



*sensors*



Article

---

# A Novel Memory and Time-Efficient ALPR System Based on YOLOv5

---

Piyush Batra, Imran Hussain, Mohd Abdul Ahad, Gabriella Casalino, Mohammad Afshar Alam, Aqeel Khaliq and Syed Imtiyaz Hassan

Special Issue

Machine Learning in Robust Object Detection and Tracking




Edited by  
Dr. Qing Guo



<https://doi.org/10.3390/s22145283>

Article

# A Novel Memory and Time-Efficient ALPR System Based on YOLOv5

Piyush Batra <sup>1</sup>, Imran Hussain <sup>1,\*</sup> , Mohd Abdul Ahad <sup>1</sup>, Gabriella Casalino <sup>2</sup> , Mohammad Afshar Alam <sup>1</sup>, Aqeel Khalique <sup>1</sup> and Syed Imtiyaz Hassan <sup>3</sup> 

<sup>1</sup> Department of Computer Science and Engineering, Jamia Hamdard, New Delhi 110062, India; piyushbatra1999@gmail.com (P.B.); aahad@jamiahamdard.ac.in (M.A.A.); aalam@jamiahamdard.ac.in (M.A.A.); aqeelkhalique@jamiahamdard.ac.in (A.K.)

<sup>2</sup> Department of Computer Science, University of Bari, 70125 Bari, Italy; gabriella.casalino@uniba.it

<sup>3</sup> Department of CS and IT, Maulana Azad National Urdu University, Hyderabad 500032, India; s.imtiyaz@gmail.com

\* Correspondence: ihussain@jamiahamdard.ac.in

**Abstract:** With the rapid development of deep learning techniques, new innovative license plate recognition systems have gained considerable attention from researchers all over the world. These systems have numerous applications, such as law enforcement, parking lot management, toll terminals, traffic regulation, etc. At present, most of these systems rely heavily on high-end computing resources. This paper proposes a novel memory and time-efficient automatic license plate recognition (ALPR) system developed using YOLOv5. This approach is ideal for IoT devices that usually have less memory and processing power. Our approach incorporates two stages, i.e., using a custom transfer learned model for license plate detection and an LSTM-based OCR engine for recognition. The dataset that we used for this research was our dataset consisting of images from the Google open images dataset and the Indian License plate dataset. Along with training YOLOv5 models, we also trained YOLOv4 models on the same dataset to illustrate the size and performance-wise comparison. Our proposed ALPR system results in a 14 megabytes model with a mean average precision of 87.2% and 4.8 ms testing time on still images using Nvidia T4 GPU. The complete system with detection and recognition on the other hand takes about 85 milliseconds.

**Keywords:** ALPR; YOLOv5; IoT; OCR; vehicle license plate detection and recognition; urban mobility



**Citation:** Batra, P.; Hussain, I.; Ahad, M.A.; Casalino, G.; Alam, M.A.; Khalique, A.; Hassan, S.I. A Novel Memory and Time-Efficient ALPR System Based on YOLOv5. *Sensors* **2022**, *22*, 5283. <https://doi.org/10.3390/s22145283>

Academic Editor: Marco Diani

Received: 6 June 2022

Accepted: 11 July 2022

Published: 14 July 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Traffic security and congestion represent some of the major problems in upcoming smart cities. Any contribution to help manage these cities will be beneficial for everyone. With the further development of smart cities and consequently intelligent transportation systems, efficient automatic license plate recognition systems are now required more than ever. These systems have gained much attraction due to their application in intelligent surveillance systems which have various use cases such as automated parking lot management, traffic surveillance, vehicular access control, etc., which represent emerging research areas under the scope of urban mobility.

Internet of Things (IoT) devices are building blocks for smart cities. These are minute digital devices used in large quantities to complete data processing tasks. Although IoT processing devices such as Google Coral, Raspberry Pi, NVIDIA Jetson Nano, etc., are more powerful than typical IoT devices, they still lack continuous processing and cooling capabilities unlike traditional machine learning workstations. This paper aims to present the developed size and time-efficient system which can be deployed on small IoT processing devices, which can contribute to the development of smart cities.

Automated License Plate Recognition (ALPR) systems usually have two parts: license plate localization and character recognition. In the localization stage, the number plate

is detected and cropped from the frame for character recognition. In the next stage, the cropped license plates are processed and given as input to a character recognition engine. Although there are plenty of ALPR systems in development, most of them are heavily dependent on high compute resources. In this paper, we present a system that is significantly smaller in size and can work on devices with low computing power.

Deep neural nets have proven to be the best choice for object detection tasks. To build a robust deep learning model, a huge amount of labelled data are often required. However, collecting and labelling data of such magnitude is time-consuming and cost-ineffective. To overcome this issue, we adopted a concept called transfer learning [1]. In this technique, weights obtained from the pre-trained model are used to initialize a new custom detector. In our case, we used weights from the model trained on the Microsoft COCO [2], which is a dataset consisting of 91 classes with over 2.5 million labelled instances in 3.25 lakh images. This technique helped us to build a better-generalized model with a low training time.

Our approach consists of two subsystems. The first sub-system of our proposed system, which is used for license plate detection, is based on the recently released YOLOv5 [3] which is a time-efficient successor of You Look Only Once [4] advanced deep learning object detection architecture. For this sub-system, we chose the lightweight version of YOLOv5, named v5small, which consists of 283 layers, 16.4 GFLOPS, and about 7 million parameters. For character recognition, we used an open-source OCR engine known as EasyOCR. It is an optical character recognition module that supports more than 80 languages, including Hindi, English, Chinese, Arabic, Dutch, and more. Moreover, this paper focuses more on license plate detection as EasyOCR itself is a state-of-the-art optical character recognition framework.

