frames. According to Figure 5, which is stored by the network during training, the network loss is decreasing with passing iterations. So, this decrease has reached dramatically less than 1 even before 1000 iterations, and after that, this decrease has continued.

Figure 5.

Given that the images in the validation set are different from the test images, the final results are just performed on the validation images. Figure 6 shows examples of these detections in different situations. As it turns out, these images are in different situations: night, day, with more than one license plate and different distances from the camera. The trained network gives as outputs a rectangle drawn around the license plate, along with the name of the class (License Plate), the network reliability of the detection performed, as well as the coordinates of the detected rectangle (including the top, bottom, left and right points relative to the whole image).

Figure 6.

Finally, for our detection, we have the correctly detected license plates as True Positives (TP), not detected license plates as False Negatives (FN), and other items known as license plates while they were not, called False Positives (FP). For the case where the license plates may not be detected, the user needs to check all the data and there is no way to warn the user. But for the case where another element is mistakenly detected as the license plate, given that this detected element enters the next step, i.e., CR, in this step, the characters are not recognized correctly and the

user only needs to check the relevant image. As can be seen from Table 3, the number of license plates that have been correctly detected is much higher than the number of license plates that have not been detected or even mistakenly considered other elements as license plates.

Table 3.

As it is clear, Figure 7 includes the results of the images taken after converting the weights trained in the Darknet framework to a model in the TensorFlow framework. As it turns out, this model can easily detect more than one license plate in the image, and can also detect images containing license plates with different conditions. The converted model in this framework likewise gives as output the name of the desired class (License Plate), the network reliability of the detection as well as the coordinates of the detected rectangle (including its top, bottom, left, and right points relative to the whole image). The coordinates of the detected rectangle are used to crop the license plate from the whole image to recognize its characters.

Figure 7.

As can be seen from Table 4 and Table 5, this method has reasonable results for the Darknet framework, it can be used for real-time applications. However, due to the fact that the main environment for implementing YOLO models is not the TensorFlow framework and requires more optimization, so the results have the same accuracy but lower speed. At the same time, the model obtained in the TensorFlow framework can be easily connected to models in other frameworks and this is a big advantage.

4.2. Character Recognition

To obtain a suitable model in the TensorFlow framework, it is first necessary to consider the training images for the training process. These images are all in the form of images that have labels of numbers and letters. Due to the structure of the networks used for this model, its training time is about 30 minutes, and this time is such that only a string of numbers and letters is needed and there is no need to segment the characters. Figure 8 shows examples of CR for some Iranian license plates with different characters. At the top center of these images, a sequence of English characters, after which, only the English letter of this sequence must be converted to its Persian equivalent. Besides, the ground truth for each one is added. As a result, at the bottom of each license plate is written a string of numbers and letters equivalent to the above prediction.

Figure 8.

4.3. End-to-End Process

Finally, the model obtained in the LPD step is merged with the model obtained for the CR step, and the results are obtained on the images and video frames in Figure 9.

Figure 9.

In this case, as it turns out, the license plate rectangles are first detected using the model converted in the TensorFlow framework, and then the characters written on them are recognized without being segmented. Eventually, only the English letters of these characters converted to their Persian equivalents. For this reason, in addition to these predictions (for images and video frames), their predicted equivalents are further included. In fact, since the projection rectangle needed to be large for better visibility, part of the prediction was not included in the image. But to ensure the output, a text file has stored the output for each frame, and the output written under each license plate is the output of the text file.

4.4. Challenges of the Process

With determination, we can overcome the challenges that arise in each step of the ALPR process. Let's explore how we can address these obstacles:
1) To ensure that license plates remain legible, it is important to keep them clean and free of debris. If a license plate becomes damaged or tilted due to an accident, it should be replaced as soon as possible to maintain its readability.
2) It's worth noting that the license plate rectangle obtained from the first step of the process may not always completely cover the license plate. As a result, this can lead to the recognition of fewer characters than usual.
3) To ensure successful license plate recognition during filming, it's important to avoid obstructions caused by incoming images and ensure that the camera view is appropriate.
4) To improve the recognition of license plate characters, it would be helpful to address their similarities, especially in cases where the input image is of low quality. By doing so, we can reduce the chances of illegibility and misrecognition, and ensure that characters such as the numbers 3 and 4 are accurately identified.

