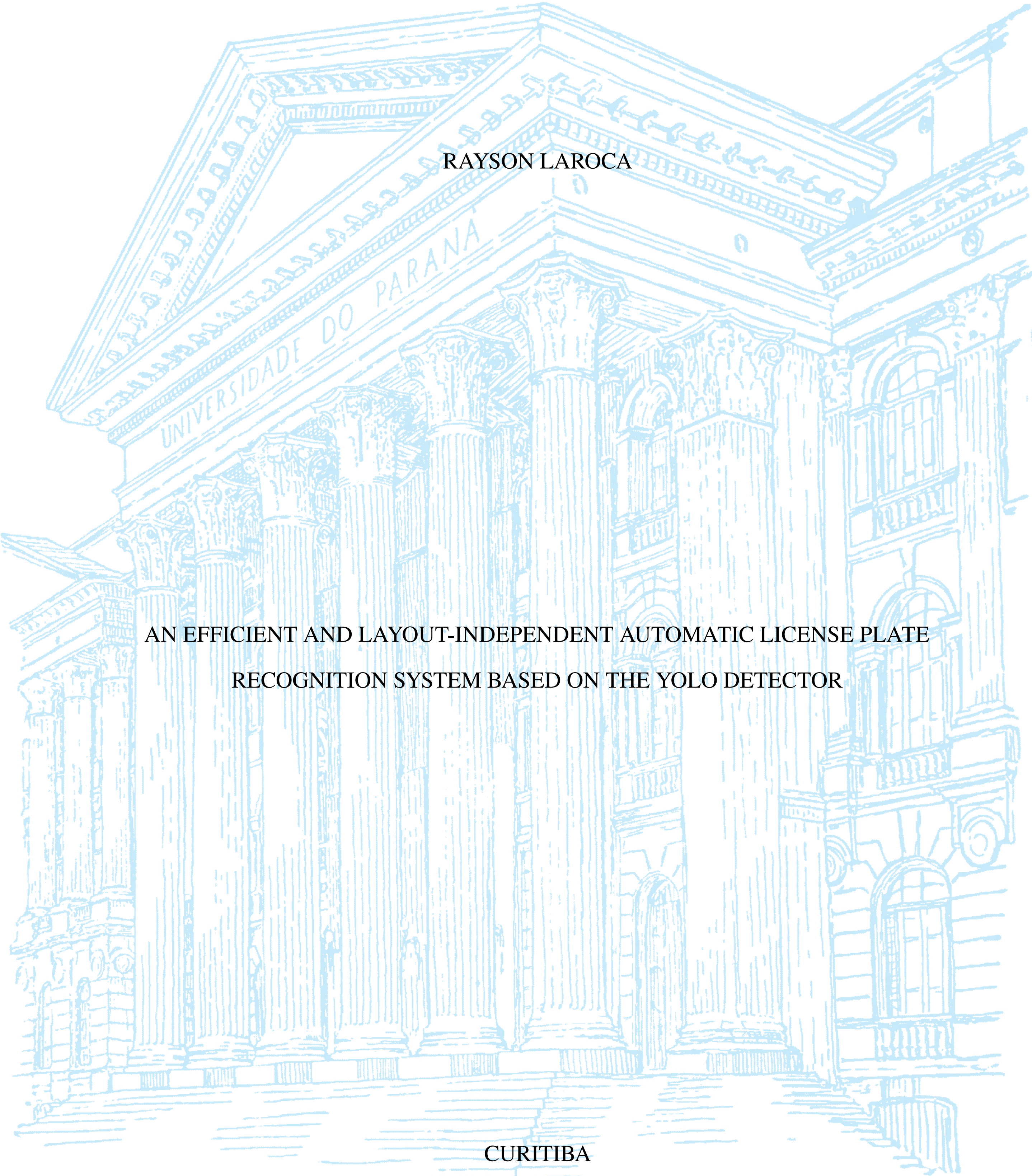
UNIVERSIDADE FEDERAL DO PARANÁ



RAYSON LAROCA

AN EFFICIENT AND LAYOUT-INDEPENDENT AUTOMATIC LICENSE PLATE RECOGNITION SYSTEM BASED ON THE YOLO DETECTOR

CURITIBA

2019

RAYSON LAROCA

AN EFFICIENT AND LAYOUT-INDEPENDENT AUTOMATIC LICENSE PLATE RECOGNITION SYSTEM BASED ON THE YOLO DETECTOR

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science at the Federal University of Paraná.

Advisor: David Menotti.

CURITIBA

2019

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

L326e

Laroca, Rayson
An efficient and layout-independent automatic license plate recognition system based on the Yolo detector [recurso eletrônico] / Rayson Laroca. – Curitiba, 2019.

Dissertação - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática, 2019.

Orientador: David Menotti .

1. Sistemas inteligentes de veículos rodoviários. 2. Sistemas de recuperação da informação – Rodovias. 3. Redes neurais (Computação). 4. Yolo (Sistema de detecção de objetos). I. Universidade Federal do Paraná. II. Menotti, David. III. Título.

CDD: 363.1256

Bibliotecário: Elias Barbosa da Silva CRB-9/1894

MINISTÉRIO DA EDUCAÇÃO
SETOR SETOR DE CIENCIAS EXATAS
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO INFORMÁTICA -
40001016034P5

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **RAYSON BARTOSKI LAROCA DOS SANTOS** intitulada: **An Efficient and Layout-Independent Automatic License Plate Recognition System Based on the YOLO Detector**, após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua Aprovação no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 08 de Março de 2019.

D Menotti
DAVID MENOTTI GOMES
Presidente da Banca Examinadora (UFPR)

EDUARDO TODT
Avaliador Interno (UFPR)

Rodrigo Minetto
RODRIGO MINETTO
Avaliador Externo (UTFPR)

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
Setor de Ciências Exatas
Coordenação de Pós-Graduação em Informática

To my parents and grandparents.

Acknowledgments

First and foremost, I would like to thank my advisor David Menotti, who has shaped my perspective as a researcher in many fundamental ways, for his guidance and support. Prof. David gave me a lot of freedom to explore new ideas and experiment new methods, while provided critical insights to guide the direction of my research.

I would also like to thank the members of my dissertation committee, Prof. Eduardo Todt and Prof. Rodrigo Minetto, for the time they spent working on my committee, and for their great help in improving the quality of this dissertation.

Many thanks to all the people who brought me to academic research, especially Prof. Alaine Guimarães, Prof. Karine Sato and all the other people at the Infoagro laboratory at the State University of Ponta Grossa. Without them, I would probably never have made up my mind to pursue a master's degree, and I would have missed this great experience in my life.

Late nights and weekends at the laboratory were brought to life by the fellowship of my colleagues and friends in the Vision, Robotics and Imaging (VRI) Research Group. I have learned so much from our discussions and I am grateful to have a group of friends who inspire each other to be the best scientists and human beings we can be.

I would like to express my gratitude to Maria Karolina Ramos, my girlfriend, for her continued and unfailing love, support and understanding for the past six years. Karol has always been around in difficult times and helped me keep things in perspective. She brought me peace, made me smile, and gave me strength all the way through. This work is also dedicated to her.

Last but not least, I would like to thank my parents, grandparents and brothers. My grandmother Neide did not get to see me complete my master's degree but would be so proud of my accomplishments. They all gave me the encouragement and motivation I needed throughout my studies and always supported me. I am grateful for their unconditional love, support, and slaps on the wrist. Their dedication has made it possible for me to become the person I am today.

This work was supported by the National Council for Scientific and Technological Development (CNPq). The Titan Xp used for this research was donated by the NVIDIA Corporation.

"Artificial Intelligence is the new electricity."
Andrew Ng

RESUMO

O reconhecimento automático de placas de veículos (ALPR, do inglês *Automatic License Plate Recognition*) tem sido um tópico frequente de pesquisa devido às muitas aplicações práticas tais como cobrança automática de pedágio e aplicação da lei de trânsito. No entanto, muitas das soluções atuais ainda não são robustas o suficiente para serem aplicadas em situações do mundo real, dependendo comumente de certas restrições como câmeras ou ângulos de visão específicos, planos de fundo simples, boas condições de iluminação, entre outras. Esta dissertação apresenta um sistema ALPR eficiente e independente de layout baseado no detector de objetos de última geração YOLO (*You Only Look Once*), que contém uma abordagem unificada para detecção de placas e classificação de layout para melhorar os resultados de reconhecimento através de regras de pós-processamento. O sistema é concebido através da avaliação e otimização de diferentes modelos com várias modificações, visando obter o melhor compromisso de velocidade/precisão em cada estágio. As Redes Neurais Convolucionais (CNNs, do inglês *Convolutional Neural Networks*) são treinadas utilizando imagens de vários conjuntos de dados, com a adição de várias técnicas de aumento de dados, para que sejam robustas sob diferentes condições (por exemplo, com variações de iluminação, posição e configurações da câmera, e tipos de veículos). Este trabalho também introduz um conjunto de dados público para ALPR, chamado UFPR-ALPR, que inclui 4.500 imagens totalmente anotadas de 150 veículos em cenários do mundo real em que tanto o veículo quanto a câmera (dentro de outro veículo) estão em movimento. Em comparação com o conjunto de dados público de placas brasileiras mais empregado para ALPR, o conjunto de dados proposto tem mais que o dobro de imagens e contém uma variedade maior em diferentes aspectos. O sistema proposto foi capaz de atingir uma taxa média de reconhecimento de ponta a ponta de 96.8% em oito conjuntos de dados públicos utilizados nos experimentos, superando tanto os trabalhos anteriores quanto os sistemas comerciais nos conjuntos de dados ChineseLP, OpenALPR-EU, SSIG-SegPlate e UFPR-ALPR. Nos demais conjuntos de dados, a abordagem proposta obteve resultados competitivos ao melhor resultado alcançado pelas linhas de base. Nosso sistema também alcançou impressionantes taxas de quadros por segundo (FPS, do inglês *Frames Per Second*) em uma unidade de processamento gráfico (GPU, do inglês *Graphics Processing Unit*) de ponta, sendo capaz de executar em tempo real mesmo quando há 4 veículos na cena.

Palavras-chave: Reconhecimento de Placas de Veículos, Redes Neurais Convolucionais, YOLO.

ABSTRACT

Automatic License Plate Recognition (ALPR) has been a frequent topic of research due to many practical applications such as automatic toll collection and traffic law enforcement. However, many of the current solutions are still not robust enough to be executed on real-world scenarios, commonly depending on certain constraints such as specific cameras or viewing angles, simple backgrounds, good lighting conditions, among others. This dissertation presents an efficient and layout-independent ALPR system based on the state-of-the-art You Only Look Once (YOLO) object detector that contains a unified approach for License Plate (LP) detection and layout classification to improve the recognition results using post-processing rules. The system is conceived by evaluating and optimizing different models with various modifications, aiming at achieving the best speed/accuracy trade-off at each stage. The Convolutional Neural Networks (CNNs) are trained using images from several datasets, with the addition of various data augmentation techniques, so that they are robust under different conditions (e.g., with variations in lighting, camera position and settings, and vehicle types). This work also introduces a public dataset for ALPR, called UFPR-ALPR, that includes 4,500 fully annotated images from 150 vehicles in real-world scenarios where both the vehicle and the camera (inside another vehicle) are moving. Compared to the public dataset of Brazilian LPs most frequently used for ALPR, our dataset has more than twice the images and contains a larger variety in different aspects. The proposed system achieved an average end-to-end recognition rate of 96.8% across eight public datasets (from five different regions) used in the experiments, outperforming both previous works and commercial systems in the ChineseLP, OpenALPR-EU, SSIG-SegPlate and UFPR-ALPR datasets. In the other datasets, the proposed approach achieved competitive results when compared to the best baselines. Our system also achieved impressive Frames Per Second (FPS) rates on a high-end GPU, being able to perform in real time even when there are 4 vehicles in the scene.

Keywords: Automatic License Plate Recognition, Convolutional Neural Networks, YOLO.

List of Figures

List of Tables

List of Acronyms

AC	Access Control
ALPR	Automatic License Plate Recognition
ANPR	Automatic Number Plate Recognition
AOLP	Application-Oriented License Plate
AP	Average Precision
API	Application Programming Interface
BFLOP	Billion Floating-Point Operations
BRNN	Bidirectional Recurrent Neural Network
CCA	Connected Component Analysis
CCPD	Chinese City Parking Dataset
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CTC	Connectionist Temporal Classification
ELM	Extreme Learning Machine
FLOP	Floating-Point Operations
FN	False Negative
FP	False Positive
FPN	Feature Pyramid Network
FPS	Frames Per Second
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
HMM	Hidden Markov Model
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IoU	Intersection over Union
ITS	Intelligent Transportation Systems
LE	Law Enforcement
LP	License Plate
LPR	License Plate Recognition
LSTM	Long Short-Term Memory
mAP	mean Average Precision
Mercosur	*Mercado Común del Sur* - Southern Common Market
MLP	Multilayer Perceptron
MNIST	Modified National Institute of Standards and Technology
NMS	Non-Maximum Suppression
OCR	Optical Character Recognition
PNG	Portable Network Graphics
PReLU	Parametric ReLU
RCNN	Region-based CNN

ReLU	Rectified Linear Unit
ResNet	Residual Network
ROI	Region of Interest
RP	Road Patrol
RPN	Region Proposal Network
SMQT	Successive Mean Quantization Transform
SNoW	Sparse Network of Winnows
SSD	Single Shot MultiBox Detector
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
VOC	Visual Object Classes
WPOD-NET	Warped Planar Object Detection Network
YOLO	You Only Look Once

Contents

1 Introduction

Automatic License Plate Recognition (ALPR) became an important topic of research since the appearance of the first works in the early 1990s [1–3]. A variety of ALPR systems and commercial products have been produced over the years due to many practical applications such as automatic toll collection, border control, traffic law enforcement, private spaces access control and road traffic monitoring [4–6].

ALPR is also known as License Plate Recognition (LPR) and Automatic Number Plate Recognition (ANPR) [6–8]. As shown in Figure 1.1, ALPR systems typically have four stages: image acquisition, License Plate (LP) detection, character segmentation and character recognition, which refer to (i) acquiring the image using a camera, (ii) locating the LP region in the acquired image, (iii) segmenting each character within the detected LP and (iv) classifying each segmented character.

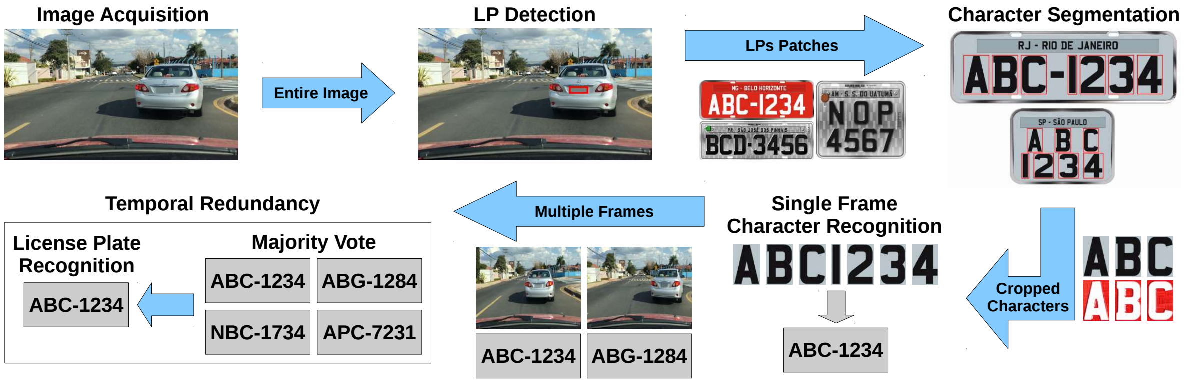


Figure 1.1: A usual ALPR system with temporal redundancy at the end.

The LP detection and character segmentation stages require higher accuracy since a failure would probably lead to another failure in the subsequent stages. Many authors have proposed approaches with a vehicle detection stage prior to LP detection, aiming to eliminate False Positives (FPs) and reduce processing time [9–11]. Regarding character segmentation, it has become common the use of segmentation-free approaches for LP recognition [8, 12, 13], as the character segmentation by itself is a challenging task that is prone to be influenced by uneven lighting, shadows, and noise [14].

Many computer vision tasks have recently achieved a great increase in performance mainly due to the availability of large-scale annotated datasets (i.e., ImageNet [15]) and hardware capable of handling a large amount of data, i.e., Graphics Processing Units (GPUs). In this scenario, deep learning techniques arise, with many machine learning competitions and challenges being won through them, even achieving superhuman visual results in some domains [16]. Despite the remarkable progress of deep learning approaches in ALPR [14, 17, 18], there are still many open challenges in this context.

1.1 Problem Statement

Although ALPR has been frequently addressed in the literature, many solutions are still not robust enough to be executed on real-world scenarios. Such solutions commonly depend on certain constraints such as specific cameras or viewing angles, simple backgrounds, good lighting conditions, search in a fixed region, and certain types of vehicles (e.g., they would not detect LPs from vehicles such as motorcycles, trucks or buses). Additionally, several approaches rely on handcrafted features that capture certain morphological and color attributes of the LPs [19–22]. These features are easily affected by noise and might not be robust for LPs of different layouts.

ALPR systems must be capable of recognizing multiple LP layouts since there might be various LP layouts in the same country or region, as illustrated in Figure 1.2. In some countries, it is possible to customize the LPs. In the United States, for example, many states sell *specialty* LPs displaying the emblems of colleges, universities, clubs, professional sports team, and fraternal organizations. It is also possible to customize the arrangement of letters and digits for an extra fee (i.e., *vanity* LPs) [23].

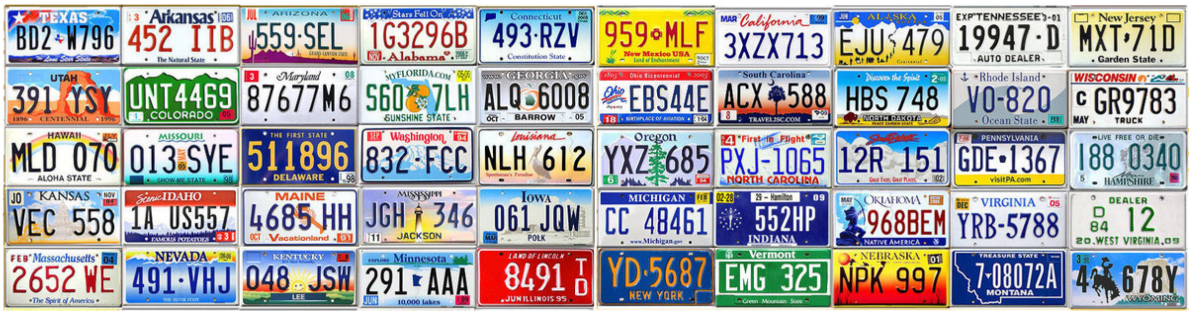


Figure 1.2: Examples of different LP layouts in the United States. Image reproduced from `http://www.ashtonrose.org/blog/new-north-dakota-license-plate`.

Even though many authors claim that their approaches could be extended with small modifications to detect/segment/recognize LPs of different layouts [24–27], this may not be an easy task. For example, a character segmentation approach designed for LPs with simple backgrounds is likely to fail on LPs with complex backgrounds and logos that touch and overlap some characters [8, 28].

In addition, ALPR systems should operate fast enough to fulfill the needs of Intelligent Transportation Systems (ITS). In technical terminology, a “real-time” operation for ALPR stands for a fast-enough operation to not miss a single object of interest that moves through the scene [4]. In the literature [13, 29, 30], generally a system is considered “real-time” if it is capable of processing at least 30 Frames Per Second (FPS) since commercial cameras usually record videos at that frame rate. Despite the importance of having a fast system in ALPR applications, many authors still propose computationally expensive approaches that are not able to process frames in real time, even when the experiments are performed on a high-end GPU [14, 17, 26].

Although major advances have been achieved in computer vision using deep learning methods [31], there is still a great demand for ALPR datasets with vehicle and LP annotations. The SSIG-SegPlate [32] is the best known public dataset of Brazilian LPs for ALPR, and it has been often used in the literature [11, 13, 33], as the bounding box of all LP characters were manually labeled by the authors, enabling the application of object detection and data augmentation techniques that require the position of each character. However, the SSIG-SegPlate dataset contains less than 800 training examples and has several constraints such as the use of a static camera mounted always in the same position, all images have very similar and relatively simple backgrounds, there are no motorcycles and only a few cases where the LPs are not well aligned. It should be noted that deep learning approaches are particularly dependent on the availability of large quantities of training data in order to generalize well and yield high classification accuracy on unseen data [34]. Higher amounts of data allow the use of more robust network architectures with more parameters and layers.

In ALPR applications, the images are acquired by cameras located alongside roads, in the entrance/exit of private spaces, on a vehicle's windshield, among others. The camera's position and its specifications (resolution, autofocus, focal length, etc.) should be considered since an ALPR approach might be robust only for a specific setup/application [35], for example, a system designed for images captured by a static camera at the entrance/exit of a parking lot will probably perform poorly on images acquired by dashboard cameras. This was illustrated in the most recent work of our colleagues [13]. Even though their approach, which is based on deep multi-task networks, achieved state-of-the-art results in the SSIG-SegPlate dataset, it did not perform well in our dataset (introduced in Section 4.1), in which the images were acquired by non-static cameras, correctly recognizing less than 60% of the frames in the test set. As pointed out in [13], the nature of non-static backgrounds might be very problematic to some LP detection approaches that work directly on the frames (i.e., without vehicle detection) since there are many different patterns on the scenes that might be confused with an LP.

You Only Look Once (YOLO) [29, 36, 37] is a real-time object detection system that achieved outstanding and state-of-the-art results in the Pascal Visual Object Classes (VOC) [38] and Common Objects in Context (COCO) [39] detection tasks. Although YOLO has already been employed in the ALPR context in previous works, a detailed assessment of its concepts or models for this task has not yet been presented, to the best of our knowledge. In [40, 41], for example, promising LP detection results were achieved through models based on YOLO, however, these works did not address LP recognition. In [11], on the other hand, all stages were handled using YOLO-based models. Although the ALPR system proposed in their work is quite fast (i.e., 76 FPS on a high-end GPU), a poor recognition rate of 63.18% was obtained in the SSIG-SegPlate dataset, which is not satisfactory for real-world applications.

1.2 Objectives

As great advances in object detection were achieved through YOLO-inspired models [42–44], we decided to specialize it for ALPR. The main objective of this work is **to design an efficient and layout-independent ALPR system using the YOLO object detector at all stages**.

In order to accomplish the main objective, some secondary or specific objectives are required, as follows:

- To eliminate several constraints commonly found in ALPR systems. All stages of the proposed approach are trained and tested using images from several datasets, which were collected under different conditions (e.g., with variations in lighting, camera position and settings, and vehicle types) and reproduce distinct real-world applications;

- To propose a layout classification stage prior to LP recognition, so that we can employ layout-specific approaches for this task in cases where the LP and its layout are predicted with a high confidence value. In other cases, a generic approach is applied;

- To evaluate different YOLO models (e.g., Fast-YOLOv2, YOLOv2 and YOLOv3) with various modifications (e.g., changes in the input size, number of filters, layers, and anchors, among others) and carefully combine them in the best way in order to achieve the best speed/accuracy trade-off at each stage;

- To propose a larger dataset for ALPR focused on usual and different real-world scenarios, which eliminates many of the constraints found in ALPR applications by using different non-static cameras to capture images from different types of vehicles (cars, motorcycles, buses and trucks) with complex backgrounds and under different lighting conditions;
- To design and apply data augmentation techniques to simulate LPs of other layouts and to generate LP images with characters that have few instances in the training set, as many examples are needed to effectively train Convolutional Neural Networks (CNNs).

1.3 Contributions

The main contributions of this work can be summarized as follows:

- A new efficient and layout-independent ALPR system using the state-of-the-art YOLO object detection CNNs[1], which outperforms previous works and two commercial systems in the ChineseLP [45], OpenALPR-EU [46], SSIG-SegPlate [32] and UFPR-ALPR datasets, and achieves competitive results to the baselines in other four public datasets;
- A public dataset for ALPR that includes 4,500 fully annotated images (with over 30,000 LP characters) from 150 vehicles in real-world scenarios where both the vehicle and the camera (inside another vehicle) are moving. Compared to the SSIG-SegPlate dataset [32], the proposed one has more than twice the images and contains a larger variety in different aspects;
- Annotations regarding the position of the vehicles, LPs and characters, as well as their classes, in the public datasets used in this work since they have no annotations or contain labels only for part of the ALPR pipeline. The annotations, which were made manually, are publicly available to the research community;
- A comparative evaluation of the proposed approach, previous works in the literature and two commercial systems in eight publicly available datasets.

All publications generated by this research are listed in Section 6.2.

1.4 Outline

The remainder of this work is organized as follows. Chapter 2 presents the theoretical foundation of deep learning, CNNs and YOLO. We briefly review related works in Chapter 3. The UFPR-ALPR dataset and the proposed ALPR system are introduced in Chapter 4. We report and discuss the results of our experiments in Chapter 5. Conclusions and future works are given in Chapter 6.

[1] The entire ALPR system, i.e., the architectures and weights, is publicly available for academic purposes at `https://web.inf.ufpr.br/vri/publications/layout-independent-alpr/`.

2 Theoretical Foundation

In this chapter, we present a theoretical basis of the concepts employed in this work. We first introduce the evaluation metrics commonly used for object detection since our ALPR system is based on the YOLO object detector. Then, we provide information on deep learning, CNNs and data augmentation. Finally, we describe YOLO, its second and third versions (i.e., YOLOv2 and YOLOv3) in detail.

2.1 Evaluation Metrics

The *precision* and *recall* evaluation metrics are commonly used in object detection [38, 39, 47] and also in the ALPR context [11, 14, 41]. These metrics are defined based on the area of the ground truth and the predicted bounding boxes in terms of FPs, False Negatives (FNs), True Positives (TPs) and True Negatives (TNs), and can be formally expressed as

$$precision = \frac{TP}{TP + FP}, \tag{2.1}$$

$$recall = \frac{TP}{TP + FN}, \tag{2.2}$$

where TPs are examples correctly labeled as positives, FPs refer to negative examples incorrectly labeled as positive, TNs correspond to examples correctly labeled as negatives and, finally, FNs refer to positive examples incorrectly labeled as negatives [48].

Precision varies in the [0,1] range and the higher its value, the smaller is the set of FPs which were computed. Recall also varies in the [0,1] range and the higher its value, the smaller is the set of TPs which were not found [49]. As pointed out in [50], neither precision nor recall alone can accurately assess the match quality. In particular, recall can be easily maximized by returning as many predictions as possible (resulting in a poor precision), e.g. predicting many vehicles/LPs in the same frame/region. On the other side, a high precision can be achieved at the expense of a poor recall by returning only a few (correct) correspondences, e.g. using a very high confidence threshold to consider a vehicle/LP detection.

The *F-measure* metric is defined as a harmonic mean of precision and recall. As shown in Equation 2.3, the most general form allows the differential weighting of precision and recall, however, commonly they are given equal weight (i.e., $\beta = 1$) [51]. The *Average Precision* (AP) [38] metric summarizes the shape of the precision/recall curve, and is defined as the average precision at a set of eleven equally spaced recall levels [0, 0.1, . . . , 1] (see

Equation 2.4). Finally, the *mean Average Precision* (mAP) is calculated by taking the mean AP over all classes.

$$F\text{-}measure = (1+\beta^2)\cdot\frac{precision\cdot recall}{(\beta^2\cdot precision)+recall}, \tag{2.3}$$

$$AP = \frac{1}{11}\sum_{r\in\{0,0.1,\ldots,1\}}\max_{\tilde{r}:\tilde{r}>r} Precision(\tilde{r})\,. \tag{2.4}$$

A metric often used to assess the quality of predictions in object detection tasks is the *Intersection over Union* (IoU), also known as Jaccard index and Jaccard similarity coefficient, which can be expressed by the formula

$$IoU = \frac{area(B_p\cap B_{gt})}{area(B_p\cup B_{gt})}, \tag{2.5}$$

where B_p and B_{gt} are the predicted and ground truth bounding boxes, respectively. Figure 2.1 illustrates this definition. The closer the IoU is to 1, the better the detection.

