An Embedded Real-Time Automatic License Plate Recognition System Using YOLO Algorithm

Farideh Sadat Jamali[a], and Majid Sadedel[b,*]
a. *Faculty of Mechanical Engineering, Tarbiat Modares University, Tehran, Iran.* (E-mail address: f.jamali@modares.ac.ir)
b. *Assistant Professor, 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. Automatic License Plate Recognition (ALPR) is crucial in Intelligent Transportation System but faces challenges like weather and light conditions, camera angles, and license plate distortion. With advances in deep learning, as well as computing platforms, particularly GPUs, these algorithms have found major applications. The task becomes even more complex with Iranian license plates due to the strong similarity of some Persian characters, and the need for real-time processing is often overlooked. Consequently, this work proposes a two-stage deep learning-based algorithm for ALPR, with impressive precision and real-time applications. The methodology involves License Plate Detection (LPD) and Character Recognition (CR) using separate fine-tuned YOLOv5 networks, extracting characters in two sequential steps. The model shows robustness under challenging scenarios such as uneven lighting, low-quality images, and noise. Experimental results show an end-to-end mean Average Precision (mAP) of 95.5% and an inference speed of 23 Frames Per Second (FPS), meeting real-time requirements. Specifically, a mAP of 98.2% is achieved in the CR stage, effectively addressing character similarity issues. The developed model is implemented on the Jetson Nano, an embedded device, using DeepStream and demonstrates strong performance. For real-time detection, the TensorRT-based model deployed on the Jetson Nano achieved 6 FPS inference speed.

KEYWORDS: Automatic License Plate Recognition, YOLO, Embedded Platform, Deep Learning, Object Detection

1. Introduction

The Automatic License Plate Recognition (ALPR) system is a crucial component of the Intelligent Transportation System (ITS), which forms the basis of a Smart City. It is evident that ALPR systems are used everywhere, and this extent of application illustrates the usefulness of this technology. This system can be used for a variety of purposes, including: automatic toll payment, parking enforcement [1], traffic management [2], stolen vehicle recognition, automatic fines for offenders, security surveillance and etc. To meet the requirements of the mentioned wide range of applications, ALPR systems have been developed with appropriate properties according to each of them. Initially, ALPR systems used image processing algorithms like Edge Detection, Fourier Transform, Wavelet Transform, which were not as effective as modern technologies, according to published papers. Today, ALPR systems use deep learning methods and neural networks, as well as image processing algorithms. Since deep learning and neural networks are constantly evolving, it is essential to use high-performance and up-to-date networks for ALPR. A majority of the methodologies presented in the published articles have an accuracy over 85%, but because their datasets and hardware are not similar, it is not possible to evaluate and compare which has a better performance. There is no doubt that many of the proposed algorithms for ALPR will perform well under controlled conditions. While extensive research has been conducted on ALPR, there are still unresolved challenges that need to be addressed. Uncontrollable conditions can present many challenges such as improper lighting, inappropriate weather and environmental conditions, number of License Plates (LPs), image blurring and LP obstructions. Consequently, to overcome these challenges, nowadays, ALPR systems employ advanced image processing and computer vision techniques. These high computation techniques have become feasible as a result of improvements in the Graphics Processing Unit (GPU) and the revolution in deep learning approaches [3]. Currently, edge devices are a popular platform for implementing deep learning-based algorithms [4]. To accomplish this, a light weight model with high inference speed must be designed. General ALPR systems consist of four stages that run back-to-back: Vehicle Detection (VD), License Plate Detection (LPD), Character Segmentation (CS), and Character Recognition (CR). These stages can be adjusted according to the performance expectations and application requirements. However, ALPR models are usually presented with only two steps, LPD and CR, in order to boost model execution speed while maintaining high accuracy. This pipeline starts by detecting the LPs in the video frames. LPD phase requires a powerful Object Detection (OD) algorithm that detects LP with high accuracy. Due to the conflict between this step and the video frames extracted from the camera, the OD

algorithm must be very fast to ensure that no frame is lost. Once the detected LP has been separated from the original image, the characters on the LP will be recognized in the next step without any segmentation methods. In this system, all stages play a complementary role in determining its final performance. However, LPD has a special effect because it determines the quality of the input data in the process and must search for LPs in complex and variable scenes at a very high speed. This means that if the LPD phase is performed well, it is likely to prevent misclassification of other objects as LP and misidentification of LP characters in the next phase. This is the reason why some researches present a model that includes the VD stage before the LPD, because the vehicle has larger dimensions and is more immediately recognizable than the LP. This work increases the model processing time, which is not suitable for deployment of the model on edge devices. Aside from that, because some Persian characters are similar to one another and have been identified less often than English characters, research on ALPR algorithms is more focused on Iranian license plates. The main objective of this paper is to present an accurate, robust, and real-time ALPR model for edge devices. The strengths of our proposed model can be summarized in four key points: high processing speed and real-time capability, remarkable accuracy in recognizing similar characters, robustness under challenging and complex conditions, and compatibility with embedded systems featuring limited computational resources (suitability for deployment on edge devices). Therefore, we investigated and verified the existing methods for establishing the model on the Jetson Nano Developer Kit in order to select the one with the best performance index. In this study, the main contributions of the work are listed as below:

- We propose a two-stage deep learning model based on You Only Look Once (YOLO) for ALPR, which achieves a precision of 98.2% in recognizing Persian characters. This model effectively addresses the challenge of distinguishing similarly-shaped characters and demonstrates outstanding performance in both precision and speed compared to existing approaches. Therefore, the high processing speed and the remarkable accuracy in recognizing highly similar characters make our proposed model well-suited for real-time applications.
- We validate our proposed model on a small dataset containing challenging scenarios, such as low-quality images, unexpected lighting conditions, nighttime and noisy images, and other complex situations. The model's accuracy dropped by only about 1.5%, demonstrating its robustness under complex conditions.
- We implement our model on the Jetson Nano using the DeepStream SDK and conduct a performance comparison between the Jetson Nano and a GPU-enabled PC. The results highlight the Jetson Nano's suitability for mobile ALPR applications, demonstrating the feasibility of deploying our proposed model on mobile devices. As a result, one of the strengths of our presented model is its compatibility with edge devices, including those with limited computational resources.

The manuscript is organized as follows: Section 2 reviews related works on LPD and CR in summary. Section 3 describes the methodology in detail. Section 4 introduces the experimental setup, datasets and model deployment. Section 5 presents results and discusses about them. Finaly Section 6 concludes the paper and proposes future work.

2. Related Works

There have been several papers published so far on ALPR, and various technological solutions have been proposed. The proposed ALPR algorithm in this study involves the two main stages of LPD and CR; therefore, in literature, only works relevant to these two steps will be discussed. A number of recent articles [5] have also demonstrated that the segmentation stage can yield significant results. Taking a general perspective, the presented solutions can be divided into two categories. One perspective incorporates image processing approaches like edge-based [6], [7], [8], color-based [9] and texture-based [10] for LPD, which mainly based on the features of the LP. In CR, most of the methods in this category are used for preprocessing. The second perspective considers deep learning algorithms along with neural networks. Several recent studies have utilized Convolutional Neural Networks (CNNs) for both the LPD and CR steps [11], [12] because they have shown significant potential for solving such problems and other similar real-world issues like Face Mask Detection [13].