Figure 10.

As Figure 10 shows, there are challenges for both steps in the proposed ALPR approach. In image a, the license plate character has been recognized less than eight, and character “X” indicates that the number of characters has a value of less than eight. Likewise, the number “4” is incorrectly recognized instead of the number “9”. In image b, the license plate and part of the car are detected as outputs. In part c, for the Peugeot 405 car, the number “4” was incorrectly recognized instead of the number “2”, and due to the incorrect license plate detection (location of the rectangular), the number “8” at the end of the license plate is not correctly located in the detection rectangle, so this number is not recognized correctly. Finally, in part d, which is part of the video evaluation, the number of characters for the Pride car is six. For example, in some cases, the license plates are dirty or the camera does not have a good view of them. In the end, for all the results to be comparable at a glance, they are all listed in Table 4. This table shows the average execution time for images with different qualities in addition to the frames per second value for videos. As the results show, this method is very suitable for images, and with this number of data has a significant accuracy. Table 5 shows these accuracies for different steps of this proposed method well.

Table 4.

Table 5.

4.5. Comparison of the Proposed Method with other Methods

As Table 6 shows, the proposed method has been able to achieve acceptable accuracy with this limited amount of data. Aside from that, this method has been able to provide a compact model that consumes limited storage space. The methods presented in previous studies have datasets with large numbers of images, higher-volume models (storage space), or datasets of images with limited conditions: that is, there is only one vehicle in the images, for example, or images are taken from very close distance. These issues and some others have all contributed to the high accuracy of these methods. While in this proposed method, an attempt has been made to obtain the best results with the least conditions. The ability of a model to perform well when faced with new or varying data is a significant aspect of its robustness. Our proposed model has been trained on a wide range of license plate images, and it can achieve acceptable accuracy when presented with new and unseen images. This demonstrates that the model is capable of recognizing license plates in a variety of real-world scenarios, including challenging lighting conditions, angles, and poor-quality images. It's worth noting that comparing the accuracy of our study with other studies in the field may not be entirely reliable due to differences in datasets. Nonetheless, we remain confident in the potential of our model to make a positive difference in the world.

Table 6.

5. Conclusions

The paper presents an innovative method for automatic recognition of Iranian license plates, which utilizes cutting-edge technologies such as CNN, CRNN, and CTC structure. Despite the lack of a suitable dataset for Iranian license plates, the researchers managed to create a diverse dataset of 3065 images, followed by a dataset of 3364 images of license plate characters. The proposed method was able to achieve impressive results using the Darknet framework for LPD, and the researchers overcame the limitations of this framework by converting the trained weights to a TensorFlow model. The unified method without CS demonstrates the power of collaboration between different frameworks and techniques. The researchers' dedication and hard work are truly inspiring, and their achievements pave the way for further advancements in license plate recognition technology. Furthermore, some main achievements obtained are as follows:

1. The small size of the proposed model (less storage space)
2. Short execution time (speed) of the proposed method
3. Obtaining a model with high average accuracy even with the limited number of images in the datasets

Considering that the use of YOLO models for the OCR step that has been proposed in previous works requires separate labeling of characters on license plates, for this reason, using other network models, for example, using a combination of deep RNNs can help to improve this part. Here we do not need to have the coordinate of each character and only a sequence of them is needed. As a result, this type of model is implemented in frameworks other than Darknet, and therefore converting weights to a model in the TensorFlow framework can be efficient.

In general, according to the results obtained in the results section and Figure 6, Figure 7, and Figure 8, as well as Table 6, it can be concluded that all the approaches of the article, have been achieved. In addition, the purpose of presenting a reliable model here is that by changing different inputs and in different conditions that the model is not familiar with, the proposed method can obtain desirable outputs. Because this demand has been fulfilled to an acceptable extent, a robust model has been obtained here.