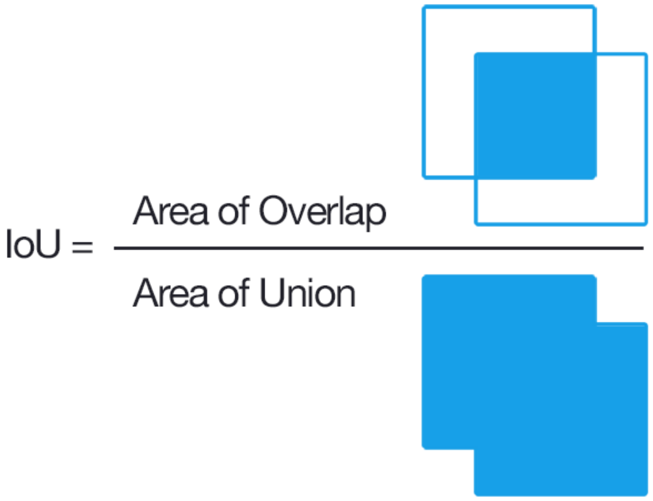


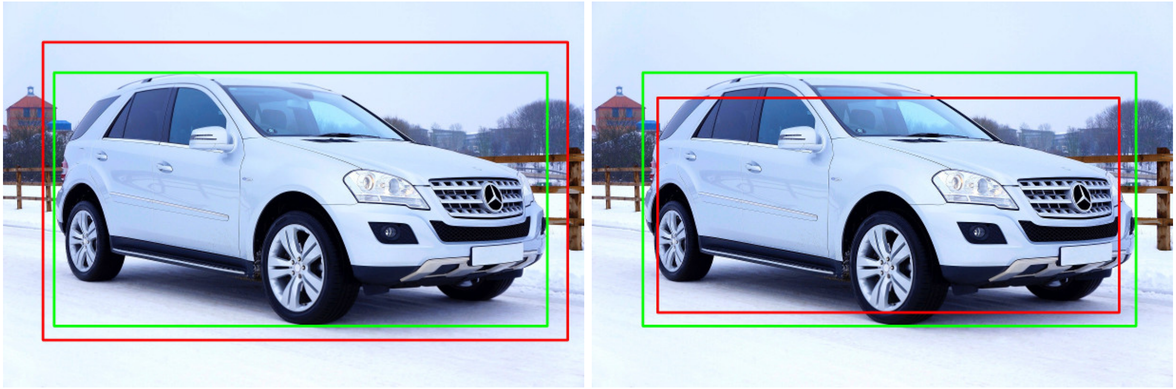
Figure 2.1: IoU is the division of the overlapping area between the bounding boxes by the union area. Image reproduced from `https://www.pyimagesearch.com/`.

The IoU is interesting because penalizes both over- and under-estimated objects, as shown in Figure 2.2. Overestimated bounding boxes might include a large amount of unnecessary information and also increase the processing time of subsequent stages. On the other hand, meaningful parts of the object might be lost in underestimated bounding boxes.

The PASCAL VOC [38] and COCO [39] object detection tasks consider a detection to be correct if the IoU between predicted and ground truth bounding boxes exceed 0.5. As stated in [38], this threshold was set deliberately low to account for inaccuracies in bounding boxes in the training data, for example, defining the bounding box for a highly non-convex object (e.g., a person with arms and legs spread) is somewhat subjective.

2.2 Deep Learning

Problems that are intellectually difficult for human beings but relatively straightforward for computers (e.g., problems that can be described by a list of formal/mathematical rules) were



(a) Overestimated vehicle (IoU = 0.8) (b) Underestimated vehicle (IoU = 0.8)

Figure 2.2: An illustration of two bounding boxes with the same IoU with the ground truth. The predicted position and ground truth are outlined in red and green, respectively. Image (without the bounding boxes) reproduced from `https://www.pexels.com`.

rapidly tackled in the early days of artificial intelligence. On the other hand, problems that humans solve intuitively, that feel automatic, such as telling the difference between pictures of cats and dogs are very challenging for artificial intelligence [52, 53].

The ability to process natural data in their raw form (such as the pixel values of an image) was limited in conventional machine learning techniques. For many years, the development of machine learning systems required a lot of effort and considerable domain expertise to transform raw data into feature vectors with both discriminative and informative features [31]. It should be noted that the choice of data representation (or features) directly determines the performance of machine learning methods [54], as demonstrated in Figure 2.3.

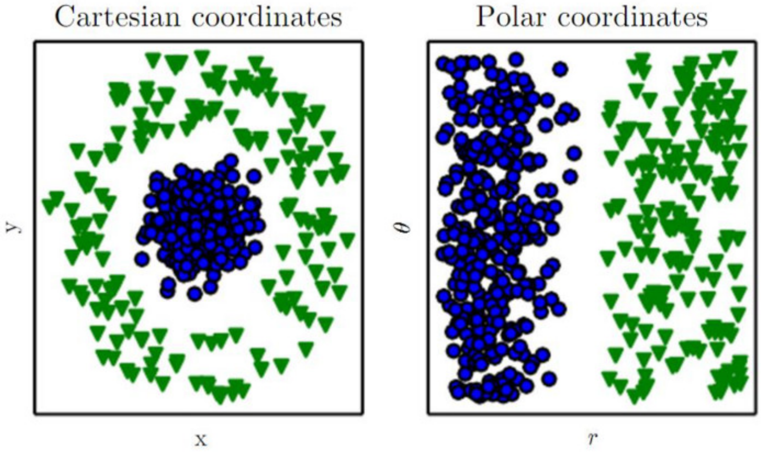


Figure 2.3: An example of different data representations. It is impossible to draw a straight line that separates two categories of data when representing them using Cartesian coordinates. On the other hand, this task becomes very simple when using Polar coordinates. Image reproduced from `http://www.deeplearningbook.org/`.

One solution to this problem is *representation learning*, which is a set of methods where the representations needed for detection or classification are automatically discovered from raw data [31]. In other words, instead of telling the system what a cat or dog looks like (through feature vectors), we provide as input a lot of images (i.e., millions or hundreds of thousands) of cats and dogs and let the system learns by itself to associate patterns and images with the correct label [53]. A string of empirical successes has been achieved both in academia and in industry with the growing interest of the scientific community on representation learning [54].

The central problem in representation learning is that it can be very difficult to extract such high-level, abstract features from raw data. *Deep learning* solves this problem by introducing representations that are expressed in terms of other, simpler representations [52]. An illustration of a deep learning model is shown in Figure 2.4. As can be seen, features regarding the presence or absence of edges at particular orientations and locations in the image are learned in the first representation layer. Next, corners and contours (i.e., collections of edges) are detected in the second layer. The third layer is where parts of objects are found, by locating specific collections of contours and corners. Finally, the subsequent layers would detect specific objects as combinations of these parts [31, 52]. As noted by LeCun et al. [31], the key aspect of deep learning is that these layers of features are learned from data using a general-purpose learning procedure, and thus it requires very little engineering by hand.

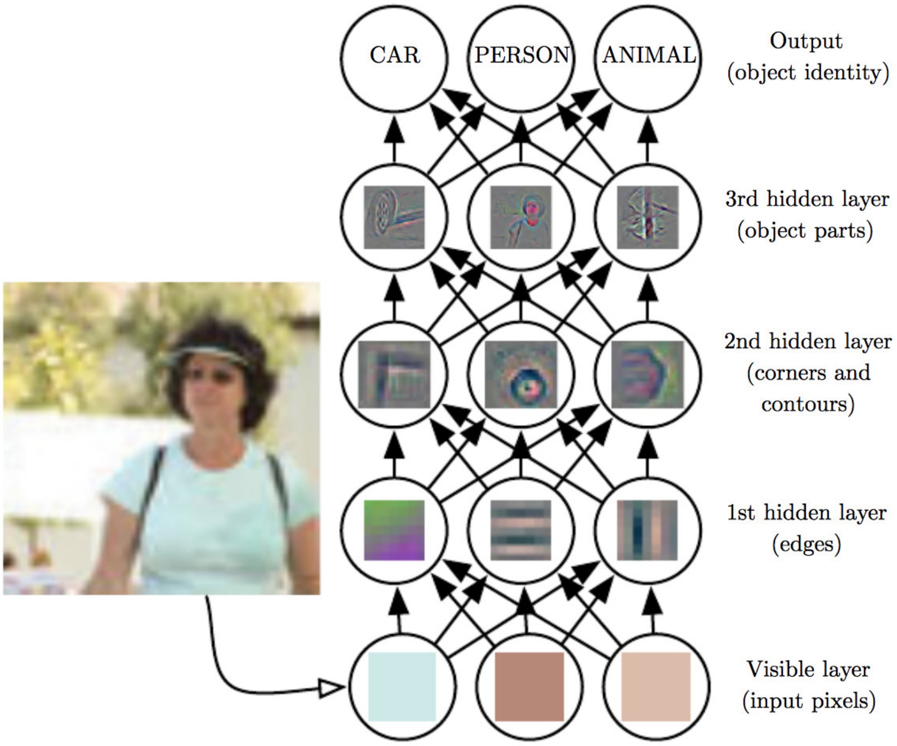


Figure 2.4: An illustration of a deep learning model. First, low-level features such as edges and curves are found, and then more abstracts concepts are built through a series of layers. Image reproduced from http://www.deeplearningbook.org/.

At first, deep learning approaches were mainly employed for the handwritten digits recognition problem, breaking the supremacy of Support Vector Machines (SVMs) in the MNIST dataset[1]. The focus shifted progressively to object recognition in natural images, increasingly attracting the attention of the scientific community since the breakthrough achieved by Krizhevsky et al. [55] on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC)[2], bringing down the state-of-the-art error rate from 26.2% to 15.3% [54].

In addition to the outstanding results achieved in several applications through deep learning, there are two other reasons for its success [31, 56]. First, the dramatically increased chip processing abilities (e.g., GPUs). Second, the fact that deep learning can easily take advantage of increases in the amount of available computation and data since it requires very little engineering by hand.

[1] The MNIST dataset is described at http://yann.lecun.com/exdb/mnist/.

[2] The ILSVRC challenge is described at http://www.image-net.org/challenges/LSVRC/.

2.2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs), also known as Convolutional Networks and ConvNets, are one of the most popular deep neural network architectures. These networks are designed to process data that have a known, grid-like topology, for example a color image composed of three 2-D arrays containing pixel intensities in the three color channels [31, 52]. It is worth noting that the impressive results reported by Krizhevsky et al. [55] were obtained using CNNs.

All CNNs perform a kind of linear operation called *convolution* (hence the name) in at least one of their layers [52]. The basic building blocks of CNNs are convolutions, pooling (downsampling) operators, activation functions [e.g., Rectified Linear Unit (ReLU)] and fully connected layers, which are essentially similar to hidden layers of a Multilayer Perceptron (MLP) [57]. Figure 2.5 shows an example of a CNN. Each one of those building blocks will be described throughout this section.

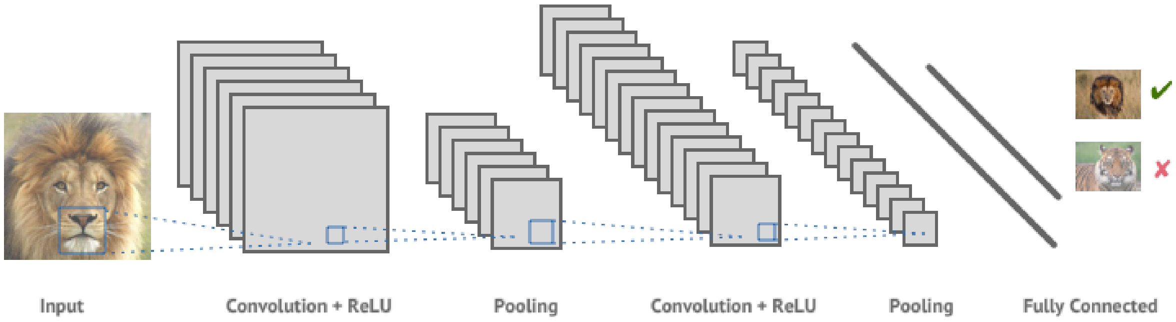


Figure 2.5: An example of a CNN, which consists of convolutional layers, activation functions and pooling layers, followed by a set of fully connected layers. Image reproduced from [58].

Convolutional Layer

The main building blocks of CNNs are the convolutional layers, which are composed of a set of *filters* (or kernels), each to be applied to the entire array of pixel values. Each filter is a matrix of weights (or values) which can be considered as a feature identifier (e.g., straight edges, simple colors, and curves). The filters produce what can be seen as an affine transformation of the input image. Each filter is slid (or convolved) around the input image with the values in the filter being multiplied by the original pixel values of the image [57]. An example of 2-D convolution is shown in Figure 2.6.

Each region that the filter processes is called *local receptive field* and an output value (pixel) is a combination of the input pixels in this local receptive field, as shown in Figure 2.7. That makes the convolutional layer different from layers of an MLP for example, where each neuron produces a single output based on all values from the previous layer (see Figure 2.8) [57].

An important aspect of CNNs is that the filter weights are shared across receptive fields, significantly reducing the number of weights that the network has to learn. As stated by LeCun et al. [31], if a feature can appear in one part of the image, it could appear anywhere, hence the idea of filters at different locations sharing the same weights and detecting the same pattern in different parts of the array.

Note that convolution is not naturally equivariant to some other transformations, such as changes in the scale or rotation of an image. Therefore, other mechanisms are necessary for handling these kinds of transformations [52].

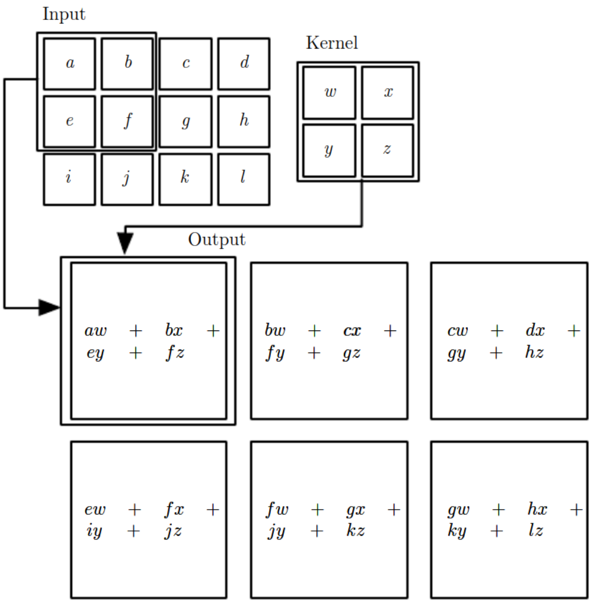


Figure 2.6: An example of 2-D convolution. The boxes with arrows were drew to indicate how the upper-left element of the output tensor is formed by applying the kernel to the corresponding upper-left region of the input tensor. Image reproduced from [52].

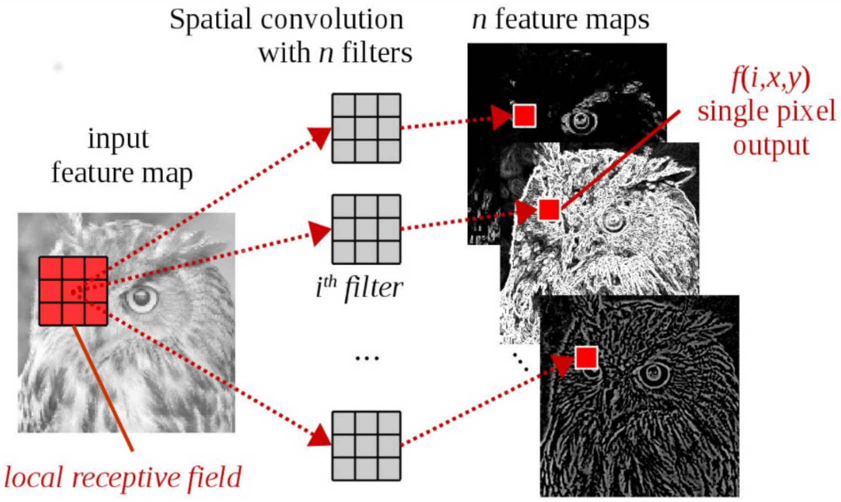
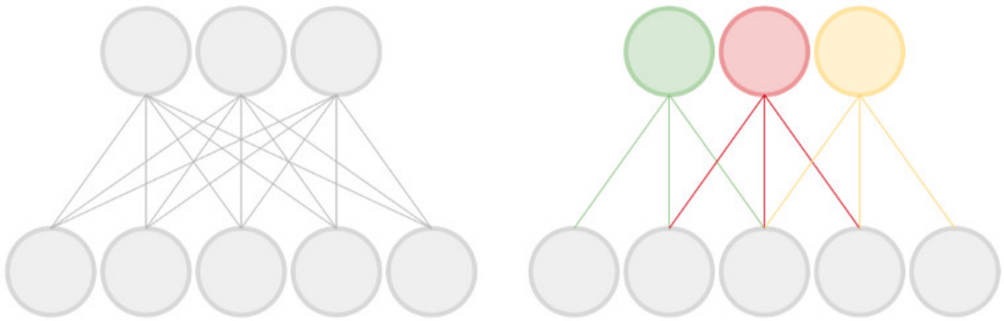


Figure 2.7: The convolution process. A convolution processes local information centred in each position (x, y): this region is called local receptive field, whose values are used as input by some filter i with weights w_i in order to produce a single point (pixel) in the output feature map $f(i, x, y)$. Image reproduced from [57].

Activation Function

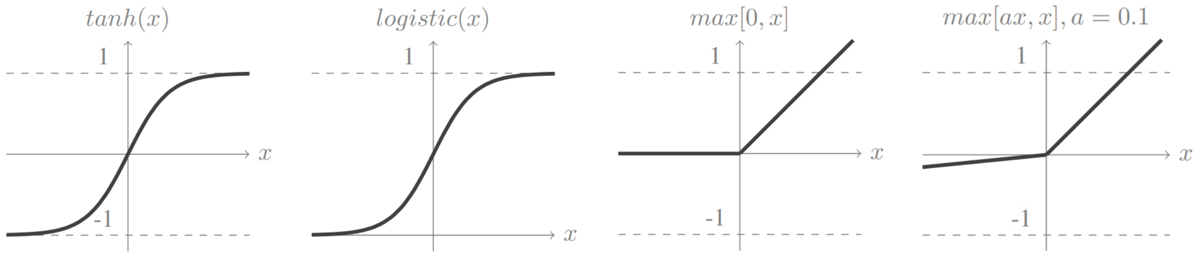
In order to go from one layer to the next, a set of units compute a weighted sum of their inputs from the previous layer and pass the result through an activation function [31]. In contrast to the use of a sigmoid function such as the logistic or hyperbolic tangent in MLPs, the *Rectified Linear Unit* (ReLU) is often used in CNNs after convolutional or fully connected layers [57]. Figure 2.9 shows plots of these functions.

Although sigmoid functions are commonly used in neural networks, their limitations are well known. For example, it is slow to learn the whole network due to weak gradients when the units are close to saturation in both directions [56]. Deep CNNs with ReLUs train several times faster than their equivalents with sigmoid functions [55]. The *Leaky* ReLU allows for a



(a) Fully connected layer (b) Convolutional layer

Figure 2.8: Comparison between fully connected and convolutional layers. In a fully connected layer, each unit is connected to all units of the previous layers. In a convolutional layer, on the other hand, each unit is connected to a constant number of units in a local region of the previous layer. Image reproduced from `https://www.quora.com/what-is-the-difference-between-a-convolutional-neural-network-and-a-multilayer-perceptron`.



(a) Hyperbolic tangent (b) Logistic (c) ReLU (d) Parametric ReLU (PReLU)

Figure 2.9: Activation functions. (a) and (b) are often used in MLP networks, while (c) and (d) are more common in CNNs. Note that a PReLU (d) with $a = 0.01$ is equivalent to Leaky ReLU. Image reproduced from [57].

small, non-zero gradient when the unit is saturated and not active. Maas et al. [59] observed that the non-zero gradient does not substantially affect training optimization and that deep networks with Leaky ReLUs converge slightly faster.

In addition to the innovations in better architectures of deep learning models, there is also a growing body of work on developing and implementing better nonlinear units [56].

Pooling

In addition to convolutions and activation functions, *pooling* operations make up another important building block in CNNs. Pooling operations reduce the size of feature maps by using some function to summarize subregions, such as taking the average or the maximum value of the contributing features [60]. Although much better linear discrimination performance was achieved with max pooling compared to average pooling in [61], the same research group showed in [62] that depending on the data and features, either max or average pooling may perform best. Then, in this section, we focus on the max-pooling operator since it is the most frequently used [57].

The role of the pooling layer is to merge semantically similar features into one, allowing representations to vary very little when elements in the previous layer vary in position and appearance [31]. In other words, the use of pooling can be viewed as adding an infinitely strong prior that the function the layer learns must be invariant to small translations [52]. See Figure 2.10 for an example of how max pooling (with stride = 1) works.

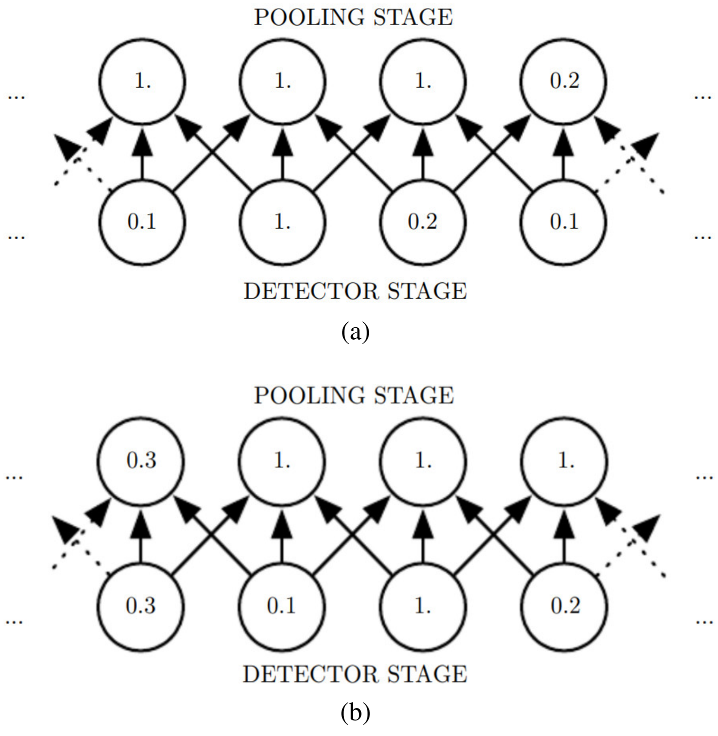


Figure 2.10: Max pooling introduces invariance. (a) shows a view of the middle of the output of a convolutional layer, and (b) shows a view of the same network, after the input has been shifted to the right by one pixel. The bottom row shows the outputs of the activation function. The top row shows the outputs of max pooling, with a stride of one pixel between pooling regions and a pooling region width of three pixels. Observe that every value in the bottom row has changed, but only half of the values in the top row have changed, because the max-pooling units are only sensitive to the maximum value in the neighborhood, not its exact location. Image reproduced from `http://www.deeplearningbook.org/`.

It is possible to use fewer pooling units than detector units (see Figure 2.11) since pooling summarizes the responses over a whole neighborhood. In this way, the computational efficiency of the network is improved because the next layer has fewer inputs to process. When the number of parameters in the next layer is a function of its input size (e.g., the next layer is fully connected and based on matrix multiplication) this reduction in the input size can also result in improved statistical efficiency and reduced memory requirements for storing the parameters [52].

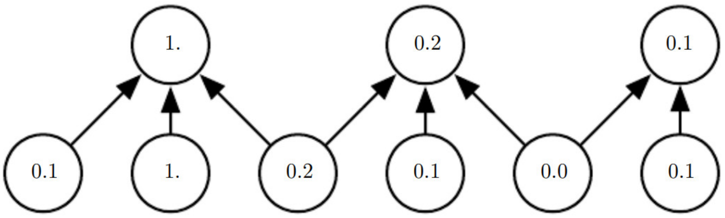


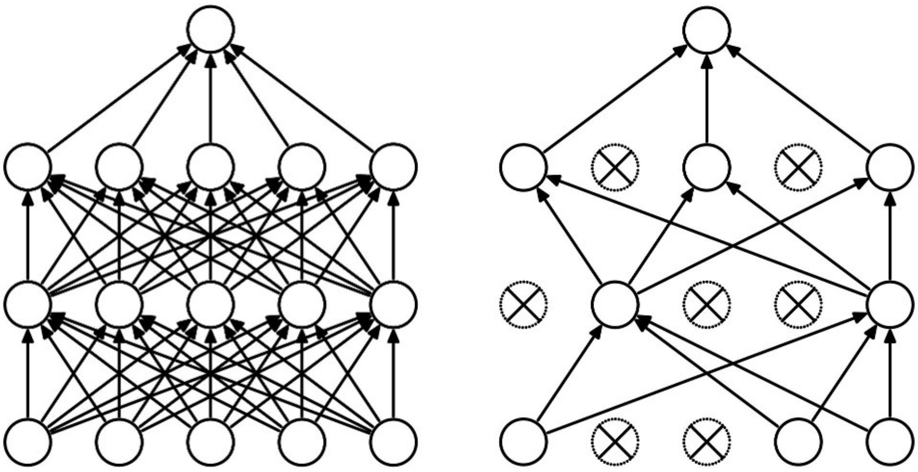
Figure 2.11: Max pooling with downsampling. When using stride = 2 between pools, the representation size is reduced by a factor of two, which reduces the computational and statistical burden on the next layer. Note that the rightmost pooling region has a smaller size, but must be included if we do not want to ignore some of the detector units. Image reproduced from `http://www.deeplearningbook.org/`.

It should be noted that generative models such as auto-encoders and Generative Adversarial Networks (GANs) shown to be harder to train with pooling layers [57]. Therefore, pooling layers might be avoided in some neural network architectures.

Fully Connected Layers and Regularization

Conventional CNNs perform convolution in the lower layers of the network. For classification, the feature maps of the last convolutional layer are vectorized and fed into fully connected layers followed by a softmax logistic regression layer [63].

However, the fully connected layers are prone to overfitting, thus hampering the generalization ability of the overall network [63]. In this sense, a technique called *dropout* [64] was introduced to limit co-adaptation. It operates as follows. On each training instance, each hidden unit is randomly omitted with a fixed probability (e.g., $p = 0.5$) [56]. The neurons that are "dropped out" do not contribute to the forward pass and do not participate in backpropagation, as illustrated in Figure 2.12. Thus, the neural network samples a different architecture every time an input is presented, but all these architectures share weights [55].



(a) Standard neural network

(b) After applying dropout

Figure 2.12: An illustration of dropout regularization. (a) shows a standard neural network with 2 hidden layers, and (b) shows an example of a thinned network produced by applying dropout to the network on (a). Image reproduced from [64].

Dropout is turned off in the test stage and the activations are rescaled by p to compensate those activations that were dropped during the training stage [57]. The benefits of dropout regularization for training deep neural networks are to make a hidden unit act strongly by itself without relying on others and to serve a way to do model averaging of different networks. These benefits are most pronounced when the training data is limited, or when the network size is disproportionally large with respect to the size of the training data [56].

Deep neural networks involve the composition of several functions or layers. Training these networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change [65]. In other words, the gradient tells how to update each parameter, under the assumption that the other layers do not change. In practice, all layers are updated simultaneously. Hence, unexpected results might happen because many functions composed together were changed simultaneously, using updates that were computed under the assumption that the other functions would remain constant [52].

This makes it notoriously hard to train models with saturating nonlinearities. Therefore, the training is slower since it requires lower learning rates and careful parameter initialization [65]. In this direction, Ioffe and Szegedy [65] proposed a regularization technique called *batch normalization* for controlling the distributions of neural network activations, thereby reducing internal covariate shift [66]. Batch normalization is a method of adaptive reparametrization in which the output of each neuron (before application of the nonlinearity) is normalized by the mean and standard deviation of the outputs calculated over the examples in the mini-batch [67]. This effectively decouples each layer's parameters from those of other layers, leading to a better-conditioned optimization problem. Deep neural networks trained with batch normalization converge significantly faster, generalize better and often do not need dropout [52, 57, 66].

2.2.2 Data Augmentation

A huge number of training examples are required to train CNNs since they often have a large set of parameters to be optimized [57]. In practice, the amount of data available is limited. One way to get around this problem is to create fake data and add it to the training set. This process is known as *data augmentation*. It is reasonably straightforward to create new fake data for some machine learning tasks [52].

Images in the same dataset usually have similar illumination conditions, a low variance of rotation, pose, etc. Therefore, one can augment the training dataset using many operations to produce several times more examples [57]. Operations like translating the training images a few pixels in each direction can often greatly improve generalization, even if the model has already been designed to be partially translation invariant by using the convolution and pooling techniques described in the previous section. Many other operations such as rotating or scaling the image have also proven quite effective [52, 57, 68]. In Figure 2.13, we show some LP images generated using the data augmentation technique proposed in [13], which consists of the permutation of the characters on the LPs in addition to random variations of scale, rotation, brightness and cropping.

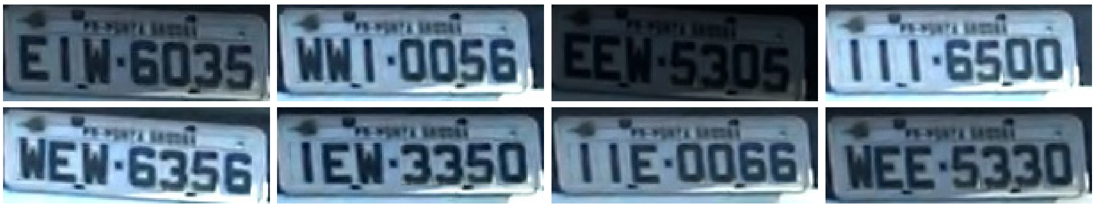


Figure 2.13: An illustration of data augmentation. The image in the upper-left corner is the original and the others were generated automatically.