2.1 License Plate Detection

Sultan *et al.* [14] developed a systematic ALPR application. This study uses morphological image processing methods for LPD. Khan *et al.* [15] proposed an efficient model to detect the LPs using YOLOv5 algorithm for unconstrained real-world environment. Vaiyapuri *et al.* [16] introduced a model relies on traditional image processing for LPD, which includes a Madian filter to develope the image and remove noise, a Sobel edge detector filter to find the edges of the LP, and a morphological method to isolate the LP from the background. The LPD process is completed using two methods, dilation and erosion. Laroca *et al.* [17] presented a modified Fast-YOLOv2 model for LPD issue. This network structure has been modified in the following ways: layer 13 filter size has been changed, layer 14 has been added with 1024 filters and output layer filters have been reduced to 50. Tourani *et al.* [18] developed a flexible method based on YOLOv3 for Iranian LPD, which achieved 97.8% in term of accuracy. The LPD stage proposed by Pustokhina *et al.* [19] uses image processing methods to detect LP. The designed model is based on the Improved Bernsen algorithm (IBA) and Connected Component Analysis (CCA). To implement ALPR, Izidio *et al.* [20] considered an embedded platform, and based on that point of view, they have used the Tiny-YOLOv3 model, which is capable of high processing speeds, for LPD. Luo and Liu [21] improved the accuracy and execution speed of the YOLOv5m network by modifying its structure, and used this improved network to solve the LPD problem. The changes include the use of K-Means++ algorithm, the use of DIOU loss function, and the removal of the 20×20 feature map. With the aim of establishing the network on edge-based devices and increasing the speed of inference, Ashrafi *et al*. [22] employed a two-part real-time model including MobileNet SSDv2 and Hierarchical Haar classifier. The proposed network achieves 82.7% precision and 27.2 Frames Per Second (FPS) inference speed. Al-batat *et al.* [23] implemented all ALPR steps using YOLO. In this respect, Tiny-YOLOv4 was trained on five renowned datasets in this field and achieved an average accuracy of 99.16%. Silva and Jung [24] introduced an Improved Warped Planar Object Detection Network (IWPOD-NET) for LPD to detect four corners coordinates of LP. WPOD-NET originally used YOLO, SSD, and Spatial Transformer Networks (STN), but this study proposes affine transformations alongside them for OD. This feature rectifies the LP view to Fronto-Parallel. In a similar manner to [24], [25] additionally addressed the issue of LP rectification using the perspective transformation method. Guatam *et al.* [25] applied regression as well as a loss function which is based on average sum of Euclidean distance to train CNNs for LP correction.

2.2 Character Recognition

In [26], morphological methods were used to identify vehicle LP characters, similar to [14]. Kaur *et al.* [26] suggested an approach based on a combination of pre-processing, morphological operations and CNN techniques for CR. In the initial stages of building the model, gray scaling, median filtering, thresholding, and masking are applied as pre-processing techniques. In order to extract features from the image, morphological operations are used after the image quality has been improved. A specific structure of CNN is then used to classify and identify the characters on the LP. Khan *et al.* [15] proposed a robust neural network architecture to recognize characters of LP based on CNN, which is made up of sixteen convolutional layers with a kernel size of 64, 128, 256 and 512 to extract advanced features, five maxpooling layers at the end of convolutional layers and two fully connected layers. Vaiyapuri *et al.* [16] proposed a deep learning-based model for CR, which uses indistinct CNN with Squirrel Search Algorithm (SSA) for parameter optimization. He and Hao [27] presented a hybrid structure that combines feature extraction, sequential modeling, and predictive modeling for CR. This structure was tested using several techniques for each term, and the best results were announced. For feature extraction, ResNet is used, followed by Bi-directional Long Short Term Memory (LSTM), and attention-based modeling is used for prediction. Izidio *et al.* [20] developed a deep learning based architecture for recognizing characters in real time. In the shallow CNN, three convolutional layers have filter sizes of 48, 64, and 128 and two fully connected layers are used to produce the outputs. The articles [28] and [29], however, consider CR in the category of Natural Language Processing (NLP), in which characters are viewed as sequential data with a special order. Shi and Zhao [29] proposed a CR method that combines Gated Recurrent Units (GRU) and Connectionist Temporal Classification (CTC), which improves convergence speed and reduces training time. The model presented by Li *et al.* [28] is similar to that presented in the previous work. However, they used LSTM instead of GRU to achieve

their goal. Chen and Hendry [3] developed a method for recognizing Taiwanese LPs, which trains separate networks for each character. This study designed and trained 36 single-class networks with similar structures to YOLO, which has a lighter architecture than YOLO, using the Sliding Window detection process.

3. Methodology

ALPR pipeline, which illustrates in Figure 1, comprises two main stages: LPD and CR. To begin with, the fine-tuned and trained YOLOv5s model, is used to detect LPs in each video frame. Based on the predicted coordinates of the bounding boxes, the detected LPs are separated from the original image. Next, the characters on the cropped LP are detected using the trained YOLOv5m model. In order to recognize the LP, the detected characters are sorted in accordance with their x coordinates. According to Figure 1, the model will output a string of numbers and letters. For both LPD and CR stages, the YOLOv5 model are used, but before proceeding, we would like to explain why this algorithm was selected. Our criteria for selecting an algorithm are fast execution speed, lightweight model, and superior precision respectively. Due to the purpose of this paper being to deploy the designed model on one types of edge devices, namely NVIDIA Jetson Nano, it is imperative that the algorithm selected can successfully run in real-time scenarios and at a high speed. Additionally, the algorithm should be small in size and capable of being implemented on the Jetson Nano board based on the technical specifications of the board (GPU compute capability, RAM uasge limitation). Similarly, to any other algorithm, high precision is also critical for us, and the designed algorithm must be implemented on the mobile devices with acceptable performance. OD refers to the process of locating and recognizing objects in a digital image. LPD is classified under the OD concept due to its similarity, and the algorithms used for it may also be used for LPD. Additionally, the proposed model treats CR as an OD task, where each character is treated as an object. In a review of other articles for CR, it has been observed that the use of algorithms such as Recurrent Neural Networks (RNNs), GRUs and LSTMs significantly increases the model execution time, which suggests that the designed model will not be suitable for real-time execution. Furthermore, since our proposed algorithm does not include the CS stage, it will be excluded as a classification task. As a result, our developed sequential algorithm is entirely based on the concept of OD.

Figure 1.

At present, YOLO [30] and Region-based Convolutional Neural Networks (R-CNN) [31] are widely used and popular OD algorithms. There are significant differences between these two algorithms in terms of their structure and performance. For structure, the YOLO family, which includes YOLOv2 [32], YOLOv3 [33], YOLOv4 [34] and YOLOv5 [35], are single-stage detectors, while the R-CNN family, which includes R-CNN [31], Fast R-CNN [36] and Faster R-CNN [37], are two-stage detectors. First, the R-CNN algorithm generates areas in which objects may be found, and then uses a neural network to assign these areas to the desired objects. In this regard, this type of detector is more accurate than single-step detectors. Despite this, the network inference speed is slower than YOLO due to the multi-steps in the detection process. The YOLO family, is well known for their fast execution speed and real-time applications. YOLO identifies and locates objects in one pipeline. Compared to R-CNN, this algorithm achieves an acceptable precision, but it is lower than R-CNN precision. Therefore, by creating a trade-off between precision and speed and considering that the model will be deployed on the edge devices, Jetson Nano Developer Kit, the YOLO algorithm is a more effective choice for this study. YOLOv5 is the latest version at the time of this study. This version has two advantages over its predecessor, YOLOv4: 1. The YOLOv5 weight file is 90% smaller in size. 2. The speed of its inference is faster. From another perspective, YOLOv4 performs slightly better in precision. However, the above advantages prove that YOLOv5 is a suitable algorithm for deploying on embedded devices for real-time implementations. In comparison with its previous versions, this version of YOLO provides higher accuracy and faster inference speed. The YOLOv5 network architecture shown in Figure 2, which was modeled based on [38], consists of three components. To extract significant features from the incoming image, CSPDarknet is employed as the backbone component. Shortly, CSPDarknet is superior due to its ability to address heavy processing calculations in its network architecture. A Path Aggregation Network (PAN) is used in the neck layer to gather feature mapping (aggregation of parameters from different levels of the backbone). YOLO is also used as the head, which gives outputs

such as class, precision score, location and size information. A major difference between YOLOv5 and other versions is the ability to automatically train bounding box anchors (auto-anchors) based on genetic algorithms and K-means techniques.