In the future, the speed of the obtained model for testing data in the TensorFlow framework could be improved. Moreover, it is possible to expand the prepared dataset with more variety of images, and as a result, the performance of the proposed method can be improved. The method used for the final step of the process can potentially be used for other image-based sequence tasks.

Acknowledgment

The use of the dataset in this paper has been restricted to individuals interested in research, keeping in mind the ethical and security implications. The images used in the article have also been sourced with permission and care. This approach demonstrates the authors' commitment to upholding ethical standards and ensuring the safety of all involved parties.

References

1. Luo, S., and Liu, J. “Research on car license plate recognition based on improved YOLOv5m and LPRNet,” *IEEE Access*, vol. **10**, pp. 93692–93700, 2022, doi: 10.1109/ACCESS.2022.3203388.
2. Laroca, R., Santos, M., Estevam, V., et al. “A first look at dataset bias in license plate recognition,” in *35th Conference on Graphics, Patterns, and Images, SIBGRAPI 2022 (2022)*, Aug. 2022. doi: 10.1109/sibgrapi55357.2022.9991768.
3. Deng, J., Dong, W., Socher, R., et al. “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.
4. Lin, T.-Y., Maire, M., Belongie, S., et al. “Microsoft coco: Common objects in context,” in *European Conference on Computer Vision*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Cham: Springer International Publishing, 2014, pp. 740–755.
5. Everingham, M., Van Gool, L., Williams, C. K. I., et al. “The pascal visual object classes (voc) challenge,” *Int J Comput Vis*, vol. **88**, no. 2, pp. 303–338, 2010, doi: 10.1007/s11263-009-0275-4.
6. Khoramdel, J., Hatami, S., and Sadedel, M. “Wearing face mask detection using deep learning during COVID-19 pandemic,” *Scientia Iranica*, vol. **30**, no. 3, pp. 1058–1067, 2023, doi: 10.24200/sci.2023.59093.6057.

