Electronic Letters on Computer Vision and Image Analysis 23(2):50-64, 2024

An Efficient Deep Learning based License Plate Recognition for Smart Cities

Swati, Shubh D. Kawa, Shubham Kamble, Darshit Desai, Pratik H. Karelia and Pinalkumar Engineer

Department of Electronics Engineering, Sardar Vallabhbhai National Institute of Technology, Surat, Gujarat, India
Received 16th of May, 2024; accepted 17th of September 2024

Abstract

With the high load of vehicle traffic, tracing and capturing vehicular information over traffic surveillance on roads, parking, or for safety concerns is tough. In the proposed method, a deep learning-based object detection model, EfficientDet-D0, has been trained with the custom dataset for license plate detection and used an optical character recognition model, *Tesseract*. In the proposed method, we have used an improved license plate extraction algorithm, which reduces false localization followed by character recognition in a pipeline manner. We have also explored the model quantization method to compress the model with reduced precision for efficient edge-based deployment for an end application. In the proposed work, we have dedicated our study to Indian vehicles, evaluated the performance with standard datasets like CCPD and UFPR, and have achieved 97.9% in license localization and 95.15% in end-to-end detection and recognition, respectively. We have implemented it on Raspberry Pi3 and NVIDIA Jetson Nano devices with improved performances. Compared with state-of-the-art, we have achieved 2×, 3.8×, and 2.5× in CPU, GPU, and edge platform, respectively.

Key Words: Deep learning, License plate detection (LPD), Optical character recognition (OCR), Smart surveillance, Edge intelligence, Model quantization

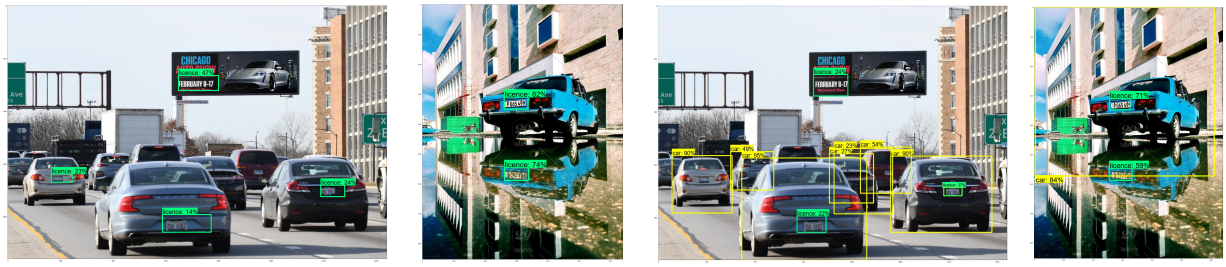
1 Introduction

Due to increasing traffic load, keeping track of all the vehicles is getting very difficult. If we have an intelligent solution to track vehicular information that is accurate and fast, then it could be a great problem solver. With the advancement in deep learning, we can extend to an implementation that can be helpful in a real-world use case and could be a potential solution. A license plate recognition (LPR) is an image processing task that first localizes the number plate in a vehicle and then extracts the characters, which assists in determining the vehicle's details. Possible use cases include traffic surveillance, automatic toll collection booths, vehicle identification, parking management systems, traffic law enforcement, criminal investigation, etc. The practical application of automatic license plate recognition (ALPR) extends notably to intelligent transportation systems (ITS), offering valuable assistance in the identification of potential security vulnerabilities and the enhancement of automation processes [1]. Despite the pervasive installation of CCTV cameras across diverse locations, the reliance on human monitoring for recording vehicle ingress and egress persists. Integrating ALPR systems is

Correspondence to: d20ec012@eced.svnit.ac.in, swati.st008@gmail.com

Recommended for acceptance by Angel D. Sappa
https://doi.org/10.5565/rev/elcvia.1917

ELCVIA ISSN:1577-5097
Published by Computer Vision Center / Universitat Autònoma de Barcelona, Barcelona, Spain



(a) Challenges

(b) Proposed solution

Figure 1: Challenges in current deep learning based license plate detection and proposed solution

a viable solution, markedly reducing the operational burden on service personnel, mitigating human-related challenges, and eradicating associated errors. By 2030, the global automatic number plate recognition (ANPR) market is expected to reach new heights, with an increased focus on data analytics and reporting for urban planning and traffic optimization. The ANPR technology will be an integral part of smart cities, contributing to a more connected and efficient urban environment [2].

Transitioning from cloud-based solutions to edge-based computation improves latency and power and will help in making real-time decisions in an efficient way [3][4]. It will be pivotal in effective traffic management with various other integrated solutions like parking facilities, vehicle access control, real-time data analytics, and many more [2]. In a real-time environment, many environmental constraints exist, such as poor lighting, bad weather, vehicles blocking the clear view of other vehicles, obstructing the license plate (LP) visibility, high-speed cars, and many more [5]. Current trends majorly deal with cloud-based solutions where the data is constantly shared to and from the mounted camera to the servers for computation [6]. The camera is stationed at a specific location and elevation that might interfere with image acquisition in certain adverse environmental conditions. It is easier to alleviate such scenarios in deep learning-based solutions as we need to enrich the dataset by incorporating diverse situations. Although there are several convolutional neural networks (CNN) based solutions, they lack a specific extraction algorithm to reduce false license plate detection (LPD) [7] [8] [9]. We have observed two existing solutions: (1) direct LP detection and (2) vehicle detection followed by LP detection being exercised, which fails to localize the LP in challenging scenarios. Our proposed implementation explores the possibility of LP and vehicle detection from a single model followed by an extraction algorithm to reduce false LP localization, and the same can be visualized in Figure 1. Here, for object detection, EfficientDet-D0 [10] was used as it is a lighter model, and our objective is to proceed to edge computation later. After LP detection, the localized LP was sent to a license plate recognition (LPR), which performs optical character recognition (OCR) to recognize the present characters. Here, we wanted to explore another candidate for LP detection. Currently, most of the implementation circles around YOLO [11], YOLO-tiny [12], and MobileNet [13] etc. The main contributions of this work are as follows:

1. Exploring the suitability of another object detection model, EfficientDet-D0 for LPD targeting edge-based implementation.