Figure 2.

The YOLOv5 network architecture consists of four models: YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x. Differences between them can be attributed to the number of feature extraction modules and the number of convolutional filters in the network. This is also apparent in the number of parameters in the model. When selecting among the four models described above, the complexity of the problem and the algorithm design requirements should be considered, for example, making a trade-off between precision and speed or deploying on edge devices.

3.1 License Plate Detection

The least complicated YOLOv5 model, YOLOv5s, is used for the LPD network. The LPD issue has one object class (License Plate) whose object is not particularly complicated due to its shape and its features are relatively straightforward to extract. Moreover, since LPD is the first step of the ALPR model, its inference speed should be as fast as possible in order to process video frames at a high rate. The present study proposes a YOLOv5s architecture that uses 640 × 640 pixels as the input image dimensions during training. As a default, YOLOv5s utilizes Stochastic Gradient Descent (SGD), which has been replaced by the Adam optimizer in our proposed algorithm. The following claims can be made based on the experiments performed using both Adam and SGD optimizers:

1. Adam optimizer's adaptive learning rate is an advantage over SGD, so it does not require as much attention to determine the optimal learning rate.
2. Training duration can be reduced and convergence can be accelerated by adjusting the optimizer to Adam.
3. The Adam optimizer's precision is approximately 3% higher than SGD when other hyperparameters are held constant.

A batch size of 16 is selected based on the GPU's memory limit. Also, weight decay and momentum should be unchanged at 0.0005 and 0.937. This will stabilize the gradient and avoid updating weights based on common features, thus reducing the impact of the imbalance dataset. Changes in data augmentation techniques are also considered. We used three techniques for data augmentation: Mosaic, Fliplr, and Scale. As Mixup and Flipud change significant features that can be extracted from input, they have not been utilized in this problem as they would result in incorrect training of the network. Table 1 contains details of the parameters set for network training.

Table 1.

When training a network, the default anchor values are checked at the beginning of the process, and if they do not fit, new anchor values are suggested so that it can be trained again using those values in the future. Despite the fact that the anchor values are completely appropriate for this problem and the data, the fit rate was reported to be 0.998. Transfer learning has been used to train the network. By using these weights, network learning is more efficient and faster, and learning is transferred more effectively.

3.2 Character Recognition

The model used for CR is YOLOv5m, which includes more parameters than the model used for LPD. In the CR issue, there are 28 object classes (Letters and Numbers), some of which have intricate and similar shapes, thus making it more challenging to extract their features. Table 2 illustrates the remarkable similarity between some Persian characters. In this study, one of the objectives was to develop a model that is capable of recognizing similar Persian characters accurately. Additionally, the blurring of the entrance LP image from the previous step requires a more sophisticated network. As CR is the second stage of the model designed for ALPR, its inference speed can be slower than that of LPD. The present study proposes a YOLOv5m architecture that uses 416 × 416 pixels as the input image dimensions during training. On the basis of the reasons mentioned in the previous section, the network optimizer is

changed to Adam for training this network. Batch size, weight decay, and momentum are set like the LPD network. As opposed to the LPD stage, the Fliplr data augmentation technique was not used to train this network, since it alters the characters and the network extracts false features as a result. Table 3 summarizes the data augmentation methods used to train the CR network and the tuned learning rate for it. According to an analysis of the CR dataset, there are few images with rotation, although rotation is likely to occur in reality. Therefore, the rotation data augmentation technique was also used to train the network. Training began with a check of the default anchors, which reported a fit rate of 1. This indicates that the default anchors were appropriate, and the network did not need to be retrained with suggested anchors. Similarly, to the previous section, transfer learning has also been used to train the network.

Table 2.

Table 3.

4. Experimental Setup

This section presents the architecture of the systems and datasets employed in this study. Also, this section describes the structure and how the designed model can be deployed on Jetson Nano with DeepStream. The training procedure and experiments were executed on computer with an Ubuntu 20.04 OS, Intel Core i7 3.50 GHz CPU, 32 GB RAM for CPU (DDR4), NVIDIA GeForce GTX 1050i GPU and 4 GB RAM for GPU. To train, fine-tune, test and deploy the YOLOv5 model, we must first clone YOLOv5 repository and install its requirements according to the documentation provided for it. Using the above PC, both networks were trained. Jetson Nano was only used for testing and deployment of the end-to-end prototype. The Jetson Developer Kit from NVIDIA is one of the most widely used edge devices for artificial intelligence applications. It is equipped with accelerators that can infer and deploy algorithms in real-time. The minicomputer runs deep learning algorithms and neural networks in parallel for applications such as image classification, OD, and segmentation using Compute Unified Device Architecture (CUDA), which enables the simultaneous execution of complex calculations. Jetson minicomputers are categorized according to their configuration, such as GPU capability, CPUs, RAM size, pinout, and other peripheral components. This research is also conducted on Jetson Nano, one of the simplest board in this family. According to the technical specifications of this board, its RAM is insufficient to deploy heavy and complex models.

4.1 Datasets

To train each sub-network of the proposed two-stage model, two separate datasets were used. The dataset used for LPD consists of 6306 samples. The dataset was compiled from three datasets. Table 4 lists the titles and percentages of each of them. Both the purchased dataset and the public dataset are available separately on the Internet. On request, researchers can access the dataset collected by the authors of this study.

Table 4.

In the purchased dataset, samples are often taken close to the vehicle, and only one LP is labeled per image. The same condition applies to the second tier (public dataset) which is publicly available online. To achieve diversity in the dataset, the manually labeled dataset has been added to it, whereas the more samples, the better the network can be trained. A variety of factors have been considered in photography, including: different lighting and environmental conditions (There are also images taken at night), a variety in the number of LPs, different distances from the vehicle. As a result of the mentioned cases, we have created a dataset with a large variety, allowing the model to see all possible states for the LP. This study used a public dataset containing 1643 images and 28 classes to recognize Persian characters. It has 18 letters and numbers 0-9 in its classes. This dataset does not include letters that have a very limited and special use, and for that reason, they are not considered. Since Persian letters cannot be used as labels in the same way as English letters, an English equivalent has been used. Based on the dataset, Figure 3 shows how many instances are present in each class. There are almost no differences in frequency between the numbers except for the number 0.

Figure 3.

An Iranian LP structure explains the frequency comparison between numbers and letters. Only three letters (alef, pe, te and malol) are less frequent than other letters due to their special use in reality. Sample distribution in this dataset is consistent with the appearance characteristics of Iranian LPs and has a reasonable degree of diversity.