It is well-known that unbalanced data (usually the case in ALPR) is undesirable for neural network classifiers since the learning of some patterns might be biased. This problem can be addressed with data augmentation, by increasing the number of images of under-represented classes to create a new set of training images, in which each class is equally represented.

It is worth noting that some frameworks (e.g., Darknet [69]) already have built-in data augmentation [29], and one must be careful not to apply transformations that would change the correct class. For example, Optical Character Recognition (OCR) tasks require recognizing the difference between 'b' and 'd' and the difference between '6' and '9', so these cases must be considered before applying horizontal flips and 180° rotations for those tasks [52].

2.3 YOLO

A core problem in computer vision is object detection. The detection task is substantially more complex than the classification one [56]. Detection pipelines generally start by extracting features from input images in a sliding window fashion or on some subset of regions in the image. Then, classifiers are used to identify objects in the feature space [29].

Unlike sliding window and region proposal-based techniques, *YOLO* [29] is a system which reframes object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Thus, YOLO reasons globally about the image when making predictions. This is achieved by unifying the separate components (e.g., generating potential bounding boxes, running a classifier on these boxes, and post-processing to refine the bounding boxes) of object detection into a single neural network [29].

As can be seen in Figure 2.14, YOLO divides the input image into an $S \times S$ grid. Each cell predicts B bounding boxes (x,y,w,h) and confidence scores for those boxes. These scores reflect how likely the bounding box contains an object (i..e, the objectness [70]) and how accurate is the bounding box. Additionally, each grid cell predicts one set C of class probabilities, regardless of the number of bounding boxes B. Class-specific confidence scores are generated by multiplying the individual bounding box's objectness score by the class probabilities [29].

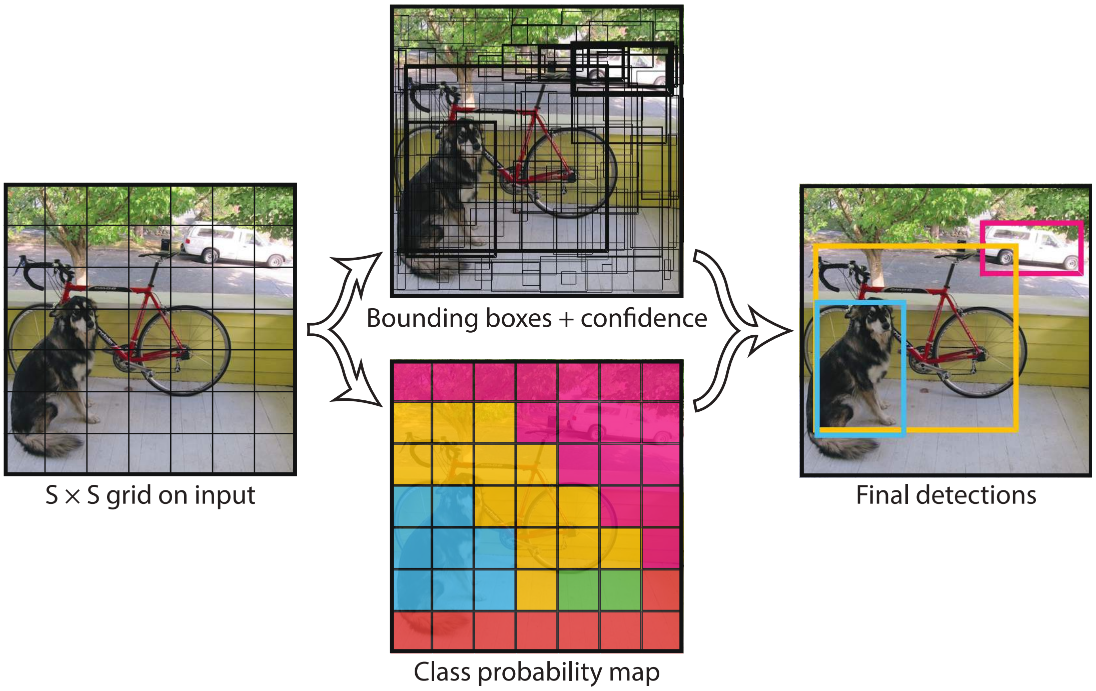


Figure 2.14: An overview of YOLO. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes (x,y,w,h), confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B \times 5 + C)$ tensor. Image reproduced from [29].

Since there are many bounding boxes predicted with very low confidence values for any class, a simple threshold is applied to reduce FPs. Furthermore, it is not always clear on which grid cell an object falls into and the objects can be well located by multiple cells (especially large objects or objects near the border of multiple cells). Therefore, a Non-Maximum Suppression (NMS) algorithm is employed to eliminate redundant detections [29].

YOLO has 24 convolutional layers followed by 2 fully connected layers (see Figure 2.15). The first 20 convolutional layers followed by both an average pooling layer and a fully connected layer were used for pre-training the classification task on the ImageNet 1000-class competition dataset. Then, four convolutional layers and two fully connected layers were added for detection. A linear activation function is used for the final layer and all other layers use Leaky ReLUs. In addition, the input resolution of the network was increased from 224 × 224 to 448 × 448 as detection usually requires fine-grained information. Lastly, there is a dropout layer with rate = 0.5 after the first connected layer to prevent co-adaptation between layers [29].

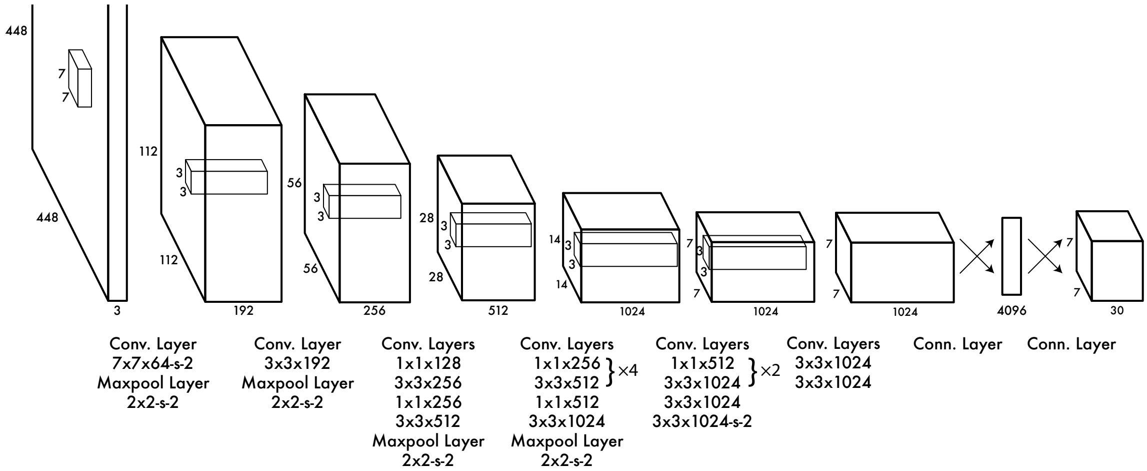


Figure 2.15: The YOLO architecture. It has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the feature space from preceding layers. The convolutional layers are pre-trained at half the resolution (224×224) and then at double the resolution for detection. Image reproduced from [29].

A fast version of YOLO, called Fast-YOLO (or Tiny-YOLO), was also designed in [29]. All training and testing parameters are the same, however, Fast-YOLO uses fewer convolutional layers (9 instead of 24) and fewer filters in those layers, as detailed in Table 2.1.

Table 2.1: The Fast-YOLO architecture.

#	Layer	Filters	Size	Input	Output
0	conv	16	$3 \times 3/1$	$448 \times 448 \times 3$	$448 \times 448 \times 16$
1	max		$2 \times 2/2$	$448 \times 448 \times 16$	$224 \times 224 \times 16$
2	conv	32	$3 \times 3/1$	$224 \times 224 \times 16$	$224 \times 224 \times 32$
3	max		$2 \times 2/2$	$224 \times 224 \times 32$	$112 \times 112 \times 32$
4	conv	64	$3 \times 3/1$	$112 \times 112 \times 32$	$112 \times 112 \times 64$
5	max		$2 \times 2/2$	$112 \times 112 \times 64$	$56 \times 56 \times 64$
6	conv	128	$3 \times 3/1$	$56 \times 56 \times 64$	$56 \times 56 \times 128$
7	max		$2 \times 2/2$	$56 \times 56 \times 128$	$28 \times 28 \times 128$
8	conv	256	$3 \times 3/1$	$28 \times 28 \times 128$	$28 \times 28 \times 256$
9	max		$2 \times 2/2$	$28 \times 28 \times 256$	$14 \times 14 \times 256$
10	conv	512	$3 \times 3/1$	$14 \times 14 \times 256$	$14 \times 14 \times 512$
11	max		$2 \times 2/2$	$14 \times 14 \times 512$	$7 \times 7 \times 512$
12	conv	1024	$3 \times 3/1$	$7 \times 7 \times 512$	$7 \times 7 \times 1024$
13	conv	256	$3 \times 3/1$	$7 \times 7 \times 1024$	$7 \times 7 \times 256$
14	connected			12544	1470
15	detection				

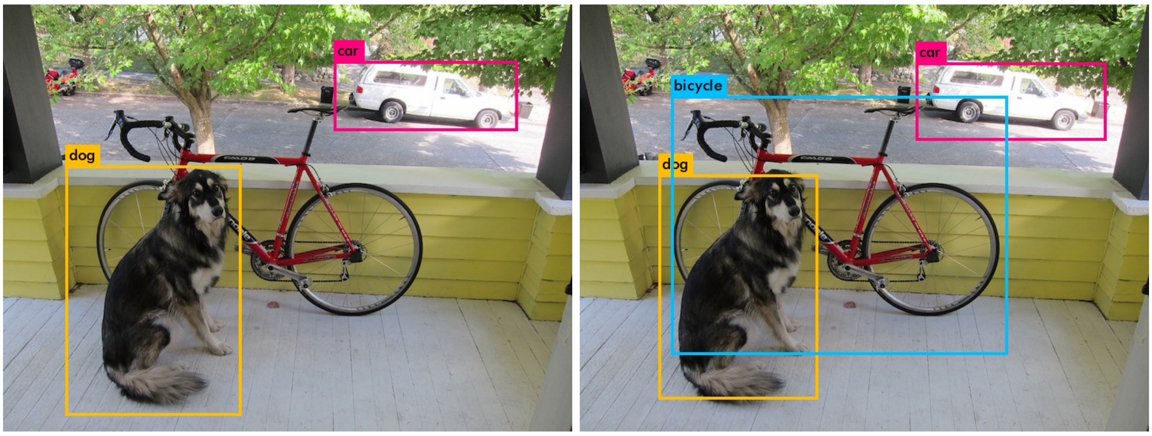
Despite being much smaller, Fast-YOLO is still able to detect some objects quite precisely. In Figure 2.16, we show the predictions obtained in the same image with Fast-YOLO and YOLO when using default parameters.

Impressive results were attained in the Pascal VOC detection dataset with both YOLO and Fast-YOLO models, as shown in Table 2.2. YOLO processes images in real time at 45 FPS, whereas Fast-YOLO processes 155 FPS while still achieving double the mAP value of other real-time detectors. Note that better mAP values were obtained with other approaches (e.g., Fast R-CNN and Faster R-CNN), but these approaches are still far from real time. These results were reported in [29].

Real-Time Detectors	Train	mAP	FPS
100Hz DPM	2007	16.0	100
30Hz DPM	2007	26.1	30
Fast-YOLO	2007 + 2012	52.7	**155**
YOLO	2007 + 2012	**63.4**	45

Less Than Real-Time	Train	mAP	FPS
Fastest DPM	2007	30.4	15
R-CNN Minus R	2007	53.5	6
Fast R-CNN	2007 + 2012	70.0	0.5
Faster R-CNN VGG-16	2007 + 2012	73.2	7
Faster R-CNN ZF	2007 + 2012	62.1	18

Table 2.2: A comparison of object detectors on Pascal VOC 2007. Fast-YOLO is the fastest detector and is still twice as accurate as any other real-time detector. YOLO is 10.7% mAP more accurate than Fast-YOLO while still well above real time in speed. Results reproduced from [29].

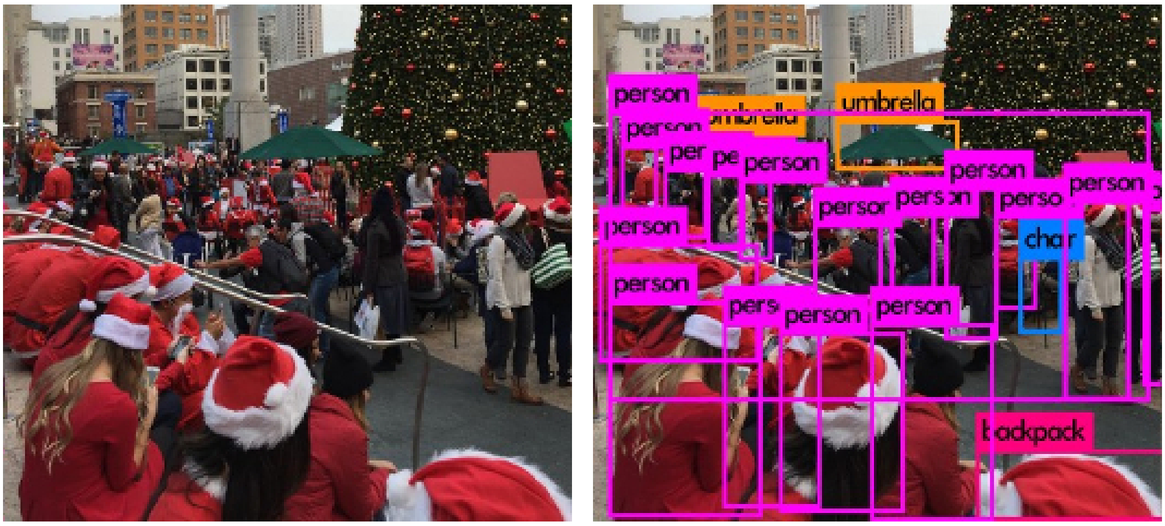


(a) Fast-YOLO

(b) YOLO

Figure 2.16: Objects predicted by both (a) Fast-YOLO and (b) YOLO in the same image. The main difference is that the Fast-YOLO model was not able to detect the bicycle. Image (without the bounding boxes) reproduced from `https://pjreddie.com/darknet/yolo`.

Despite the promising results achieved, YOLO has some limitations. First, each grid cell only predicts two boxes (by default) and can only have one class. This limits how close detected objects can be, as shown in Figure 2.17. Second, YOLO struggles to generalize to objects in new or unusual aspect ratios or configurations, as it learns to predict bounding boxes from data. This is not a major problem for ALPR since vehicles, LPs and its characters are standardized and have very similar configurations. Lastly, errors in small and large bounding boxes are treated equally in the loss function. However, a small error in a large box is generally benign but a small error in a small box has a much greater effect on IoU [29, 71].



(a) Original image

(b) Objects predicted by YOLO

Figure 2.17: A limitation of YOLO. It might miss objects that are too close. In (a), there are 9 individuals in the lower-left corner, but only 5 were detected by YOLO, as can be seen in (b). Image reproduced from [71].

2.3.1 YOLOv2

As pointed out in [36], YOLO has higher localization errors and the recall is lower when compared to some detection systems proposed later. YOLOv2 [36] is the second version of YOLO with the objective of significantly improving its accuracy while making it faster. Here we describe the concepts introduced in YOLOv2 and their impact on the mAP obtained on Pascal VOC 2007.

A new classification model, called Darknet-19, is used as the base of YOLOv2. As can be seen in Table 2.3, the model has 19 convolutional layers (hence the name) and 5 max-pooling layers. In short, the model consists of mostly 3×3 filters, and the number of channels is doubled after every pooling step. As in [63], global average pooling is used to make predictions, and 1×1 filters are employed to compress the feature representation between 3×3 convolutions [36].

Table 2.3: The Darknet-19 classification model, used as the base of YOLOv2. Table adapted from [36].

#	Layer	Filters	Size	Input	Output
0	conv	32	$3 \times 3/1$	$448 \times 448 \times 3$	$448 \times 448 \times 32$
1	max		$2 \times 2/2$	$448 \times 448 \times 32$	$224 \times 224 \times 32$
2	conv	64	$3 \times 3/1$	$224 \times 224 \times 32$	$224 \times 224 \times 64$
3	max		$2 \times 2/2$	$224 \times 224 \times 64$	$112 \times 112 \times 64$
4	conv	128	$3 \times 3/1$	$112 \times 112 \times 64$	$112 \times 112 \times 128$
5	conv	64	$1 \times 1/1$	$112 \times 112 \times 128$	$112 \times 112 \times 64$
6	conv	128	$3 \times 3/1$	$112 \times 112 \times 64$	$112 \times 112 \times 128$
7	max		$2 \times 2/2$	$112 \times 112 \times 128$	$56 \times 56 \times 128$
8	conv	256	$3 \times 3/1$	$56 \times 56 \times 128$	$56 \times 56 \times 256$
9	conv	128	$1 \times 1/1$	$56 \times 56 \times 256$	$56 \times 56 \times 128$
10	conv	256	$3 \times 3/1$	$56 \times 56 \times 128$	$56 \times 56 \times 256$
11	max		$2 \times 2/2$	$56 \times 56 \times 256$	$28 \times 28 \times 256$
12	conv	512	$3 \times 3/1$	$28 \times 28 \times 256$	$28 \times 28 \times 512$
13	conv	256	$1 \times 1/1$	$28 \times 28 \times 512$	$28 \times 28 \times 256$
14	conv	512	$3 \times 3/1$	$28 \times 28 \times 256$	$28 \times 28 \times 512$
15	conv	256	$1 \times 1/1$	$28 \times 28 \times 512$	$28 \times 28 \times 256$
16	conv	512	$3 \times 3/1$	$28 \times 28 \times 256$	$28 \times 28 \times 512$
17	max		$2 \times 2/2$	$28 \times 28 \times 512$	$14 \times 14 \times 512$
18	conv	1024	$3 \times 3/1$	$14 \times 14 \times 512$	$14 \times 14 \times 1024$
19	conv	512	$1 \times 1/1$	$14 \times 14 \times 1024$	$14 \times 14 \times 512$
20	conv	1024	$3 \times 3/1$	$14 \times 14 \times 512$	$14 \times 14 \times 1024$
21	conv	512	$1 \times 1/1$	$14 \times 14 \times 1024$	$14 \times 14 \times 512$
22	conv	1024	$3 \times 3/1$	$14 \times 14 \times 512$	$14 \times 14 \times 1024$
23	conv	1000	$1 \times 1/1$	$14 \times 14 \times 1024$	$14 \times 14 \times 1000$
24	avg			$14 \times 14 \times 1000$	1000
25	softmax				

An improvement of more than 2% in mAP was attained by adding *batch normalization* on all convolutional layers. In this way, dropout was removed without overfitting. Furthermore, an increase of almost 4% mAP was achieved through *high-resolution classification*, that is, after training the classification network on images of 224×224 pixels (as in YOLO) the network was tuned at the full 448×448 resolution for 10 epochs on ImageNet [36].

The fully connected layers from YOLO were removed and YOLOv2 uses *anchor boxes* (or priors) to predict bounding boxes. Thus, a class and objectness are predicted for each anchor box [36]. For a better understanding, consider that 5 anchor boxes with particular aspect ratios are created, as shown in Figure 2.18. Instead of predicting 5 arbitrary bounding boxes, YOLOv2 predicts offsets to each of these anchor boxes. Thus, the network does not predict the final size of the object but only adjusts the size of the nearest anchor to the size of the object [36, 71].

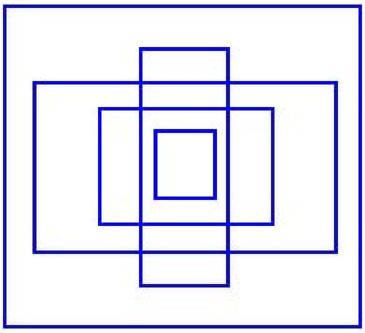


Figure 2.18: Examples of anchors boxes. Image reproduced from [71].

Predicting offsets instead of coordinates simplifies the problem and makes it easier for the network to learn [36]. As illustrated in Figure 2.19, the diversity of the predictions is maintained and each prediction focuses on a specific shape. In the real-life domain, the bounding boxes are not arbitrary, for example, cars have very similar shapes, which are different from those of pedestrians. Therefore, the training will be more stable when starting with diverse guesses (i.e., anchor boxes) that are common to real-life objects [71].

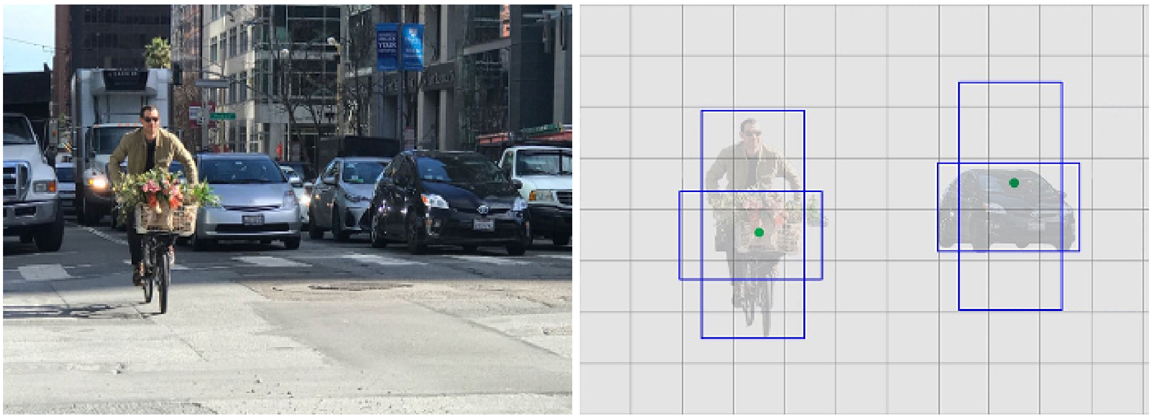
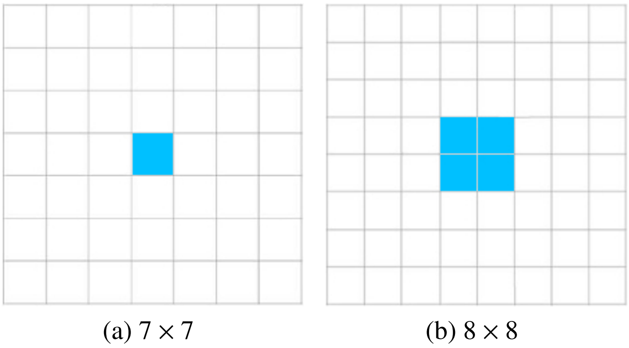


Figure 2.19: An illustration of two objects (a cyclist and a car) and two anchor boxes. Observe that each anchor box focuses on a specific shape. Images reproduced from [71].

Instead of choosing the anchor boxes by hand, YOLOv2 runs k-means clustering on the training set bounding boxes to automatically find good priors. In [36], the authors used $k = 5$ as a good trade-off between recall and model complexity. The clusters, called as *dimension priors*, are significantly different than hand-picked anchor boxes. An important piece of information is that YOLOv2 takes a long time to stabilize with random initialization, since any anchor box might end up at any point in the image, regardless of what location predicted the box. In this sense, YOLOv2 predicts location coordinates relative to the location of the grid cell, i.e. the network predicts 5 bounding boxes for each cell in the output feature map. A 5% mAP increase was achieved when using dimension priors instead of hand-picked anchor boxes [36].

In YOLOv2, the input image size is 416 × 416 (instead of 448 × 448) so there is only one center cell in the feature map. The center of the image is usually occupied by objects, so it is better to have a single location at the center to predict these objects instead of four locations that are all nearby (see Figure 2.20). Also, one pooling layer was eliminated in order to make the output of the convolutional layers higher resolution. Then, as YOLOv2 downsamples the image by a factor of 32, an input image of 416 × 416 generates an output feature map of 13 × 13 [36].



(a) 7×7

(b) 8×8

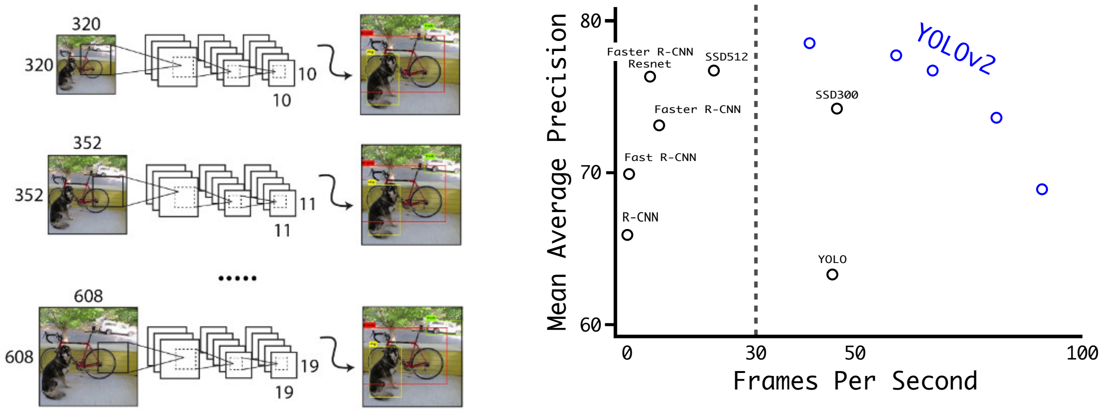
Figure 2.20: Center cells on both (a) odd and (b) even output feature maps. It is better to have a single location at the center than four locations that are all nearby. Images adapted from [71].

The YOLOv2 architecture is shown in Table 2.4. Note that some modifications were made to the Darknet-19 model for detection. The last convolutional layer was removed and three 3×3 convolutional layers with 1024 filters each were added, as well as a 1×1 convolutional layer. While a 13×13 feature map is sufficient for large objects, YOLOv2 might benefit from fine-grained features for detecting smaller objects. In this way, a *pass-through* layer was added to brings features from an earlier layer at 26×26 resolution. It reshapes the $26 \times 26 \times 512$ feature map into a $13 \times 13 \times 2048$ feature map, which can be concatenated with the original features. Running the detector on this expanded feature map gives a modest 1% mAP increase [36].

Table 2.4: The YOLOv2 architecture.

#	Layer	Filters	Size	Input	Output
0	conv	32	3 × 3/1	416 × 416 × 3	416 × 416 × 32
1	max		2 × 2/2	416 × 416 × 32	208 × 208 × 32
2	conv	64	3 × 3/1	208 × 208 × 32	208 × 208 × 64
3	max		2 × 2/2	208 × 208 × 64	104 × 104 × 64
4	conv	128	3 × 3/1	104 × 104 × 64	104 × 104 × 128
5	conv	64	1 × 1/1	104 × 104 × 128	104 × 104 × 64
6	conv	128	3 × 3/1	104 × 104 × 64	104 × 104 × 128
7	max		2 × 2/2	104 × 104 × 128	52 × 52 × 128
8	conv	256	3 × 3/1	52 × 52 × 128	52 × 52 × 256
9	conv	128	1 × 1/1	52 × 52 × 256	52 × 52 × 128
10	conv	256	3 × 3/1	52 × 52 × 128	52 × 52 × 256
11	max		2 × 2/2	52 × 52 × 256	26 × 26 × 256
12	conv	512	3 × 3/1	26 × 26 × 256	26 × 26 × 512
13	conv	256	1 × 1/1	26 × 26 × 512	26 × 26 × 256
14	conv	512	3 × 3/1	26 × 26 × 256	26 × 26 × 512
15	conv	256	1 × 1/1	26 × 26 × 512	26 × 26 × 256
16	conv	512	3 × 3/1	26 × 26 × 256	26 × 26 × 512
17	max		2 × 2/2	26 × 26 × 512	13 × 13 × 512
18	conv	1024	3 × 3/1	13 × 13 × 512	13 × 13 × 1024
19	conv	512	1 × 1/1	13 × 13 × 1024	13 × 13 × 512
20	conv	1024	3 × 3/1	13 × 13 × 512	13 × 13 × 1024
21	conv	512	1 × 1/1	13 × 13 × 1024	13 × 13 × 512
22	conv	1024	3 × 3/1	13 × 13 × 512	13 × 13 × 1024
23	conv	1024	3 × 3/1	13 × 13 × 1024	13 × 13 × 1024
24	conv	1024	3 × 3/1	13 × 13 × 1024	13 × 13 × 1024
25	route [16]				
26	reorg		/2	26 × 26 × 512	13 × 13 × 2048
27	route [26, 24]				
28	conv	1024	3 × 3/1	13 × 13 × 3072	13 × 13 × 1024
29	conv	425	1 × 1/1	13 × 13 × 1024	13 × 13 × 425
30	detection				

Another concept introduced in YOLOv2 is *multi-scale training* (see Figure 2.21a). The original YOLO uses a fixed input resolution of 448×448 pixels. Since YOLOv2 uses only convolutional and pooling layers, it can be resized on the fly. In short, every 10 batches the network randomly chooses a new image dimension size from 320×320 to 608×608 pixels (these are the default values). This approach forces the network to learn to predict well across a variety of input dimensions, increasing the mAP by 1.4% [36].