In this paper, the conceptual background regarding the technologies is given in Section 2. Section 3 discusses the previous work [5–8] completed in this field. Section 4 provides the methodology and implementation detail regarding the proposed model. Section 5 discusses the results obtained after the implementation. Finally, Section 6 discusses the conclusion and future work related to the research.

## 2. Conceptual Background

This section presents a brief background on the frameworks and models used. It describes the different versions of YOLO and offers a brief introduction to the EasyOCR library.

### 2.1. YOLOv5

“You look only once” is an advanced algorithm for real-time object detection. It is the go-to architecture for many AI computer vision engineers. YOLOv1 [4] was released in 2016 and it was a leap forward in real-time object detection research. After 1 year, in 2017, YOLO9000 was released. It is commonly known as YOLOv2. This version was faster, better, and developed to be on par with the performance of Faster-R-CNNs. After that, another darknet-based version named YOLOv3 [9] was released. This iteration is the most popular and brought down the detection errors drastically. In 2020, researchers published a much-advanced version of Yolo called YOLOv4 [10]. This version incorporated new features such as Mosaic data augmentation, Cross mini-Batch Normalization (CmBN), Cross-Stage-Partial-connections (CSP), Weighted-Residual-Connections (WRC), Self-adversarial-training (SAT), and Mish-activation. Using all these new features, YOLOv4 achieved a whopping 65.7% AP50 on the Microsoft COCO [2] dataset. Within a few days of the release of YOLOv4, PyTorch-based YOLOv5 [3] surfaced.

YOLOv5 is the newly released version of the Yolo family. It is a leading-edge deep learning algorithm that is very time performant and produces object detection models which are relatively smaller in size. For this paper, we chose its lightweight version called YOLOv5s (small). It is a 283-layers deep neural network with over 7 million trainable parameters. For more information about the layers, please refer to Figure 1 given below.

	from	n	params	module	arguments
0	-1	1	3520	models.common.Focus	[3, 32, 3]
1	-1	1	18560	models.common.Conv	[32, 64, 3, 2]
2	-1	1	18816	models.common.C3	[64, 64, 1]
3	-1	1	73984	models.common.Conv	[64, 128, 3, 2]
4	-1	1	156928	models.common.C3	[128, 128, 3]
5	-1	1	295424	models.common.Conv	[128, 256, 3, 2]
6	-1	1	625152	models.common.C3	[256, 256, 3]
7	-1	1	1180672	models.common.Conv	[256, 512, 3, 2]
8	-1	1	656896	models.common.SPP	[512, 512, [5, 9, 13]]
9	-1	1	1182720	models.common.C3	[512, 512, 1, False]
10	-1	1	131584	models.common.Conv	[512, 256, 1, 1]
11	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
12	[-1, 6]	1	0	models.common.Concat	[1]
13	-1	1	361984	models.common.C3	[512, 256, 1, False]
14	-1	1	33024	models.common.Conv	[256, 128, 1, 1]
15	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
16	[-1, 4]	1	0	models.common.Concat	[1]
17	-1	1	90880	models.common.C3	[256, 128, 1, False]
18	-1	1	147712	models.common.Conv	[128, 128, 3, 2]
19	[-1, 14]	1	0	models.common.Concat	[1]
20	-1	1	296448	models.common.C3	[256, 256, 1, False]
21	-1	1	590336	models.common.Conv	[256, 256, 3, 2]
22	[-1, 10]	1	0	models.common.Concat	[1]
23	-1	1	1182720	models.common.C3	[512, 512, 1, False]
24	[17, 20, 23]	1	16182	models.yolo.Detect	[1, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [11, 6, 90, 156, 198, 373, 326]], [128, 256, 512]]

Model Summary: 283 layers, 7063542 parameters, 7063542 gradients, 16.4 GFLOPS

Figure 1. Architecture of YOLOv5s.

YOLOV5 is a single-stage object detector with three important parts, namely, the model backbone, neck, and head. In the first part, the model backbone, important features from the provided input image are extracted. In this process, cross-stage partial networks are used to identify information-rich features from the image. The neck helps the model to perform well on unseen data. It generates feature pyramids which help the model to generalize well. It also helps to detect objects of varying sizes and scales. Lastly, the model head is used to perform the end decision. It carries the features generated by the model neck and generates the output vectors with scores and coordinates for bounding boxes.

Moreover, YOLOv5 is significantly smaller in size than any of its predecessors, yet it performs on par with all of them, in some cases better, and it is user-friendly to train and deploy. All these characteristics make it the perfect candidate for our study. Further in this paper, we will demonstrate the performance metrics that we used and the results obtained by our model.

## 2.2. EasyOCR

EasyOCR is a state-of-the-art OCR library that is efficient and supports over 80 languages. It is built using python and Facebook's PyTorch framework. It carries out detection using a character region awareness algorithm commonly known as CRAFT [11] which is a neural network-based scene text detection model. For recognition, EasyOCR uses a CRNN which consists of ResNet for feature extraction, LSTM sequence labelling, and CTC decoding. Figure 2 represents the exact working pipeline of EasyOCR. This pipeline also includes some pre-processing steps which make this engine one of the best [12].

Although this OCR engine can detect and recognize more than 80 languages, we used it in the English configuration. After detecting the license plate in the input image, we cropped and provided as the input to this OCR engine to obtain the license plate number.

