

Received 23 December 2022, accepted 12 February 2023, date of publication 8 March 2023, date of current version 24 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3254365

## TOPICAL REVIEW

# License Plate Recognition Methods Employing Neural Networks

MUHAMMAD MURTAZA KHAN<sup>1</sup>, (Member, IEEE),

MUHAMMAD U. ILYAS<sup>2,3</sup>, (Senior Member, IEEE), ISHTIAQ RASOOL KHAN<sup>1</sup>, (Member, IEEE), SALEH M. ALSHOMRANI<sup>1</sup>, AND SUSANTO RAHARDJA<sup>4,5</sup>, (Fellow, IEEE)

<sup>1</sup>Department of Computer Science and Artificial Intelligence, College of Computer Science and Engineering, University of Jeddah, Jeddah 21589, Saudi Arabia

<sup>2</sup>Department of Computer and Network Engineering, College of Computer Science and Engineering, University of Jeddah, Jeddah 23437, Saudi Arabia

<sup>3</sup>Department of Computer Science, College of Engineering and Physical Sciences, University of Birmingham Dubai, Dubai, United Arab Emirates

<sup>4</sup>School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an 710072, China

<sup>5</sup>Infocomm Technology Cluster, Singapore Institute of Technology, Singapore 138683

Corresponding author: Muhammad Murtaza Khan (mkhan@uj.edu.sa)

This work was supported by the Deanship of Scientific Research (DSR), University of Jeddah, under Grant UJ-03-18-ICP.

**ABSTRACT** Advances in both parallel processing capabilities because of graphical processing units (GPUs) and computer vision algorithms have led to the development of deep neural networks (DNN) and their utilization in real-world applications. Starting from the LeNet-5 architecture of the 1990s, modern deep neural networks may have tens to hundreds of layers to solve complex problems such as license plate detection or recognition tasks. In this article, we present a review of the state-of-the-art methods related to automatic license plate recognition. Since deep networks have demonstrated a remarkable ability to outperform other machine learning techniques, we focus only on neural network based license plate recognition methods. We highlight the particular types of networks, i.e., convolutional, residual recurrent, or long-short-term-memory, used for the specific tasks of license plate detection, extraction, or recognition in different existing works. The presented summary also highlights some of the most widely used data sets for comparison and shares the results reported in the reviewed papers. We also give an overview of the effects of fog, motion, or the use of synthetic data on license plate recognition. Finally, promising directions for future research in this domain are presented.

**INDEX TERMS** License plate, detection, recognition, deep learning, neural networks.

## I. INTRODUCTION

License plate recognition (LPR) has received much interest from the research community for the past few decades. During this time, different methods have emerged, from elementary to complex, tackling problems from detecting standard plates captured under controlled conditions to recognizing all characters and digits under challenging conditions [36]. Most methods relied on handcrafted features explicitly designed for number plates of a particular shape and size, alphanumeric font type, size and number of digits, background, and text color, and even reflective properties of specific standardized license plates [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Derek Abbott<sup>id</sup>.

To understand why these proposed methods relied on specific handcrafted features, Anagnostopoulos highlighted the main issues related to capturing, detecting, and recognizing license plates in [3]. The author stated that the factors that must be considered while capturing a license plate revolved around triggering the capturing mechanism using either hardware sensors or software-based triggers using change detection. The captured image must include a license plate image of appropriate size, i.e., the number of pixels occupied by a license plate should be large enough for alphanumeric characters to be discernible in the image. This can be determined from the distance of capturing device to license plate, the size of CCD sensor used to capture image, the tilt or pan of capturing device, and the focal length of lens. The author also emphasized using infrared sensors and illumination sources

to capture license plates at night, as many license plates are made using retro-reflective materials.

In literature, the process of recognizing license plates is generally divided into the following three steps: i) Detection and extraction of the region containing the license plate; ii) Segmentation of the license plate characters; iii) Recognition of each alphanumeric character [2]. In their review paper [17], Du et al. presented a summary of the rationale along with the pros and cons of using boundary, texture, color, character size, or a hybrid of features for extracting license plates in images. They identified that most researchers preferred pixel connectivity, projection profiles, prior knowledge of characters, character contours, or a combination of these techniques for character detection and segmentation. They observed that the methods used for character recognition varied significantly based upon the underlying algorithms, including template matching, horizontal and vertical projections, Hotelling transform [31], transitions between foreground and background pixels, Gabor filter, Kirsch edge detection, contour tracking, contour crossing, and topological features. These conventional methods focus on a specific type of license plate and the conditions in which it was captured.

Similar to license plate recognition, several other computer vision problems were also addressed with conventional learning methods using handcrafted features. However, the advent of modern neural networks, or deep neural networks (DNNs), changed this problem-solving approach [70]. AlexNet [49], proposed in 2012, demonstrated a significant increase in the accuracy of object recognition tasks in computer vision algorithms and ushered in the era of deep learning. The 10.8% gain shown by AlexNet in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) significantly outperformed the conventional methods. This difference in performance has also raised ample interest in exploring and identifying different deep-learning architectures for license plate detection and recognition. The influence of deep learning algorithms can be observed with training and testing data becoming more complex and challenging and license plate recognition rates getting higher and closer to the human level. Another advantage of DNNs is that a network designed for one data may be used directly on another data without changing the network as the features extracted by the network will be data dependent and hence do not require human intervention.

The complete body of literature on license plate recognition methods is large and is mostly based on conventional methods that do not produce the best results. In [2] and [17], the authors presented a comprehensive review of these methods. We are focusing our attention on methods that employ deep neural networks. We have divided the related literature into three principal categories, i.e., i) License plate detection and character segmentation methods, ii) Optical character recognition (OCR)-only methods, and iii) License plate recognition methods. The second category is a large research area in its own right; however, we have covered some papers which use OCR methods after detecting the

license plates and extracting their characters. The first and third categories, i.e., license plate detection and license plate recognition, are further categorized or divided based on the type of deep neural network architecture used. This is elaborated using figures presenting a general description of the type of network. A tabular comparison between different methods in each category is also presented, which contains information about the deep learning architecture, used data set, results, and, if given, efficiency in terms of execution time. A literature survey matrix has been added to present a summary of the methods and details discussed later in Section II. The scope of this work is shown in Figures 1 & 2. Figure 1 presents the methods employed for the individual tasks of license plate detection and optical character recognition, whereas Figure 2 presents methods that holistically address the complete pipeline of license plate detection and recognition.

To summarize, the main contributions of this work are:

- Providing detailed information about neural networks, specifically deep networks, employed in the area of license plate detection, character recognition and license plate recognition, in the past decade.
- Presenting a summary of deep learning methods employed for the task of license plate detection in Table 3.
- Presenting a summary of methods employed for optical character recognition, specifically in the context of license plates, in Table 4.
- Segregation of deep learning methods based upon the type of deep learning architecture in Section V and discussion on their architectural details.
- Summary of the network along with details such as data set used, country of the license plate, results, and execution time when available.
- Presentation of some key architectures in the form of block diagrams.
- Discussion on associated issues, including use and generation of synthetic images, re-identification, and denoising, in Section VI.
- Discussion on type and trends of neural networks that have been employed in the literature, in Section VII.
- Discussion on pros and cons of the different categories of neural networks in the context of license plate recognition, in Section VII.

The remainder of this paper is organized as follows. Section II presents a brief introduction to the topic and presents the literature survey matrix. Section III presents a review of papers that use neural networks for license plate detection and extraction. Next, Section IV presents a brief review of methods focusing on character recognition of license plates followed by Section V covering license plate recognition methods. Topics associated with license plates i.e., re-recognition, deformation or blur removal etc are presented in Section VI. Finally, Section VII presents a summary of the popular approaches and future directions in the area.

## II. REVIEW: LITERATURE SURVEY MATRIX

Literature on license plate recognition has thoroughly addressed the three main research areas: license plate detection, character recognition, and license plate recognition. Most earlier works that used handcrafted features were generally applied to specific license plate types. However, these conventional object detection and recognition methods have recently been replaced by neural networks based deep learning techniques in the computer vision domain. The use of neural network based methods has resulted in improved accuracy and ability to recognize license plates in more challenging environments. Therefore, it may be more interesting to focus on neural network based methods in the area of license plate recognition because of their superior performance. We attempt to present and compare different types of neural networks proposed in the literature. We highlight if the complete process of recognition has been achieved by using a neural network or if there is a preference to solve only a particular task in the recognition process using these state-of-the-art algorithms.

An overview of some papers discussed in this review is presented as a literature survey matrix in Table 1. In this matrix, we have presented the first author's name, the year of publication, and the paper's reference number. The matrix also indicates if the paper proposes a solution for character recognition, license plate detection (LPD), license plate recognition (LPR), or the complete pipeline. The matrix also presents the type of network used for the specific tasks. For example, if the authors used a convolutional neural network (CNN) for license plate detection, it is mentioned in the table. The presented matrix can be used to identify the type of data set used, the country of origin of the license plate, the F1 score or accuracy reported for the method, the availability of the data set, and the number of citations to the work in the literature. It should be noted that certain papers have reported performance results for individual data sets. In the literature review matrix, those results are averaged for presentation to save space. However, they shall be presented separately when those methods are discussed in the review. Other parameters, such as the amount of data used for training and testing, detailed results for each data set, and the size of images, are not shared here but in the subsection discussing each method.

One of the main issues associated with deep learning techniques is the requirement for large amounts of data. Since these techniques are data-dependent, the training data's variability and quantity impact performance. Without having access to some benchmark data sets, it is not possible to have a fair comparison of different methods. In this regard, the data sets mentioned in Table 1 act as a good reference for the community. As mentioned in the notes of Table 1 some of the data sets are publicly available, while others were available but have been removed by the authors. To help the reader understand the type and variety of images available in the data sets, we have presented samples from some open-source options. These images were downloaded from the links provided in the notes of Table 1.

One of the earliest available and widely used dataset is the Application Oriented License Plate (AOLP) dataset containing Taiwanese license plates. Collected by Hsu et al., it comprises of three categories named Access Control (AC), Law Enforcement (LE) and Road Patrol (RP) containing 681, 757 and 611 images respectively [36]. AC images have panning from  $-30$  to  $30$  degrees, tilting from  $0$  to  $60$  degrees, capturing distance of less than  $5\text{m}$ , the width of the license plate is  $0.2$  to  $0.25$  times of the image and was captured using a toll station type entrance. LE images have panning from  $-40$  to  $40$  degrees, tilting from  $20$  to  $70$  degrees, capturing distance of less than  $15\text{m}$ , the width of the license plate is  $0.1$  to  $0.2$  times that of the image and contains images captured when the vehicle violated a traffic rule. Finally, the RP images have panning from  $-60$  to  $60$  degrees, tilting from  $0$  to  $50$  degrees, capturing distance of less than  $15\text{m}$ , the width of the license plate is  $0.1$  to  $0.4$  times that of the image and captured either while using a handheld device or from a mounted device.

The Caltech Cars1999 data set consists of 126 car images with a resolution of  $896 \times 592$  pixels. Another data set comprising 526 images at  $360 \times 240$  pixels resolution is also available at the same link. The first data set includes parked cars during the daytime, while the second data set, Caltech Cars2001, was captured on a highway. Sample images of the data set are displayed in the first row of Table 2. A data set shared by the University of California, San Diego comprises around 10 hours of video of cars entering and leaving the campus during various times of the day. The still frames extracted from these videos were hand-labeled; hence, data for 878 cars were annotated. The data set also contains 291 car images taken in the car park using a handheld digital camera.

The Peking University data set (PKU) is created by the National Engineering Laboratory for Video Technology (NELVT), a research group at Peking University China. The data set was captured by surveillance cameras in China during both day and night. The data set consists of 221,763 images, out of which 90,196 are labeled. Images in the data set are divided into five categories, where G4 and G5 categories contain high-resolution images compared to G1, G2, and G3. Only images in the G5 category have multiple cars in an image. These images are displayed in the second, third, fourth, and fifth rows of Table 2.

Bjorklund et al. proposed using synthetic images containing license plates in natural backgrounds and typical background images with no license plates. The images used by them are presented in the row titled "Synthetic", in Table 2. The data set developed by Stanford contains approximately 16,185 images of 196 cars. These images are presented in the seventh row of Table 2.

The Chinese Car Parking Data set (CCPD) is a comprehensive data set, 12 GB in size, containing over 300,000 images of Chinese license plates captured by a handheld camera by license plate inspectors across China. The images were acquired between 0730 hrs and 2200 hrs. Another Chinese license plate data set titled Chinese Road Plate Dataset

(CRPD) was developed by Gong et al. [25] because they noticed the lack of multi-objective real-world data sets. They captured images using electronic monitoring systems. The annotated images can be obtained from Github [24]. The main challenge presented by the dataset is the availability of images with single, double and multiple license plates.

Images taken from Greek license plates were made available by a research group at the National Technical University of Athens and are presented as LPRD in Table 2. The data set has images categorized as still images, difficult images, and videos. The data is further subdivided into images captured during daytime or night. There is also a division between color, grayscale, and blurry images. Images that have shadows, are close-ups, or were taken from far away are also saved separately.

A Brazilian car data set titled “Federal University of Parana-Automatic License Plate Recognition”, UFPR-ALPR, comprising 300 vehicles, was collected by Laroca et al. [51]. The images in the data set have a resolution of  $1920 \times 1080$  pixels per image. The data contains multiple images of the same vehicle captured from another vehicle while both are moving. This data set can be beneficial in developing an LPR system for deployment in police cars. License plates used in Tunisia [78] and Saudi Arabia [45] are also presented at the bottom of the table. The images in the Saudi Arabian database have a resolution of  $1920 \times 1080$  pixels and are captured from a moving vehicle. The license plate images from Iran are also shown in the table and have been obtained from [82]. The data set comprises approximately 20,000 images, and since these images contain only license plates, their dimensions are not fixed.

Spanhel et al. [92], shared a database comprising of relatively low-quality images for recognition of license plates. The authors also provided an HDR dataset, which is relatively smaller in size and ReID database, which may be used for re-identification of vehicles. Vehicle re-identification is a topic associated with license plate recognition as LPR may be a part of the process.

Use of synthetic license plates to train large neural networks has gained some interest in the last few years and in this context a tool to generate license plates was developed by Matthew Earl and has been made available for use [19]. The author has shared the procedure for generation of synthetic license plate images and for developing a basic deep learning architecture for detection and recognition of license plates. The authors Usmankhujiev et al. have made their tool for generation of Korean license plates available on GitHub at [99]. The utility allows generation of five different types of Korean license plates, i.e., with white, yellow or green backgrounds and different shapes and sizes.

The data sets presented above focus on different challenges faced during license plate recognition. The first challenge is associated to detection of license plate in an image. In this regard synthetic data sets have been employed as they can take license plate images and merge or paste them in various

backgrounds. The background may be a simple or complex as shown in Table 2 as “Synthetic images”. The challenges associated to license plate detection also include the angle at which the license plate is being viewed. As an example, in the data set shared by Khan et al. [45], the images were taken via a moving vehicle, by placing the camera on the dash. This causes vehicles to either enter the frame from the edge and move to the front, when the car is being overtaken, resulting in a skewed plate at the beginning and a rectangular plate when the other car is in front of the car capturing the images. Videos were also captured from the side view window of the car. This causes the license plates of parked cars to appear rectangular, however, the license plates of car passing by appear non-rectangular. Thus, making the task of detection more difficult.

Further challenges addressed by the above-mentioned data sets include variability in characters. Depending upon the country of origin the license plate may contain English, Chinese, Arabic, Persian characters etc and hence the license plate recognition algorithm has to be trained on a data set containing the characters of interest. This also holds true for the numeric digits. This also visible from the diverse license plate types shown in Table 2. Examples of Persian and Arabic license plates are show the variability in the numbers and text used in the license plates as compared to English based license plates. Similar difference can be seen for Chinese license plates in the figure. The Persian language has 32 unique characters as compared to 26 for English language and 29 for Arabic. Some languages including Chinese does not have an alphabet system and hence the various number plate permutations are not dependent upon them.

Finally, the change in lighting conditions also cause difficulty in recognition of license plates. This is covered very efficiently in PKU data set. The data set not only provides images of vehicles at different times of day but also comprises of the challenging case of license plate obstruction i.e. complete license plate is not visible in the frame.

A recent dataset on Brazilian license plates was developed by Laroca et al. [53]. They introduced a publicly available dataset called RodoSol-ALPR containing 20,000 images captured at toll booths installed on Brazilian highway of car and motorcycle license plates from Mercosur (Brazil, Argentina, Paraguay and Uruguay) region. In the work they proposed a traditional-split versus leave-one-dataset-out experimental setup for assessing the cross-dataset generalization. They used 12 Optical Character Recognition (OCR) models applied to LP recognition on nine publicly available datasets containing various challenging images captured using different capture settings, image resolutions and license plate layouts.

In a recent work Laroca et al. [54] the authors have explored the bias of the existing license plate datasets. They tested deep learning algorithms on four Chinese and four Brazilian datasets and noticed that the algorithms always performed better on the dataset they were trained on.

TABLE 1. Literature survey matrix.

Author/ Year	Ref.	LP Detection	LP Recog.	Character Recog.	Data set	Avg. Accuracy / F1 Score %	Origin	Execution Time	Citations
Li et al. 2012	[58]	CNN	-	-	Self	93.2	China	-	11
Kim et al. 2017	[47]	FRCNN + CNN	-	-	a*	97.6	USA	-	40
Kurpiel et al. 2017	[50]	CNN	-	-	Self	84.9	Brazil	-	28
Fu et al. 2017	[20]	FRCNN + CNN (VGG16)	-	-	Self	94.3	China	-	8
Peker 2019	[76]	SSD + MobileNet & Resnet50 features	-	-	Self	97.9	Turkey	-	4
Liang et al. 2022	[62]	CNN	-	-	CCPD	96	China	9.5 msec	1
Lee et al. 2022	[57]	YOLO	-	-	AOLP Self	99.6 99	Taiwan Malaysia	- -	0
Gou et al. 2016	[27]	-	-	HDRBM	Self	94.1	China	720 msec	172
Radzi et al. 2011	[81]	-	-	CNN	Self	98.8	Malaysia	-	40
Zang et al. 2015	[110]	-	-	CNN + SVM	Self	98.3	China	140 msec	51
Liu et al. 2017	[66]	-	-	CNN	Self	90.1	China	-	7
Qian et al. 2018	[79]	-	-	CLNN	Self	91.8	China	-	4
Kilic et al. 2018	[46]	-	-	LSTMs	Self	88.7	Turkey	-	6
Spanhel et al. 2017	[92]	-	-	CNN	d*	98.6	Czech Republic	8 msec GTX 1080	56
Li et al. 2016	[59]	CNN	CNN	-	a*, b*	86.4	USA, Taiwan	5, 2.5 sec	163
Wang et al. 2017	[101]	CNN	LSTMs	-	Self - GAN, Carflag	94	China	87 msec	27
Li et al. 2018	[60]	FRCNN	LSTMs	-	a*, b*, c*	96.5	Taiwan, USA, China	310 msec, 400 msec, 310 msec	155
Bjorklund et al. 2017	[8]	CNN	CNN	-	Self - Synthetic / Internet	93	Italy	-	17
Bjorklund et al. 2019	[9]	CNN	CNN	-	b*, c*	98.4	Italy, Taiwan, China	640 × 480 25.5 msec, 1280 × 960 72.7 msec GTX 1080	28
Spanhel et al. 2018	[93]	Hourglass Aligner CNN	CNN	-	d*, g*	84.8	Czech Republic	1.88 msec GTX 1080	1
Peng et al. 2018	[77]	SSD	CNN	-	Self	88.5	China	-	1
Silva et al. 2017	[89]	YOLO	YOLO	-	h*	63.2	Brazil	13 msec Titan and 114.5 msec GTX M750 Nvid	124
Silva et al. 2018	[90]	CNN (WPOD-Net)	YOLO	-	b*, h*, i*, j*	89.3	Taiwan, Brazil, Europe, Various	200 msec Titan X	115
Laroca et al. 2018	[51]	YOLO	YOLO	-	h*, k*	85.9	Brazil, Brazil	21 msec, 28.6 msec	206
Laroca et al. 2019	[52]	YOLO	YOLO	-	a*, English LP, UCSD, ChineseLP, b*, OpenALPR-EU, h*, k*	94.7	USA, Europe, USA, China, Taiwan, Europe, Brazil, Brazil	13.6 msec Nvidia Titan X	28
Goncalves et al. 2018	[23]	CNN	Multi-task Learning	-	h*, k*	72.2	Brazil, Brazil	-	25

TABLE 1. (Continued.) Literature survey matrix.