(a) Multi-scale training

(b) Accuracy and speed on Pascal VOC 2007

Figure 2.21: YOLO's multi-scale training. YOLOv2 can be resized on the fly, as it uses only convolutional and pooling layers. (a) illustrates multi-scale training and (b) shows the trade-off between performance and accuracy when running YOLOv2 at different input resolutions. Images reproduced from [36].

According to Figure 2.21b, YOLOv2 operates as a cheap, fairly accurate detector at low resolutions. At high resolution, on the other hand, YOLOv2 is a very accurate detector with

Table 2.5: The path from YOLO to YOLOv2. Table reproduced from [36].

	YOLO								YOLOv2
batch normalization?		✓	✓	✓	✓	✓	✓	✓	✓
high-resolution classifier?			✓	✓	✓	✓	✓	✓	✓
fully convolutional?				✓	✓	✓	✓	✓	✓
hand-picked anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
pass-through layer?							✓	✓	✓
multi-scale training?								✓	✓
high-resolution detector?									✓
Pascal VOC 2007 mAP (%)	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	78.6

78.6 mAP while still operating above real-time speeds. A summary of the mAP improvements obtained after applying the concepts described in this section can be found in Table 2.5.

It is important to mention that the Fast-YOLOv2 model is basically the Fast-YOLO model with some of the concepts (e.g., batch normalization, dimension priors, higher resolution output, among others) described in this section. Thus, the Fast-YOLOv2 model is often referred to as Fast-YOLO [30, 72].

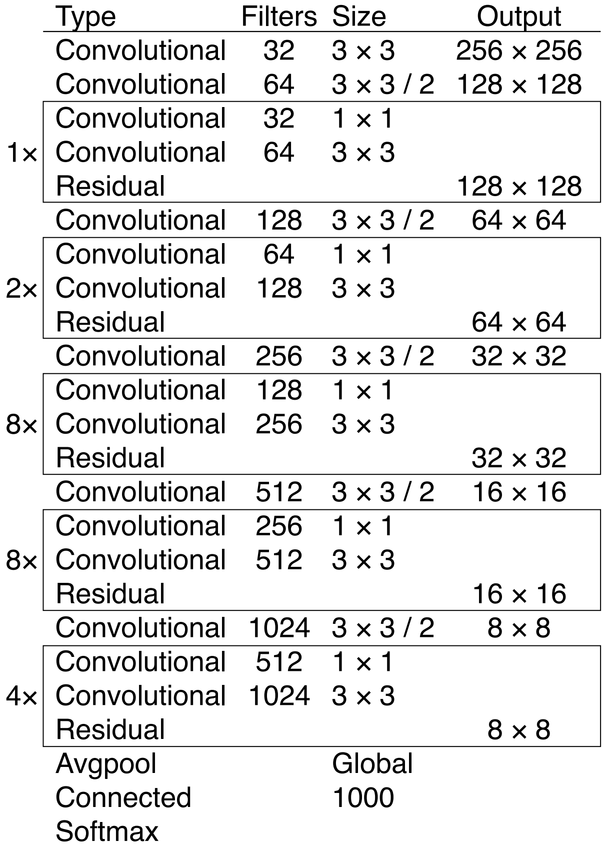
2.3.2 YOLOv3

Although outstanding results have been achieved in object detection using YOLOv2, its architecture still lacks some concepts that are currently essential in most state-of-the-art detectors such as residual blocks, shortcut connections (or skip connections), and upsampling [73]. Based on that, Redmon and Farhadi [37] introduced YOLOv3 (the latest version of YOLO), which uses various tricks to improve training and increase performance, including those concepts mentioned above, multi-scale predictions and a better backbone classifier. In YOLOv3, these tricks are employed along with many of the concepts introduced in YOLOv2 (e.g., anchor boxes, multi-scale training, and batch normalization). In this section, we do not list the impact of each new trick used in YOLOv3 on the mAP obtained on detection tasks, as this information was not provided in [37].

The softmax function, which imposes the assumption that each bounding box has exactly one class, was replaced in YOLOv3 by independent logistic classifiers in order to predict multiple labels for an object. According to [37], this formulation helps when working on more complex domains such as the Open Images dataset [74], in which there are many overlapping labels (e.g., woman and person). YOLOv3 also changes the way it calculates the loss function. The objectness score for each bounding box should be 1 if the bounding box prior overlaps a ground truth object by more than any other prior. YOLOv3 only assigns one bounding box prior for each ground truth object, and thus the prediction is ignored for other priors with overlap greater than a predefined threshold (default = 0.5). Lastly, if a bounding box prior is not assigned to a ground truth object it incurs no loss for coordinate or class predictions, only objectness [37].

A new network, called Darknet-53, is used for feature extraction in YOLOv3. As can be seen in Table 2.6, Darknet-53 has 53 convolutional layers (hence the name) and is a hybrid approach between Darknet-19 and residual networks, as it uses successive 3×3 and 1×1 convolutional layers, has some shortcut connections and is significantly larger [37]. Shortcut connections are those skipping one or more layers. In [75], shortcut connections simply perform identity mapping, and their outputs are added to the outputs of the stacked layers (see Figure 2.22). According to Han et al. [76], Residual Networks (ResNets) [75] leverage the concept of shortcut connections inside a residual block to make it possible to train much deeper network architectures.

Table 2.6: The Darknet-53 backbone classifier, used for feature extraction in YOLOv3. Darknet-53 contains a softmax layer, although it is replaced by independent logistic classifiers in YOLOv3. Table reproduced from [37].



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

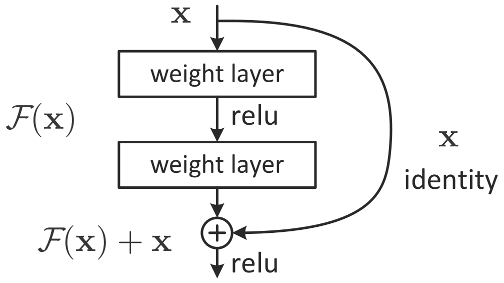


Figure 2.22: A residual block. Image reproduced from [75].

As can be seen in Table 2.7, Darknet-53 is much more powerful than its predecessor (Darknet-19) but still more efficient than ResNet-101 and ResNet-152. YOLOv3's backbone performs on par with state-of-the-art classifiers with fewer Floating-Point Operations (FLOP) and more speed [37]. In addition, Darknet-53 achieves the highest FLOP per second, which means that the network structure better utilizes the GPU, making it more efficient to evaluate and thus faster. According to [37], this is mainly because ResNets have too many layers and are not very efficient. The experiments were carried out on an NVIDIA Titan X at 256 × 256 pixels.

Table 2.7: Comparison of backbones on ImageNet [15]. Accuracy (top-1 and top-5), Billion Floating-Point Operations (BFLOP), BFLOP per second, and FPS for various networks. Table reproduced from [37].

Backbone	Top-1	Top-5	BFLOP	BFLOP/s	FPS
Darknet-19 [36]	74.1	91.8	**7.29**	1246	**171**
ResNet-101 [75]	77.1	93.7	19.7	1039	53
ResNet-152 [75]	**77.6**	**93.8**	29.4	1090	37
Darknet-53 [37]	77.2	**93.8**	18.7	**1457**	78

The most striking feature of YOLOv3 is that it predicts bounding boxes at three different scales, which are precisely given by downsampling the dimensions of the input image by 32, 16 and 8, respectively [73]. For example, using an input size of 416×416 pixels, the bounding boxes are predicted at the following scales: 13×13, 26×26 and 52×52 pixels. The features are extracted from those scales using a concept similar to Feature Pyramid Networks (FPNs) [77]. Note that detections at different layers help address the issue of detecting small objects, a frequent complaint with YOLO and YOLOv2. The 13×13 layer is responsible for detecting large objects, while the 26×26 layer detects medium objects and the 52×52 layer detects smaller objects [73].

The first detection is made by adding several convolutional layers to the base feature extractor (i.e., Darknet-53). The last of those layers predicts a 3D tensor encoding the bounding box, objectness, and class predictions. Based on experiments carried out in the COCO dataset [39], YOLOv3 predicts 3 bounding boxes at each scale so the tensor is $W \times H \times [3 \times (4 + 1 + 80)]$ for the 4 bounding box offsets, 1 objectness prediction, and 80 class predictions. The W and H variables refer to the width and height of the downsampled image, respectively. Next, the feature map from two layers previous is upsampled by 2 and merged with a feature map from earlier in the network, obtaining more meaningful semantic information from the upsampled features and finer-grained information from the earlier feature map. The second detection is made by adding a few more convolutional layers to process this combined feature map, predicting a similar tensor, although now twice the size. Finally, this process is performed one more time to predict bounding boxes for the final scale, which benefit from all the prior computation as well as fine-grained features from early on in the network [37].

As in YOLOv2, YOLOv3 predicts bounding boxes using dimension clusters as anchor boxes. Furthermore, the bounding box priors are still determined using k-means clustering. In [37], 9 clusters were selected based on the COCO dataset: (10×13), (16×30), (33×23), (30×61), (62×45), (59×119), (116×90), (156×198) and (373×326). These 9 priors are grouped into three different groups according to their scale [37,71].

In the latest edition of the COCO object detection task, the AP was averaged over multiple IoU values. Specifically, 10 IoU thresholds were used $[0.05, 0.15, \ldots, 0.95]$. This was a break from tradition, where AP is computed at a single IoU value of 0.5. The ‘old’ metric (used on Pascal VOC) is now referred to as AP_{50}. According to the organizers of the COCO detection task, averaging over IoUs rewards detectors with better localization[3]. Therefore, in [37], YOLOv3 was evaluated based on both metrics on the COCO dataset and compared to recent state-of-the-art methods. Results are presented in Table 2.8.

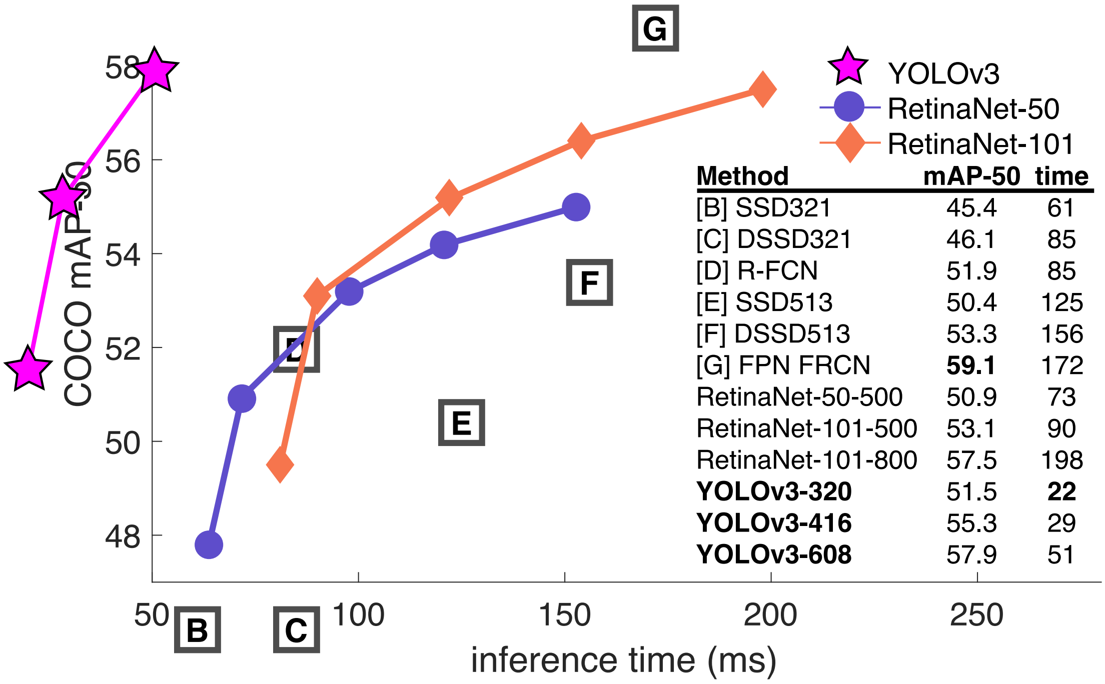
In terms of the COCO’s new AP metric, YOLOv3 is on par with the SSD variants [81,82] but is three times faster. YOLOv3 was still outperformed by other models in that metric though. On the other hand, when considering the AP_{50} metric, YOLOv3 performs almost on par with

[3] For more information about the evaluation metrics used by COCO, refer to `http://cocodataset.org/#detection-eval`.

Table 2.8: Object detection results on the COCO dataset. YOLOv3 is much better than Single Shot MultiBox Detector (SSD) variants and comparable to state-of-the-art models on the AP_{50} metric. The AP_S, AP_M and AP_L metrics refer to the AP for small, medium and large objects, respectively. Table reproduced from [37, 78].

Approach	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [75]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w/ FPN [77]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [79]	Inception-ResNet-v2	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w/ TDM [80]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [36]	DarkNet-19	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [81, 82]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [82]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [78]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [78]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608 [37]	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

other state-of-the-art object detectors such as RetinaNet [78], while being considerably faster (see Figure 2.23). Redmon and Farhadi [37] stated that this indicates that YOLOv3 is a very strong detector that excels at producing decent boxes for objects. However, performance drops significantly as the IoU threshold increases indicating that YOLOv3 struggles to get the boxes perfectly aligned with the object. As can be seen in Figure 2.23, YOLOv3 is better suited for applications that require real-time performance, for example, ALPR.



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Figure 2.23: Speed/accuracy trade-off of object detectors in the COCO dataset based on the AP_{50} metric. YOLOv3 runs significantly faster than other detection methods with comparable performance. Image reproduced from [37].

A smaller and faster model, called Fast-YOLOv3, was provided along with YOLOv3. As shown in Table 2.9, Fast-YOLOv3 predicts bounding boxes at two different scales, which are precisely given by downsampling the dimensions of the input image by 32 and 16, respectively. Fast-YOLOv3 consists of convolutional, pass-through and upsampling layers added to the first 13 layers of Fast-YOLOv2. Detections are made using 'yolo' layers, which use logistic activation.

Table 2.9: The Fast-YOLOv3 model.

#	Layer	Filters	Size	Input	Output
0	conv	16	3 × 3/1	416 × 416 × 3	416 × 416 × 16
1	max		2 × 2/2	416 × 416 × 16	208 × 208 × 16
2	conv	32	3 × 3/1	208 × 208 × 16	208 × 208 × 32
3	max		2 × 2/2	208 × 208 × 32	104 × 104 × 32
4	conv	64	3 × 3/1	104 × 104 × 32	104 × 104 × 64
5	max		2 × 2/2	104 × 104 × 64	52 × 52 × 64
6	conv	128	3 × 3/1	52 × 52 × 64	52 × 52 × 128
7	max		2 × 2/2	52 × 52 × 128	26 × 26 × 128
8	conv	256	3 × 3/1	26 × 26 × 128	26 × 26 × 256
9	max		2 × 2/2	26 × 26 × 256	13 × 13 × 256
10	conv	512	3 × 3/1	13 × 13 × 256	13 × 13 × 512
11	max		2 × 2/1	13 × 13 × 512	13 × 13 × 512
12	conv	1024	3 × 3/1	13 × 13 × 512	13 × 13 × 1024
13	conv	256	1 × 1/1	13 × 13 × 1024	13 × 13 × 256
14	conv	512	3 × 3/1	13 × 13 × 256	13 × 13 × 512
15	conv	255	1 × 1/1	13 × 13 × 512	13 × 13 × 255
16	yolo				
17	route [13]				
18	conv	128	1 × 1/1	13 × 13 × 256	13 × 13 × 128
19	upsample		2×	13 × 13 × 128	26 × 26 × 128
20	route [19, 8]				
21	conv	256	3 × 3/1	26 × 26 × 384	26 × 26 × 256
22	conv	255	1 × 1/1	26 × 26 × 256	26 × 26 × 255
23	yolo				

3 Related Work

In this chapter, we briefly review several recent works that use deep learning approaches in the context of ALPR. For relevant studies using conventional image processing techniques, please refer to [4, 6, 22, 25, 27, 35, 83–88]. We first discuss works related to each stage and then works that do not fit into the other sections. Although ALPR systems based on deep learning techniques usually address the character segmentation and recognition together, we also described some methods in which character recognition was performed separately, as there are works focused on this stage. This chapter concludes with final remarks.

3.1 License Plate Detection

Many authors have addressed the LP detection stage using object detection CNNs. Silva and Jung [11] noticed that the Fast-YOLO model achieved a low recall rate when detecting LPs without prior vehicle detection. Therefore, they used the Fast-YOLO model arranged in a cascaded manner to first detect the frontal view of the cars and then their LPs in the detected patches, attaining high precision and recall rates. Their approach is able to process 185 FPS on a high-end GPU assuming that a single vehicle is being processed.

Hsu et al. [40] customized the YOLO and YOLOv2 models exclusively for LP detection. The main customizations included (i) using a larger input size in order to detect smaller objects and (ii) adapting the bounding box estimation so that each grid cell predicts only one bounding box to eliminate FPs. Despite the fact that the modified versions of YOLO performed better and were able to process 54 FPS on a high-end GPU, we believe that LP detection approaches should be even faster (i.e., 150+ FPS) since the LP characters still need to be recognized.

Li et al. [14] trained a 4-layer CNN based on characters cropped from general text to perform a character-based LP detection. The network was employed in a sliding-window fashion across the entire image to generate a text salience map. Text-like regions were extracted based on the clustering nature of the characters. Connected Component Analysis (CCA) is subsequently applied to produce the initial candidate boxes. Then, another LP/non-LP CNN (also with 4 layers) was trained to remove FPs. Finally, the bounding boxes were refined through a projection-based method. Although the precision and recall rates obtained were higher than those achieved in previous works, this sequence of methods (see Figure 3.1) is too expensive for real-time applications, taking more than 2 seconds to process a single image when running on an NVIDIA Tesla K40c GPU.

Bulan et al. [8] first extracted a set of candidate LP regions using a weak Sparse Network of Winnows (SNoW) classifier trained with Successive Mean Quantization Transform (SMQT) features. Afterward, a strong classifier was employed to discriminate between legible and illegible LP images. The latter set includes cases where the LP is missing, partially occluded, too dark, too bright, damaged, etc. AlexNet [55] was used for extracting features of each candidate region, while a linear SVM was trained to differentiate between regions that either do or do not contain

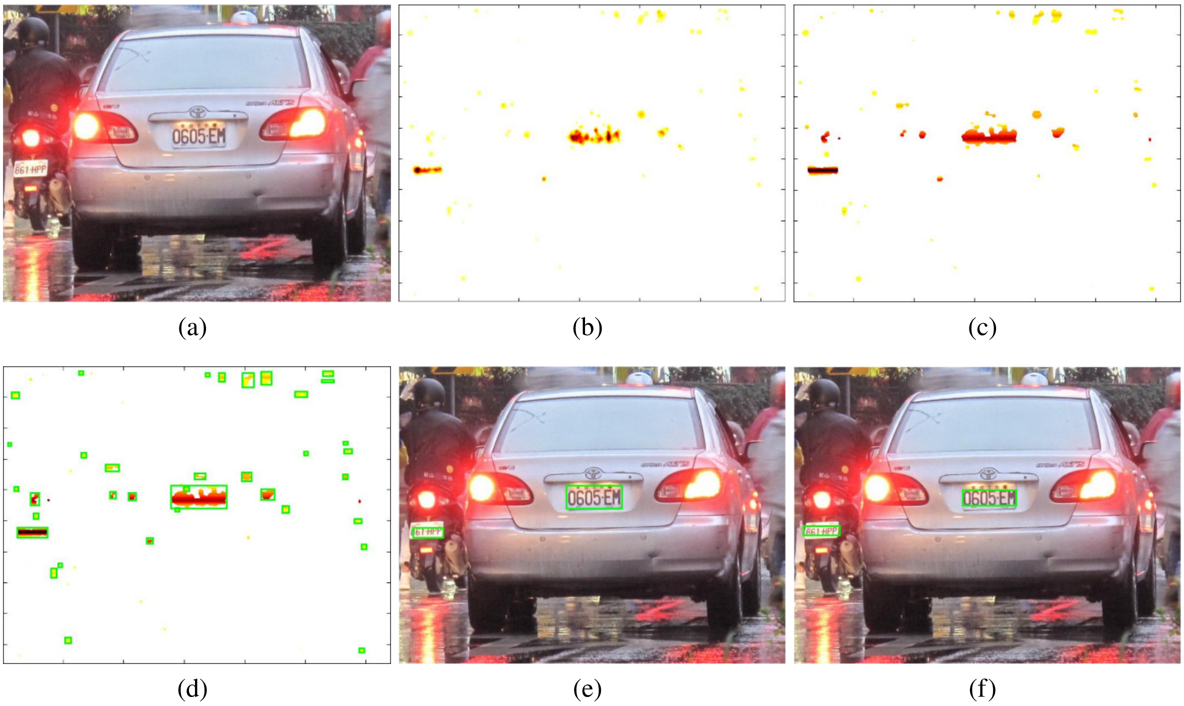


Figure 3.1: The LP detection approach proposed by Li et al. [14]. (a) input image; (b) text salience map generated after the sliding window-based detection; (c) text salience map after applying the NMS and smoothing algorithms; (d) candidate bounding boxes generated by CCA; (e) candidate bounding boxes after the elimination of FPs; (f) final bounding boxes after box refining and LP/non-LP classification. Images reproduced from [14].

an LP. Their method was configured to return 10 candidate Region of Interests (ROIs) for an input image. The top ROI returned by the strong classifier contained an LP in over 96% of the time. The authors reported that their approach processes 5 FPS on an NVIDIA GeForce GTX 570 GPU. The experiments were carried out only on a private dataset.

Rafique et al. [89] applied SVMs and Region-based CNNs (RCNNs) for LP detection, noting that RCNNs are best suited for real-time systems. Dong et al. [26] first detected LP candidates using a Region Proposal Network (RPN) and then classified the extracted patches through an RCNN. The RCNN also regresses the LP corner positions, allowing to rectify the LPs for the next stages. The reported average precision was 96.68% at 11 FPS.

Kurpiel et al. [90] partitioned the input image in sub-regions, forming an overlapping grid. A score for each region was produced using a CNN and the LPs were detected by analyzing the outputs of neighboring sub-regions. On an NVIDIA GeForce GT-740M GPU, it took 230 ms to detect LPs in images with multiple vehicles, achieving a recall rate of 83% on a challenging public dataset introduced by them. In [91], the LP detection approach relied on a 9-layer CNN trained with binary character/non-character images. For binarization, the Canny edge detector was employed. The experiments were performed in several datasets (with LPs from multiple countries), reaching accuracy rates between 93.67% and 97.34%.

Xie et al. [41] proposed a YOLO-based model to predict the LP rotation angle in addition to its coordinates and confidence value. Their network consists of 7 convolutional layers and 3 fully connected ones. Prior to that, another CNN (with the same architecture) was applied to determine the attention region in the input image, assuming that some distance will inevitably exist between any two LPs. By cascading both models, their approach outperformed all baselines in three public datasets, while still running in real time. Despite the impressive results, it is important to highlight two limitations in their work: (i) the authors simplified the problem by forcing their ALPR system to output only one bounding box per image; (ii) motorcycle LPs might be lost when determining the attention region since, in some scenarios (e.g., traffic lights), they might be very close.

Selmi et al. [92] divided the LP detection stage into preprocessing and classification. First, candidate regions were estimated through conventional techniques such as morphological operations, fine contours, geometric filtering, etc. Afterward, a 4-layer CNN was employed to classify each region as LP/non-LP. The method was evaluated in two public datasets, failing in some images with multiple vehicles and in cases where the LP was inclined or too bright/dark. This was expected since handcrafted features are easily affected by noise and might not be robust to certain variations. The execution time was not reported.

Gonçalves et al. [13] presented a 15-layer CNN to detect LPs directly in the frame, i.e. without vehicle detection. The authors showed that even at a high IoU threshold (e.g, 0.7), it is not possible to guarantee that the detected LP encloses all characters (see Figure 3.2). Therefore, a new loss function that penalizes over-segmented LPs was proposed to avoid detections on the inner side of the LP. Their approach was evaluated on public datasets and worked best on images captured by static cameras. According to the authors, this is related to the fact that non-static backgrounds contain much more patterns that can be confused with an LP.

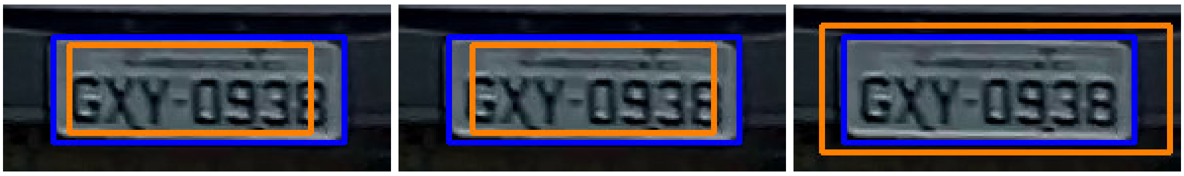


Figure 3.2: Three LPs detected with the same IoU value with the ground truth. Even at a high IoU threshold, it is not possible to guarantee that the detected LP encloses all characters. The ground truth bounding boxes are shown in blue and the hypothetical predictions are shown in orange. All three predictions have IoU = 0.7 with the ground truth, however, only the rightmost has all LP characters completely visible. Images reproduced from [13].

Silva and Jung [33] detected first the vehicles in the input image using the YOLOv2 model without any change or refinement. The outputs related to vehicles (i.e., cars and buses) were merged, whereas the outputs related to other classes were ignored. Then, they proposed a network, called Warped Planar Object Detection Network (WPOD-NET), that searches for LPs and regresses one affine transformation per detection, allowing a rectification of the LP area to a rectangle resembling a frontal view. Their approach, illustrated in Figure 3.3, was trained using many synthetically warped versions of real images to augment the training dataset allowing the network to be trained from scratch using less than 200 manually labeled images. The unwarping greatly helped the OCR task when the LP was strongly distorted. Although the experiments were performed in public datasets, the results and execution time of this particular stage were not reported. Remark that, as pointed out by the authors, the solution should be extended for motorcycles since their LPs pose new challenges due to differences in aspect ratio and layout.

3.2 Character Recognition

Menotti et al. [93] proposed the use of random CNNs to extract features for character recognition. Their CNN architecture was chosen from thousands of random possibilities and its filter weights were also set at random. By training a linear SVM on the resulting features, a significantly better performance was achieved when compared to using image pixels or learning the filters weights with backpropagation. The recognition rates reported were 98% and 96% for digits and letters, respectively. Remark that, in the ALPR context, a single mistake may imply in an incorrect

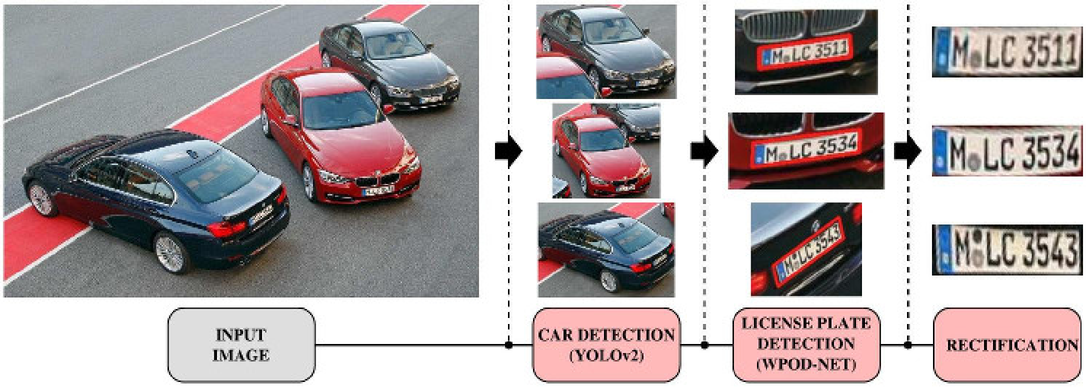


Figure 3.3: The LP detection approach proposed by Silva and Jung [33]. Note that its execution time is highly dependent on the number of vehicles detected in the input image. Image reproduced from [33].

identification of the vehicle. Thus, this approach would correctly recognize only ≈ 82% of the vehicles, considering that it was evaluated in Brazilian LPs which have 3 letters and 4 digits.

Selmi et al. [92] proposed a 37-class CNN with four convolutional layers and two fully connected layers for character recognition. In addition to the 36 classes related to letters and digits, there is a 'non-character' class to eliminate FPs. Although good results were reported in two public datasets, their CNN model was compared only with conventional image processing techniques and no information regarding processing time was provided.

Yang et al. [94] stated that most ALPR solutions do not address Chinese characters in the character recognition stage. Thus, they proposed an architecture (see Figure 3.4) to this end. A CNN with 4 convolutional layers was used for feature extraction, while a kernel-based Extreme Learning Machine (ELM) classifier was employed for recognition. The results reported were better than those obtained with other classifiers such as softmax and SVMs. According to the authors, the errors occurred mainly due to the underrepresentation of some characters in the training set. The experiments were performed on a proprietary dataset containing only Chinese characters and the execution time was not reported.