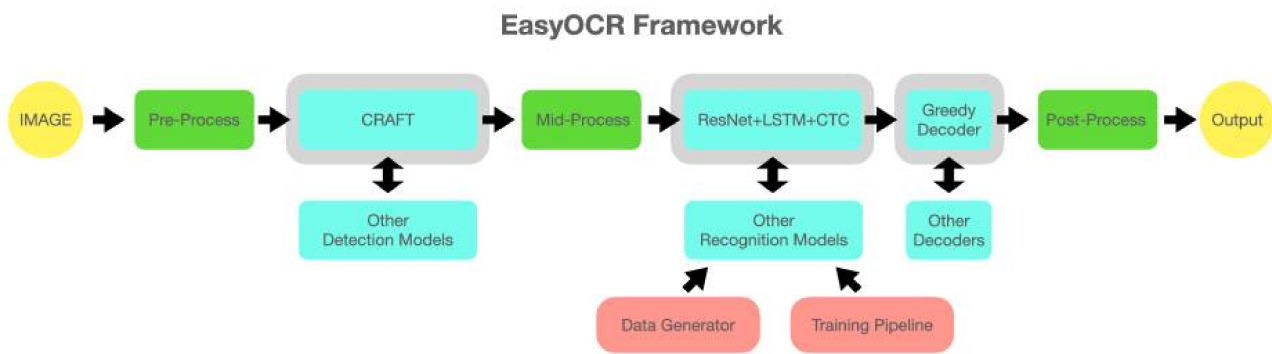


Figure 2. EasyOCR pipeline [12].

### 3. Literature Review

Vehicle license plate detection and recognition have been compelling areas of research for a long time. Computer vision researchers have devised many different approaches to solve the task at hand.

Over the past few years, many researchers have addressed the license plate detection task. For example, the authors of [13] used ResNet-50-based Faster-RCNN for Indian license plate detection and achieved a very high mAP. Mean Average Precision (mAP) is a commonly used measurement of precision for object detection models and measures the crossing of the predicted bounding box with the labelled bounding box.

The authors of [14] proposed a Fast-YOLO [15] and YOLOv2 [16]-based detector. In this approach, the authors trained two CNNs, first for vehicle detection and another for license plate detection. Their results suggested that Fast-Yolo had impressive results in both tasks. On the other hand, the authors of [17] used an exemplar SVM-based approach along with Fast and Faster-RCNN, depicting that RCNN was better suited for real-time detection. The authors of [18] also presented a YOLOv2 and Fast YOLO-based single cascaded CNN to detect both car frontal car views and licence plates, achieving high recall and precision rates. The authors of [19] used 500 images to train an AlexNet neural network and to run it on the Jetson TX1 board. Although they achieved a high accuracy, their testing set only consisted of 64 images. The authors of [20] proposed a combined deep learning end-to-end framework that eliminates the need for training separate systems for detection and recognition tasks. The authors of [21] presented a two-step approach based on InceptionV2 FasterRCNN and Long Short Term Memory (LSTM) Tesseract and managed to achieve a high accuracy in the detection task [22] using OpenALPR, which is a C++ based commercial license plate recognition library, to manage parking lot access using Raspberry Pi. The author of [23] presented a modified sliding window-based SWSCD-YOLO system that was tested for different climate conditions. This system attained test speeds of 800 ms to 1 s.

The authors of [24] used DNN to recognize the license plate numbers of cars. They used a single pass to achieve the detection and recognition of the same. In [25], the authors proposed a three-step approach for the detection and recognition of license plate numbers. They used YOLOv3. Their approach was devised to recognize number plates for different countries. The authors of [5] reviewed the state-of-the-art ALPR systems. They used different characteristics for comparison including datasets used, colors of plates, number fonts, etc. In [6], the authors proposed a CNN-based ALPR system. The claim was to achieve 98% recognition precision. The authors in [7] proposed an optimal K-means method using CNN for the automatic recognition of number plates. They performed a simulation to prove the effectiveness of their approach.

In [8], the authors proposed a statistical PRS. They used three cascading modules for the recognition and identification of number plates. The authors of [26] proposed a new method for detecting number plates using adaptive image segmentation and a probabilistic neural network (PNN) mechanism. They claimed to have achieved about 86% accuracy in prediction. The authors of [27] proposed a CNN-based approach using MD-YOLO for

multi-direction car LPR. An empirical evaluation was conducted to prove the improvement achieved in their approach. In [28], the authors proposed a method to recognize the car make and model through traffic cameras. The authors of [29] proposed a method using vertical edge detection for identifying car license plates. In [30], the researchers proposed an approach to identify license number plates for Italian vehicles. The authors claimed to have received a recognition accuracy of around 91%. An approach for real-time detection and segmentation of license plates was proposed in [31]. They claimed to have achieved an accuracy of 95% for classifying the digits. There are other researchers [32–41] who have proposed different approaches for the detection of license plates of vehicles. Table 1 provides a summary of this research.

**Table 1.** Summary of the state-of-the-art research in ALPR [24–45].

Sr No	Ref	Technique Used	Dataset Used	Accuracy Achieved
1	[24]	Unified DNN	CarFlag-Large, China	97.13%
2	[25]	YOLOv3 and YOLOv3-SPP	KarPlate, Korea	98.93%
3	[6]	Multi-task Convolutional Neural Networks (MTCNN)	Chinese City Parking Dataset (CCPD)	98%
4	[7]	OKM-CNN (Deep Learning-based K-Means) CNN	Stanford Cars dataset, FZU Cars Dataset, HumAIn 2019 Dataset	98.1%
5	[8]	Cascade Framework	SeeCar library	98.25%
6	[26]	two-layer probabilistic neural network (PNN)	Self- created sample set	86%
7	[27]	CNN-based MD-YOLO	ImageNet dataset and Application Oriented License Plate (AOLP) dataset	F Score 99.5%
8	[28]	Image processing part-based detector mechanism	Self-created dataset	99.4%
9	[29]	Vertical Edge Detection	Self-created dataset	91.6%
10	[30]	Image processing	Self Created Dataset	91%
11	[32]	Two-Step Key frames identification approach	Video Dataset	F score 91%
12	[33]	(YOLO)-darknet deep learning framework using sliding-window approach	AOLP dataset	78%
13	[34]	Two-stage Convolutional Neural Networks (CNNs) using YOLOv3 framework	JALPR dataset	87%
14	[35]	Deep Learning based real-time video monitoring	UFPRALPR dataset, SSIG-SegPlate dataset, and Low-Quality Plate-Videos dataset	95.83% for the AOLP dataset and 98.9% for CCPD datasets
15	[38]	mask region convolutional neural networks	AOLP, Caltech dataset	99.3% on AOLP and 98.9% on Caltech dataset.