7. Xu, Z., Meng, A., Lu, N., Huang, H., et al. “Towards end-to-end license plate detection and recognition: A large dataset and baseline,” In *Proceedings of the European conference on computer vision (ECCV)*, pp. 255-271. 2018.
8. Gonçalves, G. R., Silva, S. P. G. da, Menotti D., et al. “Benchmark for license plate character segmentation,” Jul. 2016, doi: 10.1117/1.jei.25.5.053034.
9. Tourani, A., Shahbahrami, A., Soroori, S., et al. “A robust deep learning approach for automatic iranian vehicle license plate detection and recognition for surveillance systems,” *IEEE Access*, vol. **8**, pp. 201317–201330, 2020, doi: 10.1109/ACCESS.2020.3035992.
10. Rakhshani, S., Rashedi, E., and Nezamabadi-pour, H. “License plate recognition using deep learning,” *Journal of Machine Vision and Image Processing*, vol. **6**, no. 1, pp. 31–46, 2019.
11. Tourani, A., Soroori, S., Shahbahrami, A., et al. “Iranis: A large-scale dataset of farsi license plate characters,” 2021, doi: 10.1109/IPRIA53572.2021.9483461.
12. Rahmani, M., Sabaghian, M., Moghadami, S. M., et al. “IR-LPR: Large scale of iranian license plate recognition dataset,” in *12th International Conference on Computer and Knowledge Engineering, ICCKE 2022*, Sep. 2022. doi: 10.48550/arxiv.2209.04680.
13. Alzubaidi L., Zhang, J., Humaidi, A. J., et al. “Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions,” *J Big Data*, vol. **8**, no. 1, p. 53, 2021, doi: 10.1186/s40537-021-00444-8.
14. Girshick, R., Donahue, J., Darrell, T., et al. “Region-based convolutional networks for accurate object detection and segmentation,” *IEEE Trans Pattern Anal Mach Intell*, vol. **38**, no. 1, pp. 142–158, 2016, doi: 10.1109/TPAMI.2015.2437384.
15. Girshick, R., “Fast r-cnn,” in *2015 IEEE International Conference on Computer Vision*, Apr. 2015, pp. 1550–5499. doi: 10.1109/ICCV.2015.169.
16. Ren, S., He, K., Girshick, R., et al. “Faster r-cnn: Towards real-time object detection with region proposal networks,” *IEEE Trans Pattern Anal Mach Intell*, vol. **39**, no. 6, pp. 1137–1149, 2015, doi: 10.1109/TPAMI.2016.2577031.
17. Redmon, J., Divvala, S., Girshick, R., et al. “You only look once: Unified, real-time object detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.
18. Redmon, J., and Farhadi, A., “YOLO9000: better, faster, stronger,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. doi: 10.1109/CVPR.2017.690.
19. Redmon, J., and Farhadi, A., “Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.
20. Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M., “Yolov4: Optimal speed and accuracy of object detection,” *arXiv preprint arXiv:2004.10934*, 2020.
21. Liu, W., Anguelov, D., Erhan, D., et al. “Ssd: Single shot multibox detector,” in *European Conference on Computer Vision*, 2016, pp. 21–37. doi: 10.1007/978-3-319-46448-0_2.
22. Shi, B., Bai, X., and Yao, C., “An end-to-end trainable neural network for Image-based sequence recognition and its application to scene text recognition,” *IEEE Trans Pattern Anal Mach Intell*, vol. PP, Jul. 2015, doi: 10.1109/TPAMI.2016.2646371.
23. Graves, A., Fernández, S., Gomez, F., et al. “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in *ICML 2006 - Proceedings of the 23rd International Conference on Machine Learning*, Jan. 2006, pp. 369–376. doi: 10.1145/1143844.1143891.
24. Massoud, M. A., Sabee, M., Gergais, M., et al. “Automated new license plate recognition in egypt,” *Alexandria Engineering Journal*, vol. **52**, pp. 319–326, Sep. 2013, doi: 10.1016/j.aej.2013.02.005.
25. Tuba, M., Akashe, S., and Joshi, A., *ICT Systems and Sustainability Proceedings of ICT4SD 2019, Volume **1***. 2019. doi: 10.1007/978-981-15-0936-0.
26. Singh, J., and Bhushan, B., “Real time indian license plate detection using deep neural networks and optical character recognition using LSTM tesseract,” in *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, 2019, pp. 347–352. doi: 10.1109/ICCCIS48478.2019.8974469.
27. Jamtsho, Y., Riyamongkol, P., and Waranusast, R., “Real-time Bhutanese license plate localization using YOLO,” *ICT Express*, Nov. 2019.
28. Lin, C.-H., and Wu, C.-H., “A lightweight, high-performance multi-angle license plate recognition model,” in *2019 International Conference on Advanced Mechatronic Systems (ICAMechS)*, 2019, pp. 235–240. doi: 10.1109/ICAMechS.2019.8861688.
29. Montazzolli, S. and Jung, C., “Real-time brazilian license plate detection and recognition using deep convolutional neural networks,” in *30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, 2017, pp. 55–62. doi: 10.1109/SIBGRAPI.2017.14.

30. Samadzadeh, A., Shayan, A. M., Rouhani, B., et al. “RILP: Robust iranian license plate recognition designed for complex conditions,” in *2020 International Conference on Machine Vision and Image Processing (MVIP)*, 2020, pp. 1–7. doi: 10.1109/MVIP49855.2020.9116910.
31. Wang, W., Yang, J., Chen, M., et al. “A light CNN for end-to-end car license plates detection and recognition,” *IEEE Access*, vol. **7**, pp. 173875–173883, 2019, doi: 10.1109/ACCESS.2019.2956357.
32. Montazzolli, S., and Jung, C., “License plate detection and recognition in unconstrained scenarios,” in *European Conference on Computer Vision*, Springer, 2018, pp. 593–609. doi: 10.1007/978-3-030-01258-8_36.
33. Al Batat, R., Angelopoulou, A., Premkumar, S., et al. “An end-to-end automated license plate recognition system using YOLO based vehicle and license plate detection with vehicle classification,” *Sensors*, vol. **22**, p. 9477, Dec. 2022, doi: 10.3390/s22239477.
34. Rashtehroudi, A., Shahbahrami, A., and Akoushideh, A., “Iranian license plate recognition using deep learning,” in *2020 International Conference on Machine Vision and Image Processing (MVIP)*, 2020, pp. 1–5. doi: 10.1109/MVIP49855.2020.9116897.