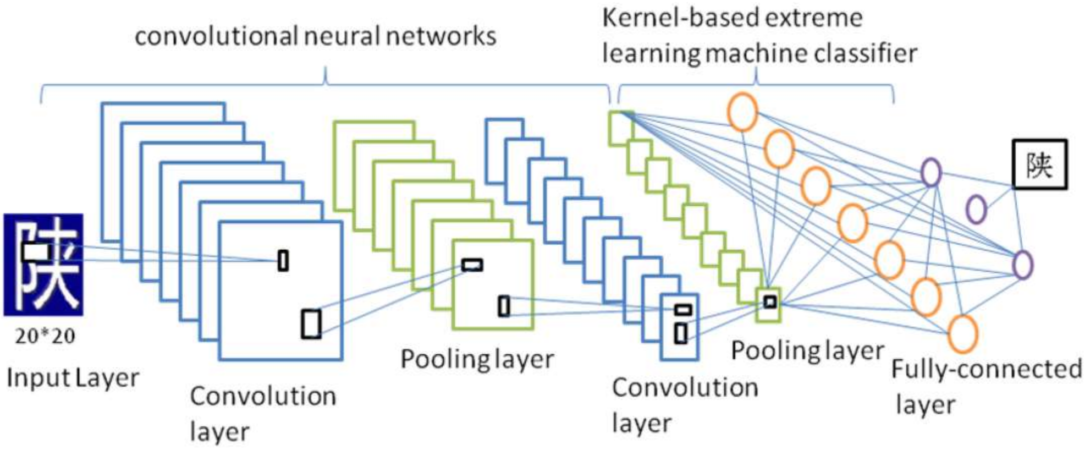


Figure 3.4: The architecture proposed by Yang et al. [94] for Chinese character recognition, which consists of two subsystems: a network with convolutional and pooling layers for feature extraction, and a fully connected classifier based on the kernel ELM algorithm for final decision making. Image reproduced from [94].

3.3 License Plate Recognition

In [11], a YOLO-based model was proposed to simultaneously detect and recognize all characters within a cropped LP. This model, called CR-NET, consists of the first eleven layers of YOLO and four other convolutional layers added to improve nonlinearity. Heuristic rules were used to adapt the results produced by CR-NET according to Brazilian LPs. While impressive FPS rates (i.e., 448 FPS on a high-end GPU) were attained in experiments carried out in the SSIG-SegPlate dataset, less than 65% of the LPs were correctly recognized. According to the authors, the accuracy bottleneck of their approach was letter recognition since the training set of characters of the SSIG-SegPlate dataset is highly unbalanced (in particular, letters).

In [33], Silva and Jung generalized CR-NET by retraining it with an enlarged training set composed of real and artificially generated images using font-types similar to the LPs of the target regions (i.e., Brazil, Europe and the United States), as shown in Figure 3.5. In this way, the retrained network became much more robust for the detection and classification of real characters on Brazilian LPs and also on LPs from other regions, outperforming previous works and commercial systems in three public datasets.

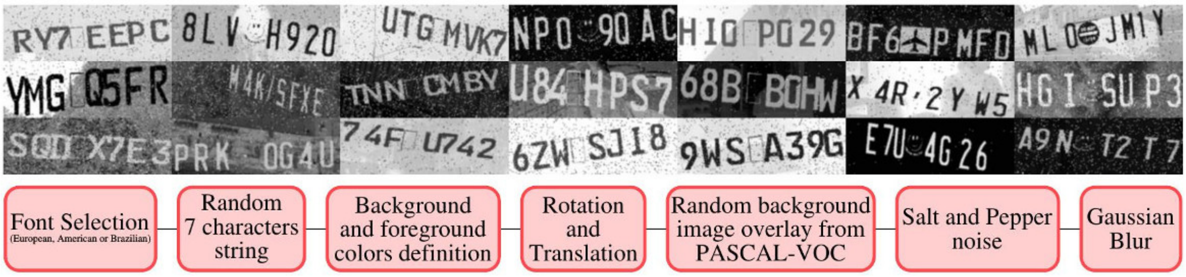

Font Selection
(European, American or Brazilian)
Random 7 characters string
Background and foreground colors definition
Rotation and Translation
Random background image overlay from PASCAL-VOC
Salt and Pepper noise
Gaussian Blur

Figure 3.5: Artificial LP samples generated in [33]. Such LPs use font-types similar to the LPs of the target regions (i.e., Brazil, Europe and the United States), which makes the network more robust for the detection and classification of real characters of LPs issued in those regions. Image reproduced from [33].

Bulan et al. [8] attained a very high accuracy in LP recognition by jointly performing the character segmentation and recognition tasks using a probabilistic inference method based on Hidden Markov Models (HMMs). The authors developed a language model (a Naive Bayes classifier) based on the LP text length and the number of times each template (i.e., a letter/digit combination) exists in the training data. The most likely LP was determined by applying the Viterbi algorithm. Despite the outstanding results obtained, this approach was evaluated only in a private dataset and takes more than one second to process each image on a GTX 570 GPU.

Li et al. [14] proposed to perform character recognition as a sequence labeling problem, also without the character-level segmentation. Sequential features were first extracted from the entire LP patch using a 9-layer CNN in a sliding window manner. Then, Bidirectional Recurrent Neural Networks (BRNNs) with Long Short-Term Memory (LSTM) were applied to label the sequential features. Lastly, Connectionist Temporal Classification (CTC) was employed for sequence decoding. Figure 3.6 illustrates the overall structure of this approach. The results showed that this method attained better recognition rates than the baselines. Nevertheless, only Taiwanese LPs were used in the experiments and the execution time was not reported.

Dong et al. [26] claimed that the method proposed in [14] is very fragile to distortions caused by viewpoint change and therefore is not suitable for LP recognition in the wild. Thus, an LP rectification step is employed first in their approach. Afterward, a CNN was trained to recognize Chinese characters, while a shared-weight CNN recognizer was used for digits and

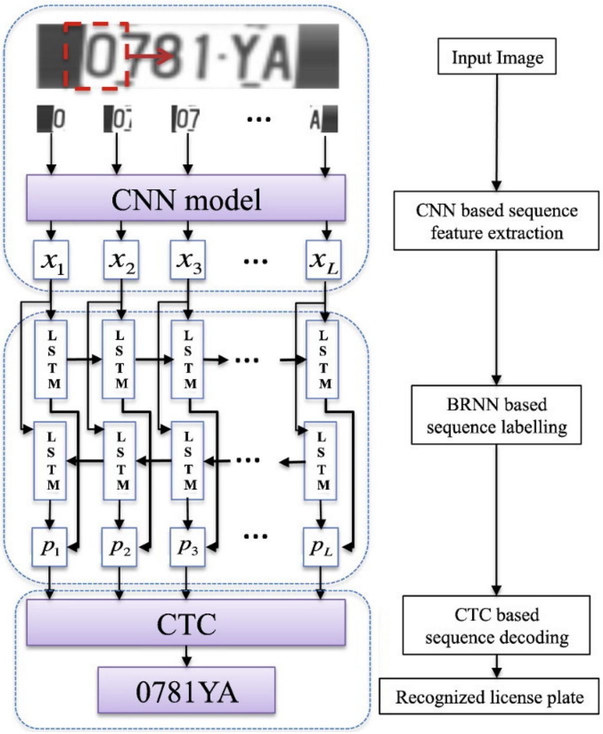


Figure 3.6: The sequence labeling-based approach proposed by Li et al. [14] for LP recognition. First, sequential features are extracted by a 9-layer CNN in a sliding window manner. Then, BRNNs with LSTM are used for sequence labeling. Lastly, CTC is employed for sequence decoding. Image reproduced from [14].

English letters, making full use of the limited training data. The accuracy rate attained for Chinese LPs was 89.05%. The authors did not report the execution time of this particular stage.

Gonçalves et al. [13] designed a multi-task CNN with 14 layers to simultaneously locate and recognize all LP characters. Promising results (in terms of both accuracy and execution time) were achieved in public datasets when taking advantage of data augmentation techniques. However, their network was designed to handle only Brazilian LPs.

Zhuang et al. [95] proposed a semantic segmentation technique followed by a character count refinement module to recognize the characters of an LP. Their framework is illustrated in Figure 3.7. For semantic segmentation, they simplified the DeepLabV2 (ResNet-101) model [96] by removing the multi-scaling process, increasing computational efficiency. According to the authors, the purpose of the multi-scaling process is to fuse hierarchical global information, however, in the LP recognition task, the semantic areas of different characters have a lower correlation. After obtaining the LP semantic map, the character areas were generated through CCA. Finally, Inception-v3 [97] and AlexNet [55] were adopted as the character classification and character counting models, respectively. The authors claimed that both an outstanding recognition performance and a high computational efficiency were attained. Nevertheless, they assumed that LP detection is easily accomplished and used cropped patches containing only the LP with almost no background as input. Furthermore, their system is not able to process images in real time (it processes 25 FPS on a high-end GPU), especially when considering the time required for the LP detection stage, which is generally more time-consuming than the recognition one.

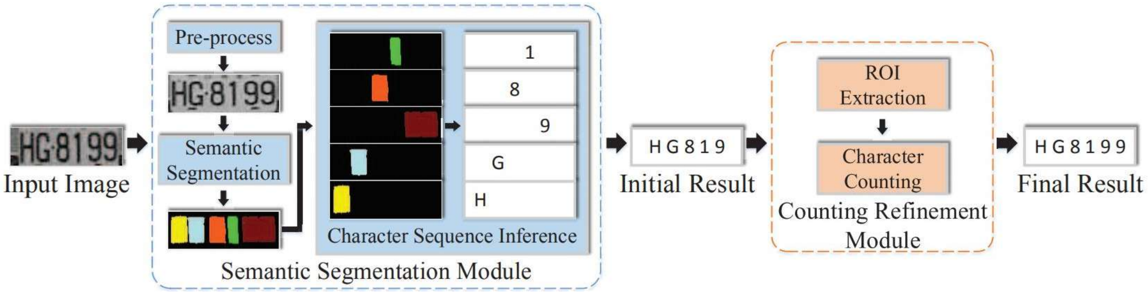


Figure 3.7: Illustration of the framework proposed by Zhuang et al. [95] for LP recognition. Their framework consists of two key modules: semantic segmentation and counting refinement. The former produces the semantic map and the initial character sequence, while the latter generates the final result (i.e., the LP text) through counting characters. Image reproduced from [95].

3.4 Miscellaneous

Some papers [98–101] focus on deblurring the LPs, which is very useful for LP recognition. Lu et al. [98] proposed a scheme based on sparse representation to identify the blur kernel, while Svoboda et al. [101] employed a text deblurring CNN for reconstruction of blurred LPs. Despite achieving outstanding qualitative results (see Figure 3.8), the additional computational cost of a deblurring stage usually is prohibitive for real-time ALPR applications.

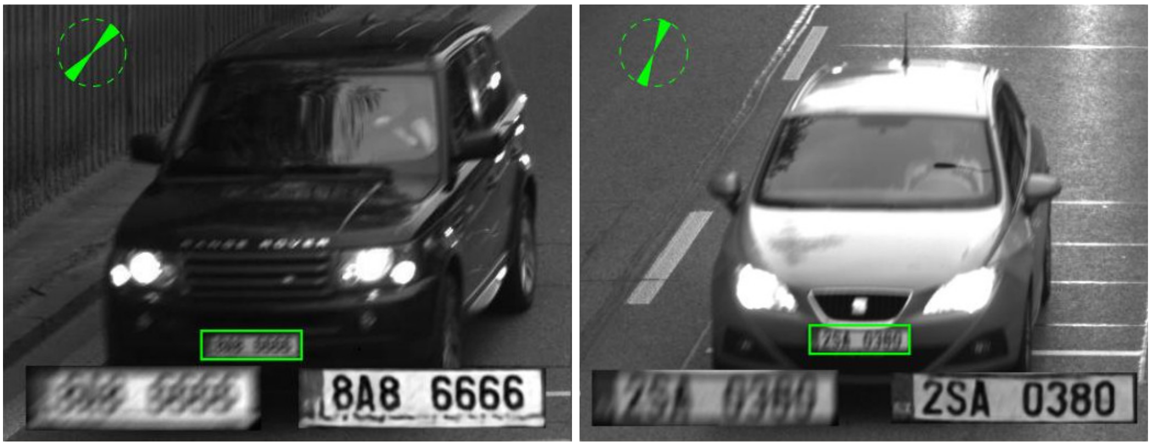


Figure 3.8: Blurred LPs captured by surveillance cameras and their respective reconstructions based on direct CNN deblurring. The network was trained for specific viewpoints using artificial data. Image reproduced from [101].

Masood et al. [18] presented an end-to-end ALPR system, called Sighthound, using a sequence of deep CNNs for LP detection, character detection (or segmentation) and character recognition. As this is a commercial system, little information is given about the network models used in each stage. For character detection, a binary network classifier was trained with LP characters as positives and symbols (e.g., wheelchair, flags, etc.) as negatives. According to the authors, the variety of character fonts and hard negative samples improved the robustness of their system, which outperformed other commercial solutions (e.g., OpenALPR [102]) in public datasets. It is worth noting that the performance rates of commercial systems are often overestimated for promotional reasons [4]. As Sighthound has a trial version through an Application Programming Interface (API), it is often used as a baseline in the ALPR literature [11, 13, 30, 33].

3.5 Final Remarks

The approaches developed for ALPR are still limited. In many studies, the authors only addressed part of the ALPR pipeline (e.g., LP detection [22, 40, 41, 90] or character/LP recognition [93–95]), or performed their experiments on datasets that do not represent real-world or challenging scenarios, making it difficult to accurately evaluate the presented methods. In addition, some authors used only private datasets to evaluate the proposed methods [8, 26, 94]. In this regard, we introduce a public dataset for ALPR that includes 4,500 fully annotated images from 150 vehicles in real-world scenarios where both the vehicle and the camera (inside another vehicle) are moving. Compared to the SSIG-SegPlate dataset [32], which is the public dataset of Brazilian LPs best known and most frequently used in the ALPR context, our dataset has more than twice the images and contains a larger variety in different aspects.

In this work, the proposed system is evaluated in eight public datasets that present a great variety in the way they were collected, with images of various types of vehicles (including motorcycles) and numerous LP layouts. It should be noted that, in most of the works in the literature, no more than three datasets were used in the experiments. In addition, despite the fact that motorcycles are one of the most popular transportation means in metropolitan areas [103], motorcycle images have not been used in the assessment of most ALPR systems in the literature.

Most of the approaches are not capable of recognizing LPs in real time (i.e., 30 FPS) [14, 17, 33, 95], making it impossible for them to be applied in some applications. Furthermore, several authors do not report the execution time of the proposed methods or report the time required only for a specific stage [14, 18, 26, 92, 94], making it difficult an accurate analysis of their speed/accuracy trade-off, as well as their applicability. In this sense, at each stage, we evaluate different YOLO models with various modifications, carefully optimizing and combining them aiming to achieve the best speed/accuracy trade-off. In our experiments, both the accuracy and execution time are reported to enable fair comparisons in future works.

It is important to highlight that although outstanding results in terms of mAP have been achieved with other object detectors such as SSD [81] and RetinaNet [78], in this work we employ YOLO since it focuses on an **extreme** speed/accuracy trade-off [78], which is essential in our domain application, being able to process more than twice as many FPS as other detectors while still achieving competitive results [36, 37].

We consider LP recognition as the current bottleneck of ALPR systems since (i) impressive LP detection results have been reported in recent works [22, 30, 41], both in terms of recall rate and execution time; (ii) OCR approaches must work as close as possible to the optimality (i.e., 100% of character recognition rate) in the ALPR context, as a single mistake may imply in incorrect identification of the vehicle [10]. Therefore, we propose a unified approach for LP detection and layout classification to improve the recognition results through post-processing rules. To the best of our knowledge, this is the first time a layout classification stage is proposed to improve LP recognition. Additionally, we design and apply data augmentation techniques to simulate LPs of other layouts and also to generate LP images with characters that have few instances in the training set, as many examples are needed to effectively train CNNs. Hence, unlike [11, 94], we avoid errors in the recognition stage due to highly unbalanced training sets of LP characters.

4 Proposal

This chapter describes our proposal and it is divided into two sections. First, we introduce the UFPR-ALPR dataset and its statistics. Then, we present the proposed ALPR system in detail, which consists of using YOLO models fine-tuned for each task, several data augmentation tricks to make our system more robust, and a unified approach for LP detection and layout classification to improve the recognition results through post-processing rules.

4.1 UFPR-ALPR Dataset

The dataset contains 4,500 images taken from inside a vehicle driving through regular traffic in an urban environment. These images were obtained from 150 videos with a duration of 1 second and frame rate of 30 FPS. Thus, the dataset is divided into 150 vehicles, each with 30 images with only one visible LP in the foreground. Figure 4.1 shows the diversity of the dataset[1].

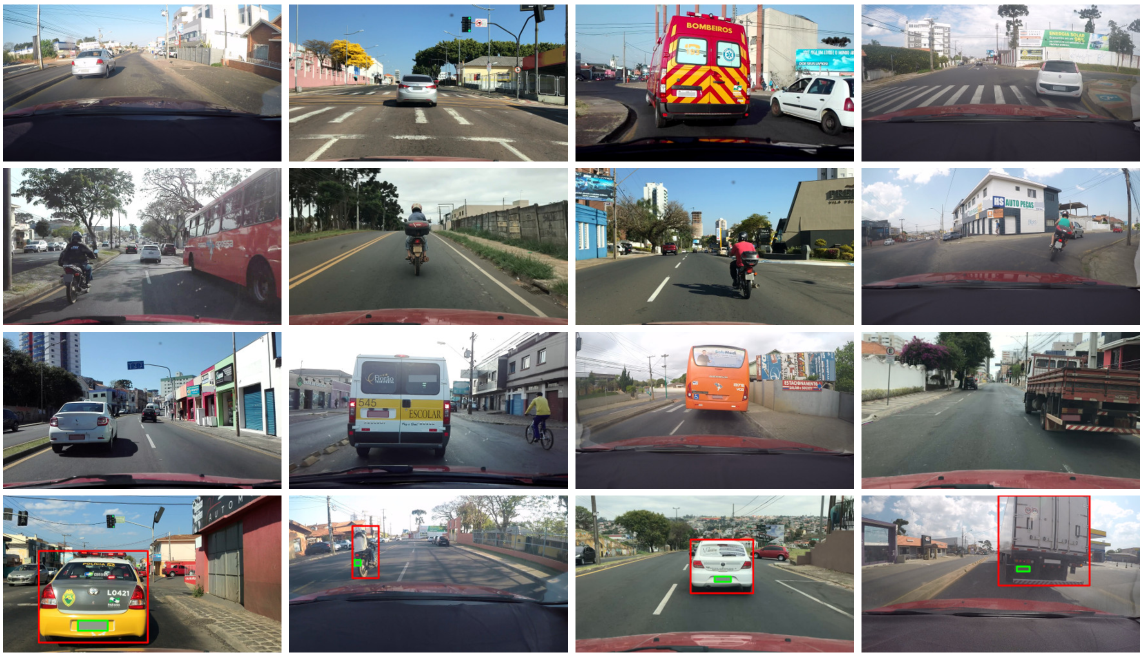


Figure 4.1: Sample images of the UFPR-ALPR dataset. First three rows show the variety in backgrounds, lighting conditions, as well as vehicle/LP positions and types. The 4th row shows examples of vehicle and LP annotations. The LPs were blurred due to privacy constraints.

The images were acquired with three different cameras and are available in the Portable Network Graphics (PNG) format with a size of 1,920 × 1,080 pixels. The cameras used

[1] The UFPR-ALPR dataset is publicly available to the research community at `https://web.inf.ufpr.br/vri/databases/ufpr-alpr/` subject to some privacy restrictions.

were: *GoPro Hero4 Silver*, *Huawei P9 Lite* and *iPhone 7 Plus*. Images obtained with different cameras do not necessarily have the same quality, although they have the same resolution and frame rate. This is due to different camera specifications, such as autofocus, bit rate, focal length, and optical image stabilization. Additional information about the dataset can be seen in Table 4.1.

Table 4.1: Additional information about the UFPR-ALPR dataset: (a) how many images were captured with each camera; (b) dimensions of vehicles, LPs and characters (width × height in pixels). It is noteworthy the great variation in the sizes of vehicles, LPs and characters.

(a)

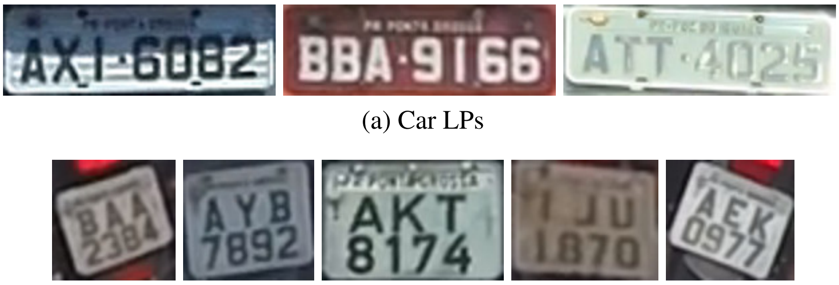
Camera	Images
GoPro Hero4 Silver	1,500
Huawei P9 Lite	1,500
iPhone 7 Plus	1,500
Total	4,500

(b)

Info	Vehicles	LPs	Characters
Minimum Size	93 × 243	53 × 15	2 × 9
Maximum Size	1,112 × 852	208 × 84	31 × 36
Average Size	421 × 408	96 × 38	11 × 17
Aspect Ratio	1.03	2.53	0.65

There are minor variations in the camera position due to repeated mountings of the camera and also to simulate a real condition, where the camera is not always placed in exactly the same position. Additionally, it should be noted that no stabilization method was used.

We collected 1,500 images with each camera (see Table 4.1a), divided as follows: 900 of cars with gray LP, 300 of cars with red LP and 300 of motorcycles with gray LP. In Brazil, the LPs have size and color variations depending on the type of the vehicle and its category. Cars' LPs have a size of 40cm × 13cm, while motorcycles LPs have 20cm × 17cm. Private vehicles have gray LPs, while buses, taxis and other transportation vehicles have red LPs. There are other color variations for specific categories such as official or older cars. Figure 4.2 shows some of the different layouts of LPs found in the dataset.



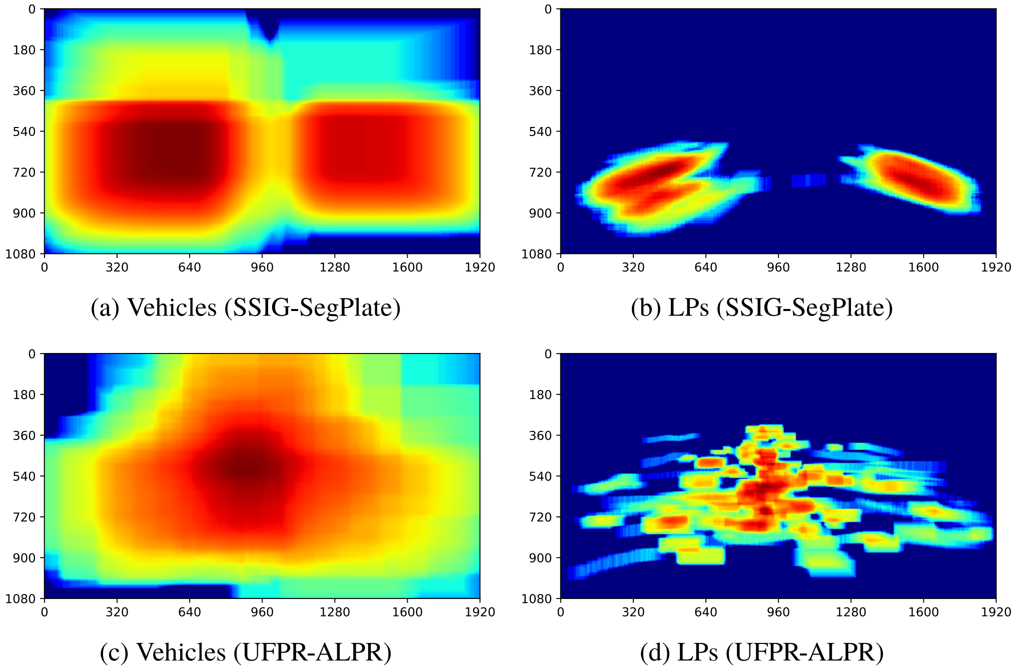
(a) Car LPs

(b) Motorcycle LPs

Figure 4.2: Examples of the LP layouts found in the UFPR-ALPR dataset. In Brazil, cars' LPs have 3 letters and 4 digits in the same row and motorcycles' LPs have 3 letters in one row and 4 digits in another.

The dataset is split as follows: 40% for training, 40% for testing, and 20% for validation. We adopt this protocol (i.e., with a larger test set) due to the fact that it has already been adopted in other public datasets [32, 72], and also to provide more samples for analysis of statistical significance. The dataset distribution was made so that each split has the same number of images obtained with each camera, taking into account the type and position of the vehicle, the color and the characters of the vehicle's LP, the distance of the vehicle from the camera (based on the height of the LP in pixels) such that each split is as representative as possible. This division is explicitly available along with the UFPR-ALPR dataset. It is worth noting that experiments carried out by us suggested that dividing the dataset multiple times and then averaging the results is not necessary, as the proposed division is representative.

The heat maps of the distribution of the vehicles and LPs for the image frame in both SSIG-SegPlate and UFPR-ALPR datasets are shown in Figure 4.3. As can be seen, the vehicles and LPs are much better distributed in our dataset.



(a) Vehicles (SSIG-SegPlate)

(b) LPs (SSIG-SegPlate)

(c) Vehicles (UFPR-ALPR)

(d) LPs (UFPR-ALPR)

Figure 4.3: Heat maps illustrating the distribution of vehicles and LPs in the SSIG-SegPlate and UFPR-ALPR datasets. The heat maps are log-normalized, meaning the distribution is even more concentrated than it appears.

In Brazil, each state uses particular starting letters for its LPs which results in a specific range. In Paraná (where the dataset was collected), LPs range from AAA-0001 to BEZ-9999. Therefore, the letters A and B have many more examples than the others, as shown in Figure 4.4.

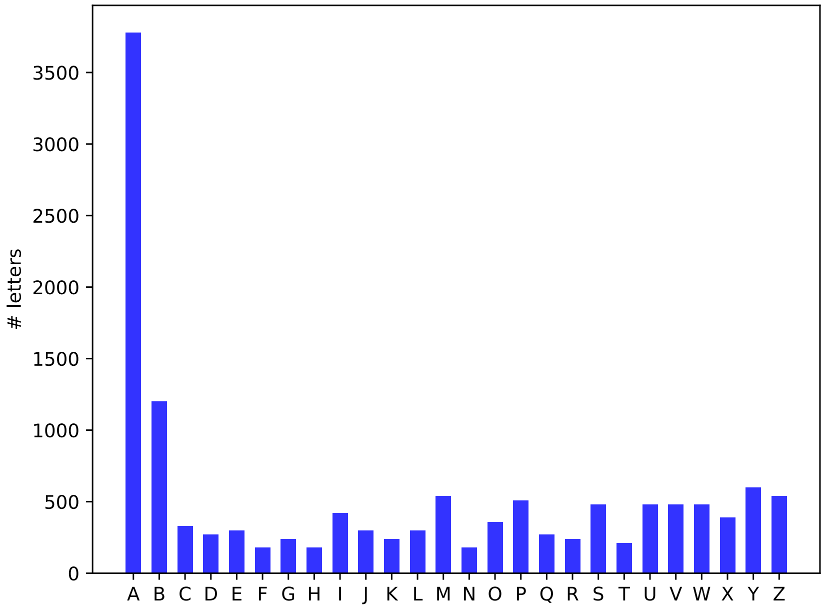


Figure 4.4: Letters distribution in the UFPR-ALPR dataset.

Every image has the following annotations available in a text file: the camera in which the image was taken, the vehicle's position and information such as type (car or motorcycle), make, model and year; the identification and position of the LP, as well as the position of its characters. In order to determine the manufacturers and models of vehicles that we were not familiar, we often made use of websites[2] [3] that allow users to enter an LP (or vehicle identification number) and return several vehicle information such as the make, model and year.

We used two open-source tools for labeling the dataset, namely *sloth*[4] and *labelImg*[5]. The most time-consuming task was the annotation of the bounding box of the characters since some of them (depending on the distance of the vehicle) are very small (see Table 4.1b), and thus even a one-pixel difference might include a lot of background noise or cut a portion of the character.

4.2 Proposed Approach

Traffic images have many textual blocks that can be confused with LPs such as traffic signs and phone numbers on storefronts. In addition, LPs might occupy very small portions of the image. Therefore, we propose to first locate the vehicles (including motorcycles) and then their LPs in the detected patches. Afterward, we detect and recognize all LP characters simultaneously by feeding the entire patch of the LP into the network. In this way, we do not need to deal with the challenging character segmentation task, which has also been avoided in related applications, such as handwritten numeral string recognition and image-based automatic meter reading [72, 104, 105].