All these results further encouraged us to use a novel YOLOv5-based deep learning approach for license plate detection in our system. Although YOLOv5 is fairly new, its performance in license plate detection tasks is remarkable.

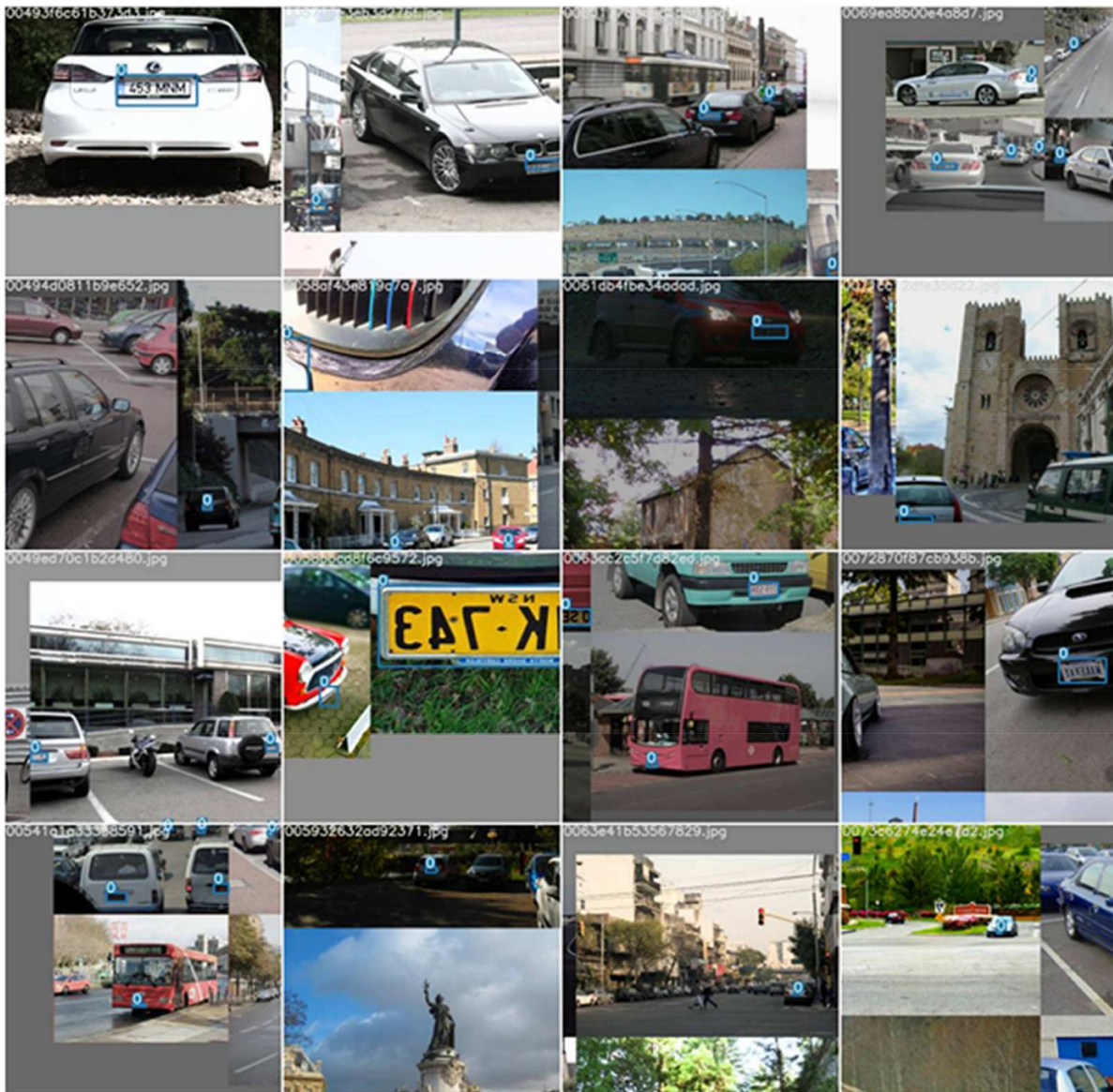
#### 4. Materials and Methods

This section presents a description of the datasets used, the performance metrics, and the methodology adopted for performing the task of automatic license plate recognition. A brief description of the training, testing, and validation data has been provided. The performance metrics such as mean average precision have been described in detail.

##### 4.1. Dataset

The dataset that we used for this paper is a combination of license plate data from the Google open Images dataset and the Indian number plates dataset that is available on

Kaggle. We put all the images from both datasets in a single folder and randomly split them into training, validation, and testing sets. Our dataset is available on Kaggle for the research community to use for further enhancement and sound comparison. This dataset contains 5991 images of vehicles in jpg format. Figure 3 represents some random samples from the aforementioned dataset.