Figures' captions:

Figure 11. The overall diagram of the proposed method

Figure 12. Examples of license plate detection dataset images under diverse environmental and distance conditions

Figure 13. The number of different labels used in the character recognition dataset: a) Numbers b) Words and letters

Figure 14. Overall diagram of the proposed character recognition stage

Figure 15. The network loss with to its iteration number

Figure 16. Examples of license plate detection using the Darknet framework

Figure 17. Examples of license plate detection using the TensorFlow framework

Figure 18. Sample outputs of the proposed character recognition model

Figure 19. Examples of the end-to-end proposed method

Figure 20. Examples of network errors

Tables' captions:

Table 7. YOLOv4-tiny network training parameter values

Table 8. Summary of network structure used for character recognition stage

Table 9. Number and distribution of detected license plates in each set

Table 10. Execution time and frames per seconds results on image and video

Table 11. Average precision on training and validation images

Table 12. Comparison of some recent methods with the proposed method

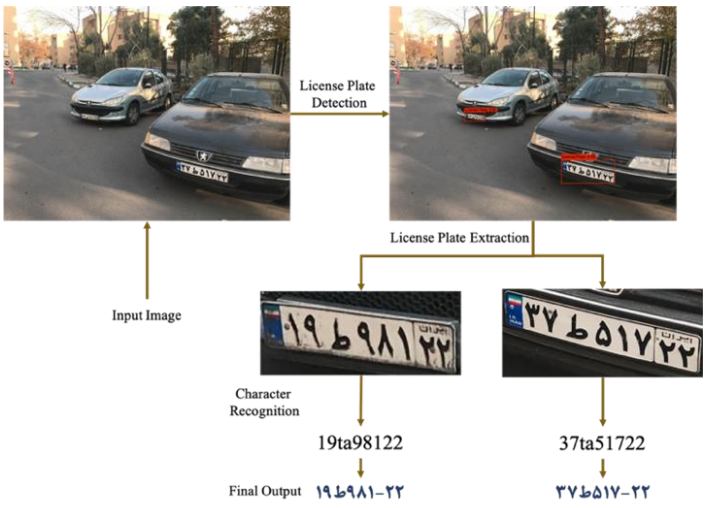


Figure 21.