Although some approaches with such characteristics (i.e., containing a vehicle detection stage prior to LP detection and/or avoiding character segmentation) have already been proposed in the literature, none of them presented robustness for different LP layouts in both accuracy and processing time, to the best of our knowledge. In [11] and [13], for instance, the authors designed real-time ALPR systems able to process more than 50 FPS on high-end GPUs, however, both systems were evaluated only on Brazilian LPs and presented poor recognition rates in at least one dataset in which they were evaluated. On the other hand, outstanding results were achieved on different scenarios in some recent works [14, 17, 33], however, the methods presented in these works are computationally expensive and cannot be applied in real time (i.e., 30 FPS). This makes them unsuitable for many real-world applications.

In order to develop an ALPR system that is robust for different LP layouts, we propose a layout classification stage after LP detection. However, instead of performing both stages separately, we merge the LP detection and layout classification tasks by training an object detection network that outputs a distinct class for each LP layout. In this way, with almost no additional cost, we employ layout-specific approaches for LP recognition in cases where the LP and its layout are predicted with a confidence value above a predefined threshold. For example, all Brazilian LPs have seven characters: three letters and four digits (in that order), and thus a post-processing method is applied to avoid errors in characters that are often misclassified, such as 'B' and '8', 'G' and '6', 'I' and '1', among others. In cases where the LP and its layout are detected with confidence below the predefined threshold, a generic approach is applied.

[2] `https://www.carcheck.com.br/`

[3] `https://carfacts.com.br/`

[4] The sloth tool is available at `https://github.com/cvhciKIT/sloth`.

[5] The labelImg tool is available at `https://github.com/tzutalin/labelImg`.

As great advances in object detection have been achieved through YOLO-inspired models [42–44], we decided to specialize it for ALPR. We use specific models for each stage[6]. Thus, we can tune the parameters separately in order to improve the performance of each task. The models used are YOLOv2, Fast-YOLOv2 and CR-NET [11], which is an architecture inspired by YOLO for character detection and recognition. We evaluated several data augmentation techniques and modifications to each network (e.g., changes in the input size, number of filters, layers and anchors) to achieve the best speed/accuracy trade-off at each stage.

This section describes the proposed approach and it is divided into three subsections, one for each stage of our ALPR system: (i) vehicle detection, (ii) LP detection and layout classification and (iii) LP recognition. Figure 4.5 illustrates the pipeline of the system, explained throughout this section. All parameters specified in this section are defined based on the validation set and presented in Chapter 5, where the experimental results are reported.

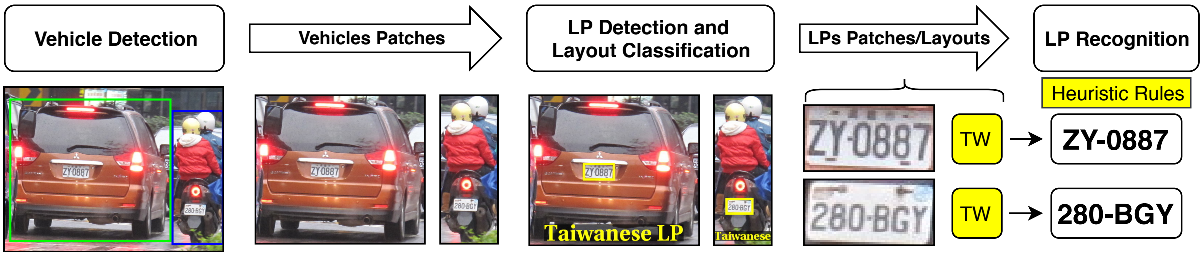


Figure 4.5: The pipeline of the proposed ALPR system. First, the vehicles are detected in the input image. Then, the LP of each vehicle is detected and its layout classified (in the example above, the LPs are Taiwanese). Finally, all characters of each LP are recognized simultaneously, with heuristic rules being applied to adapt the results according to the predicted layout class (e.g., Taiwanese LPs have 5 or 6 characters).

4.2.1 Vehicle Detection

In our previous work [30], we employed the Fast-YOLOv2 and YOLOv2 models at the vehicle detection stage to be able to handle simpler and more realistic data. For simpler scenarios, the Fast-YOLOv2 model was able to detect all vehicles correctly in much less time. However, it was not sufficiently robust on more realistic scenarios and, therefore, the YOLOv2 model was employed in those cases. Although all vehicles were correctly located in [30], that approach requires prior knowledge of which scenario is being handled and also a considerable manual effort, as the network parameters must be adjusted separately for each model/scenario.

Based on that, in this work we train a single network on several datasets, which were collected under different conditions. Thus, our network is able to detect vehicles regardless of the ALPR application or scenario, requiring no prior knowledge and considerably less manual effort since its parameters are adjusted only once for all datasets. In order not to increase the overall cost of the proposed ALPR system, we do not apply preprocessing techniques to the input image.

We conducted experiments to evaluate the following models: Fast-YOLOv2, YOLOv2, Fast-YOLOv3 and YOLOv3. Although the Fast-YOLOv2 and Fast-YOLOv3 models have correctly located the vehicles in most cases, they failed in challenging scenarios such as images in which one or more vehicles are partially occluded or appear in the background. Differently, impressive results (i.e., F-measure rates above 98% in the validation set[7]) were obtained with both YOLOv2 and YOLOv3, which successfully detected vehicles even in those cases where the smaller models

[6] In order to train all YOLO-based models, we use convolutional weights pre-trained on ImageNet [15], available at `https://pjreddie.com/darknet/yolo/`.

[7] The division of the images of each dataset into training, test and validation sets is detailed in Section 5.2.

failed. Considering that the computational cost is one of our main concerns and YOLOv3 is much more complex than its predecessor (see Section 2.3.2), we employ YOLOv2 for vehicle detection. According to our experiments and domain application, it is not necessary to use a deep model with as many layers and filters as YOLOv3 to handle the detection of one or two classes of objects.

For vehicle detection, we perform several changes to the YOLOv2 model. First, we changed the network input size from 416 × 416 to 448 × 288 pixels since the images used as input to ALPR systems generally have a width greater than height. Hence, our network processes less distorted images and performs faster, as the new input size is 25% smaller than the original. The new dimensions were chosen based on speed/accuracy assessments with different input sizes (from 448 × 288 to 832 × 576 pixels) using YOLO's multi-scale training [36]. Then, we recalculate the anchor boxes for the new input size as well as for the datasets employed in our experiments (described in Chapter 5) using the k-means clustering algorithm available in [106]. Finally, we reduced the number of filters in the last convolutional layer to match the number of classes. YOLOv2 uses A anchor boxes to predict bounding boxes each with four coordinates (x, y, w, h), confidence and C class probabilities [36], so the number of filters is given by

$$\mathit{filters} = (C + 5) \times A\,. \tag{4.1}$$

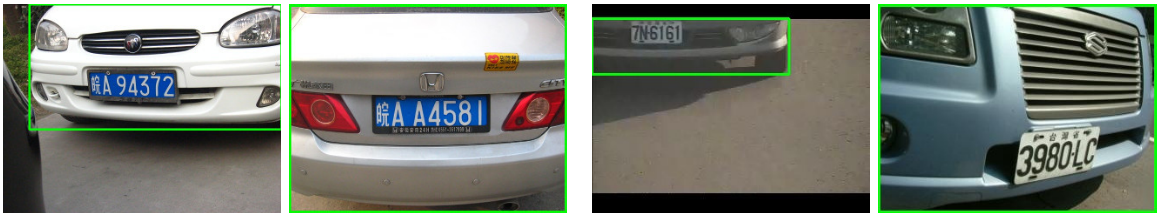
We evaluated different numbers of anchors (from 3 to 9) and the best results in the validation set were obtained using $A = 5$. As we intend to detect cars and motorcycles (two classes), the number of filters in the last convolutional layer must be 35 $((2 + 5) \times 5)$. According to preliminary experiments, the results were better when using two classes instead of just one regarding both types of vehicles.

The modified YOLOv2 architecture for vehicle detection is shown in Table 4.2. It should be noted that we evaluated many other modifications to our detector to make it even faster, e.g., we tried to remove some layers (e.g., #23 and #24) and to reduce the number of filters (between 10% and 50%) in the convolutional layers. However, after those changes, our network often failed in more realistic scenarios (similarly to Fast-YOLOv2 and Fast-YOLOv3).

Table 4.2: The YOLOv2 architecture, modified for vehicle detection. The input size was changed from 416 × 416 to 448 × 288 pixels and the number of filters in the last convolutional layer was reduced from 425 to 35.

#	Layer	Filters	Size	Input	Output
0	conv	32	3 × 3/1	448 × 288 × 3	448 × 288 × 32
1	max		2 × 2/2	448 × 288 × 32	224 × 144 × 32
2	conv	64	3 × 3/1	224 × 144 × 32	224 × 144 × 64
3	max		2 × 2/2	224 × 144 × 64	112 × 72 × 64
4	conv	128	3 × 3/1	112 × 72 × 64	112 × 72 × 128
5	conv	64	1 × 1/1	112 × 72 × 128	112 × 72 × 64
6	conv	128	3 × 3/1	112 × 72 × 64	112 × 72 × 128
7	max		2 × 2/2	112 × 72 × 128	56 × 36 × 128
8	conv	256	3 × 3/1	56 × 36 × 128	56 × 36 × 256
9	conv	128	1 × 1/1	56 × 36 × 256	56 × 36 × 128
10	conv	256	3 × 3/1	56 × 36 × 128	56 × 36 × 256
11	max		2 × 2/2	56 × 36 × 256	28 × 18 × 256
12	conv	512	3 × 3/1	28 × 18 × 256	28 × 18 × 512
13	conv	256	1 × 1/1	28 × 18 × 512	28 × 18 × 256
14	conv	512	3 × 3/1	28 × 18 × 256	28 × 18 × 512
15	conv	256	1 × 1/1	28 × 18 × 512	28 × 18 × 256
16	conv	512	3 × 3/1	28 × 18 × 256	28 × 18 × 512
17	max		2 × 2/2	28 × 18 × 512	14 × 9 × 512
18	conv	1024	3 × 3/1	14 × 9 × 512	14 × 9 × 1024
19	conv	512	1 × 1/1	14 × 9 × 1024	14 × 9 × 512
20	conv	1024	3 × 3/1	14 × 9 × 512	14 × 9 × 1024
21	conv	512	1 × 1/1	14 × 9 × 1024	14 × 9 × 512
22	conv	1024	3 × 3/1	14 × 9 × 512	14 × 9 × 1024
23	conv	1024	3 × 3/1	14 × 9 × 1024	14 × 9 × 1024
24	conv	1024	3 × 3/1	14 × 9 × 1024	14 × 9 × 1024
25	route [16]				
26	reorg		/2	28 × 18 × 512	14 × 9 × 2048
27	route [26, 24]				
28	conv	1024	3 × 3/1	14 × 9 × 3072	14 × 9 × 1024
29	conv	35	1 × 1/1	14 × 9 × 1024	14 × 9 × 35
30	detection				

Silva and Jung [33] slightly modified their pipeline by directly applying their LP detector (i.e., skipping the vehicle detection stage) when dealing with images in which the vehicles are very close to the camera, as their vehicle detector failed in several of those cases. We believe this is not the best way to handle the problem. Instead, as illustrated in Figure 4.6, we do not skip the vehicle detection stage even when only a small part of the vehicle is visible. The entire image is labeled as ground truth in cases where the vehicles are very close to the camera. Therefore, our network also learns to select the ROI in those cases.



(a) (b)

Figure 4.6: Examples of images in which only part of the vehicle is visible. The ROI (i.e., the ground truth) used for training our network is shown in green. Observe that detectors trained only on ‘entire’ vehicles would probably fail in these cases. The images in (a) and (b) were taken from the ChineseLP [45] and AOLP [107] datasets, respectively.

We exploit some data augmentation strategies (flipping, rescaling and shearing) to train our network. Thus, we prevent overfitting by creating several images with different characteristics from a single labeled one. Figure 4.7 shows an image and five new samples created from it.

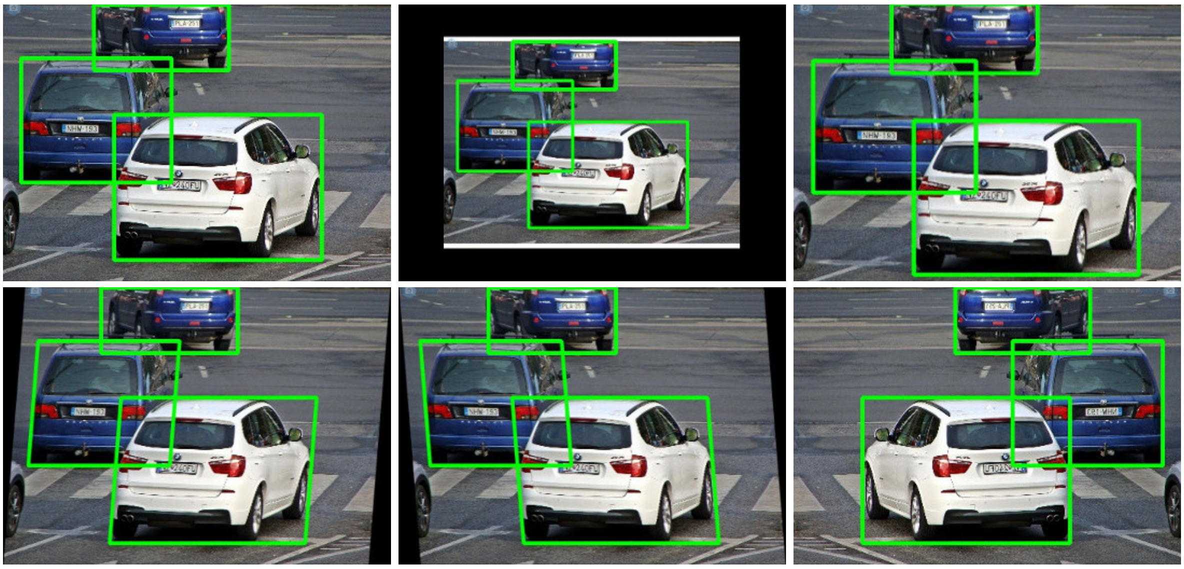


Figure 4.7: New training samples for vehicle detection created using data augmentation strategies. The upper-left image is the original (taken from `http://platesmania.com/`) and the others were generated through rescaling, shearing and horizontal flipping.

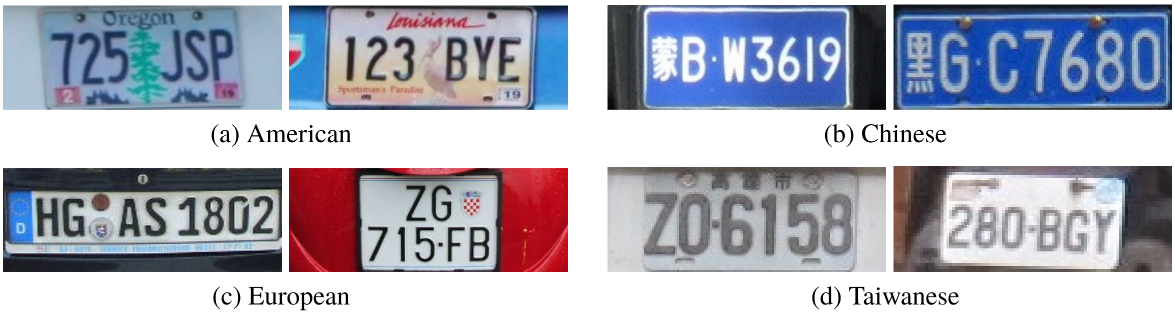
By default, YOLO only returns objects detected with a confidence of 0.25 or higher. In the validation set, we evaluate several confidence thresholds to detect as many vehicles as possible while maintaining a low FP rate. Furthermore, we apply an NMS algorithm to eliminate redundant detections (those with IoU ≥ 0.25) since the same vehicle might be detected more than once by the network. A negative recognition result is given in cases where no vehicle is found.

4.2.2 License Plate Detection and Layout Classification

In our previous work [30], we developed a real-time ALPR system that contains a character recognition module in which letters and digits are recognized separately. Specifically, a network is trained to recognize letters (26 classes) and another one to recognize digits (10 classes). In this sense, errors in characters that are often misclassified (e.g., ‘B’ and ‘8’, ‘G’ and ‘6’, ‘I’ and ‘1’, among others) are avoided. Although impressive results were achieved, that approach leverages prior knowledge of Brazilian LPs (i.e., the position of letters and digits) and, therefore, cannot be applied to LPs with other layouts. This drawback was mentioned in [33].

In order to develop a layout-independent ALPR system, we propose to detect the LP and classify its layout simultaneously, given a vehicle patch. To this end, we fine-tune an object detection network to predict one distinct class for each LP layout. In this way, we can employ layout-specific approaches for LP recognition in cases where the LP and its layout are predicted with a high confidence value. In other cases, a generic approach is applied. To the best of our knowledge, this is the first time a layout classification stage is proposed to improve LP recognition.

In this work, we classify each LP layout into one of the following classes: American, Brazilian, Chinese, European or Taiwanese. These classes were defined based on the public datasets found in the literature [30, 32, 35, 45, 46, 108–110] and also because there are many ALPR systems designed primarily for LPs of one of those regions [11, 35, 94]. It is worth noting that (i) among LPs with different layouts (which may belong to the same class/region) there is a wide variety in many factors, for example, in the aspect ratio, colors, symbols, position of the characters, number of characters, among others; (ii) we consider LPs from different jurisdictions in the United States as a single class; the same is done for LPs from European countries. LPs from the same country or region may look quite different, but still share many characteristics in common. Such common features can be exploited to improve LP recognition. In Figure 4.8, we show examples of LPs of different layouts and classes.



(a) American

(b) Chinese

(c) European

(d) Taiwanese

Figure 4.8: Examples of LPs of different layouts and classes. Observe the wide variety in different ways (e.g., aspect ratio, colors, symbols, position of the characters, number of characters, among others) on different LP layouts. The images in (a) and (b) were taken from `http://platesmania.com/`, while the images in (c) and (d) are from the EnglishLP [109] and AOLP [107] datasets, respectively. Examples of Brazilian LPs are shown in Figure 4.2.

Looking for an efficient ALPR system, in this stage we performed experiments with the Fast-YOLOv2 and Fast-YOLOv3 models, which are focused on a speed/accuracy trade-off. In the validation set, Fast-YOLOv2 obtained slightly better results than its successor (an F-measure rate about 1% higher). This is due to the fact that YOLOv3 and Fast-YOLOv3 have relatively high performance on small objects (which is not the case since we first detect the vehicles), but comparatively worse performance on medium and larger size objects [37]. Accordingly, here we employ the Fast-YOLOv2 model.

We made some changes to the Fast-YOLOv2 model to adapt it to our application and to achieve even better results. First, we changed the kernel size of the next-to-last convolutional layer from 3×3 to 1×1. Then, we added a 3×3 convolutional layer with twice the filters of that layer. In this way, the network reached better results (F-measure $\approx$ 1% higher, from 97.97% to 99.00%) almost without increasing the number of FLOP required (i.e., 5.35 $\rightarrow$ 5.53 BFLOP), as alternating 1×1 convolutional layers between 3×3 convolutions reduce the feature space from preceding layers [29, 36, 63]. As in the vehicle detection stage, we recalculate the anchors for our data and make adjustments to the number of filters in the last layer. The modified architecture is shown in Table 4.3. We use the default input size of Fast-YOLOv2 (416 × 416 pixels), as it is very similar to the average size of vehicles in the proposed dataset (421 × 408 pixels), which contains different types of vehicles.

Table 4.3: Fast-YOLOv2 with some changes for LP detection and layout classification. First, the kernel size of layer #13 was reduced from 3 × 3 to 1 × 1, and layer #14 was added. Then, the number of filters in layer #15 was reduced from 425 to 50, as we use 5 anchor boxes to detect 5 classes (see Equation 4.1).

#	Layer	Filters	Size	Input	Output	BFLOP
0	conv	16	3 × 3/1	416 × 416 × 3	416 × 416 × 16	0.150
1	max		2 × 2/2	416 × 416 × 16	208 × 208 × 16	0.003
2	conv	32	3 × 3/1	208 × 208 × 16	208 × 208 × 32	0.399
3	max		2 × 2/2	208 × 208 × 32	104 × 104 × 32	0.001
4	conv	64	3 × 3/1	104 × 104 × 32	104 × 104 × 64	0.399
5	max		2 × 2/2	104 × 104 × 64	52 × 52 × 64	0.001
6	conv	128	3 × 3/1	52 × 52 × 64	52 × 52 × 128	0.399
7	max		2 × 2/2	52 × 52 × 128	26 × 26 × 128	0.000
8	conv	256	3 × 3/1	26 × 26 × 128	26 × 26 × 256	0.399
9	max		2 × 2/2	26 × 26 × 256	13 × 13 × 256	0.000
10	conv	512	3 × 3/1	13 × 13 × 256	13 × 13 × 512	0.399
11	max		2 × 2/1	13 × 13 × 512	13 × 13 × 512	0.000
12	conv	1024	3 × 3/1	13 × 13 × 512	13 × 13 × 1024	1.595
13	conv	512	1 × 1/1	13 × 13 × 1024	13 × 13 × 512	0.177
14	conv	1024	3 × 3/1	13 × 13 × 512	13 × 13 × 1024	1.595
15	conv	50	1 × 1/1	13 × 13 × 1024	13 × 13 × 50	0.017
16	detection					

In Table 4.3, we also list the number of FLOP required in each layer to highlight how small the network is compared to others, e.g., YOLOv2 and YOLOv3. For this task, our network requires 5.53 BFLOP while YOLOv2 and YOLOv3 require 29.35 and 66.32 BFLOP, respectively. It is noteworthy that we only need to increase the number of filters in the last convolutional layer (following Equation 4.1) so that the network can detect/classify additional LP layouts.

For LP detection and layout classification, we also use data augmentation strategies to generate many other images from a single labeled one. As can be seen in Figure 4.9, given the labeled bounding box of the vehicle, new training samples are created by adding some margin to it and through rescaling and shearing. Horizontal flipping is not performed at this stage, as the network leverages information such as the position of the characters and symbols on the LP to predict its layout (besides the LP's aspect ratio, colors, and other characteristics).

For testing, we also add a small margin to the vehicle patch (10% of its width/height) to avoid losing LPs in cases where the vehicle is not very well detected, as illustrated in Figure 4.10b. In addition, we attempted to avoid large distortions in images of motorcycles and larger vehicles (e.g., trucks and buses) by enlarging the detected regions horizontally or vertically, so that each vehicle patch has an aspect ratio (w/h) between 0.75 and 1.25 (see Figure 4.10c). However, this impaired the performance of the network in preliminary experiments, as a lot of background noise is added to the vehicle patch in that way (note that the training images were also enlarged in this experiment). Therefore, we do not enlarge the vehicle patch after adding a small margin to it.

Only the detection with the highest confidence value is considered in cases where more than one LP is predicted, as each vehicle has only one LP. Then, we classify as 'undefined layout' every LP that has its position and class predicted with a confidence value below 0.75, regardless of which class the network predicted (note that such LPs are not rejected, instead, a generic approach is used in the recognition stage). This threshold was chosen based on experiments performed in the validation set, in which approximately 92% of the LPs were predicted with a

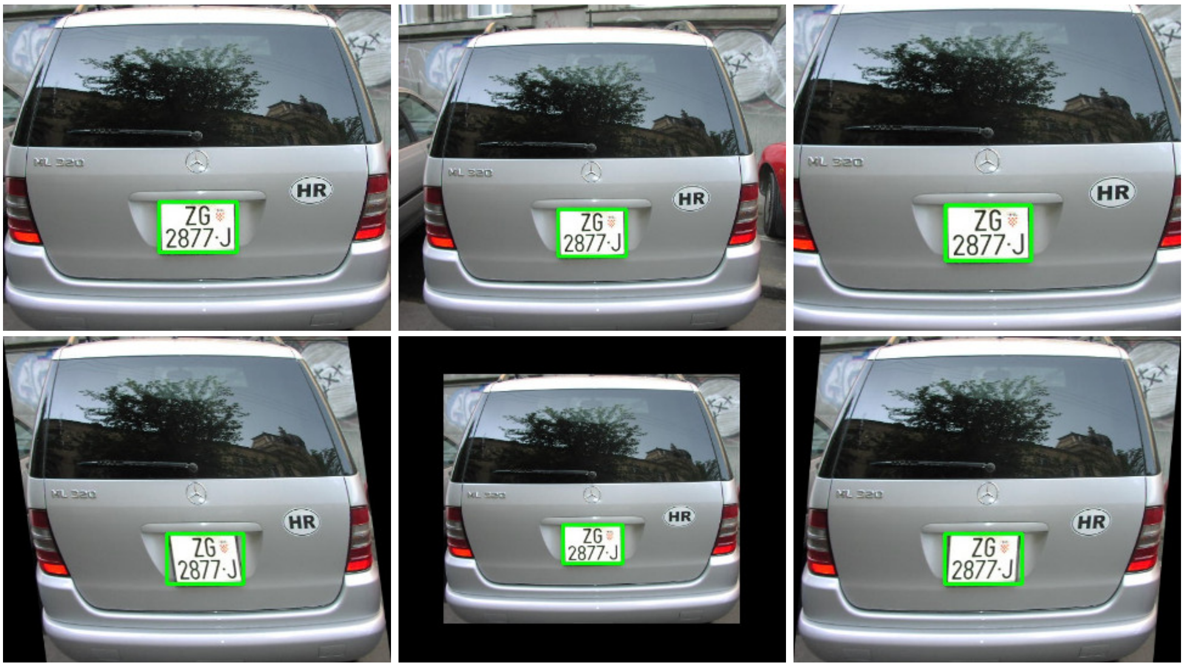
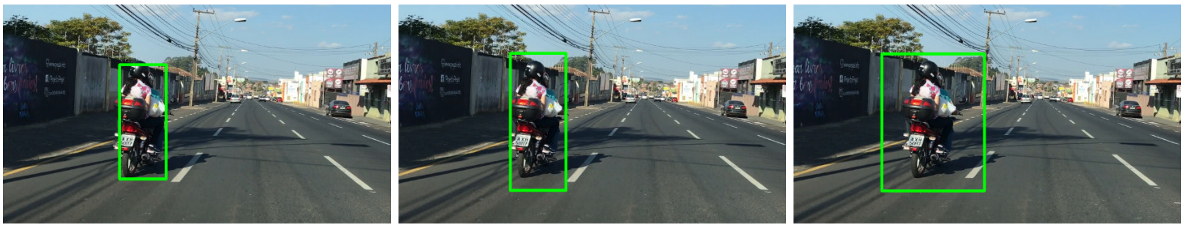


Figure 4.9: New training samples for LP detection and layout classification created using data augmentation. The upper-left image uses the ideal bounding box of the vehicle and the others were generated by adding some margin to it and through rescaling and shearing. The original image was taken from the EnglishLP dataset [109].



(a) Original (b) After adding a margin (c) After enlarging

Figure 4.10: A vehicle's bounding box after adding a small margin to it (10% of the size of the bounding box) and after enlarging it. In preliminary experiments, the results were better after (b) and worse after (c). In this way, we do not enlarge the vehicle patch after adding a small margin to it. This image belongs to the UFPR-ALPR dataset.

confidence value above 0.75. In each of these cases, the LP layout was correctly classified. A negative result is given in cases where no LP is predicted by the network.

It is worth noting that we could have trained two distinct networks for this stage: one for cars and another one for motorcycles, as we detected these vehicles separately in the previous stage. We believe that in this way it would be possible to attain even better results since each network would focus on the detection of LPs in one type of vehicle, for example, motorcycle LPs generally have similar width and height, unlike car ones. Besides, it would be possible to employ a network with a different (and smaller) input size to detect LPs in motorcycles, which have different aspect ratio from other vehicles. This approach was not carried out due to the lack of fully annotated public datasets with motorcycle images and different LP layouts.

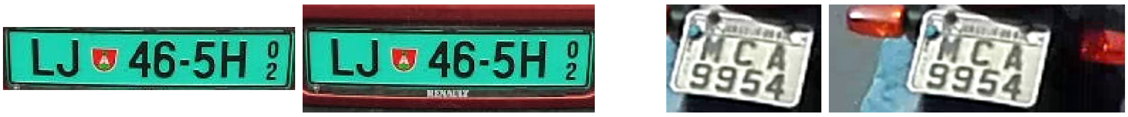
4.2.3 License Plate Recognition

Once the LP has been detected and its layout classified, we employ the CNN proposed by Silva and Jung [11], called CR-NET, for LP recognition. CR-NET is a YOLO-based model that consists of the first eleven layers of YOLO and four other convolutional layers added to improve nonlinearity. This model was chosen for two main reasons. First, it was capable of detecting and recognizing LP characters at 448 FPS in [11]. Second, very recently, it yielded the best recognition results in the context of image-based automatic meter reading [72], outperforming two segmentation-free approaches based on deep learning [13, 111].