**Figure 3.** Random samples from the dataset (<https://www.kaggle.com/datasets/daturks/vehicle-number-plate-detection>, accessed on 5 June 2022) (<https://storage.googleapis.com/openimages/web/index.html>, accessed on 5 June 2022).

The dataset was divided into 3 folders, namely train, test, and validation. The training set contained 4201 samples which roughly constituted 70% of the dataset. The validation set that was used to validate the model performance while training contained 1188 images which composed nearly 20% of the database. Lastly, the test dataset which was used to perform the final testing of the model contained 602 images.

All the images in this dataset have a text label file embedded with them. This label file contains bounding box coordinates for a number plate in YOLO annotation format. YOLO accepts the labels in a certain format given below:

$$\langle \text{object-id} \rangle \langle \text{center\_x} \rangle \langle \text{center\_y} \rangle \langle \text{width} \rangle \langle \text{height} \rangle \quad (1)$$

where:

object-id: number corresponding to object category;

Center\_x and center\_y: normalized center point of the bounding box;

Width and height: normalized width and height of the bounding box.

To create a robust system, our dataset contained images from different angles and different climate conditions. All these images were of different sizes and resolutions but the final input image to the model was provided in 640 pixels.

#### 4.2. Performance Metrics

For object detection tasks, mAP or mean average precision is often the performance metric of choice for many computer vision practitioners. Object detection tasks include identifying the relevant object in the images and then stratifying it into known classes. These tasks make predictions in the form of object class labels and a bounding box surrounding the object itself. Precision and recall are the two-core metrics that together were used to evaluate the mAP or performance of the model.

Precision is a measure of quality. It describes how often our model guessed correctly from all the guesses. Perfect precision means every guess is correct. On the other hand, recall is a measure of quantity; it indicates whether our model predicted the same result as expected or not. Perfect recall indicates that a model has made the correct number of predictions or has given full coverage.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP}) \quad (2)$$

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN}) \quad (3)$$

where:

TP: True positives;

TN: True negatives;

FP: False positives;

FN: False negatives.

The mAP was calculated by computing a series of precision-recall curves (AP) over varying Intersection over Union (IoU) levels. IoU is another performance metric that measures the accuracy of the bounding box on a particular task. In other words, IoU measures the overlap between 2 annotations. First, the average precision per class is calculated, and then it is averaged over all object categories to obtain mAP.

In this study, we evaluated mAP@0.5 and mAP@0.5:0.95 which means mean average precision with an IoU threshold of 0.5 and mean average mAP over multiple IoU thresholds, from 0.5 to 0.95, respectively.

#### 4.3. Methodology

This proposed ALPR framework consists of 2 subsystems: license plate detection and character recognition (Figure 4). For the first sub-system, the small and lightweight version of YOLOv5 is used. Encouraged by the success of previous YOLO models, in this paper, we used the YOLOv5s model for license plate detection. For training this model we used transfer learning and initialized the model with weights learned from the Microsoft COCO [2] dataset. This dataset contains 91 classes of objects with over 2.5 million labelled instances in 3.25 lakh images. In this approach, we adapted the features learned from a more extensive dataset and employed them for our use. Utilizing such learned features



allowed us to build a robust and high-performing model from the available data. Along with this, we used an SGD optimizer and trained our model for 100 epochs.

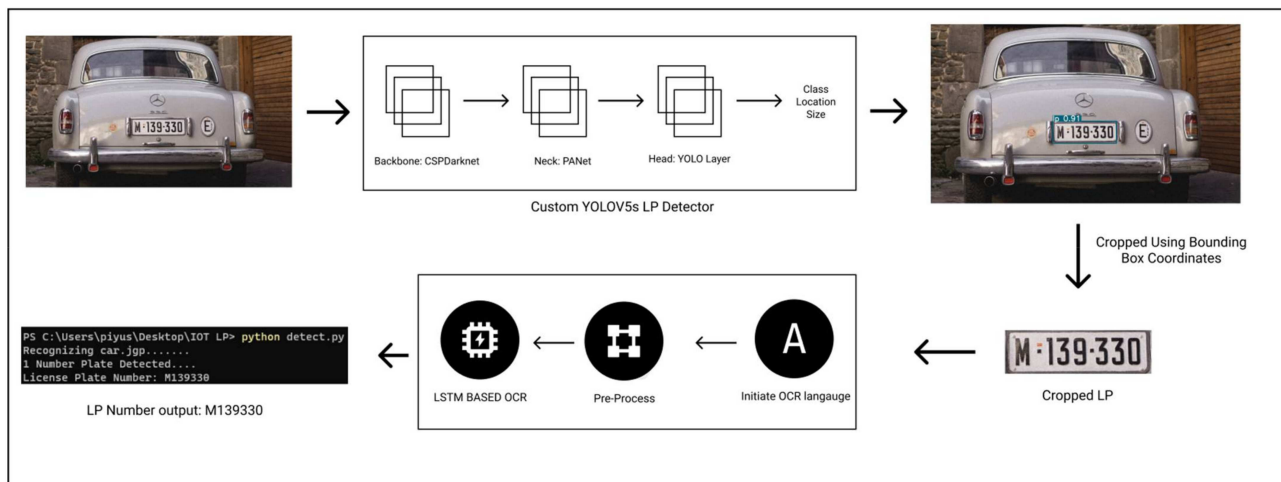


Figure 4. Overview of the methodology.

This license plate detection subsystem takes input in the form of still images or videos or a network stream and detects the license plate in every frame. The output of this system is the coordinates for the detected bounding box. After detecting the license plate from the image, we used OpenCV [46] to further process it. Using the bounding box coordinates from the first module, we cropped out the license plate and fed it to the next subsystem. This subsystem utilizes the power of the LSTM-based EasyOCR engine to recognize the license plate number.