Hendry et al. 2019	[34]	YOLO (Sliding Window Single Class)	YOLO	-	b*	78	Taiwan	900 msec GTX 970	67
Izidio et al. 2019	[41]	Tinyv3	CNN	-	Self	98	Brazil	4.9 sec RasPI	16
Usmankhujayev et al. 2019	[98]	YOLO v3	LSTMs	-	Self	95.6	Korea	-	5
Duan et al. 2019	[18]	SSD	CNN	-	Self	96.6	China	24 msec	2
Masood et al. 2017	[68]	CNN	CNN	-	a*, i*	94	USA and Europe	-	105
Lu et al. 2018	[67]	CNN + Spatial Transformer Module	CNN	-	ALPR, l*	94.7	China	250 msec	1
Zherzdev et al. 2018	[117]	CNN + Spatial Transformer Module	CNN	-	Self	95	China	3 msec	22
Xu et al. 2018	[107]	CNN	CNN (Rpnet)	-	m*	98.5	China	16 msec Nvidia Quadro P4000	77
Rui. 2019	[85]	CapsuleNN	CapsuleNN	-	Self	89.6	China	-	N/A
Selmi et al. 2020	[87]	RCNN	CNN Swish activation function	-	b*, c*	96.4	Taiwan, China	-	21
Zhuang et al. 2018	[120]	Deep Lab v2 - ResNet 101	AlexNet + Inception v3	-	b*, n*, Self	99.1	Taiwan, Greece, China	40 msec Titan X GPU	22
Rakhshani. 2020	[83]	Encoder-Decoder Pair	Logistic Regression Classifiers	-	ILPD11, b*	96.1	Iran, Taiwan	-	3
Henry. 2020	[35]	YOLO v3	YOLO v3 SPP	-	e*, b*, n*, a*, Uni of Zagreb	98.11	Korea, Taiwan, China, Greece, USA and Croatia	41.6 msec Titan X	23
Zhang. 2020	[112]	Xception CNN	LSTMs	-	b*, c*, f*	90.5	Taiwan, China	-	10
Qin et al. 2020	[80]	CNN + Inception	CNN	-	b*, m*	95.2	China, Taiwan	30 msec RTX 2080 Ti GPU	2
Jin et al. 2020	[44]	YOLO v3 + CNN	CC + NN	-	Self	94.1	N/A	-	2
Pustokhina et al. 2020	[78]	CC + K-means	CNN	-	o*, p*, q*	96.5	N/A	-	67
Vaiyapuri et al. 2021	[100]	Morphological Operators	Hough Transform	CNN	o*, p*, q*	97	N/A	-	4
Wang et al. 2021	[102]	CNN	CNN	-	m*, f*, b*, c*	96.3	China, Taiwan	6.7 msec	8
Zou et al. 2022	[121]	YOLO v3	ILPRNET	-	m*, f*, b*	93.6	China, Taiwan	-	1
Huang et al. 2021	[38]	CNN	CNN	-	r*, b*, c*	96.4	China, Hong Kong, Macao, Taiwan	-	6

CNN = Convolutional Neural Network, FRCNN = Faster Regional CNN  
 RFCNN = Region based fully CNN, CC = Connected Component  
 LSTM = Long Short Term Memory Neural Network  
 SSD = Single Shot Multibox Detector, YOLO = You Only Look Once  
 Nvid = Nvidia, RasPI = Raspberry PI  
 sec = seconds, msec = milliseconds  
 a\* = Caltech = <http://www.vision.caltech.edu/archive.html> (last accessed 1 Oct. 2021)  
 b\* = AOLP = <https://sites.google.com/site/avlabaolp/download-the-database> (last accessed 1 Oct. 2021)  
 c\* = PKU = [https://github.com/ofeeler/LPR/tree/master/pku\\_vehicle\\_dataset](https://github.com/ofeeler/LPR/tree/master/pku_vehicle_dataset)  
 d\* = ReId = <https://medusa.fit.vutbr.cz/traffic/> (Not working)  
 e\* = KarPlate = <http://pr.gachon.ac.kr/ALPR.html>  
 f\* = CLPD = [https://github.com/wangpengnorman/CLPD\\_dataset](https://github.com/wangpengnorman/CLPD_dataset)










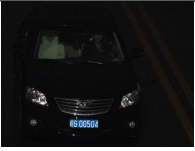

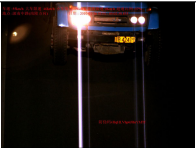





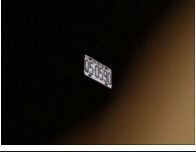

























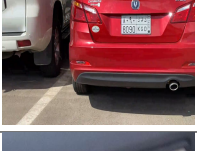
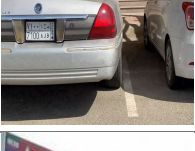
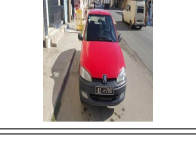




g\* = Carcam6k = <https://medusa.fit.vutbr.cz/traffic/> (Not working)  
 h\* = SSIG = <http://www.ssig.dcc.ufmg.br/ssig-segplate-database/> (Not working) <http://smartsenselab.dcc.ufmg.br/sensedatasets/>  
 i\* = OpenALPR = <http://www.inf.ufbrs.br/~crjung/alpr-datasets>  
 j\* = CDHard = [http://ai.stanford.edu/~jkrause/cars/car\\_data set.html](http://ai.stanford.edu/~jkrause/cars/car_data_set.html)  
 k\* = UFPR-ALPR = <https://web.inf.ufpr.br/vri/databases/ufpr-alpr/>  
 l\* = Rlpd = <https://github.com/ALPRRlpd/Rlpd> (Not-available)  
 m\* = CCPD = <https://github.com/detectRecog/CCPD>  
 n\* = MediaLab = <http://www.medialab.ntua.gr/research/LPRdatabase/>  
 o\* = FZU = Not available.  
 p\* = Stanford = [http://ai.stanford.edu/jkrause/cars/car\\_dataset.html](http://ai.stanford.edu/jkrause/cars/car_dataset.html)  
 q\* = HumAln = [https://github.com/mehulgupta2016154/TCS\\_HUMAIN](https://github.com/mehulgupta2016154/TCS_HUMAIN)  
 r\* = HZM (HongKong-Zhuhai-Macao) multi-style dataset (Not available)

They emphasized the need to have cross-dataset evaluations to better judge the generalization capabilities of a proposed license plate recognition algorithm.

### III. NEURAL NETWORKS FOR LICENSE PLATE DETECTION

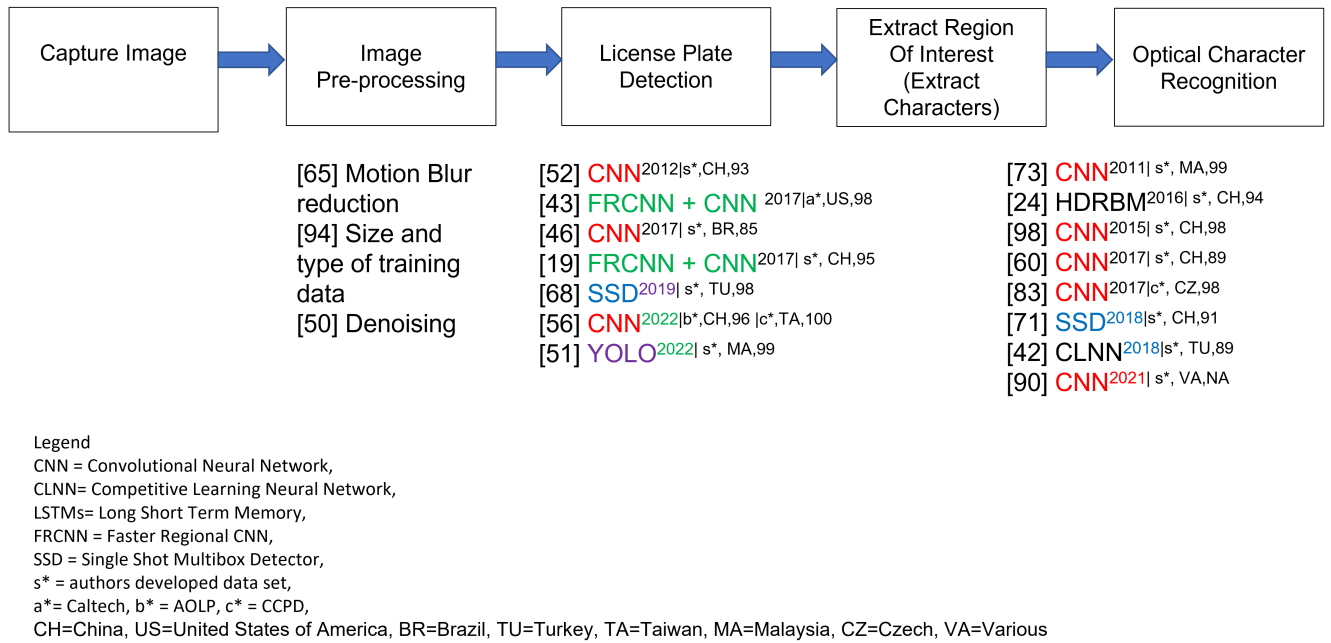
The first step in license plate recognition is detecting a license plate in an image and determining its boundaries. The bound-

TABLE 2. Sample images from data sets.

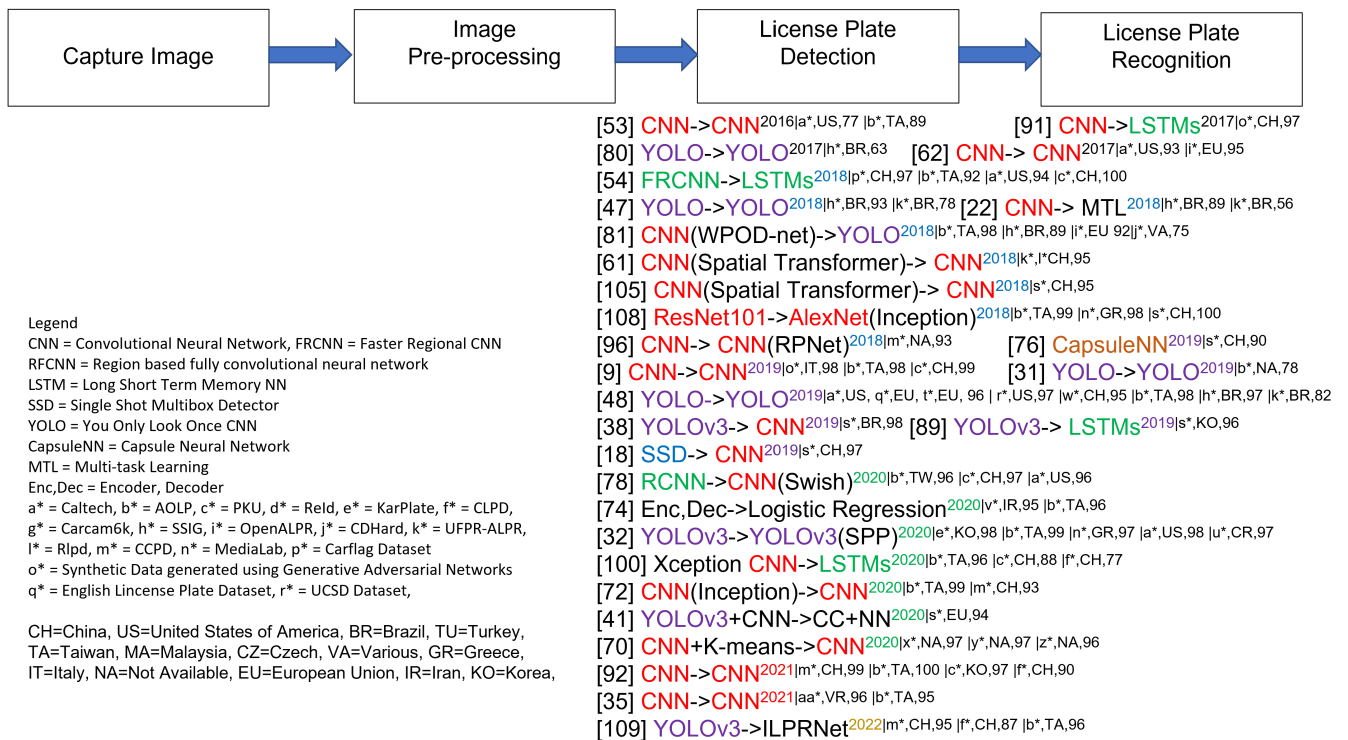
Caltech99,01					
PKU-G1,G2,G3					
PKU-G4,G5					
Synthetic					
Stanford Cars					
CCPD					
LPRD					
UFPR-ALPR					
SaudiLP					
HumAIn and Persian					

aries around the license plate are referred to as the “bounding box”. Thus, the goal is to draw a bounding box around a license plate or all license plates in an image. The success

criteria vary from finding the correct number of license plates to matching the intersection between the annotated region and the region marked by the algorithm.



**FIGURE 1.** General pipeline for license plate detection or character recognition. Methods listed below each block represent papers reviewed in this work. The papers presented here focus only on one block of the pipeline. Therefore, papers related to character recognition or region extraction assume that license plate has already been detected.

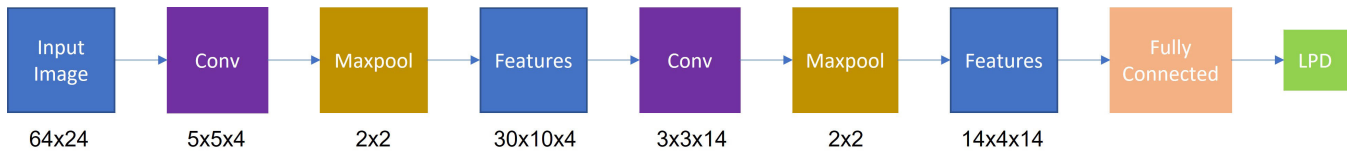


**FIGURE 2.** General pipeline for both license plate detection and recognition. Papers listed in the figure are reviewed in this work. The method mentioned on the left of the arrow is used for license plate detection while that on the right of the arrow is used for license plate recognition.

In 2012, Li et al. proposed using a multi-scale convolutional neural network for license plate detection [58]. Instead of just using the output of a layer in the next layer, they proposed using two different layers in the final classifica-

tion. The idea was to use low-level local features extracted at the earlier layer and the high-level global invariant features extracted in the deeper (later) layer as input to the fully connected classification layer. Their network comprised





**FIGURE 3.** Multi-scale CNN network comprised of an input layer of size  $64 \times 24$  pixels, convolution layer with 4 filters of size  $5 \times 5$  followed by down-sampling by 2, followed by another convolutional layer having 14 filters of size  $3 \times 3$  followed by down-sampling by 2. Thus, resulting in feature/image having dimensions  $14 \times 4 \times 14$ . After passing through the fully connected layer the output indicates if a license plate was detected or not.

two convolutional layers followed by down-sampling, thus, reducing the size of the image. The network was trained by minimizing the sum-of-squares error using the gradient descent algorithm. The authors developed their own data set by annotating 5,613 images and dividing them into 5,488 training and 125 validation images. They applied shear, rotation, flipping, brightness, and contrast variations to images to obtain 65,586 training and 1,500 validation plate samples. Next, they extracted 4,000 non-plate samples and divided them into 2,500 training and 1,500 validation images. They also generated 57,977 false positives, resulting in a total of 60,477 non-plate samples. For testing purposes, the authors collected 1,559 images containing 1,602 license plates. They reported a detection rate of 93.2%. As presented in Table 3, the above-mentioned details are summarized in terms of training and testing images. The data set used by the authors comprises Chinese license plates. However, the authors have neither made the data set publicly available nor reported the time required to detect a license plate. The proposed procedure is presented as a block diagram in Figure 3.

The success of AlexNet in 2012 led to the development of deeper convolutional networks capable of object detection in complex scenarios. This further led to the development of region-based fast convolutional neural networks. Kim et al. proposed the use of a two-step detection procedure [47]. In the first step, they proposed using faster region convolutional neural networks (FRCNN) [84] to detect cars in images. They reported that using the Caltech data set [103], the proposed FRCNN method missed only one car. Next, the region containing the detected vehicle was passed to a graph-based segmentation method that combined similar regions based on color, texture, and size. Regions not containing the ratio-based dimensions of the license plate were removed. Finally, a CNN was applied for the removal of non-plate regions. The CNN comprises a pair of convolution and max-pooling layers followed by two fully connected layers and a Softmax layer. The authors used 175,000 non-plate and 35,000 plate images from the Internet to train this network. Testing on the Caltech database [103], comprising 126 images of license plates from the USA, reported a precision rate of 98.4% and a recall rate of 96.8%.

Instead of passing the complete image to the algorithm, Kurpiel et al. proposed performing image partitioning before passing it to a CNN and then aggregating the results to locate the license plate in the image [50]. Their algorithm assumed that in a sub-region of  $120 \times 180$  pixels, only one license

plate existed and that no sub-region contained more than one license plate. They developed their own database of 1,829 images containing 4,070 Brazilian license plates and reported a precision rate of 87% and a recall rate of 83%.

Fu et al. proposed using cascaded neural networks to detect and extract license plates [20]. The idea of cascaded neural networks is to reduce candidates by combining simple features, thus, creating a shallow network. They proposed a five convolutional layer region proposal network (RPN) at the heart of FRCNN to extract cars from the images. This was followed by a VGG16 network [91] with PRN to detect the license plates in the cropped region from the previous stage. The detected plate is refined by increasing the detected area by 30%. The authors proposed using transfer learning from the ImageNet data set and training the parameters of the new layers using 128 positive and negative samples. The authors generated a Chinese license plate data set of 30,975 images and reported a precision rate of 98.4% and a recall rate of 90.5% on it.

Kim et al. [48] noticed that (in the detection stage) algorithms often mistake grilles or headlights as plates. To overcome these issues, they trained a CNN for detecting these anomalies and reported an improvement in detection accuracy from 78% to 87%. However, the authors did not share the base model that was reported to have achieved 78% detection rate.

To assess the ability of different types of object detectors, Peker tested commonly used object detectors to locate Turkish license plates in images [76]. The author compared faster region convolutional neural network (FRCNN) [84], region-based fully convolutional neural network (RFCNN) [15] and single-shot multi-box detector (SSD) [63]. The training data set comprised 200 images, while the test set contained 100. FRCNN provided the best results for license plate recognition for the tested data sets. One of the variations titled 'MobileNet-SSD' tested in this work was proposed by Peng et al. [77]. The test data set used by the authors is relatively small. A larger data set would have made the results more meaningful. This also highlights the need and importance of publicly available benchmark data sets.

In [62], the authors propose using edge guided sparse attention network (EGSNet) to detect license plates. The EGSNet comprises a VGG19 backbone network, an EGSA module, and a cascaded multi-task detection head. Instead of using all 19 layers, the authors used 12 layers of the VGG19 network with the EGSA module inserted. The advantage of

**TABLE 3.** Summary of methods employing neural networks for license plate detection.

Author/ Year	Ref.	License Plate Detection Approach	Data set image size	Data set Name	Data set link	Origin	Size Training/Testing	Acc. / F1 %
Li et al. 2012	[58]	CNN	352 × 288	Self	N/A	China	5,613 / 1,559	93.2
Kim et al. 2017	[47]	FRCNN + CNN	100 × 30	Self / Caltech99	a*	USA	210,000 / 126	97.6
Kurpiel et al. 2017	[50]	CNN	N/A	Self	N/A	Brazil	1,829	84.95
Fu et al. 2017	[20]	FRCNN+ CNN(VGG16)	2048 × 1536	Self	N/A	China	30,975	94.3
Peker 2019	[76]	SSD	480 × 360	Self	N/A	Turkey	200 / 100	97.9
Peker 2019	[76]	FRCNN	480 × 360	Self	N/A	Turkey	200 / 100	100
Liang et al. 2022	[62]	CNN	720 × 1160	CCPD	b*	China	N/A	96
			N/A	AOLP	c*	Taiwan	N/A	99.6
Lee et al. 2022	[57]	YOLO	N/A	Self	N/A	Malaysia	8000/2000	99

CNN = Convolutional Neural Network, FRCNN = Faster Regional CNN, SSD = Single Shot Multibox Detector, Self = authors developed data set, Acc. = Accuracy, a\* = <http://www.vision.caltech.edu/archive.html>, b\* = AOLP = <https://sites.google.com/site/avlabaolp/download-the-database>, c\* = CCPD = <https://github.com/detectRecog/CCPD> (last accessed 1 October 2021)

EGSA is that it uses edge contours of LP and solves the LP detection problem in real time. To assess the impact of parameter selection on license plate detection, Lee et al. propose to fine-tune the parameters of YOLOv3 network in [57]. They identified and fine-tuned the following parameters a. train test ratio b. number of epochs c. mini-batch size d. L2 regularization e. learning rate, to improve the accuracy of the YOLOv3 network for a Malaysian license plate dataset, from 87.75% to 99%.

The above-presented methods for license plate detection are summarized in Table 3. The table provides details about the authors' names, the year of publication, and the work's reference number in this paper. The generic approach employed by authors is presented with information on whether they developed their own data set for training and testing or used an open-source data set. In some cases, the authors propose more than one network for license plate detection; hence more than one network is mentioned in that case. The last column presents the license plate detection results reported by the authors. Some authors report accuracy, while others report precision and recall results. For consistency, we have combined them into accuracy and F1 scores in the table. The second last column specifies the size of training and testing data if mentioned by the authors; otherwise, the size of the entire data set is provided. The size of images in the data set and the country of origin of license plates are also noted in the table.

#### IV. NEURAL NETWORKS AS OCR FOR LICENSE PLATE RECOGNITION

Character and digit recognition has been extensively researched, and optical character recognition software have been commercially available for years. Neural networks have been employed for character recognition since the early 90s. LeNet-5, the first deep convolutional neural network, was developed for handwriting recognition for postal services. However, there is a difference in general optical character recognition and license plate character recognition because of the difference in size, resolution, orientation (skew due to tilt and pan), and noise in captured data. Efforts have been made since the 90s to use neural networks

for the recognition of characters and digits on license plates.

Among some of the earliest efforts was the method proposed by Nijhuis et al. [73] in 1995, where the authors proposed the use of Discrete-time Cellular Neural Networks (DTCNN's) for feature extraction and an ordinary Multi-layer Perceptron (MLP) network for recognition of Dutch license plates. In 2006, Anagnostopoulos et al. proposed using a fully connected, three-layer neural network for optical character recognition of license plate characters [1]. The authors proposed a segmentation technique that they baptized 'SCW' in which they described local irregularity in the image using statistics based on mean and standard deviation. The speed of fully connected neural networks in the mid-2000s was slow, and the speedup of these computationally intensive algorithms was an active research area. Caner et al. proposed an FPGA-based implementation of license plate recognition incorporating neural networks [11]. They proposed a license plate detection algorithm that applied Gabor filter to grayscale images, followed by connected component labeling, feature extraction, using min/max type operations, and information about the aspect ratio of the plate. To segment the characters, they used a vertical projection based threshold. Finally, they proposed using Self-Organizing Maps (SOM) for character recognition. The input matching is done by comparing the Euclidean distance between it and each output node. Their sample data set contained 1,436 cars passing on a highway. The proposed system, implemented on a Virtex IV FPGA, took 0.5 sec on average to recognize a license plate for vehicles going up to 90 km/h. The authors reported a license plate detection accuracy of 91.7% and stand-alone character recognition accuracy of 90.9%.