2. Proposing a novel LP extraction algorithm for adverse corner cases.

3. An end-to-end smart traffic management system solution from LP localization to recognition.

4. Model quantization technique exploration to achieve hardware-friendly models.

5. Edge-based deployment on Raspberry Pi3 with camera interface and NVIDIA Jetson Nano board.

1.1 Organizational structure of the article

Section 2 discusses the related work and background of different methods to perform LPD. Section 3 discusses the adopted methodology and how DL-based methods work. In the following sections, Section 4 and Section 5, the experimental setup with the adopted extraction algorithm and respective results are illustrated respectively. Section 6 concludes the study with future work.

2 Related Work

ALPR technique was exercised long before DL methods came into existence [6]. The earlier solution extensively used traditional image processing techniques like edge detection, contour methods, etc. Several existing solutions still practice conventional methods in smart city projects, traffic management and access control, smart parking systems, and many more applications.

2.1 Conventional Methods

It is evident that detecting a license plate and character recognition are part of image processing. The problem existed long before the CNN-based solutions came into practice. Various traditional method-based implementations have been done to perform LP detection and recognition at toll booths and other traffic surveillance systems. The conventional methods leverage the various information provided by morphological operations, contours, and various pre-processing operations to get a neat and error-free image [14].

Zied Selmi et al. [15] had shown that there is a need for certain pre-processing of images to be classified as plates/not plates and then implemented a CNN to extract the characters respectively. They performed several pre-processing steps such as morphological operations, contour detection filters, and a classifier to detect the image as plates/not plates, followed by recognition of characters. In [16], to localize a license plate in an image edge detection method has been explored. In [17], the authors have performed LP detection using morphological operations and a CNN model is used for character recognition. Another approach has been discussed, which uses blur and de-blur filters to eliminate noise in the image, which is then used for LP localization; the processed image is further fed to the character recognition, which reduces the error in recognition [18]. Feature extraction methodologies like scale-invariant feature transform (SIFT) [14], the histogram of oriented gradient (HOG) [19], and the local binary pattern (LBP) techniques are discussed in [6] for vehicle identification.

2.2 DL-based Methods

Recent developments in artificial intelligence have led to major improvements in computer vision tasks. Various deep learning models can perform object classification, detection, localization, and recognition. A deep neural network (DNN) consists of several stacked layers of convolution, max-pooling, and activation function operations. DNNs extract features layer-wise and combine low-level features to form high-level features, which can find distributed expression of data [20].

The LP detection can be achieved with a DNN model by training with a custom data set to detect a license plate region in an image. Region-based convolution neural network introduces a method to attain object detection on proposed regions to localize and detect objects [21]. Faster-RCNN implements a region proposal network that is utilized to generate detection proposals. With the introduction of single shot detectors (SSDs), there was no need for region proposal, and the object detection was done through single pass, which is adopted by various object detection models like YOLO-tiny [12], MobileNet [23] and has been utilized by multiple LP detection implementations [15] [7]. Various comparisons with object detection models have been conducted for the LPD phase, and for the LPR, a custom RPNet model was developed, which achieves an AP of 94.5% by authors in [7]. However, their dataset only consists of close-up images of cars' front or rear portions, which is difficult to localize in real-time. Here, training was done on synthetic data, then fine-tuning was performed

Table 1: Standard object detection models and their parameters

Model	Methodology	Dataset	mAP	Size	time
R-CNN [24]	two stage detector	VOC	0.54	500 MB	0.02 fps
Fast R-CNN [25]	two stage detector	VOC	0.684	145 MB	0.5 fps
Faster R-CNN [26]	two stage detector	VOC	0.704	75 MB	5 fps
Feature pyramid Network [27]	two stage detector	COCO	0.59	-	6 fps
Single Shot Detector [28]	single stage detector	VOC	0.76	-	59 fps
MobileNet v2-SSD [23]	single stage detector	VOC	0.756	32 MB	21 fps
YOLO [11]	single stage detector	COCO	0.57	140.69B	20 fps
YLOv3-tiny [12]	single stage detector	COCO	0.33	35 MB	220 fps
EfficienDet-D0 [10]	single stage detector	COCO	0.34	18 MB	83 fps

on accurate data to achieve higher accuracy. In [9], authors first performed vehicle detection, followed by LP detection, and then character recognition with 93.53% accuracy. In [22], Charan and Dubey developed a methodology for detecting two-wheeler vehicles with Yolo-v4 with Tesseract for number plate detection and recognition, respectively. In their proposed work, they have achieved 94% accuracy for number plate detection.

In a study by Wang et al., in [6], they have recommended deep neural networks (DNNs) after comparing them with traditional machine learning techniques. DNNs incorporate multiple hidden layers to learn sophisticated features, enhancing their generalization capacity. This enables impressive performance in the targeted re-identification task and extends their applicability to diverse computer vision challenges, including image classification, object detection, semantic segmentation, and video tracking. Consequently, researchers have increasingly focused on exploring deep learning approaches for vehicle re-identification in recent years.

After analyzing the existing solutions, we can understand that the conventional methods (non-DL methods) are highly efficient but constrained by a lot of pre-processing and post-processing to make them suitable for performing license plate detection and recognition. Thus, manual intervention is a requirement for performance. Image analysis needs to be done to reduce noise, blur, and contrast correction. If failed, it could lead to improper results. These methods are restricted to certain preordained environmental conditions. Here, DL-based methods have an edge over non-DL methods as the models are trained once rigorously with large, diverse datasets consisting of images in all conditions, i.e., low-light exposure, tilted, flipped, different contrast, and many more. The trained model is data-driven, so it's upon us to provide assorted data so that the trained model makes accurate decisions in any unanticipated situations. However, we cannot completely rely on the DL-based model due to some environmental constraints like rain, illumination, and several other factors. In this work, we have developed an algorithm to alleviate corner cases and avoid false detection of license plates.

We have seen models used for license plate detection as object detection models. Popular choices are RCNN [24], YoloTiny-v3 [12] used for performing license plate detections in [21] [8] models implemented after custom training on LP dataset. Here, we are exploring the possibility of a different network, EfficientDet [10], a family of object detectors recently published by **Google Brain** team. It is observed in the findings that the EfficientDet model performs efficiently in training, with reasonable accuracy and latency with smaller datasets as well [10]. The performance metrics of several other object detection algorithms are logged in Table 1 comparing with EfficientDet-D0. In this work, we explore EfficientDet-D0 as our LPD model to localize the license plates in a vehicle.