The aforementioned system takes about 4.8 ms for license plate detection and about 80 ms for detection on NVIDIA T4 GPU. Due to this fast inference time, it is an ideal system for IoT processing devices such as Jetson Nano or Google coral, or Intel NCS.

### 5. Results and Discussion

From Table 2 we can see that our model achieved an mAP of 88.4% on our validation set and 87.2% on our test set. This mAP was calculated over the intersection threshold (IoU) of 0.5. Our model also achieved a very high recall and precision. Along with achieving an mAP, our resultant model was only 14 megabytes with a testing time of 4.8 ms. This inference time was considerably fast and could help to develop a robust real-time ALPR system.

Table 2. Dataset Details.

	Train	Validation	Test	Total
License Plate	4201	1188	602	5991
	70.12%	19.83%	10.05%	

where:

R = Recall

P = Precision

mAP = Mean Average Precision with an IoU threshold of 0.5

mAP.95 = mean average mAP over multiple IoU thresholds, from 0.5 to 0.95; mAP@[0.5:0.95]

Table 3 shows the mAP values obtained using the model YOLOv5s. As we intended to deploy this proposed model on an IoT processing device, we also compared our model to the tiny version of darknet-based YOLOv4. We trained a tiny YOLOv4 model on the

same dataset to illustrate a sound comparison. As you can see from Table 4, Tiny YOLOv4 achieved an mAP of 82.68% when trained and tested on the same dataset. It also produced a 22 MB model. Moreover, the testing time on the tiny yolov4 model was dramatically higher, 23.1 ms, than our proposed model.

**Table 3.** Results.

Model	Epoch	Test (%)				Validation (%)				Size	Avg Test Time
		mAP	mAP:95	R	P	mAP	mAP:95	R	P		
YOLOv5s	100	87.2	46.5	82.2	88.2	88.4	49.5	84.3	87.8	14 MB	4.8 ms

**Table 4.** Comparison between YOLOv5s and tiny YOLOv4.

Model	Test mAP	Validation mAP	Size	Test Time
YOLOv5s	87.2%	88.4	14 MB	4.8 ms
Tiny YOLOv4	82.68%	83.95	22 MB	23.1 ms

We compared our results with other work that has been completed using similar architecture but different datasets. Due to the novelty of YOLOv5, not a lot of research has been undertaken using it. Tables 4 and 5 shows the comparison between the work in [42,47] and our model. For a sound comparison, we used their trained model and tested it on our dataset.

**Table 5.** Comparison with other work completed with YOLOv5.

Model	mAP@0.5	mAP@0.5:0.95	Recall	Precision
Our model	87.2%	46.5%	82.2%	88.2%
Other work [42]	9.1%	2.4%	18.3%	17.9%

These results show that our approach outperforms the previous work. The reason for this drastic performance difference might be that the dataset used by the author of [47] consisted of about 300 images. Some systems use OpenCV contours to detect the license plate [48], while others implement deep learning neural nets for plate localization.

Our proposed methodology for license plate detection achieved promising results. With some further changes, this system could be deployed for real-time detection.

## 6. Conclusions

This study aimed to develop a framework for license plate detection and recognition. Our model was intended for IoT processing devices. We applied the concepts of transfer learning and trained a small YOLOv5 model. We presented our results in terms of license plate detection as we used a state-of-the OCR recognition engine to carry out the character recognition task. Our results suggest that YOLOv5s is a very capable architecture and produces a time and memory-efficient model. Our approach resulted in a 14 megabytes model, which is appropriate for IoT devices with less memory. When tested, it performed well with a mean average precision of 87.2% and 4.8 ms testing time on still images using Nvidia T4 GPU.

Our model achieved a high mAP. Among other advantages, we achieved a low response time with a higher accuracy. However, there is still room for improvement. The two major limitations of our system are its ability to work under low visibility and night conditions and the lack of real-world performance measurements. In future work, we plan to modify the underlying architecture of this license plate recognition system and mold it in a way highly suitable for low processing devices. We also intend to train and test our model for low visibility conditions so that it works seamlessly during the night and bad weather. Additionally, we would also train a deep neural net for character recognition;

hence, eliminating the need to use an external OCR engine. A lightweight OCR system would help us present a single end-to-end ALPR solution for IoT devices. Lastly, we intend to deploy this model on an IoT processing device and check its real-time performance.

**Author Contributions:** Conceptualization, P.B., M.A.A. (Mohammad Afshar Alam) and I.H.; methodology, A.K., G.C. and M.A.A. (Mohd Abdul Ahad); software, P.B., S.I.H. and I.H.; validation, A.K., I.H., S.I.H. and P.B.; formal analysis, G.C. and M.A.A. (Mohd Abdul Ahad); investigation, P.B., A.K., M.A.A. (Mohammad Afshar Alam) and G.C.; resources, P.B. and I.H.; data curation, A.K., S.I.H., P.B. and G.C.; writing—original draft preparation, P.B., M.A.A. (Mohd Abdul Ahad), A.K., S.I.H., I.H. and G.C.; writing—review and editing, M.A.A. (Mohd Abdul Ahad), G.C., I.H., M.A.A. (Mohammad Afshar Alam) and P.B.; visualization, I.H., S.I.H. and A.K.; supervision, I.H., M.A.A. (Mohd Abdul Ahad), G.C., M.A.A. (Mohammad Afshar Alam) and A.K. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