The early 2010s saw the revival and evolution of neural networks because of the power and ability of convolutional architecture to solve problems related to feature extraction. Hence, the simple fully-connected three-layer neural networks were replaced by deeper neural networks for most computer vision tasks. This was reflected in the license plate recognition tasks as well. In 2016, Gou et al. utilized a Hybrid Discriminate Restricted Boltzmann Machine (HDRBM) to recognize Chinese characters, a relatively more complex task

than recognizing English characters. The proposed method was also suitable for handling variations in camera angle and zoom to recognize distorted characters [27], both during day and night. The authors proposed using conventional methods, based on the top-hat morphological operator on grayscale images, to identify edges for license plate detection. They trained two separate RBMs for character recognition, one for 31 Chinese characters and another for 34 English characters, i.e., ten digits and 24 alphabetical letters (characters 'O' and 'I' are excluded). For training, they used 12,366 labeled Chinese characters and 10,408 labeled digits and English characters. The HDRBM comprised 150 hidden nodes, and the learning rate  $\lambda$  was set to 0.01. They tested their proposed method using a database presented in [26] and extended it to 4,242 images taken at  $1936 \times 2592$ ,  $1280 \times 736$ , and  $720 \times 280$  pixel resolution. The images were taken using a traffic monitoring camera under varying lighting conditions, from day to night. The authors reported the correct license plate detection rate of 95.9%, the correct character recognition rate of 98.2%, and overall combined correct detection and recognition rate of 94.1%. One downside of the proposed method is the overall execution time, which is more than half a second (sec) per image.

Radzi et al. proposed the use of CNN for license plate recognition [81]. The proposed CNN comprises five layers: a convolution layer, a sub-sampling layer, another convolution layer, and two fully connected layers. The authors used a small data set of Malaysian license plates comprising 750 images for training and 434 for testing. The proposed CNN was trained using a variable learning rate. Since the process does not consider license plate extraction or segmentation, it returns a 98.8% accuracy for the small controlled data set. Similarly, Zhai et al. proposed using neural networks to recognize alphanumeric characters on license plates [113]. However, the presented results were not for images containing license plates but only for alphanumeric characters. Similar to Radzi et al., they also stated that their classifier considered the digit '0' (zero) and the letter 'O' as the same character, as well as the digit '1' (one) and the letter 'I'.

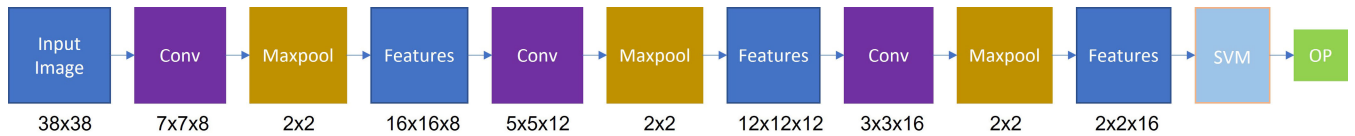
Another proposal puts forward the hybrid use of CNNs, in which a neural network is paired up with another machine-learning tool for character recognition. In this context, Zang et al. proposed using a visual saliency model to detect license plates and integrate CNN based features and Support Vector Machine (SVM) into a single framework to recognize Chinese number plates [110]. The authors used color, intensity, and orientation features to build Gaussian pyramids to generate the saliency map. The saliency map may contain objects other than the license plate; hence, the authors proposed to use information about the aspect ratio and location to refine the estimate. To segment the characters, the authors proposed using horizontal and vertical projections. The input to the CNN was an image of size  $38 \times 38$  pixels, and the network comprised three pairs of convolution and sub-sampling layers, followed by a fully connected layer comprising 64 nodes. The output of these 64 nodes was used

as the input to the SVM classifiers. The authors proposed to use 65 binary SVMs for each character. The authors tested the proposed system using 1,300 license plates captured by them and reported an F1 score of 98.3% for character recognition. The average time required for license plate detection and character recognition was approximately 140 msec. However, whether these results are for a GPU or a CPU-based implementation is not specified. The block diagram of the proposed architecture is reproduced in Figure 4.

Inspired by their earlier work [78], the authors proposed the use of Squirrel Search Algorithm (SSA)-based CNN called SSA-CNN for character recognition [100]. They utilized median filtering followed by edge detection and morphological operators to detect license plates. Next, they proposed using Hough Transform to segment license plates and SSA-based CNN to recognize characters. The authors reported an F1 score of 98% for the FZU Cars data set, an F1 score of 96.9% for the Stanford Cars data set, and an F1 score of 96.1% for the HumAIn data set.

Liu et al. proposed using CNN for Chinese license plates with white digits and characters on a blue background [66]. Chinese license plates have different background colors, depending on the type of vehicle. However, the authors only focused on the detection of one type. They stated that the letters (characters) 'I' and 'O' do not appear on Chinese license plates, thus reducing the number of possible characters from 26 to 24. Also, the first character is a Chinese character representing one of the 31 provinces. Hence, there are  $31+24+10 = 65$  possible characters that may take up seven possible spaces on a license plate. They developed an X-LN, i.e., X layer network, where each layer consists of a convolutional layer followed by a max-pooling layer, and the network concludes with two fully connected layers. The network output has  $65 \times 7$  nodes representing one of the 65 possible characters at each of the 7 locations. However, to simplify training, they trained the network to recognize the first character as one of the 31 Chinese characters. The second character is from the English alphabet and was trained to identify one of the 24 characters. Finally, the last five positions were trained for the 34 alphanumeric characters of the English language. They used an Adam optimizer to minimize the loss function during training. For training, they initially used data from their own province, which was acquired and labeled manually. It comprised 3,220 different number plates, extended to 8,192 by applying geometric transformations. They used 7,396 images for training and 796 for testing and obtained an accuracy of 88.6%. Later, they expanded their training data set by synthetically generating license plates and reported an improved accuracy of 90% for tests on synthetic images. The results reported by the authors were for different depths of the network, i.e.,  $X = 4$  and  $X = 3$ . The authors did not discuss how the license plate was detected, as their work focused only on character recognition.

Some other techniques proposed for character recognition revolved around using different types of neural networks. Qian et al. proposed using competitive neural networks to



**FIGURE 4.** CNN based network comprised of three convolution and maxpooling layers in [110]. The strides and padding were different at each stage. In the end the authors have used an SVM classifier to classify the Chinese characters.

**TABLE 4.** Summary of methods employing neural networks for license plate character recognition.

Author/ Year	Ref.	Char. Recog. (CR) Approach	Data set Image size	Data set Name	Data set link	Origin	Size Training/Testing	Acc. / F1 %	Execution time
Gou et al. 2016	[27]	HDRBM	1936 × 2592	Self	N/A	China	4,242	94.1	720 msec
Radzi et al. 2011	[81]	CNN	N/A	Self	N/A	Malaysia	750/434	98.8	N/A
Zang et al. 2015	[110]	CNN	1920 × 1080	Self	N/A	China	1,950/1,300	98.3	140 msec
Liu et al. 2017	[66]	CNN	128 × 64	Self	N/A	China	7,396/796	88.6	49 msec
Qian et al. 2018	[79]	CLNN	N/A	Self	N/A	China	/115	91.3	N/A
Kilic et al. 2018	[46]	LSTMs	640 × 360	Self	N/A	Turkey	3,754/704	88.8	N/A
Spanhel et al. 2017	[92]	CNN	1920 × 1080	c*	N/A	Czech Republic	105,924/76,142	98.6	0.82 msec GTx1080
Vaiyapuri et al. 2021	[100],	CNN	N/A	o*, p*, q*	N/A	Various	N/A	97	N/A

HDRBM = Hybrid Discriminate Restricted Boltzmann Machine, CNN = Convolutional NN, CLNN = Competitive Learning NN, LSTM = Long Short Term Memory NN, Acc. = Accuracy, c\* = <https://medusa.fit.vutbr.cz/traffic/> (Not working), o\* = FZU = Not available, p\* = Stanford = [http://ai.stanford.edu/jkrause/cars/car\\_dataset.html](http://ai.stanford.edu/jkrause/cars/car_dataset.html), q\* = HumAIn = [https://github.com/mehulgupta2016154/TCS\\_HUMAIN](https://github.com/mehulgupta2016154/TCS_HUMAIN)

recognize license plate text [79]. They assumed that the license plate had already been extracted and segmented. Hence, each character was available for recognition. The authors tested the proposed method on 115 images containing 690 characters and reported an accuracy of 91.3% for character recognition. Kilic et al. proposed using a Long Short Term Memory (LSTM) network to recognize text from license plates captured using a fixed CCTV camera [46]. For a data set of 4,693 Turkish license plates, they reported an accuracy of 88.8% for validation set images.

Spanhel et al. proposed using a cascade classifier using local binary patterns to detect license plates in Full HD quality images, extracted from video [92]. Using a Kalman filtering based approach, they also tracked the license plate in the video. They developed a database titled ‘ReId’, comprising 7,393 training tracks containing 105,924 images and 6,967 testing tracks containing 76,142 images. To recognize the license plate characters without segmentation, they proposed using a CNN with eight branches, where each branch predicted all eight characters of the license plate. Since the authors addressed the issue of the variable number of characters in certain license plates, they replaced them with a ‘#’ character during training so that the system could recognize license plates with variable numbers of characters. They reported that the proposed algorithm took approximately 5 ms for license plate detection and 8 ms for recognition of characters, minus processing overheads using a CPU. For GPU GTX 1080, they reported an overall time of 0.82 msec. They reported an accuracy rate of 98.6% for the ReId data set. However, on another data set collected by the authors, titled ‘HDR’, they reported 90.3% accuracy for license plate recognition.

The above-presented methods for license plate character recognition are summarized in Table 4. The table provides author names, the year of publication, and the

work’s reference number. The generic approach employed by authors is presented along with the information if they developed their own data set for training and testing or if they used an open-source data set. In the case of Spanhel et al., the data set produced by them was made public and is noted in the table. However, the link is not accessible now at the time of this writing. Most authors in the papers presented above report the success rates of character recognition or the complete license plate recognition process. Also, some of them report F1 scores, while others report accuracy. Therefore, the results presented in Table 4 summarize both. In several cases, authors reported the average time required by the proposed algorithm, which is shown in the last column.

## V. NEURAL NETWORKS FOR LICENSE PLATE RECOGNITION

The previous sections covered different types of neural networks used in literature to perform license plate detection, extraction of characters, or recognition of license plate characters. More recent approaches have started exploiting the prowess of the neural networks for both license plate detection and character recognition without having to perform license plate or character extraction specifically. This section presents an overview of techniques that employ single or multiple neural networks for the complete chain of license plate detection and recognition. The material below could have been presented in chronological order or based on the origin of license plate or speed of algorithm execution, e.g., slow or real-time, etc. However, we propose to divide the methods based on the type of architecture proposed by the authors. This choice highlights the community’s interest in a particular network architecture. To that end, we have divided networks into four principal categories; Convolutional Neural Networks, Recurrent Neural Networks, YOLO networks

(which are stripped-down versions of CNNs), and Other architectures.

While going through this section, please note that the figures sometimes represent license plate detection and recognition outputs separately with a common network, as in the case of Figure 5. This does not mean that LPR is independent of LPD, but this simply means that the same network parameters are used for both tasks. In other cases where LPD and LPR have two separate networks, as in the case of Figure 6, the change in input image size for LPR network compared to LPD network indicates that the region of interest extracted from LPD is fed to LPR. The figures are represented in this way to indicate if the same or different features/parameters are being used for LPD and LPR.

### A. CNN-BASED METHODS

In 2016, Li et al. proposed to address the problem of license plate recognition as a sequencing problem [59]. They proposed a three-network approach. The first CNN consists of four layers to identify text in grayscale images and divides them into 37 classes (10 digits, 26 characters, and one non-character). The architecture comprised a sequence of convolution, Rectified Linear Unit (ReLU), max-pooling, convolution, ReLU, max-pooling, fully connected, ReLU, dropout, fully connected, and Softmax layers. The second CNN classifier also has a four-layer classifier for identifying plate and non-plate regions. It comprised convolution, ReLU, max-pooling, convolution, ReLU, max-pooling, fully connected, ReLU, dropout, fully connected, and Softmax layers. The goal is to identify text in the image and determine the plate region overlapping the identified text. The authors used horizontal and vertical projections to determine the license plate's bounding box. A third neural network with nine layers is used to classify text further. The inputs to this deeper network comprise a grayscale image and Local Binary Pattern (LBP) features. Final character recognition combines the prediction results of both CNN classifiers. The authors presented recognition results on the Caltech cars data set [103] and Application Oriented License Plate (AOLP) data set [36]. The Caltech data set comprises 126 images of American license plates at a resolution of  $896 \times 592$ , while the AOLP data set has 2,049 images of Taiwanese license plates divided into 681 Access Control (AC), 757 Law Enforcement (LE) and 611 Road Patrol (RP) samples. The authors reported accuracy of 77.5% for Caltech and 92.3%, 91.3%, and 84.7% for the AOLP, AC, LE, and RP datasets, respectively. These scores are reported in Table 5.

In [8], Bjorklund et al. proposed the use of CNNs for the detection and localization of license plates and another pair of CNNs for identifying the boundaries of characters and their recognition. The authors proposed the use of synthetic images to train these neural networks. The input to the first stage is a colored image and the CNN contains 10 convolution layers and a max-pooling layer after 2<sup>nd</sup>, 4<sup>th</sup>, 7<sup>th</sup> and 10<sup>th</sup> convolutional layers. The output of the 10<sup>th</sup> layer is fed into two sub-networks. The first sub-network consists of three

fully connected layers resulting in a binary classification of images, either having a plate or not. The second sub-network consists of three fully connected layers connected to eight regression units to identify the x- and y-coordinates of the corners of the license plate. Once the license plate is detected, the second stage identifies characters and assigns a confidence score to each character while localizing the bounding box corners. This stage comprises seven convolutional layers with max-pooling after 2<sup>nd</sup>, 4<sup>th</sup> and 7<sup>th</sup> layers. Finally, the character recognition network comprises three fully connected stages with 33 classes, as the Italian license plates have only 32 possible alphanumeric characters. The authors generated synthetic images with positive and negative license plate samples. To train the network for license plates, they used 25,000 positive and 25,000 negative samples, whereas to train the network for character recognition, 175,000 positive and 175,000 negative samples were used. A pyramid of images is provided as input to the license plate detection module to obtain the best results, and the outputs are combined using intersection over union (Jaccard index). The authors then obtained multiple crops of the license plate before feeding them to the character recognition network. If the confidence score of a character is above a certain threshold, it is added to the list of candidate character boxes. The authors reported combined accuracy of around 93% on 1,000 test images downloaded from the Internet.

The authors extended their work in [9], where they discuss the generation of synthetic images in detail. The size of synthetic images for license plate detection was  $768 \times 384$  pixels, and they generated 20,000 positive and negative training images and 2,500 positive and negative validation images. For license plate detection, they modified the earlier network and proposed the use of a single feature extractor network comprising four convolution and max-pooling layers whose output is fed into two networks comprising three fully connected layers, one for localization of license plate and a second containing an additional softmax layer for differentiating between license plate and background. For character detection, a separate network with three convolutional layers and two fully connected layers was proposed. This network accepted input of size  $128 \times 64$  pixels and produced the coordinates of the top-left and bottom-right corners as the output for each character. For the character recognition module's training, the authors extracted 450,000 positive and negative samples. For testing, the authors used 1,152 images from *platesmania.com* and reported an accuracy of 98.3%. They also tested the proposed method on the AOLP Taiwanese data set and reported an F1 score of 97.2% for AC, 98.9% for LE, and 98.4% for RP images. For the Chinese license plate data set 'PKUData', they only reported LPD accuracy of 98.8%, 99%, 98.9%, and 97.7% for the G1, G2, G3, and G4 data sets, respectively. They reported an average recognition time of 25.5 msec and 72.7 msec for images of dimensions  $640 \times 480$  and  $1280 \times 960$ , respectively, on an Nvidia GTX 1080 GPU. They also reported average recognition time of 389 and 845 msec for images of dimensions  $320 \times 240$  and

640 × 480 on Jetson TX1. The advantage of this approach is that the network can be used as a plate or character detector by adapting the hyper-parameters, thus allowing the use of shared training procedures and cost functions. Another feature of the approach is the use of synthetic data for training the network, which reduces the data collection and tagging effort significantly.

Spanhel et al. performed pre-processing and applied a geometric correction to license plates before recognizing them [93]. They proposed to use the stacked hourglass CNN structure proposed in [71] for finding the corners of the license plate. This aligner CNN comprises three hourglass modules for down-sampling and up-sampling features in the spatial domain. The network used skip connections for improved performance of the gradient descent algorithm and provided a probability map of the four corners of the license plate. It was trained using images of arbitrarily rotated license plates, thus enabling the network to correct the geometric orientation of plates. The CNN for recognition was proposed by the authors earlier in [92]. The authors had access to multiple real and synthetic data sets and trained and tested the CNN using different combinations. The accuracy of recognition reported by the authors was 96% on real data. The data set was titled 'CamCar6k' by the authors and comprised 6,064 Czech license plate images divided into 2,750 training and 3,314 test images. The average recognition time of the proposed algorithm was 1.87 msec using an Nvidia GTX 1080 GPU. The authors also reported a correct recognition accuracy of 73.6% for OpenALPR data set.

Goncalves et al. used multi-task networks proposed in [115] for recognition of Brazilian license plates [23]. They used one stage for license plate detection and a second for character recognition. They observed that object detection algorithms required multi-scale features. However, this may not be necessary for license plate detection, so they only used the features from the 11<sup>th</sup> layer for detection instead of multiple layers. They also observed that a higher Intersection over Union (IoU) threshold may not necessarily encompass all characters of license plates and devised a new loss function for penalizing regression while finding the corners. For training, they used translation, vertical flipping, brightness, and contrast changes to augment the data set. They used high-resolution 1920 × 1080 images, allowing them to zoom in and out to develop a better training set. For recognition of the seven characters, they proposed to use a multi-task network and directly recognize characters instead of segmenting them first, an approach similar to [92]. The authors used SSIG and UFPR-ALPR data sets comprising 6,660 images containing 8,683 license plates from 815 vehicles with a split of 3,595 training, 705 validation, and 2,360 test images. They reported a correct detection rate of 79.3% and, assuming 100% detection, a recognition rate of 85.6%. They also reported actual results for the complete recognition process, i.e., detection and recognition, for SSIG and UFPR-ALPR data sets, at 88.8% and 55.6%, respectively. However, they

did not report the speed of recognition, although they did mention that the algorithm can process six license plates in an image in real time.

Masood et al. proposed the sequential use of three CNNs to detect license plates, segmentation of characters, and recognition of characters. However, they did not provide architectural details of the networks but made their solution available online [68]. They used the Caltech and OpenALPR data sets, containing 328 US (adding images to the Caltech99 data set) and 550 European license plates, and reported an accuracy of 93.4% and 94.5%, respectively. Lu et al. proposed using four CNNs for license plate detection, angle correction, spatial transformation, and recognition [67]. The first part comprised a neural network with ten convolutional layers, two fully connected layers, a bounding box (bbox) regression layer, and an orientation classification layer (using 0, 90, 180, and 270 degrees classes). The feature maps from 2<sup>nd</sup>, 4<sup>th</sup>, 6<sup>th</sup> convolutional layers, bbox regression, and rotation prediction layers are used for rotation correction. This is further refined by passing the result to the spatial transformation network proposed in [42]. Finally, the authors used the network proposed in [92] for character recognition. The Rotated license plate data (Rlpd) database of Chinese license plates consisting of 400,000 samples (380,000 used for training and 10,000 each for validation and testing) was used. The authors reported a recognition rate of 94.7%. The main advantage of the technique is its ability to handle rotation. However, the lack of testing on more frequently used data sets makes it difficult to compare the proposed method with existing state-of-the-art techniques.

Another approach focusing on using a spatial transformation network was proposed in [117]. The authors proposed an efficient implementation of deep neural networks that can recognize a license plate in an image in 3 msec. The proposed network consists of two parts, one for detection and another for recognition of the license plate. The authors proposed the use of a spatial transformation network proposed in [42]. The output of this network was forwarded to a CNN inspired by SqueezeNet along with inception blocks [39], [96]. They reported accuracy of 95% on a private data set containing 11,696 images of Chinese number plates. The authors claim that the proposed approach can perform license plate recognition in real time on various platforms, including FPGAs. Therefore, one of the advantages of the proposed approach is that it can be deployed on standalone hardware systems.