4.2 Model Deployment Using DeepStream SDK

The DeepStream SDK from NVIDIA is a comprehensive streaming analytics toolkit based on GStreamer for AI-based multi-sensor processing, video, audio, and image understanding. Several plugins can be added to this software development environment. Developers can create a path for data processing that includes deep learning methods, neural networks and other complex processing functions. As this advanced environment provides real-time processing of video and image data, it has been used for this study. The DeepStream SDK offers the advantage of allowing models to be run with the TensorRT optimizer. NVIDIA TensorRT is a powerful inference optimizer and runtime that offers low latency and high performance for deep learning inference applications. For this study, DeepStream pipeline and plugins have been customized. In Figure 4, we show a step-by-step breakdown of the DeepStream pipeline for deployment of LPD and CR models. Video frames are captured in real-time by the IP camera, and then the video is decoded. Video frames are captured in real-time by the IP camera, and then the video is decoded. Multiple video streams are aggregated and batched using the Gst-streammux plugin. After that, batched videos are inferred with cascaded models in order to detect the LP and the characters associated with it.

Figure 4.

Gst-nvinfer plugin, which utilizes TensorRT-based inference for detection and classification, is customized to accomplish this. According to Figure 4, the plugin is defined once in PGIE format for LPD, and once in SGIE format for CR, both of which are based on YOLOv5 trained networks. Furthermore, the Gst-Dsexample plugin is customized to save PGIE outputs (cropped detected LPs) in a specified directory. Using the Gst-nvosd plugin, the ALPR process output, which includes the specified objects and their associated text, is displayed on a video frame. By configuring the KITTI format, Kitti-output and Kitti-output-track functions, the output of each frame (labels, bounding box coordinates and precision scores) is saved in a text file according to the desired format (characters detected for each LP are listed next to each other in order).

5. Results

In this section, we analyze the effectiveness of the proposed model using various metrics. Also shows how the model performs under varied test conditions. LPD and CR models were evaluated using widely used OD metrics, such as *Precision*, *Recall*, *F1 Score* and *mean Average Precision (mAP)*. A mathematical explanation of these validation criteria can be found in Equations 1-4. In these equations, a *True Positive* indicates that the LP (or characters) is actually visible and correctly predicted as LP during the detection process. *False Positives* occur when the LP is not in the input image, but the model predicts other objects as LPs. *False Negatives* occur when the LP is present in the input image but incorrectly predicted.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (1)$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (2)$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (3)$$

$$mAP = \frac{\sum_{i=1}^{M} (\frac{1}{n} \sum_{i=1}^{n} Precision_i)}{M} \quad (4)$$

Precision indicates the percentage of correctly detected license plates out of all the plates identified by the model, highlighting its ability to avoid false positives. *Recall*, on the other hand, shows how many of the actual license plates were successfully detected, focusing on reducing false negatives. The *F1 score* combines *Precision* and *Recall* into a single metric by calculating their harmonic mean, offering a balanced perspective when both metrics are equally important. Lastly, *mAP* provides a comprehensive evaluation of the model's performance by averaging precision

values across different recall thresholds, making it a widely used standard for object detection tasks. For both networks, these metrics have been calculated.

5.1 License Plate Detection

In total, 5045 images were used to train the YOLOv5s model, and 1261 images were used for validating. In Table 5, the results of the LPD algorithm are presented. The complexity and size of the model were determined using two metrics, GFLOPs and Parameters, in addition to performance efficiency measures like mAP, recall, precision and inference speed. This table illustrates the performance of the YOLOv5s trained model for LPD. A significant aspect of the algorithm's performance is its inference speed, which is approximately 66 FPS for images and approximately 90 FPS for videos. There is a high enough speed at this stage of the model, which deals with the input, such as video frames, to avoid losing frames and LPs. Also, the model has high precision for LPD with a mAP of 97.6%, which is obtained by using problem-appropriate data augmentation techniques, a comprehensive and wide-ranging dataset, a robust Adam optimizer, and a tuned learning rate. YOLOv5s can accurately detect LPs, as shown in Figure 5.

Table 5.

Figure 5.

5.2 Character Recognition

In total, 1314 images were used to train the YOLOv5m model, and 329 images were used for validating. In Table 6, the detailed results of the CR algorithm are presented. This table illustrates the optimized performance of the YOLOv5m trained model for CR. A significant aspect of the algorithm's performance is its inference speed, which is approximately 56 FPS for images. Also, the model has high precision for CR with a mAP of 98.2%, which is obtained by using problem-appropriate data augmentation techniques, a balanced and diverse dataset, a reliable Adam optimizer, and a tuned learning rate. YOLOv5m can accurately detect characters, as shown in Figure 6.

Table 6.

Figure 6.

According to the results (Figure 6), the trained algorithm for CR has performed well in recognizing similar characters. Figure 6 illustrates scenarios where the trained YOLOv5m model successfully distinguishes between similar Persian characters. For instance, the similar characters “9” and “vav” are correctly recognized in Figure 6(c). Similarly, accurate recognition of the similar characters “5” and “he”, “9” and “1”, “6” and “ain” can be observed in Figures 6(d), 6(e), and 6(f), respectively. Although both data augmentation and the YOLOv5m model structure play significant roles in the model’s performance, data augmentation can be considered the key factor in improving the recognition of visually similar characters. Data augmentation directly impacts the diversity of the training data, exposing the model to a broader range of similar characters and challenging conditions. This technique helps the model learn critical features that enable it to distinguish between similar characters in real-world scenarios. A model trained with appropriate data augmentation techniques develops enhanced feature extraction and attention mechanisms, resulting in more accurate classification of similar characters.

Figure 7 shows the performance of the model based on the confusion matrix. This matrix provides information about the true and predicted values of the classes and indicates how precisely the model predicts the classes. The x-axis is the true labels and the y-axis is the predicted labels. For instance, the value of 0.98 highlighted in bold indicates that the model correctly predicted the character “ain” with 98% accuracy. The 0.02 value marked at the top indicates that the 0.02 value was incorrectly estimated as the number “8”. Class “ain” and class “8” have very similar appearances. Confusion matrix is helpful in understanding which classes are being confused by the model as other classes, like above example.

Figure 7.

5.3 End-to-End Process

Until now, the analysis results have been provided individually for each phase. The two trained YOLOv5 networks for LPD and CR have now been integrated and deployed hierarchically. Below are the results of an end-to-end evaluation of the suggested methodology on two platforms, the PC and the Jetson Nano board (which possess limited computational resources). As explained earlier, to carry out this process on the Jetson Nano board, the DeepStream SDK is utilized. Figure 8 illustrates the results of running the proposed end-to-end ALPR algorithm on various typical real-world scenarios on a PC.

Figure 8.

To evaluate the robustness of the proposed end-to-end ALPR model, we tested it on a challenging dataset (hard dataset) containing images with snow, noise, low-quality, nighttime, uneven lighting and other complexities. The model achieved an accuracy of 94%, demonstrating a minimal 1.5% decrease compared to its performance on the standard dataset. This slight reduction in precision is reasonable and expected due to the difficulty of the dataset, as such conditions often lead to a significant drop in model performance. The robust performance of the model can be attributed to the use of effective data augmentation techniques and the advanced architecture of YOLOv5, which enabled the model to generalize well even under challenging conditions. This result highlights our model's strength and potential for real-world applications where such complexities are common. Figure 9 illustrates the performance of the proposed model on several samples from the challenging and complex dataset. The results in this section highlight one of the key strengths of our proposed approach: its significant accuracy in challenging scenarios and complex conditions.

Figure 9.