3 Proposed Methodology

From the above discussion, the dominant methodologies are DL and non-DL-based implementation, out of which we can come to an understanding that DL-based methods have the edge over the conventional for ANPR, as they are trained for all diverse cases like poor illumination, disoriented vehicles, weather conditions, etc.

3.1 Object Detection Revisited

Object detection is a technique of localization and prediction of individual objects in an image or video. Various applications can be targeted using object detection techniques such as face recognition, traffic detection, vehicle access control, emotion recognition, autonomous driving, and more [29]. The major underlying task for object detection is to localize and classify the object with existing classes. We can assume an object detection method is the next stage task of a classification algorithm. There are two main kinds of frameworks for object detection: two-stage and single-stage detectors. First, it follows a traditional method where region proposals are generated, followed by classifying the generated region proposals into different existing categories. Second, it associates detection with a regression problem where a backbone CNN architecture is used for feature extraction and later passed through a segment to provide bounding boxes and categories. The main difference between the two methods is that the latter skips the region of interest (ROI) proposal and performs prediction in a single pass. Two-stage detectors are more accurate but get very bulky as two separate networks are connected back to back, hence the higher turnaround time. Table 1 describes different types of object detection models based on methodologies and their performance metrics. We can observe that two-stage detectors achieve better performance but at the cost of model size and speed. However, single-stage detectors offer faster response time with reduced model size and slightly compromised performance. One can decide what fits their criteria as per their requirement.

3.2 EfficientDet

EfficientDet is a comparatively newer network in object detection which proposes several key optimizations to improve efficiency. A bi-directional feature pyramid network (BiFPN), which allows easy and fast multi-scale feature fusion along with a compound scaling method that uniformly scales the resolution, depth, and width for all backbone, feature network, and box/class prediction networks simultaneously, is proposed [10]. The authors have discussed the implication of implementing bulkier models with larger parameters in real-world applications like self-driving cars, robotics, and other resource-constrained scenarios where it gets very tedious to infer with such a high degree with computations and parameters. This architecture achieves comparable accuracy with reduced parameters and latency. There are several models available in EfficientDet family D0-D7 varying in degree of computation and memory size. We have opted for EfficientDet-D0 due to its smaller memory size and floating point operations (FLOPS) [10].

Also, when targeting a real-world application, gathering a larger dataset to train the network with better accuracy and efficiency is a challenge. Since this model is trained with 300 epochs, it has efficient learned parameters, which are also utilized in transfer learning with smaller custom datasets. It is suggested that ImageNet [31] checkpoints give better training for object detection even with smaller datasets incorporating several augmentation techniques, which improve accuracy and efficiency. We can refer to Table 1, illustrating standard object detection architectures and their model parameters. Based on this, we can make an informed decision on selecting the best model. It is evident that there are several other networks with better parameters. However, we had isolated this specific model due to its smaller computation and model size with good speed. Our main intention is to have an edge implementation for real-time applications. So, our solution is for a resource-constrained system.

3.3 Proposed Implementation

Usually, in any classic ANPR solution, the standard steps are image capture, number plate localization, and preprocessing of the extracted plate, followed by character recognition. Our proposed block diagram is presented in Figure 2, which shows an end-to-end implementation of ALPR where first, the image acquisition is performed with the mounted camera hardware, after which the detection model performs localization of vehicles and license plates. We have explored and trained a combination of detection models where only single-class

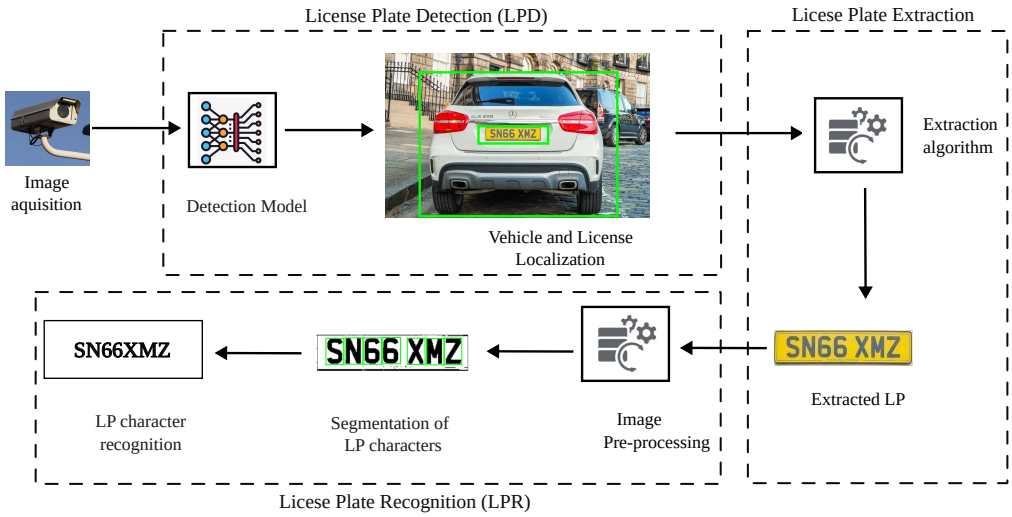


Figure 2: Proposed block diagram of our automatic number plate recognition

and two-class (vehicle and license plates) detection techniques are followed by an algorithm to verify the ascertainment of the final prediction. Next, the localized LP is extracted and followed by a pre-processing algorithm consisting of traditional image processing techniques to remove noise, channel conversion, and binarization to help in a better perception of characters to perform OCR. The pre-processed image is passed through the OCR network, which extracts characters present in LP and stores them for further tasks.