Xu et al. developed a comprehensive database titled 'Chinese City Parking Data set' (CCPD) that contained 250,000 real images with annotated license plate corners, bounding boxes, and license plate numbers [107]. Having a comprehensive data set of real images captured and annotated by inspectors, the authors proposed an efficient algorithm titled 'Roadside Parking net' (RPnet) capable of processing 60 frames per sec on NVIDIA Quadro P4000 GPU, i.e., approximately 16.7 msec per image. The authors also developed a 'CCPD-Characters' data set that contained at least

1,000 samples of each possible character on Chinese license plates. Their proposed RPnet comprised two modules. The first is the detection module containing ten convolutional layers for extracting features and feeding three fully connected layers for bounding box prediction. The second, the recognition module, uses pooling layers [84] for extracting features of interest along with classifier layers for character recognition. The two modules work as a single network, where the detection module is used as ‘attention’ [14] for the recognition module. To train the network, the authors set the license plate detection loss to smooth L1 loss function between the predicted and ground-truth bounding box, while the classification loss was set to cross entropy loss. The authors reported a detection accuracy of 92.7% and recognition accuracy of 85.1% on their proposed data set at 61 frames per sec. The work presents an excellent comparison of variation in images in available data sets to justify the development of their own data set. Their proposed technique has the advantage of detecting and recognizing images of varying dimensions because of their region of interest approach. One of Xu’s collaborators, Meng et al., proposed another method based on locating, cutting, and recognizing license plates and characters in [69]. To detect the license plate, they proposed using a network baptized ‘LocateNet’ comprising ten convolution layers, two max-pooling layers, and three fully connected layers to predict the four vertices of license plates. The detected license plate was spatially transformed to have a rectangular size of  $256 \times 72$  pixels. The horizontal and vertical projections of these images were passed through a four-layer fully connected network, titled ‘CutNet’, for character segmentation. These segmented characters were passed to AlexNet [49] for recognition. The authors tested the proposed method on CCPD data set images and reported an accuracy of 96.2%.

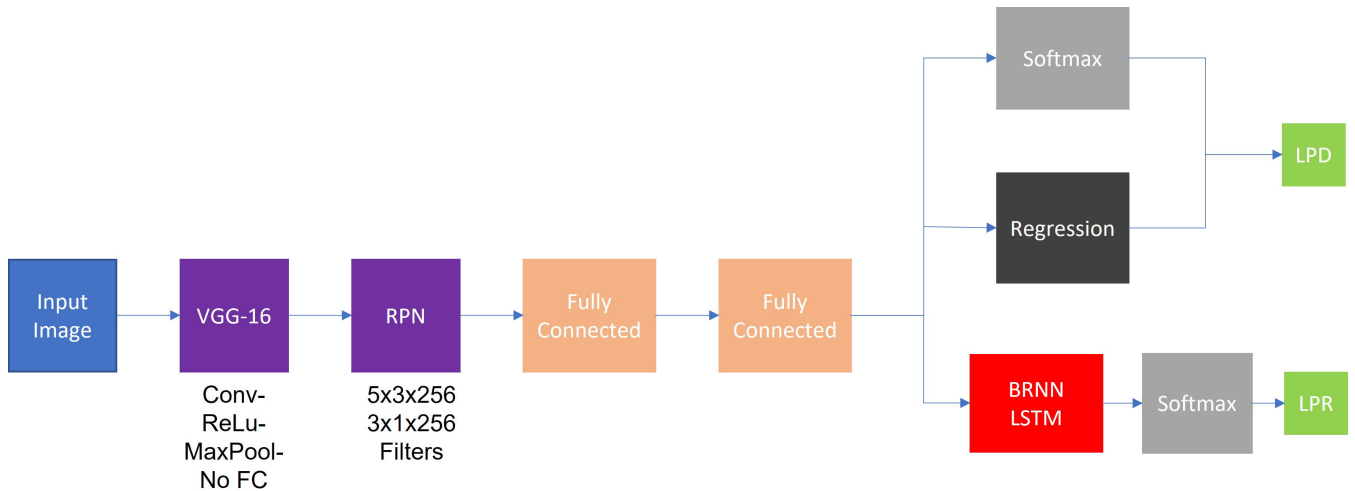
## B. RNN-BASED METHODS

Building on their prior work [59], in which they used three convolutional neural networks, Li et al. [60] proposed a solution comprising a convolutional neural network for detection and a recurrent neural network for recognition of license plates. They proposed to use a VGG-16 network proposed in [91]. The network is pre-trained on ImageNet to extract features from the input images but the authors retained only the 2<sup>nd</sup> and the 5<sup>th</sup> pooling layers. They also removed the fully connected stages of the VGG-16 network to obtain the desired features. Next, they proposed to use Region Proposal Network (RPN) by Ren et al. [84], also known as FRCNN, to obtain bounding boxes around objects in images. This was done by modifying the algorithm to apply two convolutional filters across each sliding position to get a 512-dimensional feature vector that can be fed to the two fully connected RPNs. To avoid character segmentation, the authors tackled the character recognition problem as a sequence labeling problem. They used bidirectional recursive neural networks (BRNN) and Long Short Term Memory (LSTMs) with

Connectionist Temporal Classification (CTC) loss. They tested the proposed algorithm on four data sets: ‘CarFlag-Large,’ comprising 460,000 images of Chinese license plates (322,000 training / 138,000 testing), ‘AOLP’ [36] comprising 2,049 images of Taiwanese license plates, ‘Caltech99’ comprising 126 images of American license plates that were augmented by adding 1,626 new images, and ‘PKUData’ comprising 3,977 images of Chinese license plates. They reported a recognition accuracy of 95.6% for AOLP-AC, 96.4% for AOLP-LE, 83.8% for AOLP-RP while taking 400 msec per image, 94.1% for Caltech99 data set, 99.73% for PKUData while taking 310 msec per image, and 97.1% for CarFlag-Large data set while taking 310 msec per image. Further details related to these results can be observed in Table 5, and the block diagram of their proposed algorithm is presented in Figure 5. The main advantage of the proposed approach is that no segmentation is required.

The authors in [98] proposed using two deep neural networks to detect and recognize Korean license plates. They used 8,075 images to train the YOLOv3 network to identify the position of number plate in an image, both during the day and night time. For night images, they modified the histogram so that the number plate becomes visible in the training data. To recognize the license plate, they proposed using Long Short Term Memory (LSTM) architecture. Instead of directly using the images for recognition, the authors applied geometric transformation. This was done to ensure that the characters provided to LSTM are not distorted and, therefore, result in a correct label sequence. This was specifically done for Korean license plates with two rows of characters. They trained the license plate detection algorithm on 8,075 images and the character recognition LSTM network on 110,000 images. They tested the algorithm with 1,425 images and obtained an accuracy of 95.6% for the five different types of Korean license plates. In another RNN based approach, authors proposed using CNN and gated recurrent unit (GRU, similar to LSTMs) based neural networks for license plate recognition [109]. The authors used CNN to extract features and GRU to encode and decode these features. On their own data set, the authors claimed to have achieved 99% recognition accuracy with an average execution time of 200 msec per image.

In another recent work [87], Selmi et al. proposed the use of a recurrent neural network titled Mask-RCNN [30] to detect and recognize license plates. For feature extraction, the authors used a modified version of GoogleLeNet by reducing its number of inception modules to six and adding several pooling layers. They also used the Swish activation function instead of ReLU. To generate the proposal regions, they used Mask-RCNN using 12 anchor boxes. The features are then fed to two convolutional layers for license plate detection and bounding box regression. They used non-negative maximum suppression to avoid duplicate regions of interest. To remove false positives and false negatives, the authors used the RoI-Align layer of Mask-RCNN. After using fully connected layers for the detection and estimation of the



**FIGURE 5.** Input image is passed to a convolutional neural network (VGG-16) followed by Region Proposal Network (RPN) to identify multiple regions of interest. These features are passed through two fully connected layers to extract features for License plate detection and recognition. Softmax layer on top provide probability of each region being plate/non-plate and Regression helps find a bounded box while the BRNN layer followed by Softmax proposes the probability of recognition of number plate letters and characters.

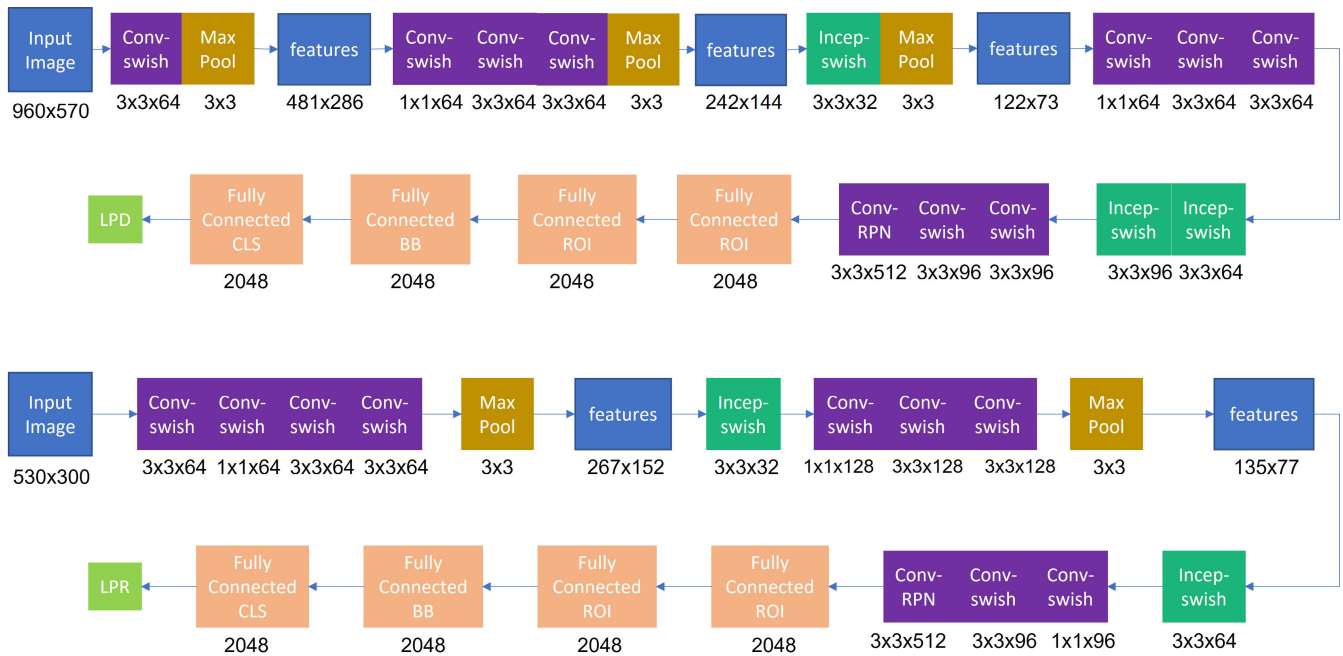
bounding box, they performed the segmentation of characters for recognition. The authors used Mask-RCNN again, but this time for recognizing characters in the cropped region of the license plate. The authors tested the proposed algorithm on the AOLP dataset and reported an F1 score of 97.1% for AOLP-AC, 96.6% for AOLP-LE, and 95.2% for AOLP-RP. On Caltech data set, they reported an F1 score of 95.9%. For the PKU data set, the reported scores are 97.9% for G1, 97.4% for G2, 97.2 for G3, 96.9% for G4, and 96% for G5. The proposed architecture is presented in Figure 6.

In [112], the authors propose to extract features using a 30-layer Xception network. Two types of features are extracted: intermediate features from the 12<sup>th</sup> layer of the Xception network and global features from the 30<sup>th</sup> layer. The global features are used as input to the LSTM network at time step 0, while intermediate features are used as input to a 2D attention model. Together they are used to identify letters on the number plate. The advantage of using a two-layer LSTM network with an attention mechanism is that there is no need to correct irregular license plate images or segment characters. Another contribution of the work is to use AsymCycleGAN to generate synthetic license plates to train the LPR algorithm. The authors used light and dark images from the CCPD data set and 800 synthetic and 800 real images for training the AsymCycleGAN algorithm. The Chinese License Plate Dataset (CLPD) developed by the authors comprises 1,200 images containing LPs of the size of  $149 \times 48$  pixels. The images are obtained from different vehicles, i.e., trucks, buses, and cars, and while some license plates have seven letters, others have eight. Empirically, the authors identified that 512 Channel CNN, along with 30 Xception layers, provides the best results. On the AOLP dataset, the authors reported an accuracy of 97.3%, 98.3%, and 91.9% for AC, LE, and RP images. For PKU and CLPD, they reported accuracy of 88.2% and 76.8%, respectively, while using three-fifths of data for training and rest for testing.

### C. YOLO-BASED METHODS

Most of the methods discussed above used at least two and sometimes three different neural networks for detection, extraction, and recognition. In most methods, the authors used both synthetic and real images for training purposes. Generally, the priority concern in the above-presented techniques was not execution speed but detection accuracy. Some authors have focused on developing algorithms that target efficient processing, i.e., they attempted to detect and recognize the license plates quickly. In this regard, Silva et al. proposed using You Only Look Once (YOLO) networks for the detection and recognition of license plates [89]. They noticed that YOLO could not detect small license plates in SSIG data set images that had a resolution of  $1920 \times 1080$  pixels [22]. Therefore, they proposed to crop the images to a smaller size so that it contains only the front of the vehicle for training the YOLO network. The authors modified the 15<sup>th</sup> layer of the network by reducing the number of filters and setting the goal to differentiate only between two classes, i.e., plate and no-plate, to speed up the detection process. The authors used transfer learning, followed by further training of the network on full-resolution cropped images. For the second stage, they used a smaller rectangular input. This stage comprises the first 11 layers from the YOLO network with four added convolutional layers and was trained to get a  $30 \times 10$  pixel output instead of  $13 \times 13$  pixels. Finally, the authors employed the heuristics that Brazilian license plates only have seven characters, out of which the last four are digits. For the test data set comprising 2,000 images, the reported accuracy of license plate recognition was 63.2% at a processing speed of 13 msec per image on Titan X GPU. The block diagram of the proposed architecture is presented in Figure 7. Due to the extraction requirement of the front plate and the heuristic stated above, the method is only specific to Brazilian license plates. In a relatively similar approach, Arsenovic et al. proposed using the YOLO algorithm to





**FIGURE 6.** The convolutional layers proposed in [87] used Swish activation function instead of ReLu. The pipeline for detection of license plate is shown at the top and for character recognition at the bottom.

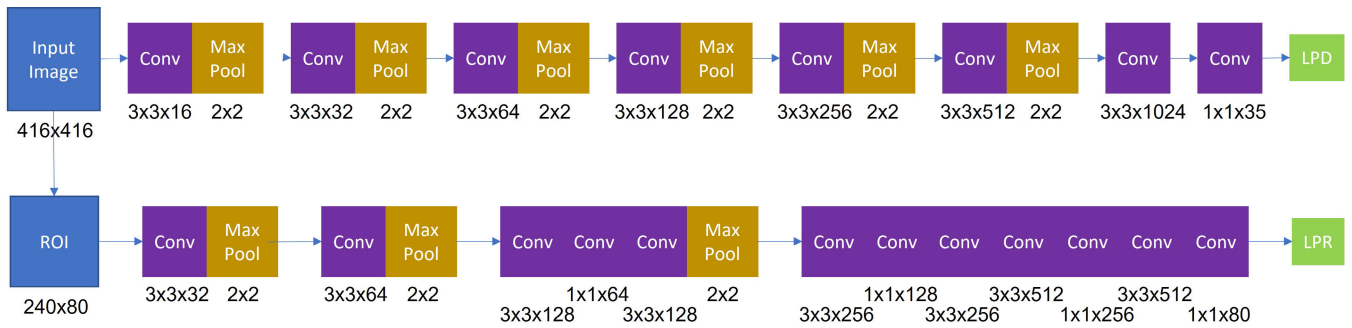
detect license plates and a CNN for recognizing digits and characters on Serbian license plates in [5]. They proposed using the YOLO algorithm for better speed because it can theoretically process 45 images per sec. For training, they generated Serbian license plates using backgrounds from the SUN [106] and Stanford [32] background databases. Their reported character recognition rate is 97%, and the digit recognition rate is 94.5%.

Laroca et al. proposed using different architectures for license plate detection, character extraction, and recognition in [51]. They tested both YOLOv2 and Fast-YOLO models for license plate detection. They noticed that they had to change the number of filters depending on whether the data set comprised only cars or both cars and motorcycles, i.e., one or two classes. The number of filters was set to 30 or 35 for one class or two classes, respectively. Following license plate detection, they proposed using CR-NET [89] in two sequential stages for character segmentation and recognition. The input to the character segmentation stage was a patch of size  $240 \times 80$  pixels. They fed both original and negative images of each license plate during training. To obtain better results, the authors modified the depth of networks for character and digit recognition. They conducted experiments with SSIG [22] and Federal University of Parana-Automatic License Plate Recognition (UFPR-ALPR) data sets [51], divided into 40% training, 20% validation, and 40% testing samples. The SSIG data set contains 2,000 frames from 101 videos, on which the proposed system achieved an accuracy of 93.5% at 47 frames per sec, i.e., 21.2 msec per image. For the UFPR-ALPR data set that comprises 4,500 frames from 150 videos with camera and vehicle moving, the method obtained an accuracy of

78.3% at 35 frames per sec, i.e., 28.6 msec per image. The proposed method may struggle to detect small license plates, as the YOLO networks generally do not detect small-sized objects very well.

In another study, the authors used a modified version of the Fast YOLO network for character recognition. Extending their previous work, they proposed YOLOv2 for vehicle detection, Fast-YOLO2 for license plate detection and layout classification, and CR-NET for recognition [52]. They modified the layers of Fast-YOLO2 to improve its performance and tested the proposed method using Caltech99 [103], English LP [94], UCSD-stills [16], Chinese LP [118], AOLP [36], Open ALPR (EU) [74], SSIG [22] and UFPR-ALPR [51] data sets and reported average processing time of 13.6 msec per image. They reported accuracy of 96.1%, 95.5%, 97.3%, 95.4%, 98.4%, 95.7%, 96.9%, and 82.5%, on these datasets, respectively. This comprehensive testing covers license plates from the United States, Europe, China, Taiwan, and Brazil. The work focused on using different YOLO algorithms and their modification to achieve faster recognition speed. In [7], a comparative analysis of different algorithms for synthetic data sets was presented.

Hendry et al. proposed using a YOLO-based neural network architecture for the detection and recognition of license plates [33], [34]. Instead of using a two-stage approach, they employed 36 tiny YOLO networks to detect ten digits, 25 characters ('o' and '0' are considered the same in Taiwanese license plates), and one network for detecting the license plate. The novelty of the method was in a sliding window based approach and a modified YOLO network with ten convolution layers, three residual layers, one average



**FIGURE 7.** Upper row comprises of convolutional and maxpooling layers transferred from YOLO-VOC network in [89]. The lower rows also comprises of convolutional and maxpooling layers, with a slightly different arrangement.

pooling layer, one fully connected layer, and one softmax layer. Since the objects were detected using a sliding window based approach, multiple objects may be detected at the same location, and objects that are not part of the license plate may be detected. Therefore, localization is essential. Since the proposed method is for Taiwanese license plates, the authors took advantage of the fact that there are six objects on these plates. The authors used the AC images of AOLP data set [36] for training and tested the system on all of the images, including AC, LE, and RP subsets. They reported a recognition speed of approximately 800 msec to 1 sec per image on an Nvidia GTX 970 GPU and an accuracy of 78%. The authors also reported a license plate detection accuracy of 98.2%. The block diagram of the proposed architecture is presented in Figure 8. The single-pass approach for detecting and recognizing individual digits requires a sliding window to be moved over the image. This results in slowing the algorithm and thus not providing real-time results. Secondly, having a separate network detecting each character or digit may result in a large memory requirement.

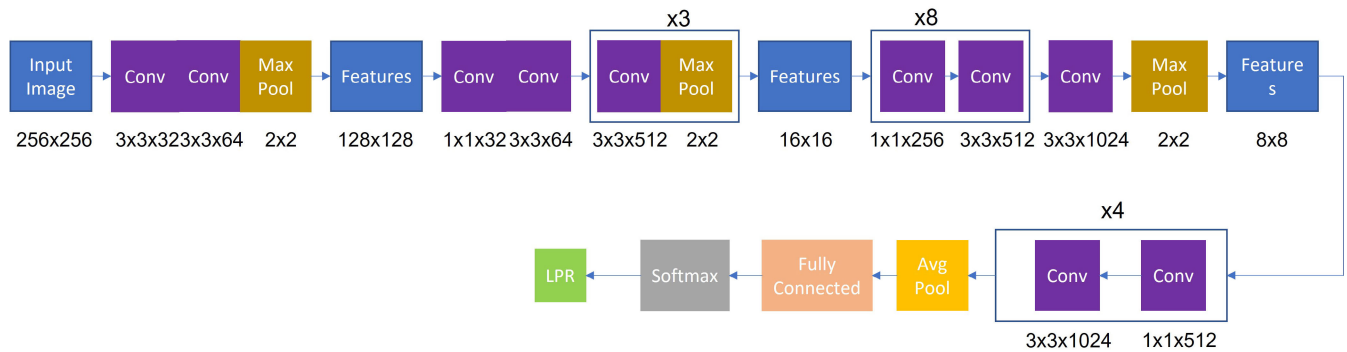
In other reported results, the authors in [97] proposed to use YOLOv2 for license plate detection and recognition and reported a processing rate of 5 frames per sec and an accuracy of 97%. In another approach, Izidio et al. proposed to use a pair of networks [41] comprising the YOLOv3 network for detecting license plates and a second neural network to recognize digits and characters. They implemented their system on a low-power Raspberry Pi3 platform and reported an overall recognition accuracy of 98.4% on images of size  $1024 \times 768$  while taking 4.9 sec on average. They passed the training images through affine transformations to augment the training data set. The advantage of the proposed approach is that it has low memory and computational power requirements, even if it is relatively slow due to implementation on less capable hardware.