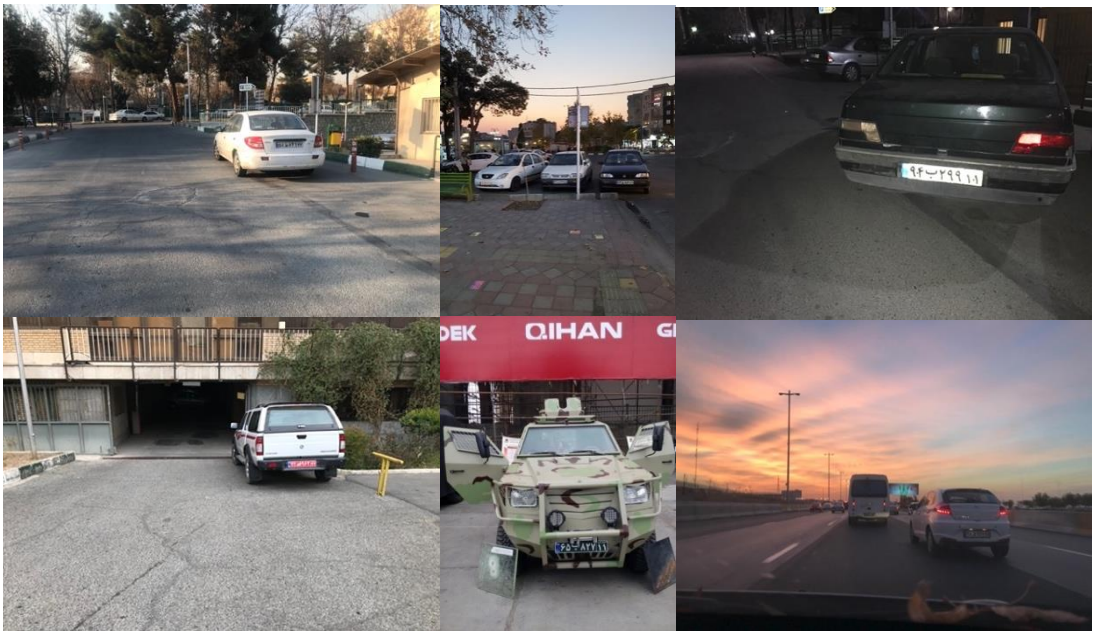
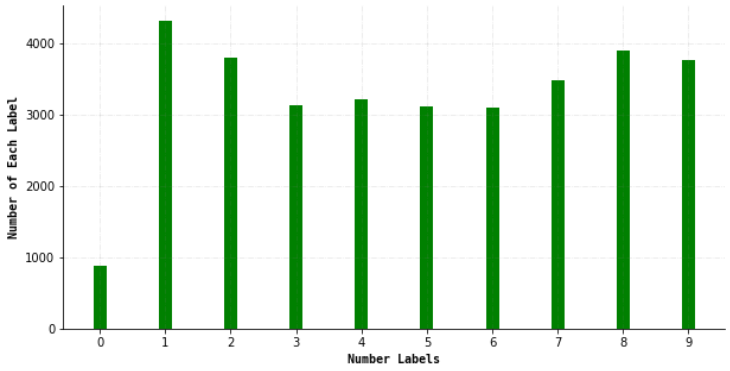
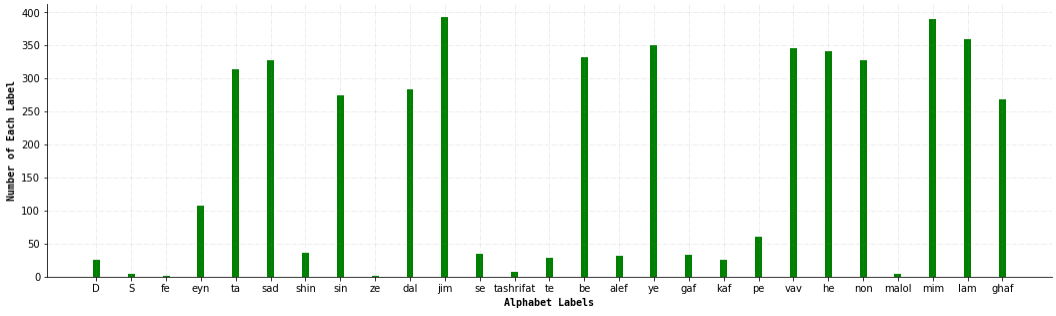


Figure 22.



(a)



(b)

Figure 23.

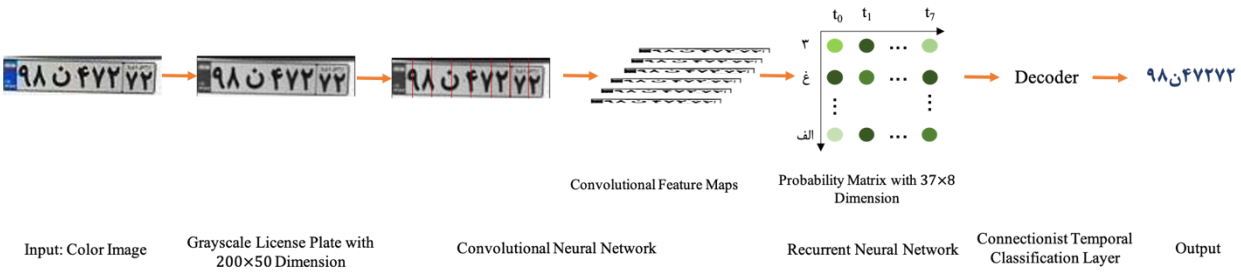


Figure 24.