Additionally, Figure 10 illustrates the output of the real-time execution of the proposed model on the Jetson Nano, and the resulting text files are saved for each frame processed. A comparison of the precision and inference speed of the ALPR algorithm on two devices, the PC and Jetson Nano, is presented in Table 7. Based on the end-to-end experiment results presented in the table, the ALPR network showed a precision of 95.5%, while the precision of the LPD module was 97.6% and the precision of the CR module was 98.2%. It is due to the quality of the LP images in the CR dataset and the cropped LP for the CR phase when tested on real-world samples. Accordingly, the quality of the LP image when cropped and given to the CR stage is lower than the quality of the CR dataset used to train the network. Because of this, the end-to-end accuracy is slightly lower than the two stages that make it up. As a comparison, the precision of the same model on the Jetson Nano board is 93%, which is about 2.5% less than the precision of the model on the PC. There are three reasons for this issue: 1. For the two devices, the experimental samples are different. 2. Testing conditions differ between the two devices (the Jetson Nano board was tested in real-time with the camera, whereas the PC test was conducted using a recorded video) 3. The DeepStream SDK has been used to deploy the model on the Jetson Nano board, and the results presented in the studies about this toolbox indicate that converting the model format to engine (using the TensorRT optimizer) results in a slight loss of accuracy. However, the developed ALPR algorithm has high and significant overall precision.

Table 7.

Figure 10.

As for execution time, the proposed algorithm achieved an overall inference speed of approximately 23 FPS on PC, which proves its real-time functionality. The algorithm's inference speed is estimated to be around 6 FPS on the Jetson Nano device. According to NVIDIA's website, the Compute Capability of the Jetson Nano device is 5.3, whereas for the PC used in this study, it is 6.1. Based on the published articles regarding the deployment of deep learning-based models on various Jetson Developer Kits [39], we can estimate that the proposed ALPR algorithm will operate at approximately 24 FPS or higher on a more advanced Jetson devices, for example Jetson Xavier NX.

5.4 Comparison of the Proposed Method with Previous Methods

We compared our proposed method with other state-of-the-art methods published recently, as shown in Table 8, in terms of accuracy and inference time. Our proposed method outperforms the methods presented in [40], [18], [41] and [42] in terms of the accuracy of both the LPD and CR stages, end-to-end accuracy, and inference time. The end-to-end precision of [18], at 95.05%, is very close to our model's precision of 95.5%. However, our model is approximately three times faster, which is a significant advantage.

[43] achieved an accuracy of 98% in the LPD stage and 98.8% in the CR stage, which are 0.4% and 0.6% higher than the respective accuracies of our method's stages. However, as emphasized throughout this paper, the strength of our method lies in its real-time capability and low execution time while maintaining high accuracy. In [43], only the execution time for the LPD stage is reported as 23 msec, whereas our model completes this stage in approximately 15.15 msec, which is about 1.5 times faster. Additionally, according to the original Faster R-CNN paper, the processing time for a single image is approximately 140 msec. Based on this estimation, the end-to-end execution time of the method proposed in [43] can be inferred to exceed 160 msec, which is roughly four times the total execution time of our proposed model and this is one of the strengths of our approach.

Table 8.

5.5 Analysis of Misclassification

Although our proposed model demonstrates high accuracy, it encounters errors in certain specific scenarios. Therefore, in this section, we analyze instances where the model misclassifies characters. This analysis aims to identify the model's weaknesses and the challenges it faces, allowing us to propose potential solutions for improvement.

Based on our research and the confusion matrix data, when the angle of the vehicle's license plate exceeds a certain threshold and the image quality is not standard, the character "he" is often misclassified, particularly as "ye", and "mim" is misclassified as the digit "9". These misclassifications are illustrated in Figure 11(a) and Figure 11(b), respectively. The first issue arises due to the limited number of "he" samples in the dataset, which prevents the model from being exposed to diverse instances of this character, leading to misclassification. The second issue is caused by the visual similarity between "mim" and "9", as both share a circular shape at the top, which confuses the model.

Figure 11.

To address the issue with the character "he", increasing the amount of training data and applying particular data augmentation techniques can be beneficial. For the problem involving "mim" and the digit "9", utilizing finer-grained features, optimizing the model architecture, and incorporating post-processing techniques could help improve performance. Implementing these strategies is likely to enhance the model's accuracy and reduce the occurrence of such errors.

6. Conclusions

This paper proposes a deep learning-based detection algorithm for automatic license plate recognition, which consists of two stages: License Plate Detection (LPD) and Character Recognition (CR). Our method offers several advantages, including high accuracy, the capability to address the challenges posed by visually similar Persian characters, real-time execution, and compatibility with edge devices. Additionally, it demonstrates significant robustness under challenging conditions and achieves superior processing speed, making it suitable for real-time applications. Our approach relies on two sequential YOLOv5 deep convolutional neural networks for this objective. Both networks were trained using a combination of transfer learning strategies and appropriate data augmentation techniques. Our algorithm has been implemented on two different devices: A Personal Computer (PC) and a Jetson Nano device. On PC, the inference speed of the trained model is 23 FPS and the precision of the trained model is 95.5%. The inference speed of the model is measured at 6 FPS by implementing the model on Jetson Nano using DeepStream and performing real-time testing. We discussed the results recorded on both devices and compared them. This study demonstrates that

the proposed ALPR model performs satisfactorily and efficiently on GPU platforms. For future work, we intend to explore advanced neural network models to address the misclassified characters and compare their results with the existing model. Furthermore, apply the developed models on other embedded platforms, including more advanced Jetson developer kit products.

Declarations

Conflict of interest The authors declare that they have no confict of interest.
Availability of data and materials Data will be made available on reasonable request.

References

[1] Dalarmelina, N. do V., Teixeira, M. A. and Meneguette, R. I., "A real-time automatic plate recognition system based on optical character recognition and wireless sensor networks for ITS," *Sensors (Basel).*, vol. **20**, no. 1, 2019, doi: 10.3390/s20010055.

[2] Kosuru, V. S., Venkitaraman, A. K., Chaudhari, V. D., et al., "Automatic identification of vehicles in traffic using smart cameras," *2022 5th Int. Conf. Contemp. Comput. Informatics*, pp. 1009–1014, 2022, doi: 10.1109/IC3I56241.2022.10072979.

[3] Hendry and Chen, R.-C., "Automatic license plate recognition via sliding-window Darknet-Yolo deep learning," *Image Vis. Comput.*, vol. **87**, pp. 47–56, 2019, doi: https://doi.org/10.1016/j.imavis.2019.04.007.

[4] Shashirangana, J., Padmasiri, H., Meedeniya, D., et al., "License plate recognition using neural architecture search for edge devices," *Int. J. Intell. Syst.*, no. January, pp. 1–38, 2021, doi: 10.1002/int.22471.

[5] Zibani, R., Sebbak, F., El Yazid Boudaren, M., et al., "Multi-attribute fusion-based approach for Algerian automatic license plate recognition," *Multimed. Tools Appl.*, vol. **83**, pp. 30233–30259, 2023, doi: https://doi.org/10.1007/s11042-023-16789-6.

[6] Massoud, M. A., Sabee, M., Gergais, M., et al., "Automated new license plate recognition in Egypt," *Alexandria Eng. J.*, vol. **52**, no. 3, pp. 319–326, 2013, doi: 10.1016/j.aej.2013.02.005.

[7] Luo, L., Sun, H., Zhou, W., et al., "An efficient method of license plate location," *First Int. Conf. Inf. Sci. Eng.*, no. 2, pp. 770–773, 2009, doi: 10.1007/978-3-642-27966-9_81.

[8] Akoushideh, A., Shahbahrami, A. J., and Afshany, A., "Parallelization of license plate localization on GPU platform," *Multimed. Tools Appl.*, vol. **83**, pp. 2551–2564, 2023, doi: https://doi.org/10.1007/s11042-023-15656-8.