In [35], the authors addressed the issue of generalization, i.e., recognition of license plates from multiple countries. They proposed using Tiny YOLOv3 for license plate detection and YOLOv3-SPP (Spatial Pyramid Pooling) to recognize license plate characters. The character recognition algorithm returns the bounding boxes of characters without their sequence; therefore, a layout detection algorithm had

to be used after character recognition to extract the correct sequence of numbers for license plates of different countries. The benefit of the proposed system is that YOLOv3 has a better mean average precision than the YOLOv2 algorithm. Also, the ability of YOLOv3 to make predictions at multiple scales helps it detect smaller objects. Contrary to the popular character segmentation and recognition method, the authors treated character recognition as an object detection problem. This was achieved by a spatial pooling pyramid that splits a feature map into  $B_i$  bins at the  $i$ -th layer. Feature maps are then pooled using max-pooling into the same size and thus producing  $N \times B_i$  vector. YOLOv3 is better at detecting smaller objects than YOLOv2 and is comparable to other CNN-based techniques; however, there is still a trade-off between accuracy and speed. It is generally recommended to significantly increase the training data size for the YOLOv3 network to reduce this trade-off.

The focus of authors in [35] was to build a robust Network for LPR and was trained on images of varying sizes for KarPlate, AOLP, Caltech, MediaLab and University of Zagreb. There were a total of 71 character classes, i.e., 35 for Korean, 10 for numerals and 26 for English.

For training and testing, the authors developed their own data set titled KarPlate, comprising 4,000 full High Definition (HD) images of Korean cars. The proposed algorithm was tested for license plates from South Korea, Croatia, the United States, Greece, and Taiwan. From the KarPlate data set, 3147 images were used for training and 850 for testing. Out of the 2049 images of AOLP, 681 + 757 images of AC and LE subsets were used for training, and 611 images of RP subset for testing. For MediaLab data, 259 images augmented by 501 images of the University of Zagreb, 108 images of OpenALPR Europe, and 1,428 images of ReId were used for training, and 431 images of MediaLab data set were used for testing. For training, 80 out of 244 images of the Caltech Car data set augmented by 244 images of the OpenALPR US data set, 108 images of the OpenALPR Europe data set, and 501 images of the University of Zagreb data set were used. The accuracy of the proposed system remained 99.41% for AOLP-AC, 97.88% for AOLP-LE, 99.51% for AOLP-RP, 96.98% for MediaLab, 97.83% for Caltech Cars, 97% for Uof Zagreb, and 98.17% for KarPlate. The processing times were



**FIGURE 8.** The authors propose sliding window single class detection in [34]. To account for increase in computations because of sliding window method they have reduced the number of layers in the architecture.

29.19 msec for AOLP, 47.02 msec for MediaLab, 34.3 msec for Caltech Cars, 32.18 msec for Uof Zagreb, and 65.12 msec for KarPlate, resulting in an average of 41.56 msec for license plate recognition.

In [44], the authors propose a super-resolution convolutional neural network for license plate detection in fog-haze environments. They hypothesize that the captured foggy image is usually brighter due to added atmospheric light than the fog-free image. They developed a joint fog-haze removal model to address this issue using a convolutional neural network. They employed YOLOv3 for feature extraction and WPOD-NET to correct the position of bounding boxes. Detected license plates may suffer from blurring or noise pollution, affecting character recognition. Therefore, they proposed to use SRCNN to obtain high-resolution images. The authors next used a combination of connected components and template matching using neural networks for character recognition. The authors collected a data set of 3000 images and added noise by rotating them and other image processing techniques to get 12000 images. They reported accuracy of 94.12% using a 7-layer network.

In [102], the authors proposed using two deep neural networks, baptized VertexNet and SCR-Net, for LPD and character recognition, respectively. The advantage of using VertexNet is that it assists in rectifying license plates and hence improves license plate detection and recognition accuracy for images captured at different angles. In VertexNet, the authors proposed to use a single detector on  $256 \times 256$  size images. The network consists of a backbone, fusion, and head networks. In the backbone, the residual neural network block is used along with the Squeeze-and-Excitation attention module to aggregate the feature maps across spatial dimensions. These features are fused by the feature pyramid network, and the data is fed to the head network for the prediction of four vertices. The SCR-Net utilizes images of dimension  $64 \times 256$  to predict the characters. The authors tested the proposed system on CCPD, AOLP, PKU, and CLPD data sets and reported an accuracy of 89.8% and 96.8% on CLPD and PKU data sets. For the AOLP data set, they reported accuracy of 99.4%, 99.9%, and 99.7% for AC, LE, and RP data sets. For the CCPD data set, they reported accuracy of 99.9%,

99.7%, 99.4%, 99.9%, 99.9%, 99.4%, 94.8% for the Base, DB, FN, Rotate, Tilt, Weather, and Challenge sub-cases, while reporting an average processing time of 6.7 msec.

In [80], authors tackled the problem of detecting and recognizing license plates containing one or two lines of text. They proposed using a lightweight convolutional neural network to extract license plate features and simultaneously perform classification (one or two lines plate) and character recognition while treating the recognition as a sequence labeling problem. The first part of the lightweight network is the backbone responsible for extracting features that are passed to the classification head, which determines if the license plate has a single or two lines. In the case of two lines, it determines the slicing parameter. The slicing parameter can be fixed if the height ratio between two lines of characters is constant. Finally, the recognition head using both features and output of the classifier head, recognizes the license plate characters. The feature extraction network is inspired by LPRNet with added inception layers for better feature extraction. License plate recognition rates for CCPD base examples were reported to be 99.1%, while they remained 87.6% for the CCPD challenge examples and 98.84% for the AOLP-RP data set.

In a recent work, Zou et al. [121] proposed using the YOLOv3 network for license plate detection and extraction. They employed an Improved License Plate Recognition Net (ILPRNET) for character localization and recognition. They only discuss the ILPRNET while using the stock YOLOv3 algorithm for license plate detection. They proposed to use a U-Net for the localization of license plate characters which incorporates a spatial attention mechanism. They reported accuracy of 96.3%, 97.9%, and 95% for the AC, LE, and RP sub-sets of the AOLP data set. For the CCPD data set, they reported accuracy of 99.2%, 98.1%, 98.5%, 90.3%, 95.2%, 97.8%, and 86.2% for the Base, DB, FN, Rotate, Tilt, Weather, and Challenge sub-sets, respectively.

Huang et al. proposed using two fully convolutional one-stage (FCOS) object detectors for simultaneously classifying license plates and characters, followed by an assembly module in [38]. They employed Res-Net50 as the backbone network for the extraction of features. Features having a

TABLE 5. Methods employing neural networks for LPR.

Author/ Year	Ref.	LP Detect.	LP Recog.	Data set	Origin	Size Train/Test	Acc. / F1 %	Avg. Exe. Time
Li et al. 2016	[59]	CNN	CNN	a* b*	USA Taiwan	/126 / 681, 757, 611	77.45 92.3, 91.4, 84.7	5sec 2.5 sec
Wang et al. 2017	[101]	CNN	LSTMs	o* o*	China China	203,774/9,986 45,139/5,925	97.6 96.2	N/A N/A
Li et al. 2018	[60]	FRCNN	LSTMs	p* b* a* c*	China Taiwan USA China	322,000/138,000 1368/681, 1,292/757, 1438/611 1,626/126 322,000/810 , 700, 743, 572, 1152	97.1 95.6, 96.4, 83.8 94.1 99.9, 99.9, 99.6, 100, 99.3	- 400 msec - 310 msec
Bjorklund et al. 2017	[8]	CNN	CNN	o*	Italy	40,000/1,000	93	-
Bjorklund et al. 2019	[9]	CNN	CNN	o* b* c*	Italy Taiwan China	40,000/5,000	98.3 96.9, 98.5, 98 98.8, 99, 98.9, 97.7	640 × 480 25.5 msec 1280 × 960 72.7 msec GTX 1080
Silva et al. 2017	[89]	YOLO	YOLO	h*	Brazil	2,000	63.2	13 msec Titan GPU
Silva et al. 2018	[90]	CNN (WPOD-Net)	YOLO	b* h* i* j*	Taiwan Brazil Europe Various	/611 /804 /212 /102	98.4 88.6 92.4 75	200 msec Titan X
Laroca et al. 2018	[51]	YOLO	YOLO	h* k*	Brazil Brazil	2,000/ 4500/	93.5 78.3	21 msec 28.6 msec
Laroca et al. 2019	[52]	YOLO	YOLO	a* q* r* w* b* t* h* k*	USA Europe USA China Taiwan Europe Brazil Brazil	126 509 291 411 2,049 /108 2,000 4,500	96.1 95.5 97.3 95.4 98.4 95.7 96.9 82.5	13.9 msec
Goncalves et al. 2018	[23]	CNN	Multi-task Learning	h* k*	Brazil Brazil		88.8 55.6	
Hendry et al. 2019	[34]	YOLO (Sliding Window SCD)	YOLO	b*	-	681 / 681,757,611	78	900 msec GTX 970
Izidio et al. 2019	[41]	Tinyv3 YOLO	CNN	s*	Brazil	858/332	98.4	2.7 sec RasPI
Usmankhujaev et al. 2019	[98]	YOLO v3	LSTMs	s*	Korea	110,000 /1,425	95.6	-
Duan et al. 2019	[18]	SSD	CNN	s*	China	175,00/6,688	96.6	24 msec
Masood et al. 2017	[68]	CNN	CNN	a* i*	USA Europe	/328 /550	93.4 94.5	- -

TABLE 5. (Continued.) Methods employing neural networks for LPR.

Lu2018	[67]	CNN Spatial Transformer	CNN	k*, l*	China	38,000/10,000	94.7	
Zherzdev et al. 2018	[117]	CNN Spatial Transformer	CNN	s*	China	11,696	95	3 msec
Xu18	[107]	CNN	CNN (Rpnet)	m*		250,000	(Average of all 8 image sets)92.7	16 msec Nvidia Quadro P4000
Rui2019	[85]		CapsuleNN	s*	China	160,000/3,000	89.6	-
Selmi et al. 2020	[87]	RCNN	CNN Swish activation function	b*	Taiwan		97.1, 96.6, 95.1	
				c* a*	China USA	3977	97.11 95.9	
Zhuang et al. 2018	[120]	Deep Lab v2	AlexNet	b*	Taiwan	/681, 757, 611	99.4, 99.3, 99	
		ResNet 101	+Inception v3	n* s*	Greece China	/706 4,057/1,000	97.89 99.7	4msec Titan X
Rakhshani et al. 2020	[83]	Encoder-Decoder	Logistic Regression	v*	Iran	7,000/4,000	95.3	0.123 sec
				b*	Taiwan	/2,049	96.1	
Henry et al. 2020	[35]	YOLO v3	YOLO v3 SPP	e*	Korea	6,834/1,179	98.17	42 msec Titan X
				b* n* a* u*	Taiwan Greece USA Croatia	1,366/683 /431 80/46 401/100	99.4, 97.9, 99.5 97 97.8 97	
Zhang et al. 2020	[112]	Xception CNN	LSTMs	b*	Taiwan	1,366/683	97.3, 98.3, 91.9	
				c* f* b*	China China Taiwan	1,352/901 1,200	88.2 76.8 98.8	30 msec RTX 2080Ti GPU
Qin et al. 2020	[80]	CNN + Inception	CNN	m* s*	China Europe	3000/	93.3 94.1	
Jin et al. 2020	[44]	YOLO v3 + CNN	CC + NN	x*			97.2	
Pustokhina et al. 2020	[78]	CC + K-means	CNN	y* z* m*			96.6 95.6 99	6.7 msec GTX 1080Ti
Wang et al. 2021	[102]	CNN	CNN	b* c* f*	Taiwan Korea China		99.4, 99.9, 99.7 96.8 89.8	
Zou et al. 2022	[121]	YOLO v3	ILPRNET	m* f* b*	China China Taiwan		95 86.6 96.3, 97.9, 95	
Huang et al. 2021	[38]	CNN	CNN	aa*  b*	China, Hong Kong, Macao Taiwan	1376/176	96.4  94.6	

CNN = Convolutional Neural Network  
 RFCNN = Region based Fully Convolutional Neural Network  
 LSTM = Long Short Term Memory NN  
 SSD = Single Shot Multibox Detector  
 YOLO = You Only Look Once CNN  
 RasPI = Raspberry PI, Acc. = Accuracy,  
 sec = seconds, msec = milliseconds

**TABLE 5. (Continued.) Methods employing neural networks for LPR.**

a\* = Caltech = <http://www.vision.caltech.edu/archive.html> (last accessed 1 Oct. 2021)  
b\* = AOLP = <https://sites.google.com/site/avlabalp/download-the-database> (last accessed 1 Oct. 2021)  
c\* = PKU = [https://github.com/ofeeler/LPR/tree/master/pku\\_vehicle\\_dataset](https://github.com/ofeeler/LPR/tree/master/pku_vehicle_dataset) (last accessed 1 Oct. 2021)  
d\* = ReId = <https://medusa.fit.vutbr.cz/traffic/> (Not Available)  
e\* = KarPlate = <http://pr.gachon.ac.kr/ALPR.html> (last accessed 1 Oct. 2021)  
f\* = CLPD = [https://github.com/wangpengnorman/CLPD\\_dataset](https://github.com/wangpengnorman/CLPD_dataset) (Not Available)  
g\* = Carcam6k = <https://medusa.fit.vutbr.cz/traffic/> (Not Available)  
h\* = SSIG = <http://www.ssig.dcc.ufmg.br/ssig-segplate-database/> (Not Available) <http://smartsenselab.dcc.ufmg.br/sensedatasets/> (last-accessed 1 Oct. 2021)  
i\* = OpenALPR = <http://www.inf.ufbrs.br/~crjung/alpr-datasets> (last accessed 1 Oct. 2021)  
j\* = CDHard = [http://ai.stanford.edu/~jkrause/cars/car\\_data\\_set.html](http://ai.stanford.edu/~jkrause/cars/car_data_set.html) (last accessed 1 Oct. 2021)  
k\* = UFPR-ALPR = <https://web.inf.ufpr.br/vri/databases/ufpr-alpr/> (last accessed 1 Oct. 2021)  
l\* = Rlpd = <https://github.com/ALPRRlpd/Rlpd> (Not Available)  
m\* = CCPD = <https://github.com/detectRecog/CCPD> (last accessed 1 Oct. 2021)  
n\* = MediaLab = <http://www.medialab.ntua.gr/research/LPRdatabase/> (last accessed 1 Oct. 2021)  
o\* = Synthetic Data generated using Generative Adversarial Networks  
p\* = Carflag Dataset (Not Available)  
q\* = English License Plate Dataset (Not Available)  
r\* = UCSD Dataset = [http://vision.ucsd.edu/belongie-grp/research/carRec/car\\_data.html](http://vision.ucsd.edu/belongie-grp/research/carRec/car_data.html) (last accessed 1 Oct. 2021)  
s\* = Dataset developed by authors  
t\* = <https://platercognizer.com/number-plate-datasets/> (last accessed 1 Oct. 2021)  
u\* = University of Zagreb Dataset (Not Available), v\* = ILPD11 (Not Available), w\* = Chinese License Plate Dataset (Not Available)  
x\* = FZU = Not available.  
y\* = Stanford = [http://ai.stanford.edu/~jkrause/cars/car\\_dataset.html](http://ai.stanford.edu/~jkrause/cars/car_dataset.html) (last accessed 1 Oct. 2021)  
z\* = HumAIn = [https://github.com/mehulgupta2016154/TCS\\_HUMAIN](https://github.com/mehulgupta2016154/TCS_HUMAIN) (last accessed 1 Oct. 2021)  
aa\* = HZM = Hong Kong-Zhuhai-Macao multi-style dataset (Not available)

resolution 1/8 with respect to (wrt.) input image were used for LP detection, while features having a resolution 1/4 wrt. to input image were used for character recognition. The advantage of using FCOS is that they removed intermediate steps like ROI proposals and segmentation and also reduced the problem of positive-negative anchor boxes imbalance. The license plate detection branch contains four sub-branches for bounding box and orientation regression, centerness regression, and classification. Similarly, the character recognition branch also has four sub-branches of bounding box regression, centerness, and character classification. The classification subbranch predicts 73 channels map (26 English, ten numeric, 36 Chinese). For the HZM data set comprising license plates from China, Hong Kong, and Macao, they reported a license plate recognition accuracy of 98.2%. They reported LPR accuracy of 95.8%, 96.6%, and 91.6% for the AC, LE, and RP subsets of the AOLP data set.

#### D. OTHER ARCHITECTURES

Wang et al. proposed using Generative Adversarial Networks (GAN) to generate synthetic training data to train their proposed networks [101] to circumvent the requirement of a large training data set for training deep networks. They proposed the use of synthetic RGB images as input to an eight-layer CNN with batch normalization in 3<sup>rd</sup>, 5<sup>th</sup>, 7<sup>th</sup> and 8<sup>th</sup> layers. Feature vectors extracted from the CNN were fed to the LSTM based RNN, which contained a fully connected layer of 68 neurons at the output (31 Chinese characters, 26 English characters, ten digits, and one non-character class). The two data sets used by the authors comprised 203,774 training plus 9,986 testing plates in data set 1 and 45,139 plus 5,925 training and testing plates in data set 2. The authors defined recognition accuracy as the number of correctly recognized license plates with respect to the given number of license plates and reported it as 98.6%, and 96.2% for data

sets 1 and 2, respectively, when trained with both real and synthetic images. The authors also collected a third data set in Suzhou, China, containing 22026 license plates. They reported accuracy of 89.4% on this data set. Since the authors have used both CNN and RNN type networks, we have added this method to other networks category.

Zhuang et al. proposed using a convolutional neural network for semantic segmentation to extract license plate characters and then used them for counting-based refinement and recognition of characters of license plates [120]. The advantage of their proposed technique is that the complete image is processed at once instead of using a sliding window based approach. The proposed semantic map based approach classifies each input image pixel into a particular class. For pre-processing, the authors clipped the sides of the license plate by taking horizontal and vertical projections of the characters. For semantic segmentation, they employed a DeeplabV2 ResNet-101 model and, to train the network, they generated 60,000 synthetic license plate images. The ground truth had a bounding box drawn around characters, and the authors did not use pixel-level segmented characters [12]. To remove any inaccuracies in case of overlap, the authors checked for overlap between the area of the current character and previous characters using Intersection over Union (IoU). To address the issue of two characters appearing as one because of close proximity, they proposed counting characters for such regions since the number of characters on a license plate is generally fixed. They assumed character counting as a classification problem and employed AlexNet. In the final step, they utilized an Inception-v3 [96] model for character recognition. They tested the proposed method on the AOLP data set [36], Media Lab data set [2], and their own data set titled Chinese License Plate Data set (CLPD), and accuracy rates of 99.4%, 99.3%, 99%, 97.9%, and 99.7%, respectively.

Peng et al. proposed using a Mobile-Net Single Shot MultiBox Detector (SSD) neural network for detecting license plates, followed by a CNN for recognition of license plates [77]. SSD has the advantage of discretizing the space in an image into a series of boxes and providing scores to each object in the box to get better fitting around the objects. The advantage of SSD is that it is faster than Faster R-CNN and YOLO. The authors used the SSD network with 47 convolutional layers. Next, they refined the box by applying Sobel operator based filtering and using the histogram and horizontal and vertical sums. The plate's top and bottom edges were obtained using Random Sample Consensus (RANSAC) algorithm. It should be noted that although SSD may be faster than R-CNN and YOLO, RANSAC is a relatively computationally intensive algorithm. However, the authors did not compare the results between these algorithms in terms of speed and accuracy. Next, the authors proposed rectifying the license plate's geometry by making the top and bottom line segments parallel to improve recognition rates. To recognize the characters, the authors separated the characters of the license plate by using vertical projections. The segmented characters were passed for recognition to a CNN, containing three convolutional layers, three inner product layers, two max-pooling layers, and one flattened layer. The input to the CNN was of size  $30 \times 14 \times 3$ . To test their network, the authors generated geometrically distorted images by vertically rotating the car image from  $60^\circ$  to  $90^\circ$  and distorting it horizontally. For a data set of 5,000 images, the authors reported accuracy of 92% for standard and 85% for distorted images and an average accuracy of 89%, including a third data set. Although the method employed a neural network for license plate detection, the authors employed conventional techniques to refine the bounding box. Since the authors propose using both CNN and SSD networks, we have categorized their work into a hybrid category and presented it in this section.

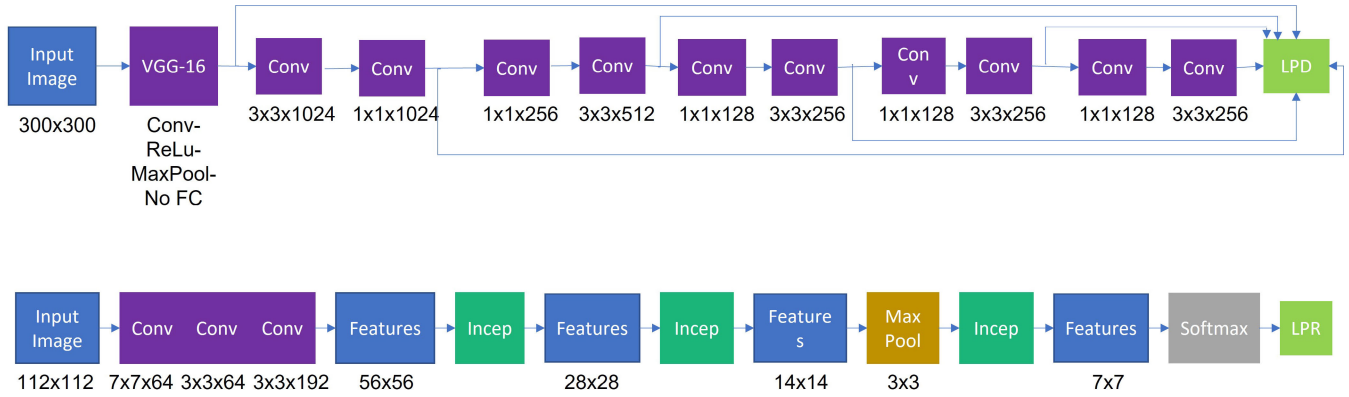
Building on their previous work [89], Silva et al. proposed using the YOLOv2 network to detect vehicles in their three-step license plate recognition system [90]. The output was scaled to account for the ratio between the license plate and car sizes before it was fed to their newly proposed Warped Planar Object Detection Network (WPOD-NET) for license plate detection and finally to an optical character recognizer (OCR). The proposed WPOD-NET combined the strengths of Spatial Transformation Networks [67], i.e., the ability to detect non-rectangular regions and multiple object detection capability of SSD [63]. WPOD-NET's eight-channel feature map contains object and non-object probabilities and affine transformation parameters. The proposed network had 21 convolutional layers, of which 14 were in residual blocks and two prior to softmax and regression layers. They used the residual blocks presented in [29]. To detect and transform the object, their loss function comprised of i) affine transformation loss and ii) probability of object / non-object loss. For recognition, they used the YOLO network proposed

in their previous work [89]. The authors tested the proposed algorithm on AOLP-RP data set containing 611 images, SSIG containing 804 images extracted by authors, OpenALPR (EU) containing 104 images, OpenALPR (Brazil) containing 108 images, and a new data set titled 'CD-HARD' containing 102 images. They reported a correct recognition rate of 93.5% for OpenALPR (EU), 91.2% for OpenALPR (Brazil), 88.6% for SSIG, 98.4% for AOLP-PR, and 75% for CD-HARD. The average recognition time reported by them is 200 msec on a Titan X GPU.