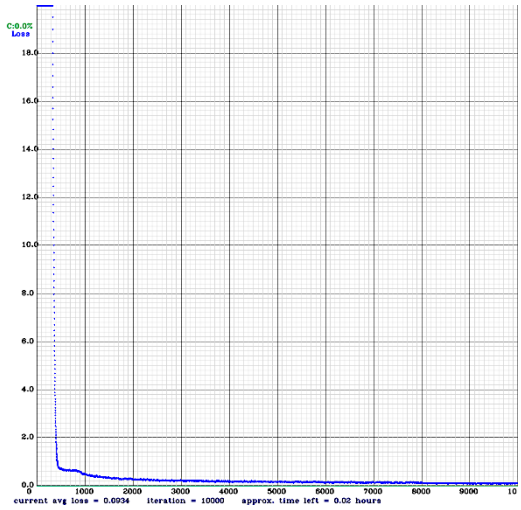


Figure 25.

Figure 26.

Figure 27.

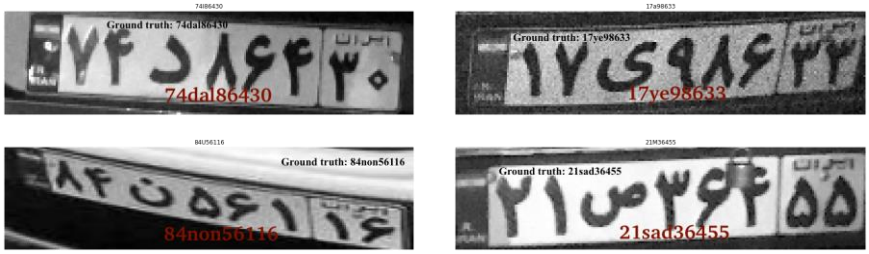
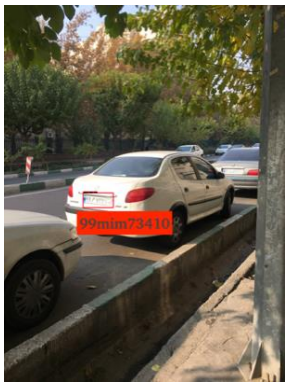


Figure 28.

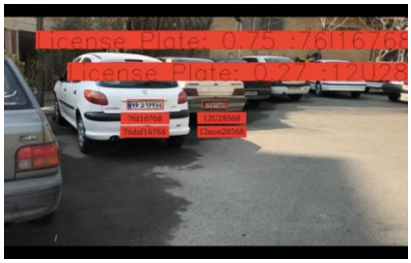


99mim73410

84non56116

Left License Plate: 0.60 **75ye64144**

Right License Plate: 0.74 **49ghaf5682**



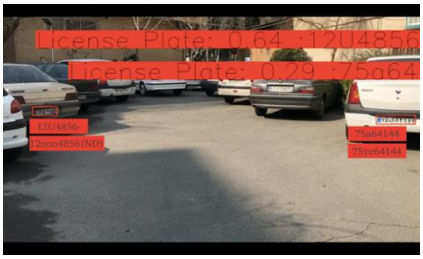
Left License Plate: 0.75 **76dal16768**

Right License Plate: 0.27 **12non28568**

Figure 29.

(a)

(b)



(c)
Left License Plate: 0.64 **12non4856(ND)**
Right License Plate: 0.29 **75ye64144**

(d)
Left License Plate: 0.69 **XXhe34838**
Right License Plate: 0.64 **76dal16768**

Figure 30.

Table 13.

Parameter	Value
Batch size	32
Subdivisions	4
Momentum	0.9
Weight decay	0.005
Learning rate	0.00261
Number of epochs	132
Number of iterations	10000
IoU threshold	0.5

Table 14.