The CR-NET architecture is shown in Table 4.4. We changed its input size, which was originally defined based on the LPs of the SSIG-SegPlate dataset, from 240×80 to 352×128 pixels taking into account the average aspect ratio of the LPs in the datasets used in our experiments (described in Section 5.1), in addition to results obtained in the validation set, where several input sizes were evaluated (e.g., 256 × 96 and 384 × 128 pixels). As the same model is employed to recognize LPs of various layouts, we enlarge all LP patches (in both the training and testing phases) so that they have aspect ratios (w/h) between 2.5 and 3.0, as shown in Figure 4.11, considering that the input image has an aspect ratio of 2.75 (352 × 128 pixels). The network is trained to predict 35 classes (0-9, A-Z, where the letter 'O' is detected/recognized jointly with the digit '0') using the LP patch as well as the class and coordinates of each character as inputs.

Table 4.4: The CR-NET model, proposed in [11]. We increased the input size from 240 × 80 to 352 × 128 pixels. The number of filters in the last convolutional layer (#14) was defined following Equation 4.1 (using $A = 5$).

#	Layer	Filters	Size	Input	Output	BFLOP
0	conv	32	3 × 3/1	352 × 128 × 3	352 × 128 × 32	0.078
1	max		2 × 2/2	352 × 128 × 32	176 × 64 × 32	0.001
2	conv	64	3 × 3/1	176 × 64 × 32	176 × 64 × 64	0.415
3	max		2 × 2/2	176 × 64 × 64	88 × 32 × 64	0.001
4	conv	128	3 × 3/1	88 × 32 × 64	88 × 32 × 128	0.415
5	conv	64	1 × 1/1	88 × 32 × 128	88 × 32 × 64	0.046
6	conv	128	3 × 3/1	88 × 32 × 64	88 × 32 × 128	0.415
7	max		2 × 2/2	88 × 32 × 128	44 × 16 × 128	0.000
8	conv	256	3 × 3/1	44 × 16 × 128	44 × 16 × 256	0.415
9	conv	128	1 × 1/1	44 × 16 × 256	44 × 16 × 128	0.046
10	conv	256	3 × 3/1	44 × 16 × 128	44 × 16 × 256	0.415
11	conv	512	3 × 3/1	44 × 16 × 256	44 × 16 × 512	1.661
12	conv	256	1 × 1/1	44 × 16 × 512	44 × 16 × 256	0.185
13	conv	512	3 × 3/1	44 × 16 × 256	44 × 16 × 512	1.661
14	conv	200	1 × 1/1	44 × 16 × 512	44 × 16 × 200	0.144
15	detection					



(a) Vertical Enlargement

(b) Horizontal Enlargement

Figure 4.11: Two illustrations of enlargement of the LPs detected in the previous stage. In this way, a single network is trained to recognize LPs of different layouts, regardless of their aspect ratios. The LP patches in (a) and (b) were taken from the EnglishLP [109] and UFPR-ALPR datasets, respectively.

It is worth noting that the first character in Chinese LPs (see Figure 4.8b) is a Chinese character that represents the province in which the vehicle is affiliated [94, 112]. Following [17], our network was not trained/designed to recognize Chinese characters, even though Chinese LPs are used in the experiments. In other words, only digits and English letters are considered. The reason is threefold: (i) there are less than 400 images in the ChineseLP dataset [45] (only some of them are used for training), which is employed in the experiments, and some provinces are not represented; (ii) labeling the class of Chinese characters is not a trivial task for non-Chinese people (we manually labeled the position and class of the LP characters in the ChineseLP dataset); and (iii) to fairly compare our system with others trained only on digits and English letters. Remark that in the ALPR literature the approaches capable of recognizing Chinese characters, digits and English letters were evaluated, for the most part, on datasets containing only Chinese LPs [25, 94, 112, 113].

As the LP layout is classified in the previous stage, we employ some heuristic rules to adapt the results produced by CR-NET according to the predicted class. Based on the datasets employed in this work, we defined the minimum and the maximum number of characters to be considered in LPs of each layout (see Table 4.5). Brazilian and Chinese LPs have a fixed number of characters, while American, European and Taiwanese LPs do not. Initially, we consider all characters predicted with a confidence value above a predefined threshold. Afterward, as in the vehicle detection stage, an NMS algorithm is applied to remove redundant detections (i.e., those with IoU ≥ 0.25). Finally, if necessary, we discard the characters predicted with lower confidence values or consider others previously discarded (i.e., ignoring the confidence threshold) so that the number of characters considered is within the range defined for the predicted class. We consider that the LP has between 4 and 8 characters in cases where its layout was classified with a low confidence value (i.e., undefined layout).

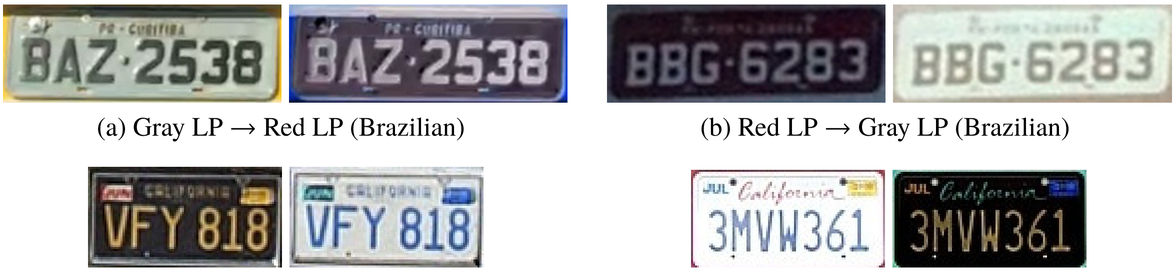
Table 4.5: The minimum and maximum number of characters to be considered in LPs of each layout.

LP Layout	# Characters	
	Min.	Max.
American	4	7
Brazilian	7	7
Chinese	6	6
European	5	8
Taiwanese	5	6

Additionally, we swap digits and letters on Brazilian and Chinese LPs, as there are fixed positions for digits or letters in those layouts. In Brazilian LPs, the first three characters correspond to letters and the last four to digits; while in Chinese LPs the second character is a letter that represents a city in the province in which the vehicle is affiliated. This swap approach, inspired by [11], is not employed for LPs of other layouts since each character position can be occupied by either a letter or a digit in American, European and Taiwanese LPs. The specific swaps are given by [1 ⇒ I; 2 ⇒ Z; 4 ⇒ A; 5 ⇒ S; 6 ⇒ G; 7 ⇒ Z; 8 ⇒ B] and [A ⇒ 4; B ⇒ 8; D ⇒ 0; G ⇒ 6; I ⇒ 1; J ⇒ 1; Q ⇒ 0; S ⇒ 5; Z ⇒ 7]. In this way, we avoid errors in characters that are often misclassified.

The LP characters might also be arranged in two rows instead of one. We distinguish such cases based on the predictions of the vehicle type, LP layout, and character coordinates. In our experiments, only two datasets have LPs with the characters arranged in two rows. These datasets were captured in Brazil and Croatia. In Brazil, car and motorcycle LPs have the characters arranged in one and two rows, respectively. Thus, we look at the predicted class in the vehicle detection stage in those cases. In Croatia, on the other hand, cars might also have LPs with two rows of characters. Therefore, for European LPs, we consider that the characters are arranged in two rows in cases where the bounding boxes of half or more of the predicted characters are located entirely below another character. In our tests, this simple rule was sufficient to distinguish LPs with one and two rows of characters even in cases where the LP is considerably inclined.

In addition to using the original LP images, we employ various data augmentation techniques to train the CR-NET model and improve its robustness. First, we double the number of training samples by creating a negative image of each LP, as we noticed that in some cases negative LPs are very similar to LPs of other layouts. This is illustrated with Brazilian and American LPs in Figure 4.12. We also generate many other images by randomly rescaling the LP patch and adding a margin to it, simulating more or less accurate detections of the LP in the previous stage.



(a) Gray LP → Red LP (Brazilian)

(b) Red LP → Gray LP (Brazilian)

(c) Black LP → White LP (American)

(d) White LP → Black LP (American)

Figure 4.12: Examples of negative images created to simulate LPs of other layouts. (a) and (b) show Brazilian LPs, while American ones are shown in (c) and (d). In Brazil, private vehicles have gray LPs, while buses, taxis and other transportation vehicles have red LPs. In the United States, old California LPs featured gold characters on a black background. Currently, they have blue characters on a white background.

The datasets for ALPR are generally very unbalanced in terms of character classes due to LP allocation policies. In Brazil, for example, one letter can appear much more often than others according to the state in which the LP was issued [30, 32]. It is well-known that unbalanced data is undesirable for neural network classifiers since the learning of some patterns might be biased. To address this issue, we employ the data augmentation technique proposed in [13], which consists of permuting on the LPs the characters overrepresented in the training set by those underrepresented. Using this technique, we are able to create a balanced set of images in which the order and frequency of the characters on the LPs are chosen to uniformly distribute them along the positions, maintaining the initial arrangement of letters and digits of each LP. In this way, the network might also learn the positions of letters and digits in certain LP layouts. It should be noted that the coordinates of each character (i.e., its bounding box) are necessary to apply this data augmentation technique.

Figure 4.13 shows some artificially generated images when applying the aforementioned method to LPs of different layouts. Following [13, 72], we also perform random variations of brightness, rotation and cropping to increase even more the diversity of the generated images.

As mentioned in [72], the adjustment of parameters is of paramount importance for the effectiveness of this technique since the presence of very large variations in brightness, rotation or cropping, for example, might impair the recognition through the generation of images that do not match real scenarios. Therefore, we empirically adjusted the parameters through visual inspection, i.e., brightness variation of the pixels [0.85; 1.15], rotation angles between $-5°$ and $5°$ and cropping from -2% to 8% of the LP size. Once these ranges were established, new images were generated using random values within those ranges for each parameter.

Remark that it would be possible to design/train a specific network for LPs of each country or region, as a large number of images (i.e., hundreds of thousands, or millions) can be generated using data augmentation strategies. However, a lot of manual effort would be required since the model and its parameters would have to be adjusted separately for each layout class. For example, a network fine-tuned to recognize Taiwanese LPs would probably have a different input size from a network designed for European LPs, as the aspect ratios of the LPs issued in those regions are generally quite different. As another example, shallower models could be employed specifically for LPs with simpler backgrounds (e.g., Brazilian and Chinese LPs).

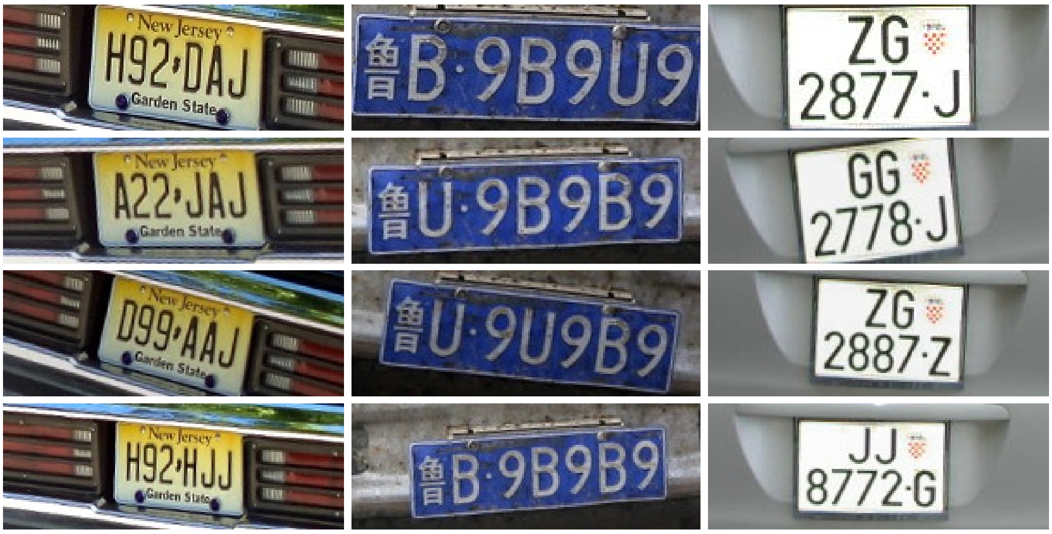


Figure 4.13: Examples of LP images generated using the data augmentation technique proposed in [13]. The images in the first row are the originals and the others were generated automatically. In the columns, LPs of different layouts are shown. From left to right: American, Chinese and European LPs.

5 Experimental Results

In this chapter, we report the experiments carried out to verify the effectiveness of the proposed ALPR system. First, we present the datasets and evaluation protocol used in our experiments. Afterward, the results achieved (in terms of both accuracy and execution time) are reported and compared with those obtained in previous works and by commercial systems. All experiments were performed on a computer with an AMD Ryzen Threadripper 1920X 3.5GHz CPU, 32 GB of RAM and an NVIDIA Titan Xp GPU (3,840 CUDA cores and 12 GB of RAM).

The Darknet framework [69] was employed to train and test our networks. However, we used the AlexeyAB's version of Darknet [106], which has several improvements over the original[1], including improved neural network performance by merging two layers into one (convolutional and batch normalization), optimized memory allocation during network resizing, and many other code fixes. For more details on this repository, refer to [106].

We also made use of the Darknet's built-in data augmentation, which creates a number of randomly cropped and resized images with changed colors (hue, saturation, and exposure). We manually implemented the flip operation only for the vehicle detection stage, as this operation would probably impair the layout classification and LP recognition tasks. Similarly, we disabled the color-related data augmentation for the LP detection and layout classification stage, as explained in Section 5.3.2.

All image resizing operations were performed using bilinear interpolation, which is implemented in the Darknet framework and used by default in the OpenCV library [114].

5.1 Datasets

In addition to the UFPR-ALPR dataset, the experiments were carried out in seven publicly available datasets: *Caltech Cars*, *EnglishLP*, *UCSD* (*Stills* subset), *ChineseLP*, *AOLP*, *OpenALPR-EU* and *SSIG-SegPlate*. These datasets are often used to evaluate ALPR systems, contain multiple LP layouts and were collected under different conditions/scenarios (e.g., with variations in lighting, camera position and settings, and vehicle types). In the following subsections, these datasets are presented and briefly described in chronological order. An overview of the datasets (including the proposed one) is presented in Table 5.1. It is noteworthy that in most of the works in the literature no more than three datasets were used in the experiments.

In our experiments, we did not make use of two datasets proposed recently: AOLPE [40] (an extension of the AOLP dataset) and Chinese City Parking Dataset (CCPD) [117]. The former has not yet been made available by the authors, who are collecting more data to make it even more challenging. The latter, although already available, does not provide the position of the vehicles and the characters in its 250,000 images and it would be impractical to label them to train/evaluate our networks (in [117], more than 100,000 images were used for training).

[1] The AlexeyAB's GitHub repository is forked from `https://github.com/pjreddie/darknet`.

Table 5.1: An overview of the datasets used in our experiments.

Dataset	Year	# Images	Resolution	LP Layout	Evaluation Protocol
Caltech Cars [108]	1999	126	896 × 592	American	No
EnglishLP [109]	2003	509	640 × 480	European	No
UCSD-Stills [110, 115]	2005	291	640 × 480	American	Yes
ChineseLP [45]	2012	411	Various	Chinese	No
AOLP [35, 107]	2013	2,049	Various	Taiwanese	No
OpenALPR-EU [46]	2016	108	Various	European	No
SSIG-SegPlate [32, 116]	2016	2,000	1,920 × 1,080	Brazilian	Yes
UFPR-ALPR	2018	4,500	1,920 × 1,080	Brazilian	Yes

5.1.1 *Caltech Cars*

The Caltech Cars dataset [108] was recorded in 1999 by Markus Weber during his doctorate at the California Institute of Technology. The dataset is composed of 126 rear-view images taken in parking lots with a resolution of 896 × 592 pixels. All images have only one car in the foreground and were captured during the daytime with the same hand-held camera at approximately the same distance. Figure 5.1 shows some images belonging to this dataset. There are no motorcycles and larger vehicles such as buses and trucks.

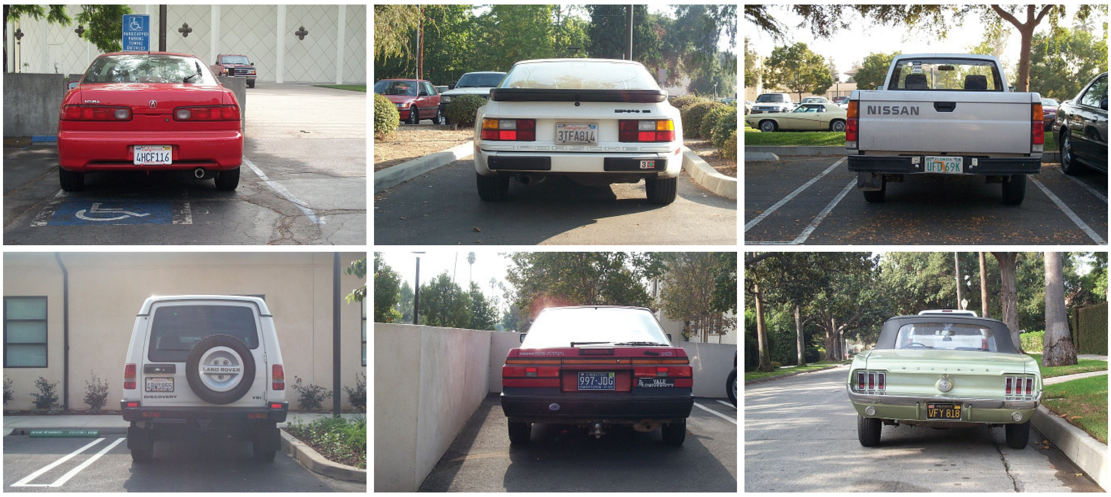


Figure 5.1: Some sample images of the Caltech Cars dataset [108]. Despite the fact that there are LPs of different layouts in this dataset, most vehicles have California-issued LPs.

Although it has been widely used in the ALPR context [14, 27, 45, 118, 119], the Caltech Cars dataset was created to validate object recognition algorithms and has no annotations. Therefore, it is necessary to manually label the LP position and characters in order to evaluate ALPR algorithms. As this dataset also does not have an evaluation protocol, other datasets were employed to train the proposed algorithms in some works [14, 45, 118], whereas in others the 126 images were randomly split into training and test sets. In [119–121], for example, 80 images were used for training and 46 for testing.

5.1.2 *EnglishLP*

In 2003, Vlasta Srebrić created the EnglishLP dataset [109] to evaluate procedures for improving the contrast of grayscale images [122]. Even though no particular name was given to the dataset, it is commonly referred to as EnglishLP dataset because it has only characters of the English alphabet [22, 87, 88]. This dataset, collected in Croatia, consists of 509 images (640 × 480 pixels) acquired using a hand-held camera under different weather conditions (sunny, cloudy, rainy) at different times of the day (morning, afternoon, evening, night). The images are divided into six folders, one for each day the images were collected. As can be seen in Figure 5.2, there are several LP layouts and different types of vehicles such as cars, buses and trucks (one vehicle per image). Nevertheless, there are no motorcycles.

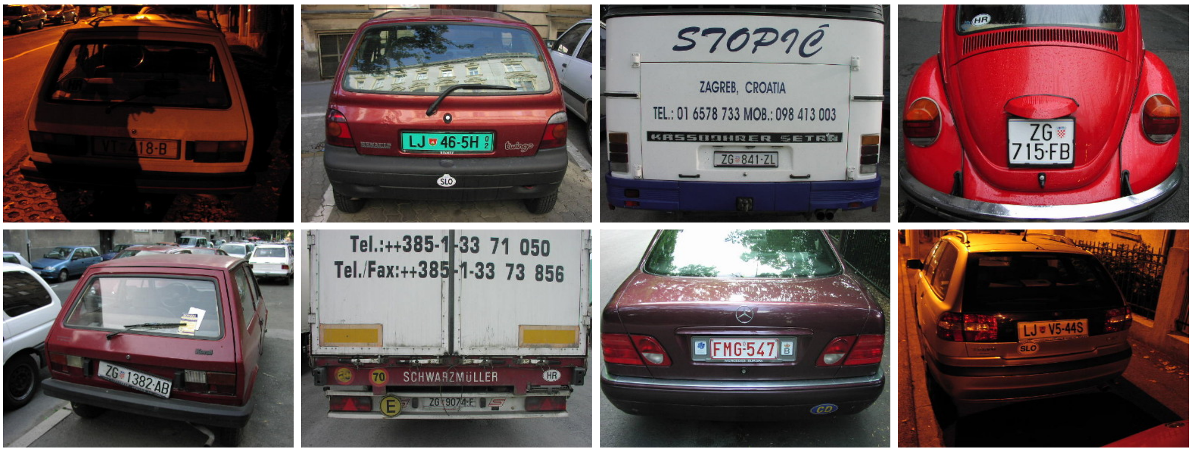


Figure 5.2: Some examples from the EnglishLP dataset [109]. All images are from the rear of the vehicle.

The EnglishLP dataset has neither annotations nor evaluation protocol. As expected, different protocols were employed in previous works to evaluate their approaches. For example, about 80% of the images were used for training and the remainder for testing in [87]. In [18, 22], on the other hand, the dataset was combined with others to create larger training and test sets. Examples of ALPR works using this dataset are [18, 22, 87, 88, 123].

5.1.3 *UCSD-Stills*

The UCSD dataset [110] was introduced in 2005 by Louka Dlagnekov as part of his master's thesis at the University of California, San Diego [115]. This dataset was created to assist the development and evaluation of algorithms for LP, make and model recognition.

The dataset, which is available by request, is divided into three subsets: *Gilman*, *Regents* and *Stills*. The first two subsets were captured by cameras mounted on top of street lamp poles overlooking stop signs. The Stills subset, on the other hand, contains images taken with a hand-held camera in various parking lots at a closer distance. As the LPs are not legible in the Gilman and Regents subsets, only the 291 images from the Stills subset, hereinafter referred to as UCSD-Stills dataset, are used in this work. Some sample images are shown in Figure 5.3.

All images of the UCSD-Stills dataset were taken during the daytime with a resolution of 640 × 480 pixels. The images are from passenger cars only, i.e., there are no motorcycles nor larger vehicles such as buses and trucks. As in the Caltech Cars dataset, there are LPs of different layouts in this dataset, however, most vehicles have LPs issued in California, United States.

The UCSD-Stills dataset is divided into a training set (186 images) and a test set (60 images). Thus, there are 45 images that are not related to any subset. We assume that these images

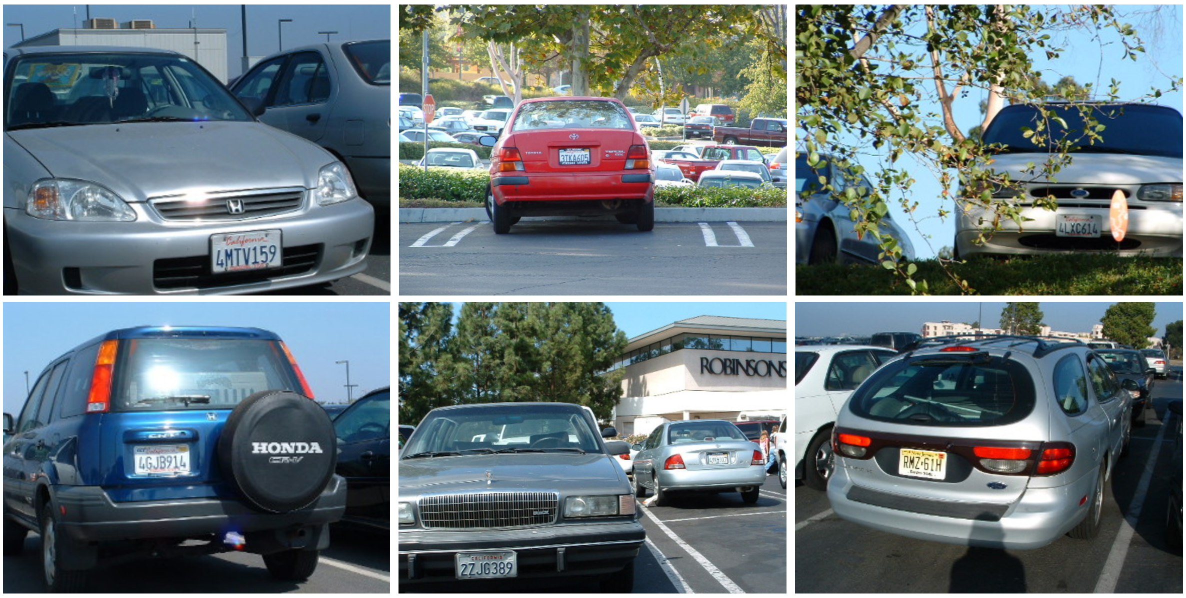


Figure 5.3: Sample images of the UCSD-Stills dataset [110]. There are images captured from the front view and others from the rear view of the vehicles. In addition, there might be vehicles in the background (see the middle image on the bottom row), however, only the position of the LP in the foreground is labeled in each image.

are intended to be used for validation. Regarding the annotations, the position (x,y,w,h) of the LP (the one in the foreground) was labeled in each image. To the best of our knowledge, this was the first dataset to provide annotations on the positions of the LPs. Examples of works that used this dataset are [41, 118, 124]. Further information can be seen in [110, 115].

5.1.4 *ChineseLP*

Zhou et al. [45] built a dataset containing 411 vehicle images (mostly of passenger cars) with Chinese LPs to evaluate their LP detection approach based on principal visual word discovery. The dataset, hereinafter referred to as ChineseLP, was created in 2012 and is available by request.

The ChineseLP dataset consists of 252 images captured by the authors and 159 images downloaded from the internet. In this way, the images present great variations in resolution (from 143 × 107 to 2048 × 1536 pixels), illumination and background. This dataset was acquired mostly by hand-held cameras, although cameras mounted on the dashboards of vehicles were also used in some cases. As shown in Figure 5.4, the images were taken during the daytime, from both front and rear views, multiple vehicles might exist in one image, and there are no motorcycles.

Despite the fact that the dataset has no evaluation protocol or annotations, it has been employed (commonly jointly with other datasets) to train and also to evaluate ALPR algorithms [14, 24, 125, 126]. One can refer to [45] for more information regarding this dataset.

5.1.5 *AOLP*

Hsu et al. [35] collected the Application-Oriented License Plate (AOLP) dataset [107] to verify the proposition that ALPR is better handled in an application-oriented way. This dataset, created in 2013, is categorized into three subsets: Access Control (AC), traffic Law Enforcement (LE), and Road Patrol (RP). The AC subset (681 images) refers to cases in which a vehicle traverses in a fixed passage at a reduced speed or with a full stop, such as at a toll station or the entrance/exit of private spaces. The LE subset (757 images) makes reference to situations in which a vehicle violates traffic laws and is captured by a roadside camera. Finally, the RP subset (611 images)

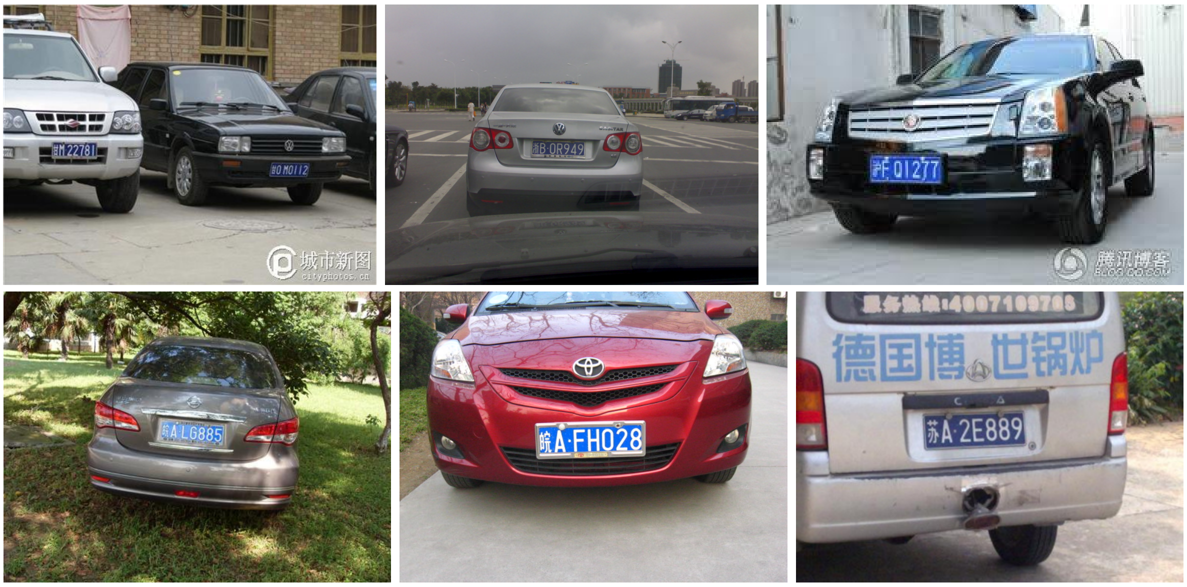


Figure 5.4: Example images from the ChineseLP dataset [45]. The images from the first row were captured by the authors, while the images from the second row were downloaded from the internet.

refers to cases in which the camera is installed on a patrol car, which takes images of vehicles with arbitrary viewpoints and distances [35]. Figure 5.5 shows sample images from each subset.