3.3.1 License Plate Detection and Extraction

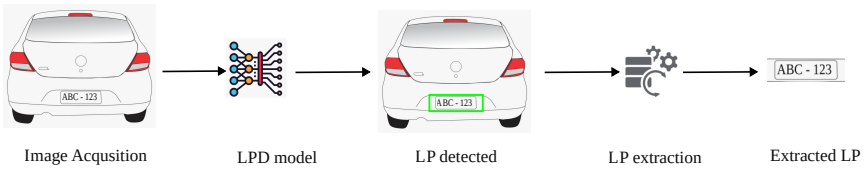
In Figure 2, the proposed methodology is illustrated. After image acquisition, the image is fed to an object detection model, which performs localization of the vehicle and license plate, respectively. Based on the above discussion, the detection model finalized is EfficientDet-D0 due to its smaller size, faster response time, and efficient training with a smaller dataset. We have proposed a novel LP extraction algorithm to ensure accurate LP localization.

As discussed in previous sections, we have performed an analysis with training models in three scenarios. Our main goal is to avoid any false detection of LP, which is used for further processing. The standard output of an object detection model is the coordinates of the detected object as a bounding box and a label with an accuracy score. We have trained and evaluated the solution in three different ways, and the same is illustrated in Figure 3 and described below:

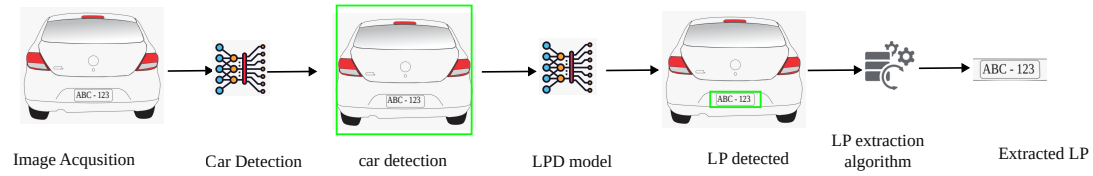
1. ***Single model performing only LP detection***
   The model here is trained to detect license plates only. We have witnessed a couple of corner cases and false predictions with license plates in constraint scenarios like reflection, blurry images, and other cases. So, we selected combining vehicle/car detection with LP detection to see if this alleviates the problem.

2. ***Two models performing only vehicle and LP detection respectively***
   Here, two models are trained separately with single prediction labels, i.e., license plates and vehicle/car, respectively. Again, the results are passed through the extraction algorithm to get the final LP detection.

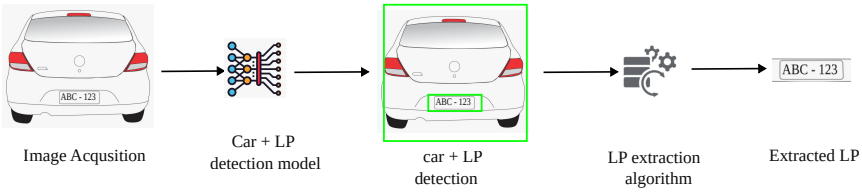
3. ***Single model performing vehicle and LP detection (2 classes)***
   Here, a single model is trained with two classes to predict, i.e., license plates and car/vehicle, respectively. After performing the object detection algorithm, the results are passed through an extraction algorithm to suppress false detection.



(a) Single detection model performing LP detection only



(b) Two detection models performing vehicle and LP detection in a sequential manner



(c) Proposed single detection model performing vehicle and LP detection in a single pass

Figure 3: Different scenarios of license plate detection and extraction

After the Car + LP detection model, we receive multiple bounding boxes (BBs) for cars and LP in a single pass. We need some mechanism for efficient localization of both vehicle and LP. The novel aspects of our proposed extraction algorithm:

- Categorizes distinct bounding boxes associated with vehicle and LP on the basis of score
- Establishes a link between bounding boxes of LP and vehicle for effective localization
- Generates plausible bounding box pairs for (vehicle, LP) followed by LP extraction algorithm
- Algorithm 1 highlights the process of extraction where inputs are bounding box pairs for (vehicle, LP) and the input image
- Post LP extraction, further processing for LPR is initiated

Algorithm 1: Proposed extraction algorithm

```
Data: Input image
Result: Extracted LP
while getting detections do
    if detected LP coordinates ⊂ detected vehicle coordinates then
        extract LP from image for further processing
    end
end
```

Figure 4: Different pre-processing techniques for OCR and respective extracted characters from license plates

3.3.2 License Plate Recognition (LPR)

License plate recognition (LPR) comes under the image processing technique commonly known as optical character recognition (OCR). It is a method where the text is recognized in a digital image. This technique is vastly used in document analysis, where texts are recognized from scanned copies of hard documents and processed for further analysis. We use an open-source OCR engine here, Tesseract [34]. It is a C-based package invoked using the PyTesseract command in the Python environment. PyTesseract or Python-Tesseract is a python-based wrapper for the C-based Tesseract OCR engine, which makes it easier to work within the same environment. We have performed a combination of pre-processing steps for efficient OCR. With Tesseract, some several other conditions and parameters enable the tool to read the image and convert the recognized characters into text, string, etc.

Pre-processing techniques in OCR

Pre-processing techniques are necessary for OCR, as, in general, the image is binarized before performing OCR; minor deviations in an image could lead to erroneous recognition. Several techniques and other techniques are available that assist in better OCR; some of them are described here and illustrated in Figure 4. RGB scale to grayscale conversion where 3, 8-bit channels input are converted to the single 8-bit channel for better readability of the LP. Binarization is where the 8-bit channel is converted to 1-bit containing only 0's and 1's in the image with some thresholding operation to eliminate extra noise and discrepancies. There are morphological operations that add or remove pixels from boundaries called dilation or erosion, respectively. These operations are implemented to modify the geometric structure in the image and aid the further decision process. Several other pre-processing techniques like gaussian blur, edge detection, gamma correction etc perform pixel-level manipulation to ensure correct recognition.

4 Experimental Setup

Both LPD and LPR models have been trained on CPU with processor AMD Ryzen 7 6800H with GPU 4 GB GeForce RTX 3050 graphics and 16 GB RAM, respectively. TensorFlow deep learning framework version 2.10 was used in the back end, and the entire scripting was done with Python 3.9. Later, the models were converted and quantized with TensorFlow-Lite, and inference was evaluated on various platforms. For edge-based deployment, Raspberry Pi3 and NVIDIA Jetson Nano were used, and the results and observation of all the above experiments are logged in Section 5.