[9] Wang, F., Man, L., Wang, B., et al., "Fuzzy-based algorithm for color recognition of license plates," *Pattern Recognit. Lett.*, vol. **29**, pp. 1007–1020, 2008, doi: 10.1016/j.patrec.2008.01.026.

[10] Wang, Y., Lin , W., and Horng, S., "A sliding window technique for efficient license plate localization based on discrete wavelet transform," *Expert Syst. Appl.*, vol. **38**, pp. 3142–3146, 2011, doi: 10.1016/j.eswa.2010.08.106.

[11] Silva, S. M. and Jung, C. R., "Real-time Brazilian license plate detection and recognition using deep convolutional neural networks," *Proc. - 30th Conf. Graph. Patterns Images, SIBGRAPI 2017*, no. June 2018, pp. 55–62, 2017, doi: 10.1109/SIBGRAPI.2017.14.

[12] Silva, S. M. and Jung, C. R., "Real-time license plate detection and recognition using deep convolutional neural networks," *J. Vis. Commun. Image Represent.*, vol. **71**, 2020, doi: 10.1016/j.jvcir.2020.102773.

[13] Khoramdel, J., Hatami, S., and Sadedel, M., "Wearing face mask detection using deep learning through COVID- 19 pandemic," *Sci. Iran.*, vol. **30**, no. 3, pp. 11058–1067, 2023, doi: 10.24200/sci.2023.59093.6057.

[14] Sultan, F., Khan, K., Shah, Y. A., et al., "Towards automatic license plate recognition in challenging conditions," *Appl. Sci.*, vol. **13**, no. 6, 2023, doi: 10.3390/app13063956.

[15] Khan, I. R., Ali, S., Siddiq, A., et al., "Automatic license plate recognition in real-world traffic videos captured in unconstrained environment by a mobile camera," *Electronics*, vol. **11**, no. 1408, 2022, doi: https://doi.org/10.3390/electronics11091408.

[16] Vaiyapuri, T., NandanMohanty, S., Sivaram, M., et al., "Automatic vehicle license plate recognition using

optimal deep learning model," *Comput. Mater. Contin.*, vol. **67**, no. 2, pp. 1881–1897, 2021, doi: 10.32604/cmc.2021.014924.

[17] Laroca, R., Zanlorensi, L. A., Gonc, G. R., et al., "An efficient and layout-independent automatic license plate recognition system based on the YOLO detector," *IET Intell. Transp. Syst.*, 2021, doi: 10.1049/itr2.12030.

[18] 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.

[19] Pustokhina, I. V., Pustokhin, D., Rodrigues, J., et al., "Automatic vehicle license plate recognition using optimal K-means with convolutional neural network for intelligent transportation systems," *IEEE Access*, 2017, doi: 10.1109/ACCESS.2020.2993008.

[20] Izidio, D., Ferreira, A., Medeiros, H., et al., "An embedded automatic license plate recognition system using deep learning," *Des. Autom. Embed. Syst.*, vol. **24**, pp. 23–43, 2019, doi: https://doi.org/10.1007/s10617-019-09230-5.

[21] 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.

[22] Ashrafee, A., Irbaz, M. S., Al Nasim, A., et al., "Real-time Bangla license plate recognition system for low resource video-based applications," *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, pp. 479–488, 2022, doi: http://dx.doi.org/10.1109/WACVW54805.2022.00054.

[23] 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**, no. 23, 2022, doi: 10.3390/s22239477.

[24] Silva, S. M., and Jung, C. R., "A flexible approach for automatic license plate recognition in unconstrained scenarios," *IEEE Trans. Intell. Transp. Syst.*, vol. **23**, no. 6, pp. 5693–5703, 2021, doi: https://doi.org/10.1109/TITS.2021.3055946.

[25] Gautam, A., Rana, D., Aggarwal, S., et al., "Deep learning approach to automatically recognise license number plates," *Multimed. Tools Appl.*, vol. **82**, no. 20, pp. 31487–31504, 2023, doi: 10.1007/s11042-023-15020-w.

[26] Kaur, P., Kumar, Y., Ahmed, S., et al., "Automatic license plate recognition system for vehicles using a cnn," *Comput. Mater. Contin.*, vol. **71**, no. 1, pp. 35–50, 2022, doi: 10.32604/cmc.2022.017681.

[27] He, M. X., and Hao, P., "Robust automatic recognition of Chinese license plates in natural scenes," *IEEE Access*, vol. **8**, pp. 173804–173814, 2020, doi: 10.1109/ACCESS.2020.3026181.

[28] Li, H., Wang, P., You, M., et al., "Reading car license plates using deep neural networks," *Image Vis. Comput.*, vol. **72**, pp. 14–23, 2018, doi: 10.1016/j.imavis.2018.02.002.

[29] Shi, H., and Zhao, D., "License plate recognition system based on improved YOLOv5 and GRU," *IEEE Access*, vol. **11**, pp. 10429–10439, 2023, doi: 10.1109/ACCESS.2023.3240439.

[30] Redmon, J., Divvala, S., Girshick, R., et al., "You only look once: Unified, real-time object detection," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. **2016-Decem**, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.

[31] 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.

[32] Redmon, J., and Farhadi, A., "YOLO9000: Better, faster, stronger," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. **2017-Janua**, pp. 6517–6525, 2017, doi: 10.1109/CVPR.2017.690.

[33] Redmon, J., and Farhadi, A., "YOLOv3: An incremental improvement," *arXiv Prepr. arXiv1804.02767.*, 2018, doi: https://doi.org/10.48550/arXiv.1804.02767.

[34] Bochkovskiy, A., Wang, C.-Y., and Liao, H. M., "YOLOv4: Optimal speed and accuracy of object detection," *arXiv Prepr. arXiv2004.10934.*, 2020, doi: https://doi.org/10.48550/arXiv.2004.10934.

[35] Jocher, G., "YOLOv5 Code Repository," *Github*, 2020. https://github.com/ultralytics/yolov5

[36] Girshick, R., "Fast R-CNN," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. **2015** Inter, pp. 1440–1448, 2015, doi: 10.1109/ICCV.2015.169.

[37] Girshick, R., Donahue, J., Darrell, T., et al., “Rich feature hierarchies for accurate object detection and semantic segmentation,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 580–587, 2014, doi: 10.1109/CVPR.2014.81.

[38] Xu, R., Lin, H., Lu, K., et al., “A forest fire detection system based on ensemble learning,” *Forests*, vol. **12**, no. 2, pp. 1–17, 2021, doi: 10.3390/f12020217.

[39] Guney, E., Bayilmis, C., and Cakan, B., “Correction: An implementation of real-time traffic signs and road objects detection based on mobile GPU platforms,” *IEEE Access*, vol. **10**, pp. 86191–86203, 2022, doi: 10.1109/ACCESS.2022.3209832.

[40] Hatami, S., Jamali, F. S., and Sadedel, M., “Iranian license plate recognition using a reliable deep learning approach,” *Sci. Iran.*, pp. 1–17, 2024, doi: 10.24200/SCI.2024.61312.7245.

[41] Tabrizi, S. S., and Cavus, N., “A hybrid KNN-SVM model for iranian license plate recognition,” *Procedia Comput. Sci.*, vol. **102**, no. August, pp. 588–594, 2016, doi: 10.1016/j.procs.2016.09.447.

[42] Ashtari, A. H., Nordin, M. J., and Fathy, M., “An Iranian license plate recognition system based on color features,” *IEEE Trans. Intell. Transp. Syst.*, vol. **15**, no. 4, pp. 1690–1705, 2014, doi: 10.1109/TITS.2014.2304515.