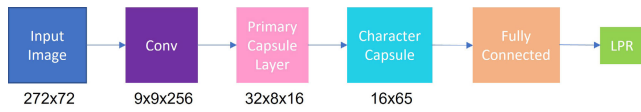
Duan et al. [18] proposed a real-time license plate recognition system without performing character extraction. The proposed systems used color, texture, depth, and morphological features along with an SSD [63] for license plate detection. The VGG16-Atrous network based SSD was compared by authors to YOLO and Faster RCNN, and they noticed that the performance of SSD made it a better option. They concluded this because the results produced by SSD were similar in quality to FRCNN while being computationally comparable to YOLO [18]. To improve the performance of SSD, the authors suggested using randomly clipped and flipped versions of the images along with the original. For license plate recognition, the authors proposed using GoogleLeNet and sped up its non-tensor layers, for real-time performance, by using DeepRebirth [61] based model. The authors collected 17,500 real license-plate images comprising five plate colors, 34 Chinese characters, 24 English characters, and ten digits. The authors used 360 images to test the results of the detection algorithm and 6,688 samples to test recognition and reported a correct recognition rate of 96.6% while taking approximately 24 msec to process one image. The structure of the proposed system is presented in Figure 9.

In [85], the author proposed using Capsule networks to recognize 31 Chinese characters, 24 English language characters (minus 'O' and 'I'), and ten digits. It is assumed that the input to the proposed neural network will be a license plate of size  $272 \times 272$  pixels. The network comprises five layers: input layer, convolution layer, capsule layer, character capsule layer, and fully connected layer, as shown in Figure 10. The character capsule layer consists of 65 character capsules, i.e., one for each character to be recognized. The author generated 160,000 training images, collected 3,000 real images for testing, and reported an accuracy of 89.6%.

In [83], the authors proposed using an encoder-decoder pair to extract features of license plates and eight parallel classifiers to recognize the number plate characters. They collected a database of 11,000 license plates and achieved an accuracy of 96% on 4,000 test images. The input to the encoder-decoder network pair is a grayscale image, while the target is a binary image. The segmented characters are passed to eight parallel classifiers trained to recognize the characters. This is done by extracting features from the encoder and passing them to the classifier. One classifier identifies the ten numerals, while the other classifies the 19 characters. Images captured for the data set have  $1280 \times 960$  resolution and are



**FIGURE 9.** DNN architecture proposed in [18] passes the input image to a Single Shot Detector (SSD) for license plate detection. SSD comprises of a VGG-16 network followed by multiple convolution layers with each layer contributing features for license plate detection. The detected region is passed to a layer of convolution and inception layers followed by a softmax layer for recognition of characters.



**FIGURE 10.** Capsule network based architecture, proposed in [85], with input layer of  $272 \times 72$  pixels, followed by Convolution layer containing 256 filters of size  $9 \times 9$ , followed by 32 capsule containing 8 filters providing input to 16 dimensional Character capsule layer which outputs a 65-dimensional vector.

captured from a height of 15m during day, night, and twilight. The authors used 1,000 images from the ILPD11 data set to train the encoder for license plate binarization. On average, a license plate took 123 msec to be recognized. The character recognition part was trained using 7,000 images from the ILPD11 data set, and 4,000 images were used for testing. The character recognition accuracy of the proposed method is 95.3%. For the AOLP dataset, the proposed method results in an average accuracy of 96.1%. The proposed architecture is shown in Figure 11.

In another approach, Pustokhina et al. [78] proposed to use the Improved Bernsen Algorithm (IBA) and connected components (CCA) for localization and detection of license plates. Next, they proposed using optimal K-means clustering with Krill Herd algorithm for license plate segmentation followed by Convolutional Neural Network (CNN) for recognition of characters. They tested the proposed method on FZU, Stanford, and HumAIn data sets. On the FZU Cars data set, the reported precision was 97.3%, recall 97.9%, and F1 score 97.2%. These values remained at 96.5%, 97.0%, and 96.6% on the Stanford Cars data set and 95.4%, 96.3%, and 95.6% on the HumAIn Cars data set.

The above-presented methods for license plate recognition are summarized in Table 5.

**VI. NEURAL NETWORKS FOR ADDRESSING ISSUES RELATED TO LICENSE PLATE RECOGNITION**

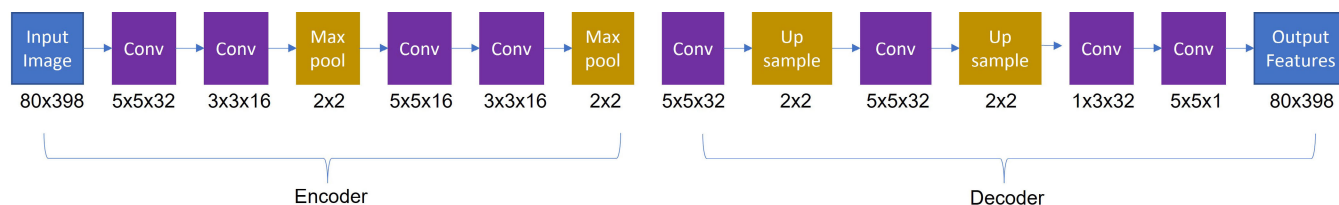
While solving a problem, insights lead to some interesting questions. In the case of license plate recognition, one

of those questions may be: “how does motion blur affect the process of license plate recognition?”. Other interesting questions would be: “how many license plates are necessary to train a deep learning network?”, and “will synthetic images produce the same results as training with real images?”. Some researchers discussed these topics in their works; some insights are summarized below in this section.

One of the main issues with the training of license plate recognition algorithms is data availability. Data sets are generally divided into real and synthetic types. Real data was traditionally considered better for training neural networks; however, it is painstakingly slow to collect and label images. On the other hand, synthetic images are easy to produce, variability can be easily managed, and their labeling is relatively easy. However, generally, they are not very good representations of real-world data. To address this issue, Wang et al. proposed using Generative Adversarial Networks (GAN) to generate synthetic training data to train their proposed networks [101]. In this regard, the authors generated license plate images and then used GAN to transform these synthetic images into real images. They proposed to use CycleGAN network [119] with Wasserstein distance loss [4]. The generator network comprised two convolutional layers with stride two, six residual blocks, and two fractionally strided convolutions with stride  $\frac{1}{2}$ . The discriminator network comprised  $70 \times 70$  PatchGAN network, which classifies a patch as fake or real. To avoid getting the same result every time, the goal was to generate the real image from the synthetic image but also to have a cycle consistency, i.e., when the output of the generator for a synthetic image is passed to the discriminator, it produced the synthetic image.

In [72], authors treated the problem of motion blur due to the movement of camera or vehicles in license plate recognition. They tried to estimate the deblurred license plate image using a Generative Adversarial Network (GAN) without estimating the blur kernel. The goal of the generator was to produce a sharp image. The CNN generator comprised a  $3 \times 3$  convolutional layer followed by ten dense blocks, each consisting of a leaky Rectified Linear Unit (LReLU),





**FIGURE 11.** Encoder Decoder pair architecture proposed by Rakhshani et al. [83] for binarization of license plate images to separate the characters from the background. They propose to use regression based classifiers for recognition of license plate characters.

$1 \times 1$  convolution, batch normalization, LReLU,  $3 \times 3$  convolution layer, and batch normalization. In the second part of the network, a discriminator attempts to identify whether an image passed to it is the generator's output or a real image. Thus, the generator tries to deceive the discriminator while the discriminator tries not to be fooled. The discriminator proposed by Nguyen et al. comprised four  $3 \times 3$  convolutional layers, batch normalization, and LReLU. This is the same architecture as the PatchGAN classifier. The authors used a data set of 500 images having resolution  $128 \times 128$  pixels and a second data set of 80 images of size  $128 \times 192$  pixels. The authors did not report the results, as the algorithm only produced a binarized deblurred image. The idea presented in [72] was inspired by the work of Svoboda et al., who proposed using CNN to deblur license plate images [95]. Svoboda et al. proposed a 15-layer convolutional network comprising convolutional and ReLU layers. The network was trained using actual patches and artificially blurred patches. Their database comprised 140,000 images divided into 14,000 test images and 126,000 training images. The process does not detect license plates or segment them but only focuses on character recognition and yields an accuracy of 91%.

In [104], Wu et al. tried to answer the question about the type and size of the data set required to train a deep neural network for license plate recognition. They generated synthetic data and augmented it using various data augmentation techniques to train and test a modified form of DenseNet [37]. Inspired by the CycleWGAN proposed in [101], the authors proposed to use an improved version of the CycleWGAN network, titled WGAN-GP [28], which improved training stability. The license plates generated by the proposed network were cropped and flipped horizontally and vertically for data augmentation. The proposed DenseNet comprised a convolution layer, followed by two pairs of dense block and transition layers, followed by another dense layer. The last dense layer is connected to a fully connected layer consisting of 68 neurons for the class labels (31 Chinese, 26 English, ten digits, one blank). The network was trained using Adam optimizer and stochastic gradient descent. To train the GAN, the authors used 1,000 fake license plates as the source and 100 real plates from the data set used in [101] as the target. They employed the CyclesWGAN-GP network to obtain 80,000 license plates and then used dilation and erosion, motion blurring, uneven lighting, stretching, affine transformation, downsampling, and addition of Gaussian noise to generate

an augmented data set. The 200,000 images obtained are transformed to grayscale and inverted to obtain a total of 400,000 images. The authors reported that when the proposed DenseNet, was trained with 300 real license plate images along with generated and augmented images, the results were similar to a baseline method trained with 200,000 real images. The authors reported 97.5% detection and 99.3% character recognition accuracy. The authors also highlighted that data generation using CycleWGAN-GP produced an increase of 1% accuracy in character recognition and a further 5% increase in accuracy because of data augmentation. On the AOLP data set, the authors reported recognition accuracy of 96.6%, 97.8%, and 91% for AC, LE, and RP images of the data set.

In a recent work, Lee et al. proposed denoising and rectification networks to improve the quality of license plate recognition [56]. The proposed algorithm is comprised of prediction networks for denoising and rectification. They proposed auxiliary prediction networks for count classification and segment prediction, and recognition networks for text detection and classification. For denoising, the authors proposed to use U-Net-based architecture [40] with the addition of skipped connections to share low-level information across network layers. The U-Net-based encoder and decoder pair produced an output that is fed to the rectification network, which is also comprised of an encoder-decoder pair. The output from this network would be denoised and rectified. The authors further proposed using binary segmentation and count estimation using the features extracted by the denoising and rectifier encoders. The authors used the segment decoder based on U-Net architecture to produce a license plate segment with values indicating the probability of pixels belonging to the license plate. Similarly, the sum of features from the last layer of denoising and rectifier encoders is fed to the counting decoder, which predicts the number of characters in the image. The authors tested the proposed method on AOLP data set [36] and Korean data set comprising 10,650 images, divided into 6,400/4,250 training/testing images titled 'VTLP'. Authors reported an accuracy of 99.2% on AOLP-RP and 93.1% on VTLP.

Another interesting topic associated with license plate recognition is vehicle re-identification, which may require more than just license plate recognition. However, it is still an important constituent of the process. In 2016, Liu et al. discussed the issue of re-identification of a vehicle at

multiple places [64]. They proposed the use of CNN for coarse identification of a vehicle and Siamese Neural Networks for matching plates in two images captured at different locations. The goal was not to recognize the characters of the license plates but to match the license plates across various samples. The proposed algorithm is comprised of a CNN-based coarse appearance matching. Next, the search is refined by using a Siamese Neural Network for matching license plates. Finally, spatio-temporal properties are used to re-rank the similarity between vehicles for possible identification. The Siamese Neural Networks used by the authors comprise two CNNs with two convolution and max-pooling layers followed by three fully connected layers. The authors also developed a comprehensive data set titled ‘VeRi-776’ with 50,000 images, with a high recurrence of cars, i.e., 776 vehicles, from 20 surveillance cameras. The test data set was separated from the original data set and comprised 11,579 images containing 200 vehicles, thus leaving 37,781 images of 576 vehicles for training and 1,678 images for query purposes of Siamese Neural Network. The authors concluded that the proposed method could be combined with another method titled ‘FACT’ [65] (combination of AlexNet and BOW-CN [116]), comprising of fusion of attributes and color features, to obtain the best results of 61%. It should be noted that the proposed work focuses on re-identification and not recognition. This idea was further extended by Shen et al. in [88], where they proposed to develop a more comprehensive path proposal using LSTM. They also tested their method on VeRi-776 database and reported matching accuracy of 83.5%, compared to 61% reported by Liu et al. [64].

Since license plate recognition is a real-world application it has to perform in real-time to be applicable in most situations related to access control and policing. Therefore, a lot of the focus is on developing real-time systems. In this regard, Duan et al. [18] proposed a real-time license plate recognition system without performing character extraction. The proposed systems used color, texture, depth, and morphological features along with an SSD [63] for license plate detection. The VGG16-Atrous network based SSD was compared by authors to YOLO and Faster RCNN, and they noticed that the performance of SSD made it a better option. They concluded this because the results produced by SSD were similar in quality to FRCNN while being computationally comparable to YOLO [18]. To improve the performance of SSD, the authors suggested using randomly clipped and flipped versions of the images along with the original. For license plate recognition, the authors proposed using GoogleLeNet and sped up its non-tensor layers, for real-time performance, by using DeepRebirth [61] based model.

Recently, Padmasiri et al. [75] presented a hardware-efficient ALPR system for detection and recognition tasks to be performed power and memory constrained devices. The authors found a number of efficient neural networks using Neural Architecture Search (NAS) strategies to carry out the operation in real time. Their approach is based on Facebook-Berkeley-Nets (FB-NET) [105] and Partially Con-

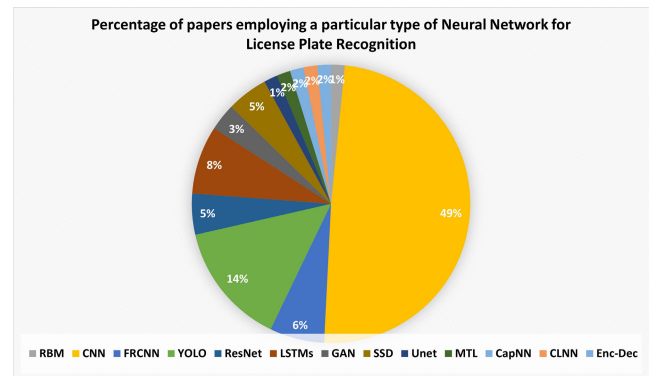


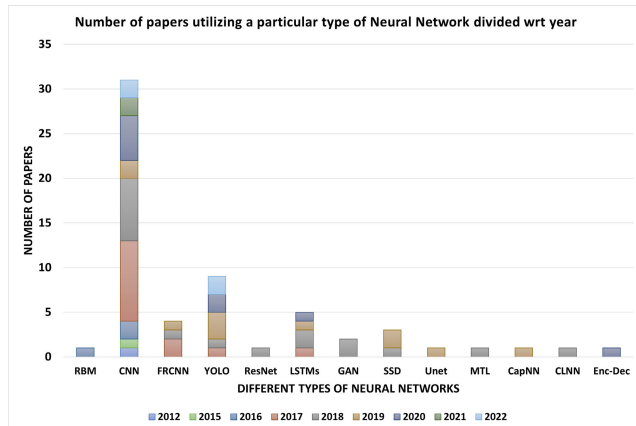
FIGURE 12. Relative percentage of type of neural network employed for license plate related tasks reviewed in this paper.

ected Differentiable architecture search (PC-DARTS) [108]. These algorithms sample a small part of a super-network to reduce the redundancy in exploring the network space, thereby achieving higher computational efficiency compared to reinforcement learning and evolutionary algorithms. The authors also converted the daytime RGB images to thermal images and used the synthesized images for training to achieve a better LPR accuracy during night time. Based on the reported results, the system performance is at par with the Roadside Parking Net (RPNet) which is an advanced system designed to work on server-grade hardware.

## VII. ANALYSIS AND DISCUSSION

The preceding sections discussed different deep neural networks employed for license plate detection and recognition. In this section, we present our observations related to these methods. First, we would like to highlight that Convolutional Neural Networks have been the most influential type of deep learning networks in the domain of computer vision. Figure 12 shows this is also true for the problem of license plate recognition, as approximately 41% of the papers reviewed in this work utilize them for license plate detection, extraction, or recognition. It should be noted that this statistic would have been higher if FRCNN and YOLO networks were not placed in further sub-categories. Another interesting observation is that classic neural networks were used from 1995 to 2008 but never after 2008. On the other hand, YOLO, LSTMs, and GAN have been explored more recently between 2017 and 2019. These observations can be verified by observing Figure 13.

Two main factors involved in any discussion on license plate recognition revolve around the method used and the data set on which the method was tested. Therefore, data sets are essential for identifying the strengths of the proposed methods. Researchers have developed and used various data sets for quantitative quality assessment of license plate recognition algorithms. Therefore, it is in the research community’s interest to identify which License Plate Recognition (LPR) algorithm produces the best result on these data sets. In Table 6, we have presented a summary of F1 and accuracy



**FIGURE 13.** Number of papers per neural network type and the year of publication.

scores as reported by authors in their work. The first column presents a non-exhaustive list of commonly used data sets. The papers that have used these data sets for training and testing purposes are presented in the second column. From the table, it is clear that Caltech, AOLP, Open ALPR, SSIG, and PKU data sets have been used by researchers quite often to test the efficacy of their LPR algorithm. It should be noted that the researchers either use the F1 score or accuracy as the quantitative quality measure; hence, they are reported separately. As an example, for the Caltech data set, the best F1 score, 95.9%, was reported by Selmi et al. in [87] while the best accuracy of 96.1% was reported by Laroca et al. in [52].

As a second example, the results for Application Oriented License-Plate (AOLP) data set have been presented separately for each subset of “Access Control (AC)”, “Law Enforcement (LE)” and “Patrolling (RP)”. Each subset presents different challenges, and all researchers do not provide results for all the subsets. For all three subsets of the AOLP data set, the best F1 score was reported by [9], while the best accuracy score for all three subsets was reported by [35].

It should be highlighted here that the results presented in Table 6 are presented assuming that the authors have used the complete images in the data set for testing. One of the challenges of result comparison lies in the fact that the size of training, validation, and test data is not the same. Also, many methods are tested on proprietary data sets, i.e., data sets developed by the authors themselves. Such data sets are not available to other researchers. To summarize, it is difficult to directly compare the effectiveness of proposed algorithms quantitatively because:

- Same data sets are not available to all researchers.
- There is no agreed-upon convention on how to split the data sets into training and testing sets.
- For a direct comparison, authors require a reference implementation of the existing algorithm, which may not be available and, therefore, it must be developed by each researcher individually. Therefore, the results of the direct comparison may vary.

Alongside the best-reported results, another point of interest is the processing speed. LPR systems are usually deployed for access control or law enforcement; hence, their results should be real-time. In this regard, we present the minimum processing time required for different data sets as reported by authors in the literature. Please note the reported results are for different processing setups; hence, the type of GPU used for reporting the results is also presented in Table 7. Among papers listed in the second column, the method proposed in [87] required the least 13.7 msec to recognize the license plate in the Caltech99 data set using a Titan X GPU. The best processing time of 25.5 msec for the AOLP data set was reported by Bjorklund et al. [9] on a GTX 1080 GPU.

While considering a deep learning architecture for license plate recognition, it is difficult to conclude which architecture is better for which problem. From Table 7 it is observable that generally, for most data sets, YOLO based algorithms result in the minimum processing time. However, it is difficult to draw a similar conclusion from Table 6 regarding the performance of the algorithms. Therefore, we have grouped algorithms based on their type and averaged their performance on popular data sets. For this comparison, only those data sets were selected, for which, the results were reported in at least two papers.

The first five methods in Table 8 can be categorized as CNN based methods, while the next three as RNN based methods. Please note that for this comparison, the F1 and accuracy scores are considered the same measure. Among the first five CNN based methods, only two have reported the accuracy of their method on the Caltech data set. Therefore, the average accuracy of these two methods is presented as “Average CNN Accuracy”. Thus, looking at the accuracy scores, it can be seen that for the Caltech data set YOLO based methods perform the best, followed closely by RNN based methods. However, it should be noted that the reported accuracy on the Caltech data set of the 2016 paper by Li et al. is significantly low as compared to the other reported results. However, the CNN based methods produce the best results for the AOLP data set. The table has been provided for the convenience of readers and should not be considered a conclusion on which architecture should be used for a particular data set.