4.1 Dataset Preparation

The entire ANPR solution combines two tasks: detection and recognition. The dataset was gathered for both learning tasks. For LPD, the gathered dataset for training was a combination of the Indian vehicle dataset [35] and a combination of UFPR [9] and the CCPD dataset [7], which were around 2000 in total. The EfficientDet-D0 training algorithm automatically applies several augmentation methods like flipping, rotation, and transformation techniques like translation, scaling, brightness adjustments, noise addition, etc. The discussed datasets are standard and open source, with some being licensed and provided for research. We used 180 images from the same dataset combination for testing and evaluated the performance measures/metrics. Further, data is annotated with **labelImg** [30] software to produce an *.xml* file with the label and bounding box coordinates related to individual images, respectively. The entire dataset was split into a 75:25 ratio for the training and

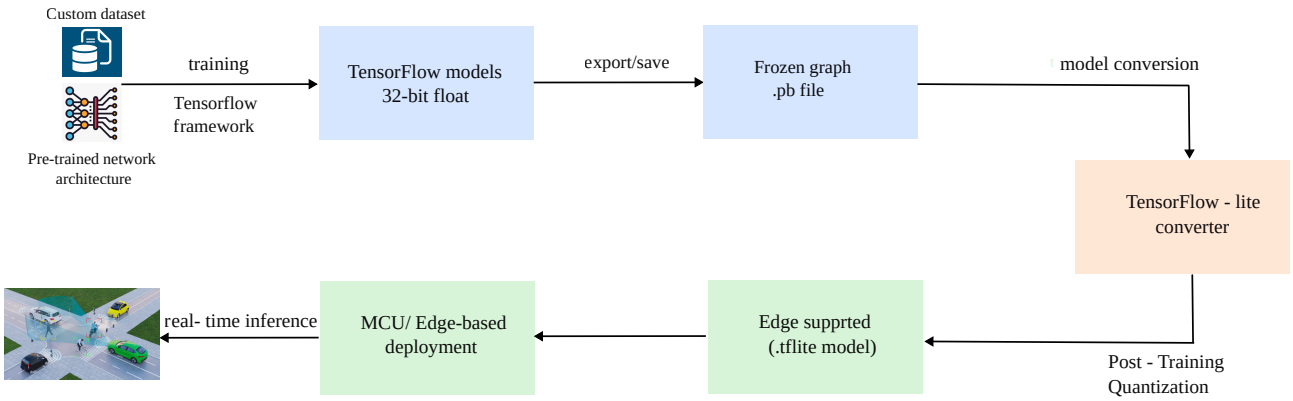


Figure 5: Model Conversion steps for Edge-based deployment

testing phase while training the LPD model. Later, the same trained models were evaluated with entirely different annotated datasets to compute and compare the evaluation performance metrics. Here, for LPD, we have developed an enhanced extraction algorithm to double-check the presence of the license plate in a vehicle to reduce false localization.

4.2 Edge-based Deployment with TensorFlow-Lite

TensorFlow-Lite or TF-lite is an open-source deep learning framework for edge-based inference. It provides tools to run trained models on embedded platforms like mobile, IoT devices, embedded Linux, androids, iOS, and microcontroller units (MCUs) [36]. Figure 5 describes the steps in performing model conversion steps with TensorFlow-Lite for edge-based deployment.

In general, models are built and trained with TensorFlow. Later, the model is converted to TF-lite models, reducing the model size, parameters, memory footprint, and computational costs. TF-lite models are compressed FlatBuffer files, an efficient cross-platform data serialization library supported by various platforms and languages. It has a lower code footprint and better efficiency and speed, which is extended here for edge deployment. There are further optimization methods supported by TF-lite called quantization. Model quantization is a model compression technique where model parameters (weights and activations) are converted from high-precision floating-point representation (64-bit or 32-bit) to fixed-point representation (16-bit or 8-bit). Generally, post-training quantization (PTQ) is used to run inference. The advantage of edge-based deployment over the cloud is that it severs the link between the internet and the host machine, thus avoiding network latency and creating a faster turnaround time. We have leveraged the model quantization method to implement it on edge devices such as Raspberry Pi3 and NVIDIA Jetson Nano boards.

4.3 Performance Metrics

1. ***Intersection over Union (IoU)***
   This is a metric based on the similarity between two data, where the area overlapping between the detected/ predicted bounding box and the ground truth bounding box is divided by the total area of the union of them. Figure 6 a describes the IoU effectively. The two different color bounding boxes are of predicted and ground truth objects, respectively. After obtaining the IoU measure, a comparison is performed with an agreed threshold to decide the outcome of the prediction. If $IoU \geq threshold$, the prediction is correct. If $IoU < threshold$, the prediction is incorrect.

2. ***Fundamental performance metrics of an object detection model***
   The fundamental metrics to evaluate the performance of many object detection models. In any computer vision task, the key features for measuring proficiency are precision, recall, and F1-score. Figure 6 b

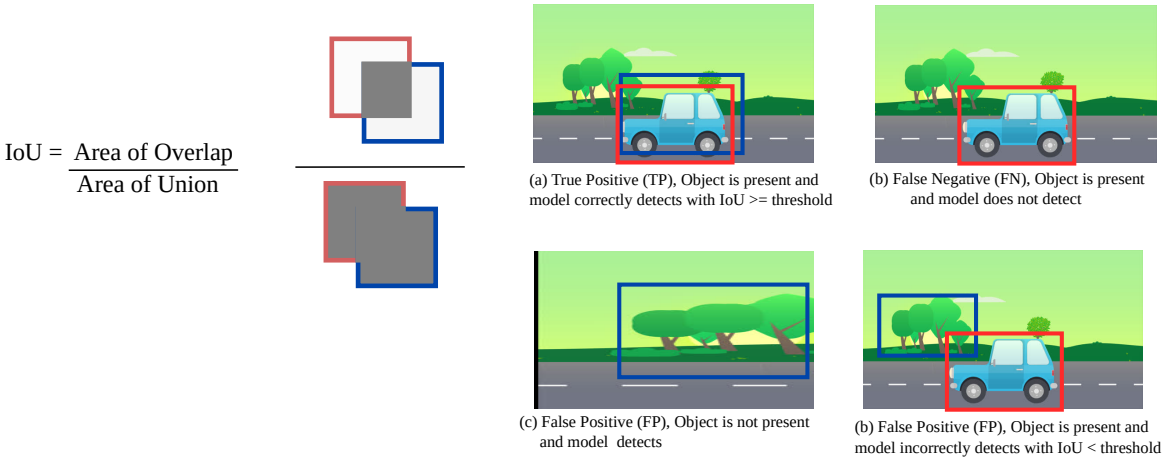


Figure 6: (a) Intersection over Union (b) Prediction outcomes of a CNN-based object detection model

describes the basics of various concepts dealing with object detection in computer vision, bounding boxes with red and blue colors representing ground truth and predicted area, respectively.