Abbreviation	Explanation
ALPR	Automatic number-plate recognition
LSTM	Long Short-Term Memory
OCR	Optical Character Recognition
YOLO	You Only Look Once
IoT	Internet of Things
COCO	Common Objects in Context
CRAFT	Character-Region Awareness For Text detection
CRNN	Convolutional recurrent neural network
RCNN	Regions with convolutional neural networks
DNN	Deep neural networks
MTCNN	Multi-Task Cascaded Convolutional Neural Networks
SGD	Stochastic Gradient Descent
mAP	mean average precision

## References

- Pan, S.J.; Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **2009**, *22*, 1345–1359. [[CrossRef](#)]
- Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft Coco: Common Objects in Context. In *Computer Vision—ECCV 2014*; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2014; Volume 8693.
- Jocher, G.; Stoken, A.; Borovec, J.; Christopher, S.T.; Laughing, L.C. Ultralytics/yolov5: v4.0-nn.SiLU() activations; Weights & Biases logging, PyTorch Hub integration. *Zenodo* **2021**. [[CrossRef](#)]
- Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27 June 2016; pp. 779–788.
- Du, S.; Ibrahim, M.; Shehata, M.; Badawy, W. Automatic License Plate Recognition (ALPR): A State-of-the-Art Review. *IEEE Trans. Circuits Syst. Video Technol.* **2012**, *23*, 311–325. [[CrossRef](#)]
- Wang, W.; Yang, J.; Chen, M.; Wang, P. A Light CNN for End-to-End Car License Plates Detection and Recognition. *IEEE Access* **2019**, *7*, 173875–173883. [[CrossRef](#)]
- Pustokhina, I.V.; Pustokhin, D.A.; Rodrigues, J.J.; Gupta, D.; Khanna, A.; Shankar, K.; Seo, C.; Joshi, G.P. Automatic Vehicle License Plate Recognition Using Optimal K-Means with Convolutional Neural Network for Intelligent Transportation Systems. *IEEE Access* **2020**, *8*, 92907–92917. [[CrossRef](#)]
- Wang, S.-Z.; Lee, H.-J. A Cascade Framework for a Real-Time Statistical Plate Recognition System. *IEEE Trans. Inf. Forensics Secur.* **2007**, *2*, 267–282. [[CrossRef](#)]

9. Farhadi, A.; Redmon, J. Yolov3: An incremental improvement. In Proceedings of the Computer Vision and Pattern Recognition Conference, Salt Lake City, UT, USA, 18 June 2018; pp. 1–6.
10. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
11. Baek, Y.; Lee, B.; Han, D.; Yun, S.; Lee, H. Character region awareness for text detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15 June 2019; pp. 9365–9374.
12. GitHub-JaidedAI/EasyOCR: Ready-to-Use OCR with 80+ Supported Languages and all Popular Writing Scripts Including Latin, Chinese, Arabic, Devanagari, Cyrillic, etc. Available online: <https://github.com/JaidedAI/EasyOCR> (accessed on 24 April 2022).
13. Ravirathinam, P.; Patawari, A. Automatic License Plate Recognition for Indian Roads Using Faster-RCNN. In Proceedings of the 11th International Conference on Advanced Computing (ICoAC), Chennai, India, 18–20 December 2019; pp. 275–281.
14. Laroca, R.; Severo, E.; Zanlorensi, L.A.; Oliveira, L.S.; Goncalves, G.R.; Schwartz, W.R.; Menotti, D. A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–10. [CrossRef]
15. Shaifee, M.J.; Chywl, B.; Li, F.; Wong, A. Fast YOLO: A Fast You Only Look Once System for Real-time Embedded Object Detection in Video. *arXiv* **2017**, arXiv:1709.05943. [CrossRef]
16. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21 July 2017; pp. 7263–7271.
17. Rafique, M.A.; Pedrycz, W.; Jeon, M. Vehicle license plate detection using region-based convolutional neural networks. *Soft Comput.* **2018**, *22*, 6429–6440. [CrossRef]
18. Montazzolli, S.; Jung, C. Real-time brazilian license plate detection and recognition using deep con-volutional neural networks. In Proceedings of the 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), Niterói, Brazil, 17 October 2017; pp. 55–62.
19. Lee, S.; Son, K.; Kim, H.; Park, J. Car plate recognition based on CNN using embedded system with GPU. In Proceedings of the 10th International Conference on Human System Interactions (HSI), Ulsan, Korea, 17–19 June 2017; pp. 239–241.
20. Wang, H.; Li, Y.; Dang, L.-M.; Moon, H. Robust Korean License Plate Recognition Based on Deep Neural Networks. *Sensors* **2021**, *21*, 4140. [CrossRef]
21. Singh, J.; Bhushan, B. Real Time Indian License Plate Detection using Deep Neural Networks and Optical Character Recognition using LSTM Tesseract. In Proceedings of the International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 18–19 October 2019; pp. 347–352. [CrossRef]
22. Buhus, E.R.; Timis, D.; Apatean, A. Automatic parking access using openlpr on raspberry pi3. *Acta Tech. Napoc.* **2016**, *57*, 10.
23. Setiyono, B.; Amini, D.A.; Sulistyaningrum, D.R. Number plate recognition on vehicle using YOLO-Darknet. *J. Phys. Conf. Ser.* **2021**, *1821*, 012049. [CrossRef]
24. Li, H.; Wang, P.; Shen, C. Toward End-to-End Car License Plate Detection and Recognition with Deep Neural Networks. *IEEE Trans. Intell. Transp. Syst.* **2018**, *20*, 1126–1136. [CrossRef]
25. Henry, C.; Ahn, S.Y.; Lee, S.-W. Multinational License Plate Recognition Using Generalized Character Sequence Detection. *IEEE Access* **2020**, *8*, 35185–35199. [CrossRef]
26. Anagnostopoulos, C.N.E.; Anagnostopoulos, I.E.; Loumos, V.; Kayafas, E. A license plate-recognition algo-rithm for intelligent transportation system applications. *IEEE Trans. Intell. Transp. Syst.* **2006**, *7*, 377–392. [CrossRef]
27. Xie, L.; Ahmad, T.; Jin, L.; Liu, Y.; Zhang, S. A New CNN-Based Method for Multi-Directional Car License Plate Detection. *IEEE Trans. Intell. Transp. Syst.* **2018**, *19*, 507–517. [CrossRef]
28. He, H.; Shao, Z.; Tan, J. Recognition of Car Makes and Models from a Single Traffic-Camera Image. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 3182–3192. [CrossRef]
29. Al-Ghaili, A.M.; Mashohor, S.; Ramli, A.R.; Ismail, A. Vertical-Edge-Based Car-License-Plate Detection Method. *IEEE Trans. Veh. Technol.* **2012**, *62*, 26–38. [CrossRef]
30. Comelli, P.; Ferragina, P.; Granieri, M.; Stabile, F. Optical recognition of motor vehicle license plates. *IEEE Trans. Veh. Technol.* **1995**, *44*, 790–799. [CrossRef]
31. Corneto, G.L.; Da Silva, F.A.; Pereira, D.R.; De Almeida, L.L.; Artero, A.O.; Papa, J.P.; Albuquerque, V.H.C.; Sapia, H.M. A New Method for Automatic Vehicle License Plate Detection. *IEEE Lat. Am. Trans.* **2017**, *15*, 75–80. [CrossRef]
32. Rajkumar, C.; Mahendran, S. Enhanced key frame identification and background subtraction methods for automatic license plate recognition of motorcyclists without helmets. *Mater. Today Proc.* **2021**. [CrossRef]
33. Hendry; Chen, R.-C. Automatic License Plate Recognition via sliding-window darknet-YOLO deep learning. *Image Vis. Comput.* **2019**, *87*, 47–56. [CrossRef]
34. Alghyaline, S. Real-time Jordanian license plate recognition using deep learning. *J. King Saud Universi-Ty-Comput. Inf. Sci.* **2020**, *34*, 2601–2609. [CrossRef]
35. Zhang, C.; Wang, Q.; Li, X. V-LPDR: Towards a unified framework for license plate detection, tracking, and recognition in real-world traffic videos. *Neurocomputing* **2021**, *449*, 189–206. [CrossRef]
36. Silva, S.M.; Jung, C.R. Real-time license plate detection and recognition using deep convolutional neural net-works. *J. Vis. Commun. Image Represent.* **2020**, *71*, 102773. [CrossRef]
37. Slimani, I.; Zaarane, A.; Al Okaiishi, W.; Atouf, I.; Hamdoun, A. An automated license plate detection and recognition system based on wavelet decomposition and CNN. *Array* **2020**, *8*, 100040. [CrossRef]