Table 9, presents a comparison of some of the data sets commonly used in the literature. The table presents a summary of the total number of images in the data set and each image’s size. If a data set has subsets, they are also presented in the table. For example, for AOLP data set, the details of access control (AC), road patrolling (RP), and law enforcement (LE) subsets are presented separately. The country of the license plate is also mentioned in the table along with a brief description of the type of images available.

Our general observation regarding the presented material is that, initially, authors focused on proposing methods employing deep neural networks for the individual tasks of license plate detection, extraction of plate or characters, or recognition of characters (optical character recognition). However,

**TABLE 6. Best F1 and accuracy scores reported in literature for different datasets.**

Dataset	Papers using dataset	Best F1 Score (%)	Ref. of Best F1	DNN Type for Best F1 Score	Best Accuracy (%)	Ref. of Best Accuracy	DNN Type for Best Accuracy
Caltech	[59], [60], [52], [68], [87]	95.9	[87]	RCNN + CNN Swish	96.1	[52]	YOLO + YOLO
AOLP-AC	[59], [60], [9], [23], [87], [35], [112]	97.2	[9]		99.4	[35]	CNN + CNN
AOLP-LE	[59], [60], [9], [23], [87], [35], [112]	98.9	[9]		97.9	[35]	CNN + CNN
AOLP-RP	[59], [60], [9], [90], [23], [87], [35], [112]	98.4	[9]		99.5	[35]	CNN + CNN
PKU G1	[60], [87], [112]	99.9	[60]	FRCNN LSTMs +	97.9	[87]	RCNN + CNN Swish
PKU G2	[60], [87], [112]	99.9	[60]	FRCNN LSTMs +	97.4	[87]	RCNN + CNN Swish
PKU G3	[60], [87], [112]	99.6	[60]	FRCNN LSTMs +	97.2	[87]	RCNN + CNN Swish
PKU G4	[60], [87], [112]	100	[60]	FRCNN LSTMs +	96.9	[87]	RCNN + CNN Swish
PKU G5	[60], [87], [112]	99.3	[60]	FRCNN LSTMs +	96	[87]	RCNN + CNN Swish
Platesmania	[9]	N.A.	N.A.		98.3	[9]	CNN + CNN
OpenALPR	[68], [52], [90], [93]	N.A.	N.A.		95.7	[52]	YOLO + YOLO
UCSD	[52]	N.A.	N.A.		97.3	[52]	YOLO + YOLO
SSIG	[89], [90], [52], [23]	N.A.	N.A.		96.9	[52]	YOLO + YOLO
CCPD	[107]	N.A.	N.A.		98.5	[107]	CNN + CNN (Rpnet)
CLPD	[112]	N.A.	N.A.		76.8	[112]	Xception CNN + LSTM
Media Lab Greece	[120], [35]	N.A.	N.A.		97.9	[120]	ResNet + Inception v3
KarPlate Korea	[35]	N.A.	N.A.		98.2	[35]	YOLO v3 + YOLO v3 SPP
University of Zagreb	[35]	N.A.	N.A.		97	[35]	YOLO v3 + YOLO v3 SPP

**TABLE 7. Minimum processing time, reported in literature, for different datasets.**

Dataset	Papers using dataset	Minimum Processing Time (msec)	Processing Hardware	Ref. of Best Time	DNN Type for Best Time
Caltech	[59], [60], [52], [68], [87]	13.7	Titan X	[52]	YOLO + YOLO
AOLP	[59], [60], [9], [90], [23], [87], [35], [112]	25.5	GTX 1080	[9]	CNN + CNN
PKU	[60], [87], [112]	310	Titan X	[60]	FRCNN + LSTMs
OpenALPR	[68], [52], [90], [93]	13.7	Titan X	[52]	YOLO + YOLO
UCSD	[52]	13.7 msec	Titan X	[52]	YOLO + YOLO
SSIG	[89], [90], [52], [23]	13.7	Titan X	[52]	YOLO + YOLO
CCPD	[107]	16.4 msec	Quadro P4000	[107]	CNN + CNN (Rpnet)
Media Lab Greece	[120], [35]	40	Tesla K80	[120]	ResNet + Inception v3
KarPlate Korea	[35]	65.1	Titan X	[35]	YOLO v3 + YOLO v3 SPP
University of Zagreb	[35]	32.2	Titan X	[35]	YOLO v3 + YOLO v3 SPP

more recently, the focus has shifted to solving the complete chain of detection and recognition using deep networks. For that, authors generally choose between RCNN, YOLO, and SSD algorithms for detecting license plates in images. The reasons for choosing among these methods are presented below.

The design of RCNN, region-convolutional neural networks, focuses on object detection and localization. As the name indicates, it is a CNN capable of classifying objects in images with the additional capability of localizing them. RCNN speeds up the process of identifying objects by proposing around 2,000 candidate regions. Since 2,000 regions proposals per image is a large number, RCNN based

classification is generally slow. Its variant, Fast RCNN, takes an image as input instead of features; hence, CNN convolutions have to be performed once for the image instead of the 2,000 candidate regions, thus, making it faster. However, both RCNN and Fast RCNN suffer from selective search, which is slow. Faster RCNN allows region proposals to be predicted by a parallel network, thus speeding up calculations. Variants of RCNN have improved prediction speed; however, only Faster RCNN can produce near real-time results. Their training requires multiple phases, and overall the network is slow to predict.

To address these problems, the YOLO and SSD algorithms were proposed. As the name suggests, you only look

TABLE 8. Average of F1 or accuracy scores, reported in literature, for different datasets.

Paper	Caltech	AOLP	PKU	OpenALPR	SSIG	MediaLab Greece
<b>CNN based methods</b>						
Li 2016, [59]	77.45	89.4	N.A.	N.A.	N.A.	N.A.
Bjorklund 2019, [9]	N.A.	98.2	N.A.	N.A.	N.A.	N.A.
Spanhel 2018, [93]	N.A.	N.A.	N.A.	73.6	N.A.	N.A.
Goncalves 2018, [23]	N.A.	N.A.	N.A.	N.A.	88.8	N.A.
Masood 2017, [68]	93.4	N.A.	N.A.	94.5	N.A.	N.A.
<b>Average CNN Accuracy</b>	<b>85.4</b>	<b>95.3</b>	N.A.	<b>84.05</b>	<b>88.8</b>	<b>96</b>
<b>RNN based methods</b>						
Li 2018, [60]	94.1	91.9	99.7	N.A.	N.A.	N.A.
Selmi 2020, [87]	95.9	96.2	97.1	N.A.	N.A.	N.A.
Zhang 2020, [112]	N.A.	95.8	88.2	N.A.	N.A.	N.A.
<b>Average RNN Accuracy</b>	<b>95</b>	<b>94.6</b>	<b>95</b>	N.A.	N.A.	N.A.
<b>YOLO based methods</b>						
Silva 2018, [90]	N.A.	98.4	N.A.	93.5	88.6	N.A.
Laroca 2019, [52]	96.1	98.4	N.A.	95.7	96.9	N.A.
Hendry 2019, [34]	N.A.	78	N.A.	N.A.	N.A.	N.A.
Henry 2020, [35]	97.8	98.9	N.A.	N.A.	N.A.	97
<b>Average YOLO Accuracy</b>	<b>96.9</b>	<b>93.4</b>	N.A.	<b>94.6</b>	<b>92.75</b>	<b>97</b>
<b>Other methods</b>						
Zhuang 2018, [120]	N.A.	N.A.	N.A.	N.A.	N.A.	97.9
<b>Average Other Accuracy</b>	N.A.	N.A.	N.A.	N.A.	N.A.	<b>97.9</b>

TABLE 9. Comparison of car license plate data sets.

Dataset	Data set size	Size of images	Origin	Type of images
Caltech 1999	126	896 × 592	USA	Day, Back of Car, outdoor
Caltech 2001	526	360 × 240	USA	Day, Back of Car, outdoor
AOLP-AC	681		Taiwan	Day, night, indoor, outdoor
AOLP-LE	757		Taiwan	Day, night, indoor, outdoor
AOLP-RP	611		Taiwan	Day, night, indoor, outdoor
PKU G1	810	1082 × 727	China	Day, Evening, outdoor
PKU G2	700	1082 × 727	China	Day, Evening, outdoor
PKU G3	743	1082 × 727	China	Evening, Night, outdoor
PKU G4	572	1600 × 1236	China	Day, Night, outdoor
PKU G5	1152	1600 × 1200	China	Day, Evening, Night, Multiple cars
OpenALPR	115, 108, 222		Brazil, EU, USA	Day, outdoor
UCSD	291	640 × 480	USA	Day, outdoor
SSIG	2000	1920 × 1080	Brazil	Day, outdoor
CCPD	250000		China	
Media Lab Greece	716		Greece Europe	Shadows, Low-light, blur, dirt
KarPlate Korea - LPD	4267	1920 × 1080	Korea	Day, outdoor
KarPlate Korea - LPR	4267		Korea	Day, outdoor
KarPlate Korea - EER	929	1920 × 1080	Korea	Day, outdoor
University of Zagreb	501		Croatia	
UFPR-ALPR	4500	1920 × 1080	Brazil	Day, outdoor
	(900 cars, 300 cars, 300 bikes)			

once (YOLO) is an efficient algorithm for finding objects in images. Since objects are identified in different grids and grids representing similar objects are subsequently combined, the algorithm does not require multiple passes. This results in a fast algorithm. However, the resizing of the input image and the size of the grid determines if the algorithm would suffer from localization errors or not. Also, compared to other algorithms, the results are not very accurate.

Comparatively, Single Shot Detector (SSD) performs better than YOLO in the localization of objects. SSD builds on the VGG-16 architecture while discarding its fully connected layers. The addition of auxiliary convolutional layers increases features at multiple spatial scales. SSD builds upon the philosophy of multibox based bounding box proposals but is capable of classifying objects. SSD adds more boxes

for identification compared to YOLO, resulting in higher accuracy at the cost of speed. VGG-16 takes most of the processing time, which may be more than 75% of the total processing time. The use of multi-scale feature extraction benefits SSD in terms of localization; however, the downside is that smaller objects that may not appear across all feature maps may not get detected.

While covering license plate recognition methods, we observed that the three most frequent architecture choices were CNN, YOLO, and LSTMs. Some authors proposed using Capsule networks or Alexnet with inception layers or even simple logistic regression based classification for recognition of characters on the detected license plate region. However, CNN, YOLO, and LSTMs remained the three main choices for the task of recognition. As for the license plate

detection, YOLO-based methods for character recognition performed efficiently with respect to time; however, the quality of results was not as accurate as some CNN or LSTM based methods.

Convolutional neural networks from LeNet-5 to date have been specifically used for the task of character recognition. They are more successful in approaches where the license plate has already been detected and hence do not have to localize the sequence of characters. Thus, it was observed that in most of the cases where CNN was proposed for character recognition, the license plate had already been detected. LSTMs have proven to be very efficient for text-based natural language processing (NLP). However, the problem for license plate text recognition is slightly different from NLP since the characters appear randomly, and nearly all combinations exist. This is unlike natural language processing, where the sequence of characters depends upon the language vocabulary. Therefore, the only advantage, in this context, that LSTMs may have is their ability to determine how many characters and numbers appear on a license plate and in which order. This can help reduce errors, especially when zero '0' is mixed up with the English alphabet 'o', and similarly '9' and 'g', etc.

It was also observed in the literature that instead of relying only on real data, the focus has shifted to exploiting synthetic data. This is generated using conventional image processing techniques or more recently developed generative adversarial networks (GANs). To get better results, authors trained their networks on huge data sets comprising positive and negative examples of license plates and their characters. However, it seems that the trend of generating huge data sets will not continue as the focus has shifted to fine-tuning for new applications instead of complete training. The idea is to train a huge common network on large amounts of data and then use applications or use-case specific examples to fine-tune this common model. This has resulted in significant gains in both computer vision and natural language processing based tasks.

## VIII. CONCLUSION

Current state-of-the-art literature on License plate recognition has focused mainly on the detection, extraction, and recognition of license plates. Some authors have also focused on deblurring, denoising, and geometric transformations to improve recognition accuracy. Neural networks were initially employed to solve the individual problems of detection, extraction, or optical character recognition. However, later efforts focused on employing them for the complete process. In this aspect, researchers have proposed using two, three, four, and even five networks to identify if an image contains a license plate or not. Over the years, various authors have employed custom CNNs, generative adversarial neural networks, recurrent neural networks, and single shot multi-box to solve the problem of license plate recognition. The focus on increasing the speed of these networks has revolved around the use and modification of the YOLO network and its evolution with time.

Compared to conventional deep learning strategies, the focus has recently shifted to semi-supervised learning. The current trend is to use large amounts of unsupervised data and a small amount of supervised data to improve model performance. This is done by optimizing two loss functions, one with supervised data and another with unsupervised data. The advantage of semi-supervised methods is the reduced time required for data annotation and preparation, as fewer data samples have to be marked manually. An interesting area to explore could be the minimum number of labeled samples required to train a license plate recognition system using a semi-supervised approach.

Transformers have recently seen much interest in the domain of natural language processing. Similar to it, Google has introduced a vision transformer for image classification. Transformers address the problem of requiring large annotated/tagged data sets for supervised learning. They employ self-supervised learning where large unlabeled data sets are used for training huge networks with billions of parameters [13]. These pre-trained models cannot be used for any specific task; however, re-trained or fine-tuned using a small data set, they can produce state-of-the-art results. Such networks have already been used for biomedical image classification [6] and fine-grained image classification [10]. Since many image data sets comprise characters and numbers, such pre-trained networks may be trained for license plate recognition by using a reduced fine-tuning data set.

Alongside these directions, methods like YOLO will continue to produce better results. Recently N-YOLO [43] was proposed for real-time object detection and tracking. The idea was to use fixed-size image patches instead of reducing the image size. This approach may help in the license plate recognition task since the license plate is generally a smaller object in the complete image frame. Most of the approaches in the literature are top-bottom approaches, i.e., a significantly huge list of objects is considered as hypotheses, rectangular boxes are identified, and then this list is reduced. Recently, the focus has shifted to bottom-up approaches as well, where objects emerge from combining sub-objects or parts. An example is CornerNet [55] where corners are detected first, and objects are then detected based on detected corners. HoughNet [86] comprises a CNN connected to three branches predicting dimensions of the object bounding box, object center location, and visual evidence scores calculated using log-polar based voting. The reason HoughNet is interesting is that it has demonstrated promising results for the detection of smaller objects.

The algorithms presented in this work have increased the accuracies of license plate recognition considerably, so much so, that for the Caltech data set an accuracy of 96% was achieved using YOLO architecture. For AOLP data set an average accuracy of 99% was achieved using CNN architecture. For the PKU dataset an average F1 score of 99% was achieved using RCNN and LSTM architectures. For OpenALPR, UCSD, SSIG, KarPlate and University of Zargeb data sets accuracies of 96%, 97%, 97%, 98% and 97% were

achieved using YOLO type networks while for CLPD an accuracy of 77% was achieved using Xception CNN and LSTM architecture.

License plate recognition has proven to be extremely beneficial in tasks related to access control, traffic monitoring, automatic violation ticketing, paid metering, toll booth ticketing, pollution control, security and surveillance. The birth of smart cities will require more and more automation tasks and this will require license plate recognition algorithms that are both efficient and streamlined. More and more cameras will increase the requirement of re-identification of cars for mapping their trajectories. An open challenge, with regards to license plate recognition, are those countries where standardized license plates are still not used by all the vehicles.

## ACKNOWLEDGMENT

The authors would like to thank Deanship of Scientific Research (DSR), University of Jeddah, for technical and financial support.

## REFERENCES

- [1] C. Anagnostopoulos, I. Anagnostopoulos, V. Loumos, and E. Kayafas, "A license plate-recognition algorithm for intelligent transportation system applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 3, pp. 377–392, Sep. 2006.
- [2] C.-N.-E. Anagnostopoulos, I. E. Anagnostopoulos, I. D. Psoroulas, V. Loumos, and E. Kayafas, "License plate recognition from still images and video sequences: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 3, pp. 377–391, Sep. 2008.
- [3] C. E. Anagnostopoulos, "License plate recognition: A brief tutorial," *IEEE Intell. Transp. Syst. Mag.*, vol. 6, no. 1, pp. 59–67, Spring. 2014.
- [4] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. Int. Conf. Mach. Learn.*, vol. 70. Aug. 2017, pp. 214–223.
- [5] M. Arsenovic, S. Sladojevic, A. Anderla, and D. Stefanovic, "Deep learning driven plates recognition system," in *Proc. 17th Int. Sci. Conf. Ind. Syst.*, 2017, pp. 1–4.
- [6] S. Azizi, B. Mustafa, F. Ryan, Z. Beaver, J. Freyberg, J. Deaton, A. Loh, A. Karthikesalingam, S. Kornblith, T. Chen, V. Natarajan, and M. Norouzi, "Big self-supervised models advance medical image classification," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 3458–3468.
- [7] S. Barreto, J. Lambert, and F. Vidal, "Using synthetic images for deep learning recognition process on automatic license plate recognition," in *Proc. Mex. Conf. Pattern Recognit.* Cham, Switzerland: Springer, 2019, pp. 115–126.
- [8] T. Björklund, A. Fiandrotti, M. Annarumma, G. Francini, and E. Magli, "Automatic license plate recognition with convolutional neural networks trained on synthetic data," in *Proc. IEEE 19th Int. Workshop Multimedia Signal Process. (MMSp)*, Oct. 2017, pp. 1–6.
- [9] T. Björklund, A. Fiandrotti, M. Annarumma, G. Francini, and E. Magli, "Robust license plate recognition using neural networks trained on synthetic images," *Pattern Recognit.*, vol. 93, pp. 134–146, Sep. 2019.
- [10] F. Al Breiki, M. Ridzuan, and R. Grandhe, "Self-supervised learning for fine-grained image classification," 2021, *arXiv:2107.13973*.
- [11] H. Caner, H. S. Gecim, and A. Z. Alkar, "Efficient embedded neural-network-based license plate recognition system," *IEEE Trans. Veh. Technol.*, vol. 57, no. 5, pp. 2675–2683, Sep. 2008.
- [12] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.
- [13] D. Chen, Y. Chen, Y. Li, F. Mao, Y. He, and H. Xue, "Self-supervised learning for few-shot image classification," in *Proc. IEEE ICASSP*, Jun. 2021, pp. 1745–1749.
- [14] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention based models for speech recognition," in *Proc. Adv. Neural Inf. Process. (NIPS)*, 2015, pp. 577–585.
- [15] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 379–387.
- [16] L. Dlagnekov and S. Belongie. (2005). *UCSD Dataset*. Accessed: Apr. 7, 2020. [Online]. Available: [http://vision.ucsd.edu/belongie-grp/research/carRec/car\\_data.html](http://vision.ucsd.edu/belongie-grp/research/carRec/car_data.html)
- [17] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, "Automatic license plate recognition (ALPR): A state-of-the-art review," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 2, pp. 311–325, Feb. 2013.
- [18] N. Duan, J. Cui, L. Liu, and L. Zheng, "An end to end recognition for license plates using convolutional neural networks," *IEEE Intell. Transp. Syst. Mag.*, vol. 13, no. 2, pp. 177–188, Summer. 2021.
- [19] M. Earl. (2016). *Deep APNR*. Accessed: Nov. 7, 2022. [Online]. Available: <https://github.com/matthewearl/deep-anpr>
- [20] Q. Fu, Y. Shen, and Z. Guo, "License plate detection using deep cascaded convolutional neural networks in complex scenes," in *Proc. Int. Conf. Neural Inf. Process.* Cham, Switzerland: Springer, 2017, pp. 696–706.
- [21] J. Gao, L. Sun, and M. Cai, "Quantifying privacy vulnerability of individual mobility traces: A case study of license plate recognition data," *Transp. Res. C, Emerg. Technol.*, vol. 104, pp. 78–94, Jul. 2019.
- [22] G. Goncalves, S. Da Silva, D. Menotti, and W. Schwartz, "Benchmark for license plate character segmentation," *J. Electron. Imag.*, vol. 25, no. 5, pp. 34–53, 2016.
- [23] G. R. Gonçalves, M. A. Diniz, R. Laroca, D. Menotti, and W. R. Schwartz, "Real-time automatic license plate recognition through deep multi-task networks," in *Proc. 31st SIBGRAPI Conf. Graph., Patterns Images (SIBGRAPI)*, Oct. 2018, pp. 110–117.
- [24] Y. Gong. (2022). *Chinese Road Plate Dataset*. Accessed: Nov. 7, 2022. [Online]. Available: <https://github.com/yxgong0/CRPD>
- [25] Y. Gong, L. Deng, S. Tao, X. Lu, P. Wu, Z. Xie, Z. Ma, and M. Xie, "Unified Chinese license plate detection and recognition with high efficiency," *J. Vis. Commun. Image Represent.*, vol. 86, Jul. 2022, Art. no. 103541.
- [26] C. Gou, K. Wang, B. Li, and F.-Y. Wang, "Vehicle license plate recognition based on class-specific ERs and SaE-ELM," in *Proc. 17th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Qingdao, China, Oct. 2014, pp. 2956–2961.
- [27] C. Gou, K. Wang, Y. Yao, and Z. Li, "Vehicle license plate recognition based on extremal regions and restricted Boltzmann machines," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1096–1107, Apr. 2016.
- [28] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of Wasserstein GANs," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 5767–5777.
- [29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
- [30] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.
- [31] H. A. Hegt, R. J. De La Haye, and N. A. Khan, "A high performance license plate recognition system," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Feb. 1998, pp. 4357–4362.
- [32] G. Heitz, S. Gould, A. Saxena, and D. Koller, "Cascaded classification models: Combining models for holistic scene understanding," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2009, pp. 641–648.
- [33] R.-C. Chen, "A new method for license plate character detection and recognition," in *Proc. 6th Int. Conf. Inf. Technol., IoT Smart City*, Dec. 2018, pp. 204–208.
- [34] R.-C. Chen, "Automatic license plate recognition via sliding-window darknet-YOLO deep learning," *Image Vis. Comput.*, vol. 87, pp. 47–56, Jul. 2019.
- [35] C. Henry, S. Y. Ahn, and S. Lee, "Multinational license plate recognition using generalized character sequence detection," *IEEE Access*, vol. 8, pp. 35185–35199, 2020, doi: [10.1109/ACCESS.2020.2974973](https://doi.org/10.1109/ACCESS.2020.2974973).
- [36] G. Hsu, J. Chen, and Y. Chung, "Application-oriented license plate recognition," *IEEE Trans. Veh. Technol.*, vol. 62, no. 2, pp. 552–561, Feb. 2013.
- [37] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269.
- [38] Q. Huang, Z. Cai, and T. Lan, "A single neural network for mixed style license plate detection and recognition," *IEEE Access*, vol. 9, pp. 21777–21785, 2021.
- [39] F. Iandola, S. Han, M. Moskewicz, K. Ashraf, W. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size," 2016, *arXiv:1602.07360*.