- Precision is a performance metric that assesses the ability to correct the prediction of the model.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{All\ detections} \tag{1}$$

- Recall is mainly termed as the sensitivity of the object detection algorithm, which measures the ability to find all positive predictions. From an object detection perspective, true negative (TN) cases are not considered.

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{All\ ground\ truths} \tag{2}$$

- The precision vs. recall curve can be seen as a trade-off between precision and recall for different confidence values associated with the bounding boxes generated by a detector [29]. The average precision (AP) is the area under the precision-recall curve. Mean average precision (mAP) is simply an average of AP over a number of classes, N. It is a metric used to measure the accuracy and correctives of an object detection model.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{3}$$

Interpolated AP - For object detection challenge, popular dataset like ImageNet [31], COCO[32], PASCAL VOC [33] and many more. For the PASCAL VOC dataset, positive prediction is for IoU ≥ 0.5. For COCO, a positive prediction is considered for AP@[0.5:0.95] for average AP or mAP.

- F1-score is computed by combining the precision and recall metrics of the model and is a parameter for determining how many times the model has made a positive prediction,

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{4}$$

5 Results and Discussion

As discussed earlier, training has been carried out in 3 different ways, i.e., LPD only, vehicle detection, single model performing LPD, and vehicle detection, and their training accuracy and other metrics are shown in

Table 2: Performance metrics of models in training and testing environment respectively

(a) Training performance metrics

Model Specification		Accuracy	mAP	mAR	F1-score
Model	Class				
LP detection	1	95.7 %	0.618	0.656	0.64
vehicle	1	93.5 %	0.611	0.625	0.62
LP + vehicle	**2**	**97.8** %	**0.662**	**0.689**	**0.68**

(b) Testing performance metrics

Model Specification		Accuracy	mAP	mAR	F1-score
Model	Class				
LP detection	1	95.8 %	0.406	0.645	0.5
vehicle	1	92.6 %	0.651	0.554	0.6
LP + vehicle	**2**	**97.9** %	**0.576**	**0.713**	**0.64**

Table 3: Evaluation results of models with same testing dataset in different model quantization format

Specification	TensorFlow (float)				TensorFlow-Lite (float)				TensorFlow-Lite (quantized 16-bit)			
	Accuracy	mAP	time (s)		Accuracy	mAP	time (s)		Accuracy	mAP	time (s)	
			CPU	GPU			CPU	GPU			CPU	GPU
LPD only	95.8%	0.406	0.436	0.3	73.4%	0.396	0.161	0.155	64.5%	0.315	0.158	0.153
vehicle followed by LPD	93.2%	0.52	1.27 s	0.613	79.4%	0.53	0.321	0.314	72.5%	0.325	0.316	0.306
vehicle + LP	**97.9%**	**0.576**	**0.832 s**	**0.315**	**85.3%**	**0.446**	**0.158**	**0.158**	**77.6%**	**0.348**	**0.16**	**0.154**

Table 2. This table reports training and evaluation performance metrics for all three scenarios where models are trained and evaluated on the same training and testing dataset, respectively. The final reported accuracy is for accurate LP localization in every scenario. We have categorized positive prediction if and only if IoU $\geq$ 0.5 and have marked such cases as accurate. Our proposed method, where a single model performs LPD and vehicle detection, surpasses accuracy, mAP, mAR, and F1-score due to double thresholding, ensuring positive localization most of the time.

Illustrated in Figure 3, there are three different scenarios to achieve LPD. We implemented all of them and logged the results. We can observe in Table 3 that all the discussed methods of achieving LPD are mentioned. They are LPD only, vehicle detection sequentially followed by LPD, and our proposed method performs vehicle and LP detection in a single pass. It can be inferred from the results that our proposed method performs with the least latency and good precision throughout, as both detections (vehicle and LP) are achieved in a single pass compared to sequentially achieving vehicle detection and LPD. When comparing the discussed methods, our proposed method achieves improved performance with reduced processing time across the different quantization formats in software-based implementation on CPU with processor AMD Ryzen 7 6800H with GPU 4 GB GeForce RTX 3050 graphics, respectively. The time logged here is the mean/ average time taken when performing the evaluation on the entire testing dataset across all the quantization formats.

TensorFlow models were in floating precision around **37 MB** (frozen graph(.pb) 18 MB and variable 19 MB). The main intent was to have an end-user application on edge devices, so we leveraged the model conversion technique supported by TF-Lite with no quantization and 16-bit fixed-point quantization, respectively. The model size was reduced to **22MB** and **11 MB**, resulting in model size reduction by 1.68$\times$ and 3.36$\times$, respectively, compared to the original model. Since the model quantization converts the model parameters from high-precision to low-precision, the reduction in accuracy score is inevitable. There is no API similar to TensorFlow in TF-Lite to achieve performance metrics, so we need to create an interpreter instance of the TF-Lite flat-buffer model. The TF-Lite interpreter is run on the same testing dataset of 180 images, and for positive prediction, we have restricted IoU $\geq$ 0.5 to calculate all the metrics reported in the Table 3. We have witnessed performance loss across the different quantization methods. However, due to our proposed method of double-checking the LP with our enhanced extraction algorithm, we have maintained that our proposed implementation performs better among all the above-discussed methods. Also, our proposed method ensures that even if we further quantize our model to 8-bit or lower due to double-checking, the false detection could be prevented with a single pass. Here, when comparing with the state-of-the-art implementation, we have compared the quantized 16-bit TensorFlow-Lite models on edge devices, and they performed the best when compared with the other two models in terms of latency and memory.