38. Selmi, Z.; Ben Halima, M.; Pal, U.; Alimi, M.A. DELP-DAR system for license plate detection and recognition. *Pattern Recognit. Lett.* **2020**, *129*, 213–223. [[CrossRef](#)]
39. Jintasuttisak, T.; Edirisinghe, E.; Elbattay, A. Deep neural network based date palm tree detection in drone imagery. *Comput. Electron. Agric.* **2022**, *192*, 106560. [[CrossRef](#)]
40. Wu, W.; Liu, H.; Li, L.; Long, Y.; Wang, X.; Wang, Z.; Li, J.; Chang, Y. Application of local fully Convolutional Neural Network combined with YOLO v5 algorithm in small target detection of remote sensing image. *PLoS ONE* **2021**, *16*, e0259283. [[CrossRef](#)]
41. Dewi, C.; Chen, R.-C.; Yu, H.; Jiang, X. Robust detection method for improving small traffic sign recognition based on spatial pyramid pooling. *J. Ambient Intell. Humaniz. Comput.* **2021**, 1–18. [[CrossRef](#)]
42. Wang, Z.; Jin, L.; Wang, S.; Xu, H. Apple stem/calyx real-time recognition using YOLO-v5 algorithm for fruit automatic loading system. *Postharvest Biol. Technol.* **2022**, *185*, 111808. [[CrossRef](#)]
43. Kocer, H.E.; Çevik, K.K. Artificial neural networks based vehicle license plate recognition. *Procedia Comput. Sci.* **2011**, *3*, 1033–1037. [[CrossRef](#)]
44. Davix, X.A.; Christopher, C.S.; Judson, D. Detection of the vehicle license plate using a kernel density with default search radius algorithm filter. *Optik* **2020**, *218*, 164689. [[CrossRef](#)]
45. Ma, L.; Zhang, Y. Research on vehicle license plate recognition technology based on deep convolutional neural networks. *Microprocess. Microsyst.* **2021**, *82*, 103932. [[CrossRef](#)]
46. Bradski, G. The openCV library. *Dr. Dobb's J. Softw. Tools Prof. Program.* **2000**, *25*, 120–123.
47. Baldota, S. How YOLOv5 Solved an Ambiguity Encountered by YOLOv3, Robust Indian License Plate Detection Using YOLOv5. Jun 21, 2020. Available online: [https://github.com/sid0312/anpr\\_yolov5](https://github.com/sid0312/anpr_yolov5) (accessed on 22 March 2022).
48. Abedin, Z.; Nath, A.C.; Dhar, P.; Deb, K.; Hossain, M.S. License plate recognition system based on contour properties and deep learning model. In Proceedings of the EEE Region 10 Humanitarian Technology Conference, Dhaka, Bangladesh, 21–23 December 2017; pp. 590–593. [[CrossRef](#)]