[43] Shahidi Zandi, M., and Rajabi, R., “Deep learning based framework for Iranian license plate detection and recognition,” *Multimed. Tools Appl.*, vol. **81**, no. 11, pp. 15841–15858, 2022, doi: 10.1007/s11042-022-12023-x.

Figures' captions:

Tables's captions:

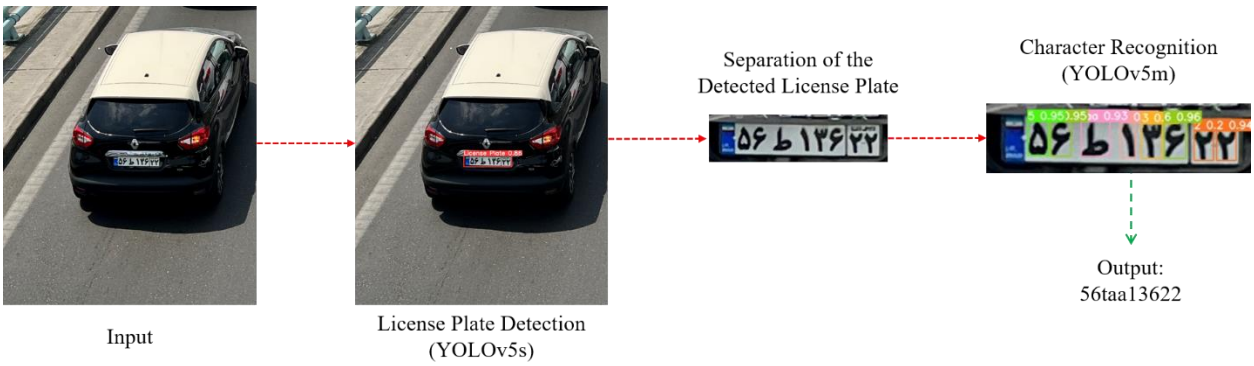

Figure 23.

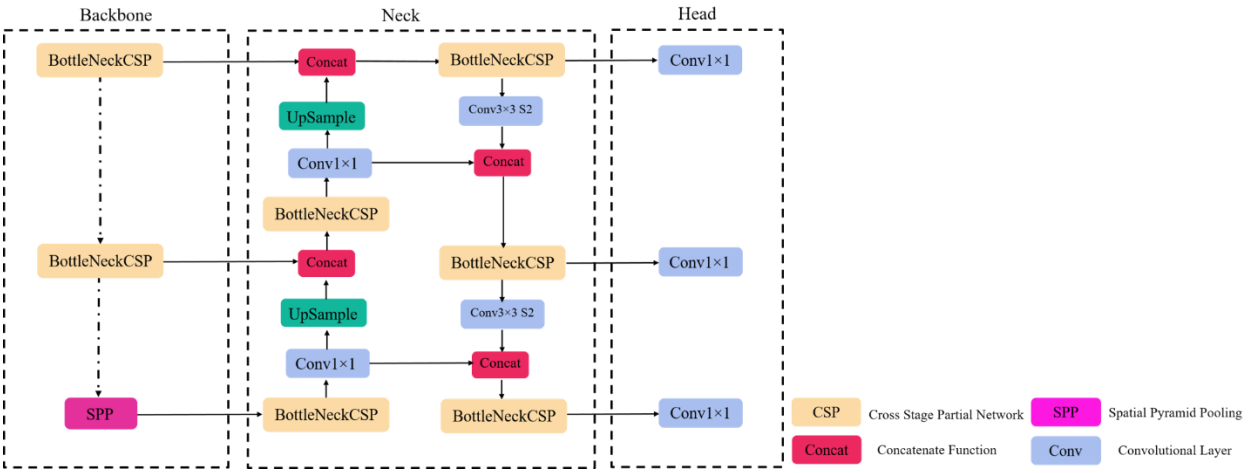

Figure 24.

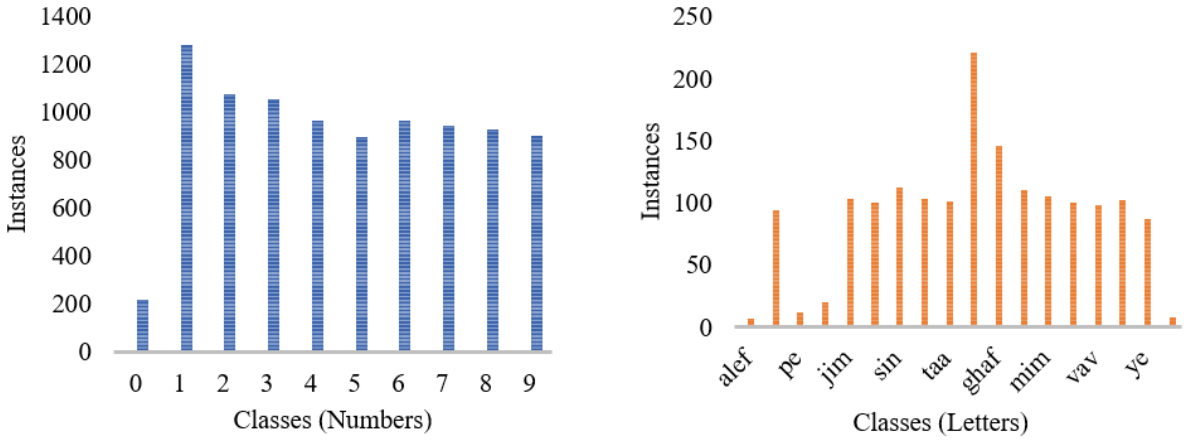

Figure 25.

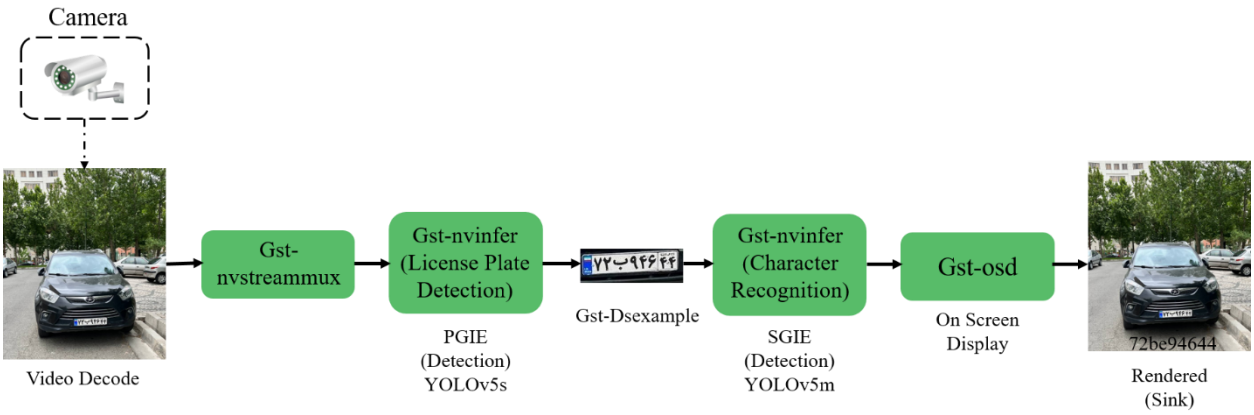

Figure 26.

Figure 27.