Table 4: Comparison with existing work across different platform and performance

Work	Implementation	Accuracy	Platform	Time (s)
Khan et.al. [37]	LPD	90.94 %	CPU	0.99
			GPU	0.42
Proposed	LPD	**97.9 %**	CPU	**0.26**
			GPU	**0.21**
Li et.al [38]	LPD	97.3 %	NVIDIA Tesla K40c	3
Proposed	LPD	**97.9 %**	NVIDIA Jetson Nano	**1.2**
Izidio et.al [8]	LPD + LPR	96.38 %	Raspberry Pi3	2.70
Proposed	LPD + LPR	**95.15 %**	Raspberry Pi3	**2.596**

5.1 Comparison with state-of-the-art

We have made a comparison with the existing implementation and have reported our findings. Table 4 has the comparison across various platforms like CPU, GPU, and edge devices, respectively. The reported time is the least time taken to achieve a single prediction. Upon comparing with similar software implementation with [37], our proposed method has **1.07**× improved accuracy with **3.8**× and **2**× speed ups in CPU and GPU based execution respectively. When discussing edge-based implementation, we have implemented it on Raspberry Pi3 and NVIDIA boards [38], and their findings are logged. Comparing with the NVIDIA development board, we have compared the LPD task, and we have achieved improved accuracy marginally but a massive speedup of about **2.5**× in terms of latency by being able to run the LPD task on Quad-core ARM Cortex-A57 MPCore processor on NVIDIA Jetson Nano 2GB. We have also successfully implemented our end-to-end solution on Raspberry Pi3 from camera acquisition to license plate detection and recognition in 1.736 s and 0.86 s, respectively. We have used a Pi NOIR2 camera with Raspberry Pi3 for real-time image acquisition for size (1024,780) and performed real-time prediction with our proposed design in a pipelined fashion. Compared with Raspberry Pi3 [8], we have improved throughput by **3.85%** in an end-to-end realization for a real-time solution. Our method has implemented two class detection for LPD (vehicle + car) along with out-of-the-box Tesseract for LPR. In existing works, the images in the dataset are either close-up images of vehicles with clear LP visibility or single vehicle presence in an image, which simplifies the detection process. However, in our proposed method, we have designed a dataset consisting of multiple vehicles, and performing LPD and LPR in that sequence is more complex, which justifies the slight performance loss.

6 Conclusion

This study presents a solution for automatic license plate detection and recognition using deep learning methodology with our proposed novel extraction algorithm. We have combined EfficientDet-D0 for performing license plate detection (LPD) and Tesseract for license plate recognition (LPR) in a pipeline fashion. As per our findings, this is the first implementation with EfficientDet-D0 as an LPD application, and we have witnessed that with a smaller dataset and training, the model competes with other existing object detection algorithms. Our proposed solution reduces false detection by getting the vehicle and LP detection in a single pass, followed by our proposed extraction algorithm incorporating double-checking LPs in various backgrounds. We have also explored the suitability of edge implementation by performing model optimization and quantization with TensorFlow-Lite, and we have reduced model size with comparable accuracy loss. Our proposed method has improved performance and reduced latency with similar benchmarks. We have also executed end-to-end implementation, from image acquisition to LPD and LPR, extracting characters present in a license plate.

In future work, we will target other embedded platforms with model optimization methods to improve performance with further model compression, like quantization-aware training, pruning, etc. In the future scope,

the acceleration of these solutions on FPGA-based devices looks promising, with reduced power consumption and faster results that could be a breakthrough in real-time solutions for end applications.

References

[1] Zi Yang, Lilian S.C. Pun-Cheng,"Vehicle Detection in Intelligent Transportation Systems and Its Applications Under Varying Environments: A Review", *Image and Vision Computing*, Volume 69, 2018, doi: *https://doi.org/10.1016/j.imavis.2017.09.008*.

[2] Kumar P., "ANPR Camera: Latest Trends and Forecast 20242030 in the global market". [Online] Available: *https://kotaielectronics.com/anpr-camera-latest-trends-in-2024-2030/*.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges", in IEEE Internet of Things Journal, vol. 3, Oct. 2016, doi: *https://doi.org/10.1109/JIOT.2016.2579198*.

[4] G. Yan and Q. Qin, "The Application of Edge Computing Technology in the Collaborative Optimization of Intelligent Transportation System Based on Information Physical Fusion", IEEE Access, 2020, doi: *https://doi.org/10.1109/ACCESS.2020.3008780*.

[5] P Mukhija, P Dahiya, (2021). "Challenges in Automatic License Plate Recognition System: An Indian Scenario", doi: *http://dx.doi.org/10.1109/CICT53244.2021.00055*

[6] H. Wang, J. Hou and N. Chen, "A Survey of Vehicle Re-Identification Based on Deep Learning," in IEEE Access, vol. 7, pp. 172443-172469, 2019, doi: *https://doi.org/10.1109/ACCESS.2019.2956172*.

[7] Z. Xu, W. Yang, A. Meng, N. Lu, H. Huang, C. Ying, e. V. Huang, Liusheng, M. Hebert, C. Sminchisescu, and Y. Weiss, "Towards end-to-end License Plate Detection and Recognition: A Large Dataset and Baseline", in *Computer Vision – ECCV 2018*, Springer Intern. Publishing, 2018, pp.261–277, doi: *https://doi.org/10.1007/978-3-030-01261-8_16*.

[8] D. M. F. Izidio, A. P. A. Ferreira, and E. N. S. Barros, "An Embedded Automatic License Plate Recognition System Using Deep Learning" in *2018 VIII (SBESC)*, 2018, pp. 38–45, doi: *https://doi.org/10.1109/SBESC.2018.00015*.

[9] R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Gonçalves, W. R. Schwartz, and D. Menotti, "A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector," in *International Joint Conference on Neural Networks (IJCNN)*, July 2018, pp. 1–10, doi: *https://doi.org/10.1109/IJCNN.2018.8489629*.

[10] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection", *CoRR*, vol. abs/1911.09070, 2019. [Online]. doi: *https://doi.org/10.1109/CVPR42600.2020.01079*.

[11] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *CoRR*, vol. abs/1506.02640, 2015. [Online], doi: *https://doi.org/10.1109/CVPR.2016.91*.