Type	Configurations
Input Image	200×50 Gray scale
Convolution layer	Units: 32, Kernel size: 3×3, Activation function: relu, Kernel initializer: he_normal, Padding: same
Maxpooling	Pool size: 2×2, Stride: 2
Convolution layer	Units: 64, Kernel size: 3×3, Activation function: relu, Kernel initializer: he_normal, Padding: same
Maxpooling	Pool size: 2×2, Stride: 2
Dense layer	Units: 32, Activation function: relu
Dropout	Rate: 0.2
Bidirectional LSTM	Units: 128
Dropout	Rate: 0.25
Bidirectional LSTM	Units: 64
Dropout	Rate: 0.25
Dense	Units: 38, Activation function: softmax
Transcription	-

Table 15.

Detection type	Training images	Validation images
TP	2255	539
FP	226	50
FN	315	105

Table 16.

Image	Step	Framework	Execution time (sec)	Video	Step	Framework	Frames per seconds (fps)
	LPD	Darknet	0.0074		LPD	Darknet	141
		TensorFlow	0.128			TensorFlow	7.8
	CR	TensorFlow	0.127		ALPR	TensorFlow	2.3
	ALPR	Darknet + TensorFlow	0.255 (separate calculation)				
		TensorFlow	0.435				

Table 17.

Type of data	LPD	CR	ALPR
	Darknet	**TensorFlow**	**TensorFlow**
Training images	88.66%	91.01%	77.61%
Validation images	87.81%	87.22%	75.14%

Table 18.

Step	Method	Pros	Cons	AP & Time
LPD + CR [31]	CNN + CNN, RNN, CTC	Without character segmentation, use a dataset with more than 250,000 images and a model 15 times lighter than the YOLOv3 model	Use a highly computed chain network (with three sections) for LPD	98.8%
LPD + CS + CR [9]	(Preprocessing) CNN (YOLOv3) + CNN (YOLOv3)	Using a dataset with more than 15,000 images (data with appropriate diversity), improving images using histogram equalization	Using preprocessing, using two YOLOv3 networks with high computational calculations, having only one car and one license plate per image of the dataset, a three steps process	95.05%, 119.73msec
LPD + CS + CR [30]	CNN (SSD300) + (Preprocessing) Using CRAFT for detecting text blobs and characters + CNN	Using a method for CR with consuming less storage space	Using a dataset with a limited number of images, a three steps process	95%, 66msec (per license plate)
CR [34]	(Preprocessing) CNN (YOLOv3) + Tesseract	Use small number of images in dataset to reach a good AP	Focus on one step (CR), use preprocessing, use a network with high computational calculations	99.2%
LPD + CR (The Proposed Method)	**CNN (YOLOv4-tiny) + CNN, RNN, CTC**	**No need for CS, no preprocessing, a model with consuming less storage space, real-time application for LPD step, using small number of images in dataset to reach a good AP**	**Time consuming of the CR step**	**75.14%, 435msec**

Biographies

Soheila Hatami received her MSc degree in Mechatronics from Tarbiat Modares University, Tehran, Iran, in 2022. She obtained her BSc degree from University of Mohaghegh Ardabili, Ardabil, and Imam Khomeini International University (as a guest student), Qazvin, in 2017, all in Electrical Engineering (power branch). Her research interests are in deep learning, image processing, industrial automation, and autonomous vehicles.

Farideh Sadat Jamali received her MSc degree in Mechatronics from Tarbiat Modares University, Tehran, Iran, in 2023. She obtained her BSc degree in Electrical Engineering from Alzahra University, in 2020. Her research interests are in deep learning, image processing, computer vision, and autonomous vehicles.

Majid Sadedel is currently an Assistant Professor at Tarbiat Modares University, Tehran, Iran. He received his PhD degree from Tehran University, Tehran in 2016, MSc degree from Sharif University of Technology, Tehran in 2011, and BSc degree from Amirkabir University of Technology, Tehran in 2009, all in Mechanical Engineering. His research interests are in robotics, artificial intelligence, mechatronics, and industrial automation.