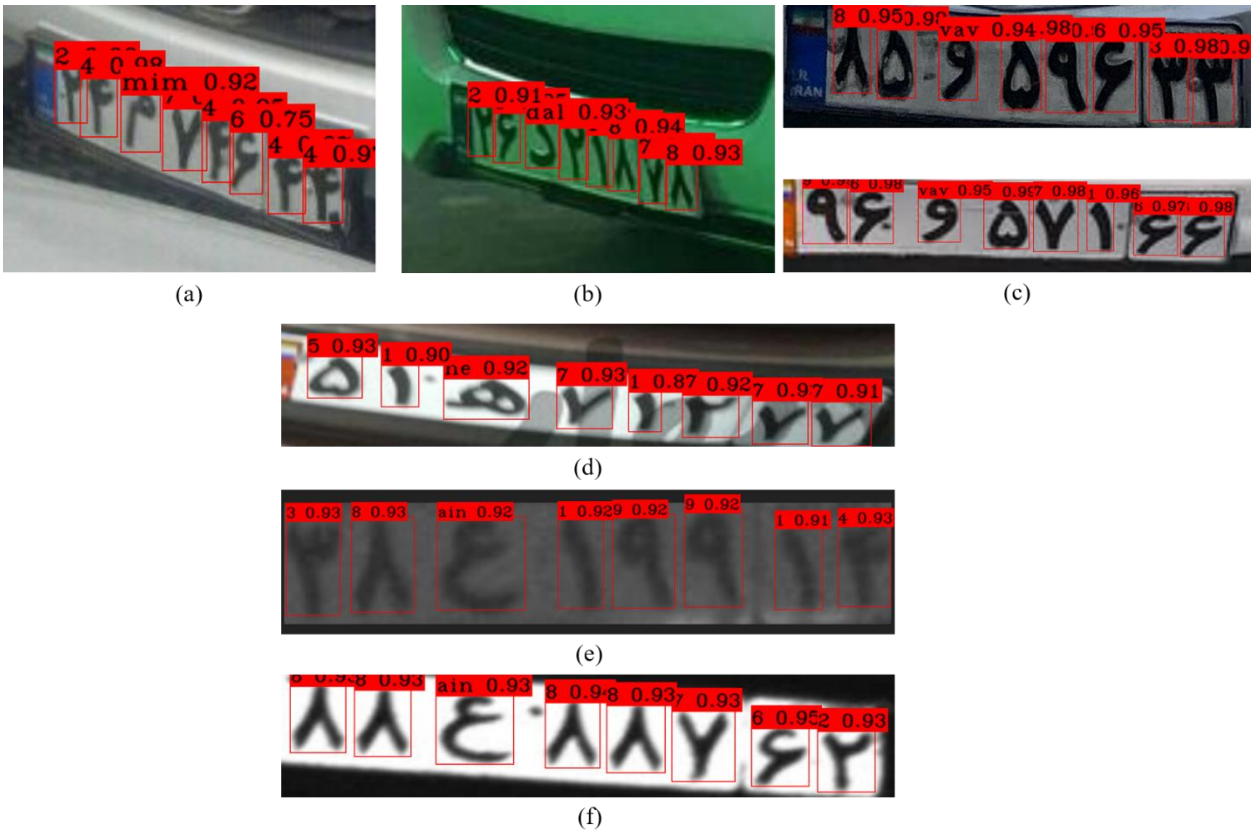

Figure 28.

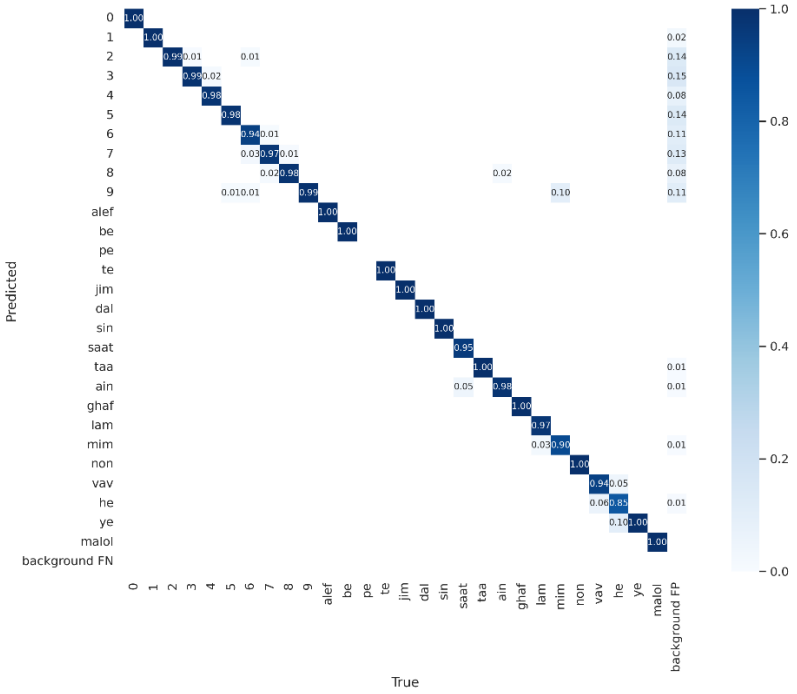

Figure 29.

Figure 30.

Figure 31.

Figure 32.

(a) (b)

Figure 33.

Table 17.

Data augmentation techniques	Optimizer	Learning rate
Scale, Mosaic, Fliplr	Adam	1×10^{-4}

Table 18.

Similar Persian Characters	
1-9	2-3
3-sin	6-ain
9-vav	pe-be
saat-sin	non-ghaf

Table 19.

Data augmentation techniques	Optimizer	Learning rate
Scale, Mosaic, Degrees (Rotation)	Adam	5×10^{-5}

Table 20.

Dataset Type	No. Samples	Participation Percentage (%)
Purchased Dataset (Private)	1466	23
Public Dataset	2840	45
Collected Dataset	2000	32
Custom Dataset (combinatorial)	6306	100

Table 21.

Model	Parameters	GFLOPs	Precision(%)	Recall (%)	mAP (%)	F1 (%)	Inference Speed (FPS)	
YOLOv5s	7022326	15.8	95.5%	94.9%	97.6%	95.2%	Video	Image
							90	66

Table 22.

Model	Parameters	GFLOPs	Precision(%)	Recall (%)	mAP (%)	F1 (%)	Inference Speed (FPS)
YOLOv5m	20962041	48.2	96.8%	98.8%	98.2%	98%	Image
							56

Table 23.

Hardware	Inference Speed (FPS)	Precision (%)
PC	23	95.5
Jetson Nano	6	93

Table 24.

Studies	Methods	Accuracy & Time		
		LPD	CR	End-to-End
Hatami et al. [40]	CNN (YOLOv4-tiny), CNN+RNN+CTC	87.81%	87.22%	75.14%, 435msec

Tourani et al. [18]	CNN (YOLOv3), CNN (YOLOv3)	97.8%	97.9%	95.05%, 119.73msec
Shahidi and Rajabi [43]	CNN (YOLOv3), Faster R-CNN	98%, 23msec	98.8%	-
Tabrizi and Cavus [41]	-, SVM-KNN	-	97.03%	-
Ashtari et al. [42]	Traditional Algorithm (Modified Template Matching + Color Features Extraction), Decision Tree + SVM	96.6%	95.85%	92.6%, , -
Our proposed method	**CNN (YOLOv5s), CNN (YOLOv5m)**	**97.6%, 15.15msec**	**98.2%, 17.86msec**	**95.5%, 43.47msec**

Biographies

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 computer vision, deep learning, image processing, and autonomous vehicles.

Majid Sadedel is currently an Assistant Professor at Tarbiat Modares University, Tehran, Iran. He received his PhD degree from University of Tehran, 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.