[12] J. Redmon and A. Farhadi, "Yolov3: An Incremental Improvement," 2018. [Online]. doi: *https://doi.org/10.48550/arXiv.1804.02767*.

[13] A. G. Howard, M. Zhu, et.al., "Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *CoRR*, vol. abs/1704.04861, 2017. [Online]. doi: *https://doi.org/10.48550/arXiv.1704.04861*.

[14] D. G. Lowe, "Object Recognition from Local Scale-Invariant Features", in Proc. ICCV, Kerkyra, Greece, 1999, pp. 1150-1157, doi: *https://doi.org/10.1109/ICCV.1999.790410*.

[15] Z. Selmi, M. Ben Halima, and A. M. Alimi, "Deep Learning System for Automatic License Plate Detection and Recognition", in *2017 14th IAPR (ICDAR)*, vol. 01, 2017, pp. 1132–1138 , doi: *https://doi.org/10.1109/ICDAR.2017.187*.

[16] B. Hongliang and L. Changping, "A Hybrid License Plate Extraction Method based on Edge Etatistics and Morphology" in *Proceedings of the 17th ICPR 2004.*, vol. 2, 2004, pp. 831–834 Vol.2, doi: *https://doi.org/10.1109/ICPR.2004.1334387*.

[17] S. Paneerselvam, P. Gurudath, R. Prithvi, and V. Ananth, "Automatic License Plate Recognition using Image Processing and Neural Network", *ICTACT J. on Image and Video Proc.*, vol. 8, 05 2018, doi: *http://dx.doi.org/10.21917/ijivp.2018.0251*

[18] V. Koval, V. Turchenko, V. Kochan, A. Sachenko, and G. Markowsky, "Smart License Plate Recognition System Based on Image Processing Using Neural Network", 10 2003, pp. 123 – 127, doi: *https://doi.org/10.1109/IDAACS.2003.1249531*

[19] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection", in Proc. CVPR, San Diego, CA, USA, 2005, pp. 886-893, doi: *https://doi.org/10.1109/CVPR.2005.177*.

[20] H. Yi, S. Shiyu, D. Xiusheng, and C. Zhigang, "A Study on Deep Neural Networks Framework", in *2016 IEEE Adv. Inf. Management, Communicates, Elect. and Aut. Control Conf. (IMCEC)*, 2016, pp. 1519–1522, doi: *https://doi.org/10.1109/IMCEC.2016.7867471*.

[21] C.-H. Lin, Y.-S. Lin, and W.-C. Liu, "An Efficient License Plate Recognition System Using Convolution Neural Networks," in *2018 IEEE Intern. Conf. on Applied System Invention (ICASI)*, 2018, pp. 224–227, doi: *https://doi.org/10.1109/ICASI.2018.8394573*.

[22] R. S. Charran and R. K. Dubey, "Two-Wheeler Vehicle Traffic Violations Detection and Automated Ticketing for Indian Road Scenario", in IEEE Transactions on Intelligent Transportation Systems, Nov. 2022, doi: *https://doi.org/10.1109/TITS.2022.3186679*.

[23] Y. C. Chiu, C. Y. Tsai, M. D. Ruan, G. Y. Shen and T. T. Lee, "Mobilenet-SSDv2: An Improved Object Detection Model for Embedded Systems," ICSSE, 2020, doi: *https://doi.org/10.1109/ICSSE50014.2020.9219319*.

[24] Girshick, Ross, et al. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. 2014 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2014, pp. 58087, doi: *https://doi.org/10.1109/CVPR.2014.81*.

[25] K. Lenc and A. Vedaldi, "R-CNN minus R", CVPR 2017, doi: *https://doi.org/10.48550/arXiv.1506.06981*.

[26] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *CoRR*, 2015. doi: *https://doi.org/10.48550/arXiv.1506.01497*.

[27] Tsung-Yi Lin, Piotr Doll ar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. "Feature Pyramid Networks for Object Detection", CVPR, 2017, doi: *https://doi.org/10.48550/arXiv.1612.03144*

[28] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot multibox detector", in ECCV, 2016, doi: *https://doi.org/10.1007/978-3-319-46448-0_2*.

[29] Z. -Q. Zhao, P. Zheng, S. -T. Xu and X. Wu, "Object Detection With Deep Learning: A Review," in IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 11, pp. 3212-3232, Nov. 2019, doi: *https://doi.org/10.1109/TNNLS.2018.2876865*.

[30] Tzutalin, "LabelImg", Git code (2015). Available: *https://github.com/tzutalin/labelImg*, Accessed: September 2023

[31] J. Deng, W. Dong, R. Socher, L. J. Li, Kai Li and Li Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," 2009 IEEE CVPR, Miami, FL, USA, 2009, pp. 248-255, doi: *https://doi.org/10.1109/CVPR.2009.5206848*

[32] Lin, TY. et al., "Microsoft COCO: Common Objects in Context", Computer Vision ECCV 2014. Lecture Notes in Computer Science, vol 8693. Springer, Cham. doi: *https://doi.org/10.1007/978-3-319-10602-1_48*

[33] Everingham M., Van Gool L., Williams C. K. I., Winn, J. and Zisserman, A., "PASCAL Visual Object Classes (VOC) Challenge", International Journal of Computer Vision, 88(2), 303-338, 2010, doi: *https://doi.org/10.1007/s11263-009-0275-4*

[34] Python-Tesseract OCR Engine, Available: *https://pypi.org/project/pytesseract/*, Accessed:Feb 2024.

[35] Car and License Plate Detection, Available: *https://www.kaggle.com/datasets/riotulab/car-and-license-plate-detection*, June 2023.

[36] TensorFlow-Lite ML for Mobile and Edge, Available: *https://www.tensorflow.org/lite* , Accessed: Dec 2023.

[37] H. Wang, J. Hou and N. Chen, "A Novel Deep Learning Based ANPR Pipeline for Vehicle Access Control", in IEEE Access, vol. 7, pp. 64172-64184, 2022, doi: *https://doi.org/10.1109/ACCESS.2022.3183101.*

[38] H. Li and C. Shen, "Reading Car License Plates Using Deep Convolutional Neural Networks and LSTMs" in CoRR,1601.05610 , 2016, doi: *https://doi.org/10.1016/j.imavis.2018.02.002.*