- [40] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5967–5976.
- [41] D. M. F. Izidio, A. P. A. Ferreira, H. R. Medeiros, and E. N. D. S. Barros, "An embedded automatic license plate recognition system using deep learning," *Design Autom. Embedded Syst.*, vol. 24, no. 1, pp. 23–43, Mar. 2020.
- [42] M. Jaderberg, K. Simonyan, and A. Zisserman, "Spatial transformer networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2015, pp. 2017–2025.
- [43] S. Jha, C. Seo, E. Yang, and G. Joshi, "Real time object detection and tracking system for video surveillance system," *Multimedia Tools Appl.*, no. 80, pp. 3981–3996, Jan. 2020.
- [44] X. Jin, R. Tang, L. Liu, and J. Wu, "Vehicle license plate recognition for fog-haze environments," *IET Image Process.*, vol. 15, no. 6, pp. 1273–1284, May 2021.
- [45] I. R. Khan, S. T. A. Ali, A. Siddiq, M. M. Khan, M. U. Ilyas, S. Alshomrani, and S. Rahardja, "Automatic license plate recognition in real-world traffic videos captured in unconstrained environment by a mobile camera," *Electronics*, vol. 11, no. 9, p. 1408, Apr. 2022.
- [46] I. Kilic and G. Aydin, "Turkish vehicle license plate recognition using deep learning," in *Proc. Int. Conf. Artif. Intell. Data Process. (IDAP)*, Sep. 2018, pp. 1–5.
- [47] S. G. Kim, H. G. Jeon, and H. I. Koo, "Deep-learning-based license plate detection method using vehicle region extraction," *Electron. Lett.*, vol. 53, no. 15, pp. 1034–1036, Jul. 2017.
- [48] B. Kim, T. Won, S. Park, and J. Heo, "Anomaly detection for deep-learning based license plate recognition in real time video," in *Proc. Conf. Res. Adapt. Convergent Syst.*, Sep. 2019, pp. 123–124.
- [49] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
- [50] F. D. Kurpiel, R. Minetto, and B. T. Nassu, "Convolutional neural networks for license plate detection in images," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 3395–3399.
- [51] 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 *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2018, pp. 1–10.
- [52] R. Laroca, L. A. Zanlorensi, G. R. Gonçalves, E. Todt, W. R. Schwartz, and D. Menotti, "An efficient and layout-independent automatic license plate recognition system based on the YOLO detector," 2019, *arXiv:1909.01754*.
- [53] R. Laroca, E. Cardoso, D. Lucio, V. Estevam, and D. Menotti, "On the cross-dataset generalization in license plate recognition," in *Proc. 17th Int. Joint Conf. Comput. Vis., Imag. Comput. Graph. Theory Appl.*, 2022, pp. 166–178.
- [54] R. Laroca, M. Santos, V. Estevam, E. Luz, and D. Menotti, "A first look at dataset bias in license plate recognition," 2022, *arXiv:2208.10657*.
- [55] H. Law and J. Deng, "CornerNet: Detecting objects as paired keypoints," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 734–750.
- [56] Y. Lee, J. Lee, H. Ahn, and M. Jeon, "SNIDER: Single noisy image denoising and rectification for improving license plate recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 1017–1026.
- [57] Y. Y. Lee, Z. A. Halim, and M. N. Ab Wahab, "License plate detection using convolutional neural network-back to the basic with design of experiments," *IEEE Access*, vol. 10, pp. 22577–22585, 2022.
- [58] J. Li, C. Niu, and M. Fan, "Multi-scale convolutional neural networks for natural scene license plate detection," in *Proc. Int. Symp. Neural Netw.* Cham, Switzerland: Springer, 2012, pp. 110–119.
- [59] H. Li and C. Shen, "Reading car license plates using deep convolutional neural networks and LSTMs," 2016, *arXiv:1601.05610*.
- [60] H. Li, P. Wang, and C. Shen, "Toward end-to-end car license plate detection and recognition with deep neural networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 3, pp. 1126–1136, Mar. 2019.
- [61] D. Li, X. Wang, and D. Kong, "Deeprebirth: Accelerating deep neural network execution on mobile devices," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–9.
- [62] J. Liang, G. Chen, Y. Wang, and H. Qin, "EGSNet: Edge-guided sparse attention network for improving license plate detection in the wild," *Applied Intell.*, vol. 52, no. 4, pp. 4458–4472, 2022.
- [63] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. Berg, "SSD: Single shot MultiBox detector," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 21–37.
- [64] X. Liu, W. Liu, T. Mei, and H. Ma, "A deep learning-based approach to progressive vehicle re-identification for urban surveillance," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 869–884.
- [65] X. Liu, W. Liu, H. Ma, and H. Fu, "Large-scale vehicle re-identification in urban surveillance videos," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2016, pp. 1–6.
- [66] J. Liu, X. Li, H. Zhang, C. Liu, L. Dou, and L. Ju, "An implementation of number plate recognition without segmentation using convolutional neural network," in *Proc. IEEE 19th Int. Conf. High Perform. Comput. Commun., IEEE 15th Int. Conf. Smart City, IEEE 3rd Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Dec. 2017, pp. 246–253.
- [67] N. Lu, W. Yang, A. Meng, Z. Xu, H. Huang, and L. Huang, "Automatic recognition for arbitrarily tilted license plate," in *Proc. 2nd Int. Conf. Video Image Process.*, Dec. 2018, pp. 23–28.
- [68] S. Z. Masood, G. Shu, A. Dehghan, and E. G. Ortiz, "License plate detection and recognition using deeply learned convolutional neural networks," 2017, *arXiv:1703.07330*.
- [69] A. Meng, W. Yang, Z. Xu, H. Huang, L. Huang, and C. Ying, "A robust and efficient method for license plate recognition," in *Proc. 24th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2018, pp. 1713–1718.
- [70] N. M. Murad, L. Rejeb, and L. B. Said, "The use of DCNN for road path detection and segmentation," *Iraqi J. Comput. Sci. Math.*, vol. 3, no. 2, pp. 119–127, 2022.
- [71] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Cham, Switzerland: Springer, 2016, pp. 483–499.
- [72] V. Nguyen and D. L. Nguyen, "Joint image deblurring and binarization for license plate images using deep generative adversarial networks," in *Proc. 5th NAFOSTED Conf. Inf. Comput. Sci. (NICS)*, Nov. 2018, pp. 430–435.
- [73] J. A. G. Nijhuis, M. H. T. Brugge, K. A. Helmholt, J. P. W. Pluim, L. Spaanenburg, R. S. Venema, and M. A. Westenberg, "Car license plate recognition with neural networks and fuzzy logic," in *Proc. Int. Conf. Neural Netw.*, 1995, pp. 2232–2236.
- [74] OpenALPR. (2016). *OpenALPR-EU Dataset*. [Online]. Available: <https://github.com/openalpr/benchmarks/tree/master/endoend/au>
- [75] H. Padmasiri, J. Shashirangana, D. Meedeniya, O. Rana, and C. Perera, "Automated license plate recognition for resource-constrained environments," *Sensors*, vol. 22, no. 4, p. 1434, Feb. 2022.
- [76] M. Peker, "Comparison of tensorflow object detection networks for licence plate localization," in *Proc. 1st Global Power, Energy Commun. Conf. (GPECOM)*, Jun. 2019, pp. 101–105.
- [77] X. Peng, L. Wen, D. Bai, and B. Peng, "Reformative vehicle license plate recognition algorithm based on deep learning," in *Proc. Int. Conf. Cognit. Syst. Signal Process.* Cham, Switzerland: Springer, 2018, pp. 243–255.
- [78] I. V. Pustokhina, D. A. Pustokhin, J. J. P. C. Rodrigues, D. Gupta, A. Khanna, K. Shankar, C. Seo, and G. P. Joshi, "Automatic vehicle license plate recognition using optimal K-means with convolutional neural network for intelligent transportation systems," *IEEE Access*, vol. 8, pp. 92907–92917, 2020.
- [79] J. Qian and B. Qu, "Fast license plate recognition method based on competitive neural network," in *Proc. 3rd Int. Conf. Commun., Inf. Manag. Netw. Secur. (CIMNS)*, 2018, pp. 114–117.
- [80] S. Qin and S. Liu, "Efficient and unified license plate recognition via lightweight deep neural network," *IET Image Process.*, vol. 14, no. 16, pp. 4102–4109, Dec. 2020.
- [81] M. Rahmani, M. Sabaghian, S. M. Moghadami, M. M. Talaie, M. Naghibi, and M. A. Keyvanrad, "IR-LPR: Large scale of Iranian license plate recognition dataset," 2022, *arXiv:2209.04680*.
- [82] S. Radzi and M. Khalil-Hani, "Character recognition of license plate number using convolutional neural networks," in *Proc. IVIC*, 2011, pp. 45–55.
- [83] S. Rakhshani, E. Rashedi, and H. Nezamabadi-Pour, "Representation learning in a deep network for license plate recognition," *Multimedia Tools Appl.*, vol. 79, pp. 13267–13289, May 2020, doi: [10.1007/s11042-019-08416-0](https://doi.org/10.1007/s11042-019-08416-0).
- [84] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.
- [85] H. Rtui, "License plate recognition using capsule networks with an improved dynamic routing algorithm," M.S. thesis, Dept. Comput. Sci., Hanyang Univ., South Korea, 2019.



- [86] N. Samet, S. Hicsonmez, and E. Akbas, "HoughNet: Integrating near and long-range evidence for bottom-up object detection," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2020, pp. 406–423.
- [87] Z. Selmi, M. B. Halima, U. Pal, and M. A. Alimi, "DELP-DAR system for license plate detection and recognition," *Pattern Recognit. Lett.*, vol. 129, pp. 213–223, Jan. 2020.
- [88] Y. Shen, T. Xiao, H. Li, S. Yi, and X. Wang, "Learning deep neural networks for vehicle re-ID with visual-spatio-temporal path proposals," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1918–1927.
- [89] S. Montazzolli and C. Jung, "Real-time Brazilian license plate detection and recognition using deep convolutional neural networks," in *Proc. 30th SIBGRAPI Conf. Graph., Patterns Images (SIBGRAPI)*, Oct. 2017, pp. 55–62.
- [90] S. Silva and C. Jung, "License plate detection and recognition in unconstrained scenarios," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 580–596.
- [91] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
- [92] J. Spanhel, J. Sochor, R. Juránek, A. Herout, L. Marsík, and P. Zecík, "Holistic recognition of low quality license plates by CNN using track annotated data," in *Proc. 14th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS)*, Aug. 2017, pp. 1–6.
- [93] J. Spanhel, J. Sochor, R. Juránek, and A. Herout, "Geometric alignment by deep learning for recognition of challenging license plates," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 3524–3529.
- [94] V. Srebric. (2003). *EnglishLP Database*. Accessed: Oct. 1, 2021. [Online]. Available: [http://www.zemris.hr/projects/LicensePlates/english/baza\\_slika.zip](http://www.zemris.hr/projects/LicensePlates/english/baza_slika.zip)
- [95] P. Svoboda, M. Hradis, L. Marsík, and P. Zecík, "CNN for license plate motion deblurring," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 3832–3836.
- [96] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1–7.
- [97] T. Tsai, Z. Lu, and C. Huang, "License plate recognition system based on deep learning," in *Proc. IEEE Int. Conf. Consum. Electron.*, May 2019, pp. 1–2.
- [98] S. Usmanhujaev, S. Lee, and J. Kwon, "Korean license plate recognition system using combined neural networks," in *Proc. Int. Symp. Distrib. Comput. Artif. Intell.* Cham, Switzerland: Springer, 2019, pp. 10–17.
- [99] S. Usmanhujaev. (2021). *Korean Car Plate Generator*. Accessed: Nov. 7, 2022. [Online]. Available: <https://github.com/Usmanhujaev/KoreanCarPlateGenerator>
- [100] T. Vaiyapuri, S. N. Mohanty, M. Sivaram, I. V. Pustokhina, D. A. Pustokhin, and K. Shankar, "Automatic vehicle license plate recognition using optimal deep learning model," *Comput., Mater. Continua*, vol. 67, no. 2, pp. 1881–1897, 2021.
- [101] X. Wang, Z. Man, M. You, and C. Shen, "Adversarial generation of training examples: Applications to moving vehicle license plate recognition," 2017, *arXiv:1707.03124*.
- [102] Y. Wang, Z. Bian, Y. Zhou, and L. Chau, "Rethinking and designing a high-performing automatic license plate recognition approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 8868–8880, Jul. 2022.
- [103] M. Weber. (1999). *Caltech Plate Dataset 1999*. Accessed: Oct. 1, 2021. [Online]. Available: <http://www.vision.caltech.edu/html-les/archive.html>
- [104] C. Wu, S. Xu, G. Song, and S. Zhang, "How many labeled license plates are needed?" in *Proc. Chin. Conf. Pattern Recognit. Comput. Vis. (PRCV)*. Cham, Switzerland: Springer, 2018, pp. 334–346.
- [105] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10726–10734.
- [106] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2010, pp. 3485–3492.
- [107] Z. Xu, W. Yang, A. Meng, N. Lu, H. Huang, C. Ying, and L. Huang, "Toward end-to-end license plate detection and recognition: A large dataset and baseline," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 255–271.
- [108] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong, "PC-DARTS: Partial channel connections for memory-efficient architecture search," 2019, *arXiv:1907.05737*.
- [109] F. You, Y. Zhao, and X. Wang, "Combination of CNN with GRU for plate recognition," *J. Phys., Conf.*, vol. 1187, no. 3, Apr. 2019, Art. no. 032008.
- [110] D. Zang, Z. Chai, J. Zhang, D. Zhang, and J. Cheng, "Vehicle license plate recognition using visual attention model and deep learning," *J. Electron. Imag.*, vol. 24, no. 3, pp. 1–10, May 2015.
- [111] M. Zha, G. Meng, C. Lin, Z. Zhou, and K. Chen, "RoLMA: A practical adversarial attack against deep learning-based LPR systems," in *Proc. Inf. Secur. Cryptol., 15th Int. Conf. (Inscrypt)*. Cham, Switzerland: Springer, 2020, pp. 101–117.
- [112] L. Zhang, P. Wang, H. Li, Z. Li, C. Shen, and Y. Zhang, "A robust attentional framework for license plate recognition in the wild," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 11, pp. 6967–6976, Nov. 2021, doi: [10.1109/TITS.2020.3000072](https://doi.org/10.1109/TITS.2020.3000072).
- [113] X. Zhai, F. Bensaali, and R. Sotudeh, "OCR-based neural network for ANPR," in *Proc. IEEE Int. Conf. Imag. Syst. Techn.*, Jul. 2012, pp. 393–397.
- [114] Y. Zhao, Z. Yu, X. Li, and M. Cai, "Chinese license plate image database building methodology for license plate recognition," *J. Electron. Imag.*, vol. 28, Jan. 2019, Art. no. 013001.
- [115] Y. Zhang and Q. Yang, "A survey on multi-task learning," 2017, *arXiv:1707.08114*.
- [116] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable person re-identification: A benchmark," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1116–1124.
- [117] S. Zherzdev and A. Gruzdev, "LPRNet: License plate recognition via deep neural networks," 2018, *arXiv:1806.10447*.
- [118] W. Zhou, H. Li, Y. Lu, and Q. Tian, "Principal visual word discovery for automatic license plate detection," *IEEE Trans. Image Process.*, vol. 21, no. 9, pp. 4269–4279, Sep. 2012.
- [119] J. Zhu, T. Park, P. Isola, and A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017.
- [120] J. Zhuang, S. Hou, Z. Wang, and Z. Zha, "Towards human-level license plate recognition," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 306–321.
- [121] Y. Zou, Y. Zhang, J. Yan, X. Jiang, T. Huang, H. Fan, and Z. Cui, "License plate detection and recognition based on YOLOv3 and ILPRNET," *Signal, Image Video Process.*, vol. 16, no. 2, pp. 473–480, Mar. 2022.



**MUHAMMAD MURTAZA KHAN** (Member, IEEE) received the B.Sc. degree (Hons.) in electrical engineering from the University of Engineering and Technology Taxila, Taxila, Pakistan, in 2000, the M.S. degree in computer software engineering from the EME College Rawalpindi, National University of Sciences and Technology, Islamabad, Pakistan, in 2005, and the M.S. and Ph.D. degrees in image processing from the Institut National Polytechnique de Grenoble, Grenoble, France, in 2006 and 2009, respectively. From 2000 to 2004, he was a Software Engineer and a Senior Software Engineer with Streaming Networks Pvt Ltd., Islamabad, where his responsibilities revolved around developing video drivers and implementation of optimized video codec solutions for Philips TriMedia processor. He has been an Associate Professor in computer science and artificial intelligence with the College of Computer Science and Engineering, University of Jeddah, Jeddah, Saudi Arabia, since 2015, and also he has been an Assistant Professor in electrical engineering with the School of Electrical Engineering and Computer Science, National University of Sciences and Technology, since 2010.



**MUHAMMAD U. ILYAS** (Senior Member, IEEE) received the B.E. degree (Hons.) in electrical engineering from the National University of Sciences and Technology, Islamabad, Pakistan, in 1999, the M.S. degree in computer engineering from the Lahore University of Management Sciences, Lahore, Pakistan, in 2004, and the M.S. and Ph.D. degrees in electrical engineering from Michigan State University, East Lansing, MI, USA, in 2007 and 2009, respectively. He was a Postdoctoral Research Associate appointed jointly by the Electrical and Computer Engineering Department and the Computer Science and Engineering Department, Michigan State University. He worked under the joint supervision of Dr. H. Radha and Dr. A. Liu at East Lansing. He has been an Assistant Professor and the Program Director of PG programs with the University of Birmingham Dubai, United Arab Emirates, since 2022. Prior to this, he has been an Associate Professor with the Department of Computer and Network Engineering, College of Computer Science and Engineering, University of Jeddah, Jeddah, Saudi Arabia, since 2016, and an Assistant Professor in electrical engineering with the School of Electrical Engineering and Computer Science, National University of Sciences and Technology, since 2011.



**ISHTIAQ RASOOL KHAN** (Member, IEEE) received the B.Sc. degree in electrical engineering from the University of Engineering and Technology Taxila, Taxila, Pakistan, in 1992, the M.S. degree in systems engineering from Quaid-i-Azam University, Islamabad, Pakistan, in 1994, and the M.S. degree in information engineering and the Ph.D. degree in digital signal processing from Hokkaido University, Japan, in 1998 and 2000, respectively. He was a JSPS Fellow with Hokkaido University (2000–2002). He was with The University of Kitakyushu, Japan, Kyushu Institute of Technology, Japan, and Institute for Infocomm Research, A\*STAR, Singapore. Currently, he is a Professor with the College of Computer Science and Engineering, University of Jeddah. His research interests include high dynamic range imaging, image quality assessment, artificial intelligence, medical image enhancement, data analytics, and digital filtering.



**SALEH M. ALSHOMRANI** received the bachelor's (B.Sc.) degree in computer science from King Abdulaziz University, Saudi Arabia, in 1997, the master's degree in computer science from Ohio University, USA, in 2001, and the Ph.D. degree in computer science from Kent State University, OH, USA, in 2008. He is currently a Professor with the Information Systems Department, University of Jeddah. His research interests include web distributed systems and internet computing, data mining, and algorithms.



**SUSANTO RAHARDJA** (Fellow, IEEE) received the B.Eng. degree in electronic engineering from the National University of Singapore, in 1991, and the M.Eng. and Ph.D. degrees in electronic engineering from Nanyang Technological University, Singapore, in 1993 and 1997, respectively. He is currently a Chair Professor with Northwestern Polytechnical University (NPU) under the Thousand Talent Plan of People's Republic of China. His research interests include multimedia, signal processing, wireless communications, discrete transforms and signal processing algorithms, implementation, and optimization. He was a recipient of numerous awards, including the IEE Hartree Premium Award, the Tan Kah Kee Young Inventor's Open Category Gold Award, the Singapore National Technology Award, the A\*STAR Most Inspiring Mentor Award, the Finalist of the 2010 World Technology Summit Award, the Nokia Foundation Visiting Professor Award, and the ACM Recognition of Service Award.

...