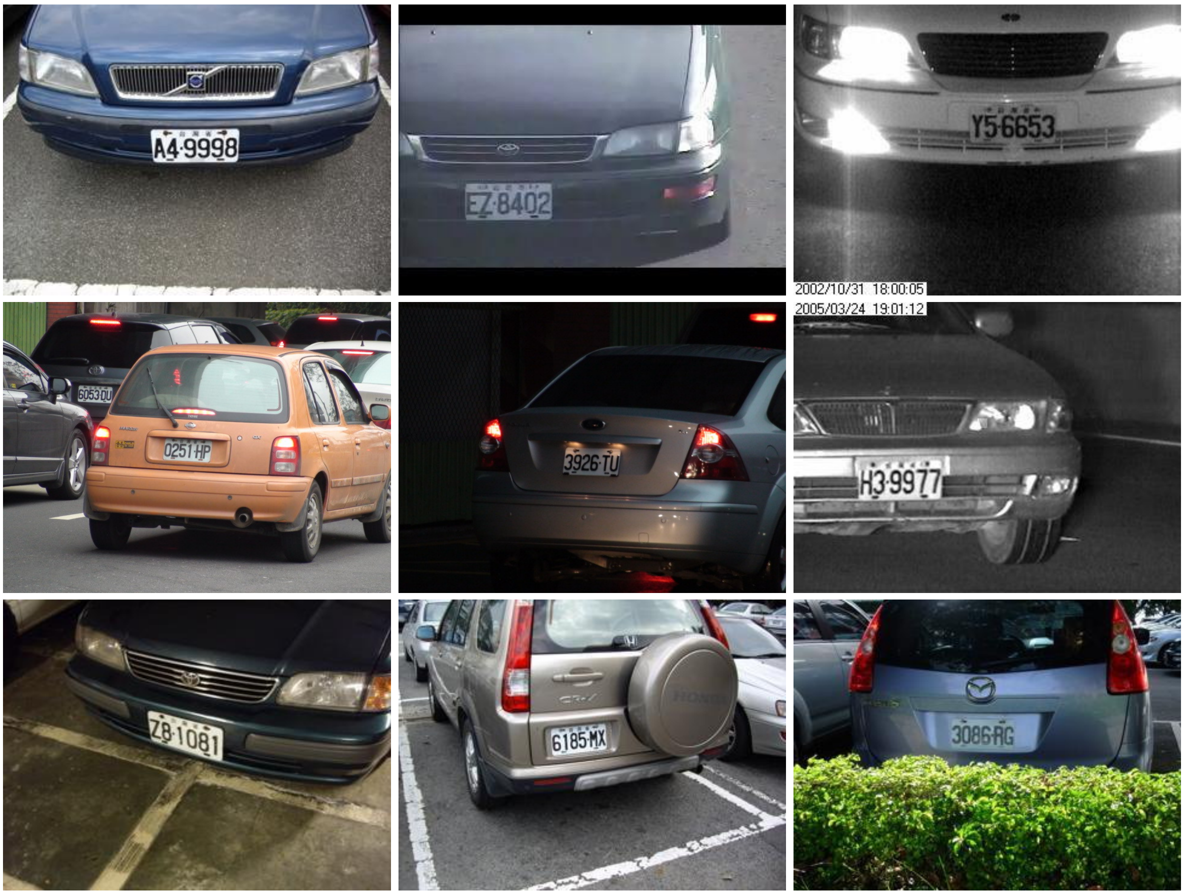


Figure 5.5: Sample images from each subset of the AOLP dataset [35]. The first row shows images of the AC subset, while the second and third rows show images of the LE and RP subsets, respectively.

All 2,049 images were collected in Taiwan, from front/rear views and various locations, time, traffic, and weather conditions. The images have a resolution between 320 × 240 and 640 × 480 pixels. To the best of our knowledge, this dataset was the first to contain annotations regarding the LP position and also its text, assisting the development and evaluation of new ALPR approaches. Note that, in some cases, there is more than one vehicle in the image. Furthermore, LPs of motorcycles are not labeled, even when they are in the foreground.

The AOLP dataset does not have a well-defined evaluation protocol. Therefore, the dataset was divided in several ways in previous works. In [35, 118], for example, 387 images were randomly chosen for training and 1,662 for testing. Alternatively, Xie et al. [41] randomly divided the data into training and test sets with a 2:1 ratio, whereas Li et al. [17] used images from different subsets for training and testing, for example, they used images from the LE and RP subsets to train their network, and evaluated it on the AC subset. These two protocols were employed by Zhuang et al. [95] to assess their approach. Other authors [14, 92, 127] chose to train their methods on images from other public datasets and evaluate them on all images of the AOLP dataset.

A license agreement is required for downloading the dataset, and more details can be found in [35, 107].

5.1.6 *OpenALPR-EU*

OpenALPR [102] is an ALPR library written in C++ with bindings in other languages, such as C#, Java and Python, that is distributed in both commercial and open source versions. In 2016, Matthew Hill (i.e., the founder of OpenALPR) provided a public dataset containing 108 images of vehicles with European LPs, hereinafter referred to as OpenALPR-EU [46], to support the development of the open source version. Some images are shown in Figure 5.6.

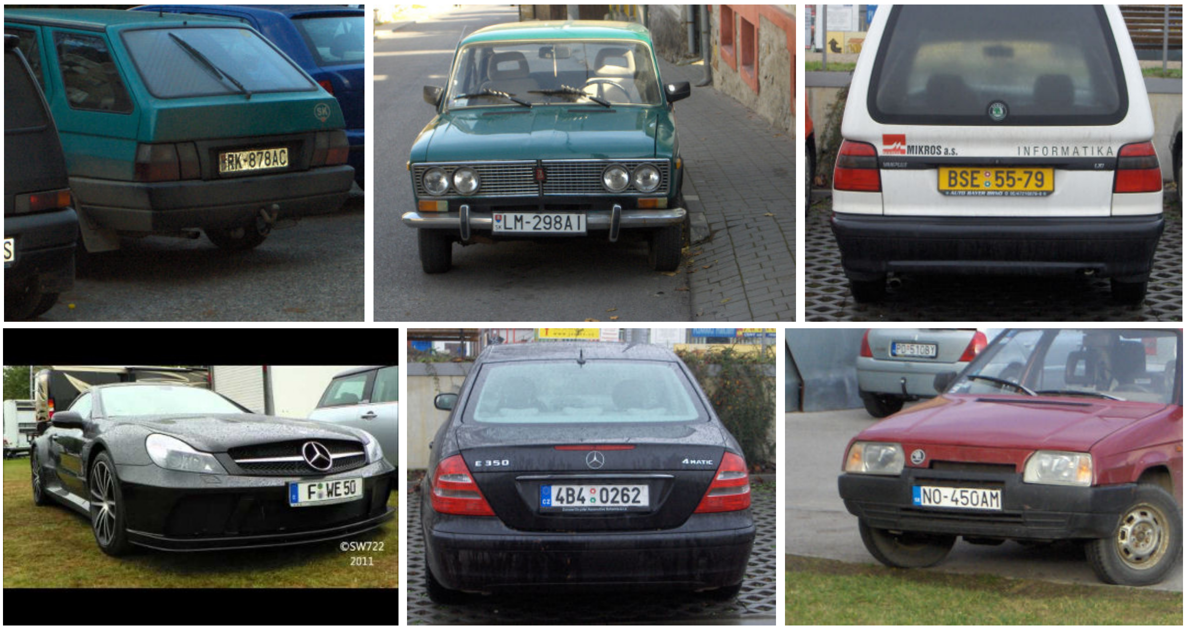


Figure 5.6: Some images from the OpenALPR-EU dataset [46]. In general, the vehicle occupies a large portion of the image and is well centered. Although in some cases there are legible LPs of other vehicles in the background (e.g., the bottom right image), only the position and text of the LP in the foreground are labeled in this dataset.

The images, which have a resolution from 326 × 249 to 2048 × 1536 pixels, were acquired during the daytime, predominantly by hand-held cameras, from the front and rear views of the vehicles. There are vehicles, mostly passenger ones, with LPs issued in several European countries such as Slovakia, Germany, Czech Republic, Norway, among others. Remark that the vehicles are well centered and there are no motorcycles in this dataset.

As ground truth, the position and text of the LP were manually labeled in each image. Although in some cases there are legible LPs of other vehicles in the background, only information regarding the LP in the foreground are provided. Considering that the OpenALPR-EU dataset contains only 108 images and has no evaluation protocol, all its images were used for testing in [18, 33], while other datasets were employed for training and validation.

5.1.7 *SSIG-SegPlate*

In 2016, Gonçalves et al. [32] pointed out that existing ALPR datasets did not provide annotations referring to the bounding boxes of the LP characters, which are essential to evaluate character segmentation techniques. Therefore, they proposed a benchmark for LP character segmentation, including an evaluation measure that is more suitable for this problem, and the SSIG-SegPlate dataset [116]. To the best of our knowledge, this is the first public dataset for ALPR to provide manual annotations on the position of both the LPs and the characters, as well as the character classes. This is quite important since it allows a quantitative evaluation of both character segmentation and recognition methods.

The SSIG-SegPlate dataset consists of 2,000 high-resolution frames (1,920 × 1,080) from 101 vehicles (passenger cars, buses and trucks; there are no motorcycles) taken with a static camera. There are several frames of each vehicle (19.80, on average). In this way, redundant information may be used to improve the recognition results [32]. All images were captured from the front view of the vehicles during the daytime on a campus of the Federal University of Minas Gerais, Brazil. Three sample frames are shown in Figure 5.7. In some cases, there are more than one vehicle/LP visible in the foreground, however, only one LP was labeled in each image.

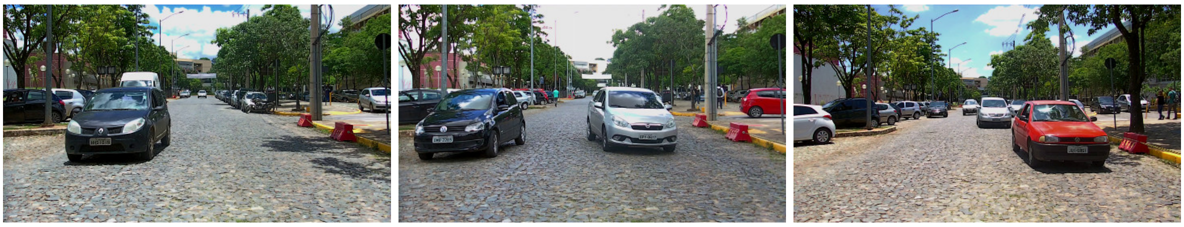


Figure 5.7: Sample frames of the SSIG-SegPlate dataset [32]. It should be noted that only one LP is labeled in each frame, that is, there are vehicles/LPs in the background (even in the foreground) that do not have annotations.

The SSIG-SegPlate dataset is divided into several folders (each folder has images of only one vehicle) and uses the following evaluation protocol: 40% of the images for training, 20% for validation and 40% for testing. According to the authors, this protocol was adopted because many character segmentation approaches do not require model estimation and a larger test set allows the reported results to be more statistically significant [32].

A license agreement is needed for downloading the SSIG-SegPlate dataset, and further information can be seen in [32, 116]. This dataset was employed for ALPR in [11, 13, 30, 33].

5.1.8 Discussion

As the datasets employed in this work were proposed over the last 20 years by different research groups from different countries, the datasets present a great variety in the way they were collected. Thus, the proposed system is evaluated in images with unique characteristics, which simulate distinct real-world scenarios and applications. It should be noted that only public datasets were used for both training and testing our approach to enable fair comparisons in future works.

Most datasets have no annotations or contain labels for a single stage only (e.g., LP detection), despite the fact that they are often used to train/evaluate algorithms in the ALPR context. Therefore, in all images of these datasets, we manually labeled the position of the vehicles (including those in the background where the LP is also legible), LPs and characters, as well as their classes[2]. In the AOLP dataset, unlike the others, the exact region containing the characters was originally labeled as the position of the LP, without any margin and not including the LP frame. Due to that, we discard those labels and annotated the positions of the LPs in that dataset to avoid inconsistency among the labels of different datasets.

In addition to using the training images of the datasets presented in this section, we downloaded and labeled more 772 images from the internet to train all stages of our ALPR system. This procedure was adopted to eliminate biases from the datasets employed in our experiments. For example, the datasets collected in the United States (i.e., Caltech Cars and UCSD-Stills) have similar characteristics (e.g., there is one vehicle per image, the vehicle is centered and occupies a large portion of the image, and the resolutions of the images are not high), which are different from those of the other datasets. Moreover, there are many more examples of Brazilian and Taiwanese LPs in our training data (note that the exact number of images used for training, testing and validation in each dataset is detailed in the next section). Therefore, we downloaded images containing vehicles with American, Chinese and European LPs so that there are at least 500 images of LPs of each class/region to train our networks. Specifically, we downloaded 257, 341, and 174 images containing American, Chinese and European LPs, respectively[3].

It is important to emphasize that these additional images are essential for the robustness of the proposed ALPR system, as many of them were acquired under conditions different from those generally found in public datasets such as rainy or snowy days, as well as images captured at night. In Figure 5.8, some examples are shown.

5.2 Evaluation Protocol

To evaluate the stages of (i) vehicle detection and (ii) LP detection and layout classification, we report the precision and recall rates (described in Section 2.1) achieved by our networks. Each metric has its importance since, for system efficiency, all vehicles/LPs must be detected without many false positives. Note that the precision and recall rates are equal in the LP detection and layout classification stage because we consider only one LP per vehicle.

We consider as correct only the detections with IoU greater than 0.5 with the ground truth. This bounding box evaluation, defined in the PASCAL VOC Challenge [38] and employed in previous works [11, 17, 27], is interesting since it penalizes both over- and under-estimated objects. In the LP detection and layout classification stage, we assess only the predicted bounding box on LPs classified as undefined layout (see Section 4.2.2). In other words, we consider as correct the predictions when the LP position is correctly predicted but not its layout, as long as the LP (and its layout) has not been predicted with a high confidence value (i.e., below 0.75).

In the LP recognition stage, we report the number of correctly recognized LPs divided by the total number of LPs in the test set. A correctly recognized LP means that all characters on

[2] All annotations made by us are publicly available to the research community at https://web.inf.ufpr.br/vri/publications/layout-independent-alpr/.

[3] The images were downloaded from www.platesmania.com. Their download links and annotations are also publicly available at https://web.inf.ufpr.br/vri/publications/layout-independent-alpr/.

Figure 5.8: Examples of images downloaded from `www.platesmania.com` that were used to train our system. Some of them were acquired under conditions different from those generally found in public datasets such as rainy or snowy days, as well as images captured at night. In this way, we prevent overfitting in certain scenarios.

the LP were correctly recognized. As pointed out by Gonçalves et al. [10], OCR approaches must work as close as possible to the optimality in the ALPR context, as a single character recognized incorrectly may imply in incorrect identification of the vehicle.

As detailed in the previous section, only three of the eight datasets used in this work contain an evaluation protocol (defined by the respective authors) that can be reproduced perfectly: UCSD-Stills, SSIG-SegPlate and UFPR-ALPR. Thus, we split their images into training, validation, and test sets according to their own protocols (described in Section 4.1 and Section 5.1). We randomly divided the other five datasets using the protocols employed in previous works, aiming at a fair comparison with them. In the next paragraph, such protocols (which we also provide for reproducibility purposes) are specified.

We used 80 images of the Caltech Cars dataset for training and 46 for testing, as in [119–121]. Then, we employed 16 of the 80 training images for validation (i.e., 20%). The EnglishLP dataset was divided in the same way as in [87], with 80% of the images being used for training and the remainder for testing. Also in this dataset, 20% of the training images were employed for validation. Regarding the ChineseLP dataset, we did not find any previous work in which it was split into training/test sets, that is, all its images were used either to train or to test the methods proposed in [14, 24, 125, 126], often jointly with other datasets. Thus, we adopted the same protocol of the SSIG-SegPlate and UFPR-ALPR datasets, in which 40% of the images are used for training, 40% for testing and 20% for validation. As pointed out in the previous section, the AOLP dataset has been divided in several ways in the literature. In this work, we randomly divided each subset of the AOLP dataset into training and test sets with a 2:1 ratio, following [41, 95]. Then, 20% of the training images were employed for validation. Lastly, all images belonging to the OpenALPR-EU dataset were used for testing in [18, 33], while other public or private datasets were employed for training. Therefore, we also did not use any image of this dataset for training or validation, only for testing. An overview of the number of images used for training, testing and validation in each dataset can be seen in Table 5.2.

Table 5.2: An overview of the number of images used for training, testing and validation in each dataset.

Dataset	Training	Validation	Testing	Discarded	Total
Caltech Cars [108]	62	16	46	2	126
EnglishLP [109]	326	81	102	0	509
UCSD-Stills [110, 115]	181	39	60	11	291
ChineseLP [45]	159	79	159	14	411
AOLP [35, 107]	1,093	273	683	0	2,049
OpenALPR-EU [46]	0	0	108	0	108
SSIG-SegPlate [32, 116]	789	407	804	0	2,000
UFPR-ALPR	1,800	900	1,800	0	4,500

We discarded a few images from the Caltech Cars, UCSD-Stills, and ChineseLP datasets[4]. Despite the fact that most images in these datasets are reasonable, there are a few exceptions where (i) it is impossible to recognize the vehicle's LP due to occlusion, lighting or image acquisition problems, among other factors; (ii) the image does not represent real ALPR scenarios, for example, a person holding an LP. Three examples are shown in Figure 5.9. Such images were also discarded in [18].

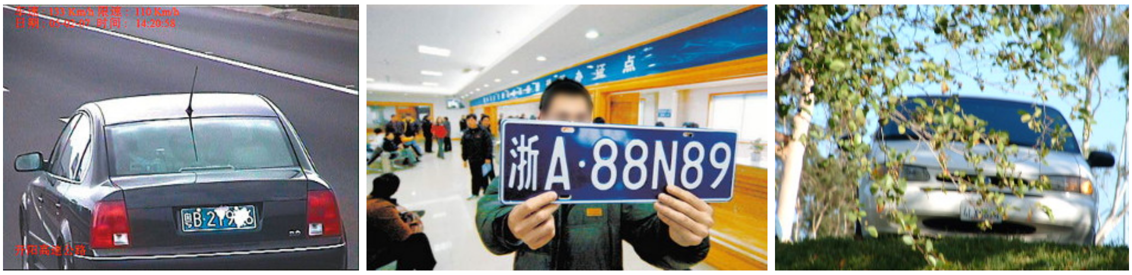


Figure 5.9: Examples of images discarded in our experiments.

It is worth noting that we did not discard any image from the test set of the UCSD-Stills dataset and used the same number of test images in the Caltech Cars dataset. In this way, we can fairly compare our results with those obtained in previous works. In fact, we used fewer images from those datasets to train and validate our networks. In the ChineseLP dataset, on the other hand, we first discard the few images with problems and then split the remaining ones using the same protocol as the SSIG-SegPlate and UFPR-ALPR datasets (i.e., 40/20/40% for training, validation and testing, respectively) since, in the literature, a division protocol has not yet been proposed for the ChineseLP dataset, to the best of our knowledge.

To avoid an overestimation or bias in the random division of the images into the training, validation and test subsets, we report in each stage the average result of five runs of the proposed approach. Therefore, at each run, the images of the datasets that do not have an evaluation protocol were randomly redistributed into each subset (training/validation/test). In the UCSD-Stills, SSIG-SegPlate and UFPR-ALPR datasets, we employed the same division (i.e., the one proposed along with the respective dataset) in all runs.

As pointed out in Section 5.1.8, we manually labeled the vehicles (including motorcycles) in the background of the images in cases where their LPs are legible. Nevertheless, in the testing

[4] The list of all images not used in our experiments can be found at `https://web.inf.ufpr.br/vri/publications/layout-independent-alpr/`.

phase, we considered only the vehicles/LPs originally labeled in the datasets that have annotations (e.g., AOLP and SSIG-SegPlate) to perform a fair comparison with previous works.

5.3 Results

In this section, we first assess the detection stages separately since the regions used in the LP recognition stage are from the detection results, rather than cropped directly from the ground truth. This is done to provide a realistic evaluation of the entire ALPR system, in which well-performed vehicle and LP detections are essential for achieving outstanding recognition results.

Afterward, the entire ALPR system is evaluated and the results are compared with those obtained in previous works and by commercial systems.

5.3.1 Vehicle Detection

In this stage, we employed a confidence threshold of 0.25 (defined empirically) to detect as many vehicles as possible, while avoiding high FP rates and, consequently, a higher cost of the proposed ALPR system. The following parameters were used for training the network: 60K iterations (max batches) and learning rate = [10^{-3}, 10^{-4}, 10^{-5}] with steps at 48K and 54K iterations.

The vehicle detection results are presented in Table 5.3. According to the detection evaluation described in Section 5.2, in the average of five runs, our approach achieved a recall rate of 99.92% and a precision rate of 98.37%. It is remarkable that the YOLOv2 model was able to correctly detect all vehicles (i.e., recall = 100%) in 5 of the 8 datasets used in the experiments. Some detection results are shown in Figure 5.10. As can be seen, well-located predictions were attained on vehicles of different types and under different conditions.

Table 5.3: Vehicle detection results achieved by the YOLOv2 model in all datasets.

Dataset	Precision (%)	Recall (%)
Caltech Cars	100.00 ± 0.00	100.00 ± 0.00
EnglishLP	99.04 ± 0.96	100.00 ± 0.00
UCSD-Stills	97.42 ± 1.40	100.00 ± 0.00
ChineseLP	99.26 ± 1.00	99.50 ± 0.52
AOLP	96.92 ± 0.37	99.91 ± 0.08
OpenALPR-EU	99.27 ± 0.76	100.00 ± 0.00
SSIG-SegPlate	95.47 ± 0.62	99.98 ± 0.06
UFPR-ALPR	99.57 ± 0.07	100.00 ± 0.00
Average	**98.37 ± 0.65**	**99.92 ± 0.08**

To the best of our knowledge, with the exception of our previous work [30], there is no other work in the ALPR context where both cars and motorcycles are detected at this stage. This is of paramount importance since motorcycles are one of the most popular transportation means in metropolitan areas, especially in Asia [103]. Although motorcycle LPs may be correctly located by LP detection approaches that work directly on the frames, they can be detected with fewer false positives if the motorcycles are detected first [9].

The precision rates obtained by the network were only not higher due to unlabeled vehicles present in the background of the images, especially in the AOLP and SSIG-SegPlate datasets. Three examples are shown in Figure 5.11a. In fact, one of the reasons we did not train

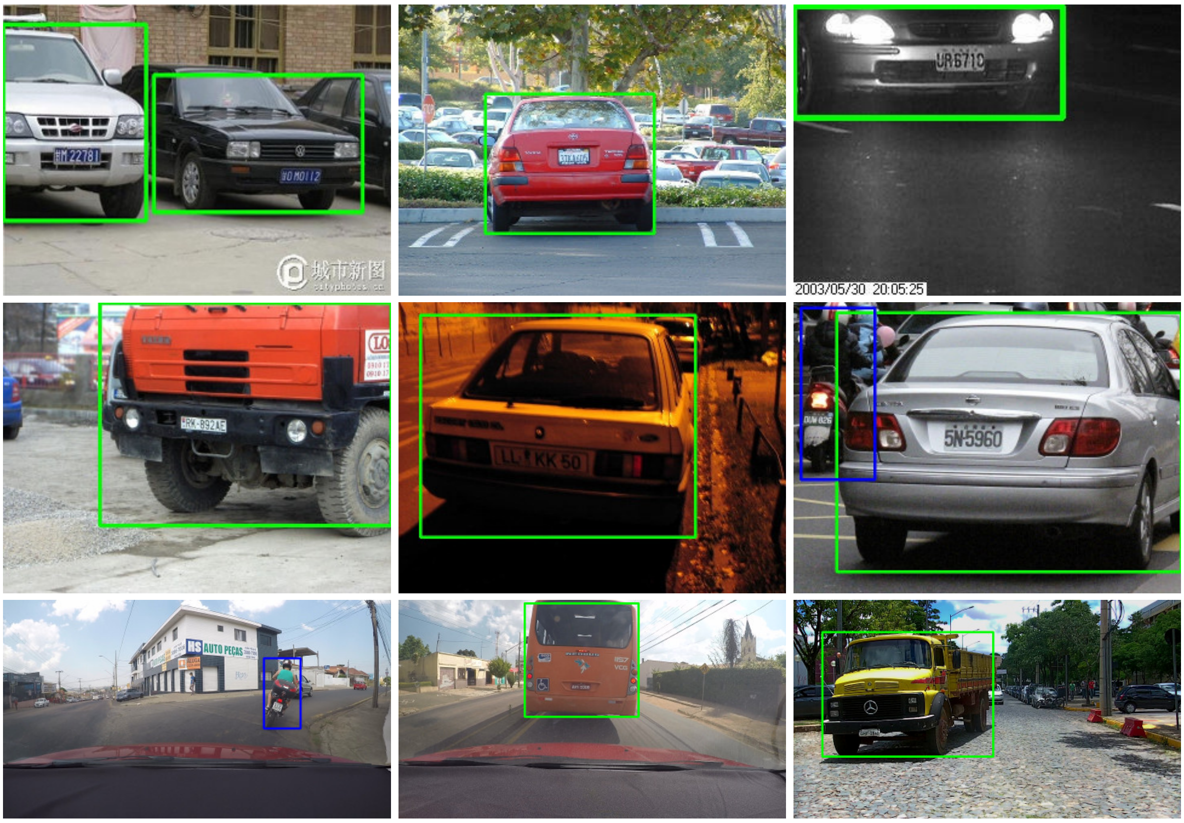


Figure 5.10: Some detection results achieved by the YOLOv2 model in different datasets. For better viewing, motorcycles' bounding boxes were drawn in blue, while the bounding boxes of other vehicles were drawn in green. Observe that vehicles of different types (cars, motorcycles, buses and trucks) were correctly detected regardless of lighting conditions (daytime and nighttime), occlusion, camera distance, and other factors.

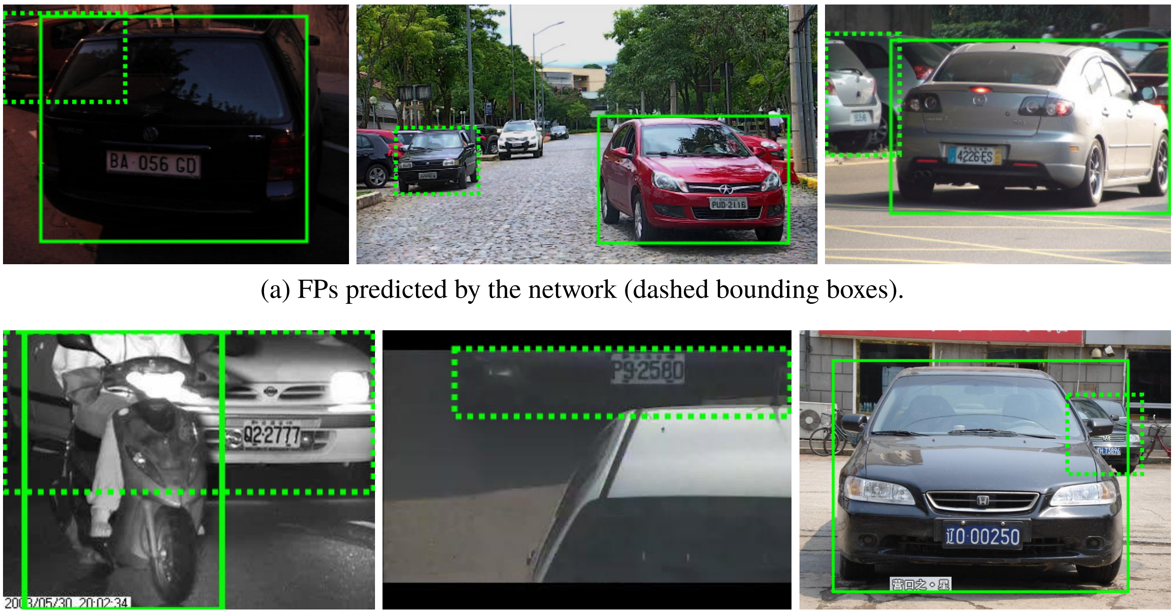
our vehicle detector using the large-scale CompCars dataset [128] is that many vehicles in the background (including those in which the LP is not visible/legible) would also be detected.

In Figure 5.11b, we show some of the few cases where our network failed to detect one or more vehicles in the image. As can be seen, such cases are challenging since only a small part of each undetected vehicle is visible. It is worth mentioning that in situations where a vehicle is detected with IoU ≤ 0.5 with the ground truth, it is still possible to detect its LP in the next stage since the LP may be within the ROI after adding a margin to it, as explained in Section 4.2.2.

5.3.2 License Plate Detection and Layout Classification

In Table 5.4, we report the results obtained by the modified Fast-YOLOv2 network in the LP detection and layout classification stage. As we consider only one LP per vehicle image, the precision and recall rates are identical. The average recall rate obtained in all datasets was 99.51% when disregarding the vehicles not detected in the previous stage and 99.45% when considering the entire test set. This result is particularly impressive since we considered as incorrect the predictions in which the LP layout was incorrectly classified with a high confidence value, even in cases where the LP position was predicted correctly (as explained in Section 5.2).

According to Figure 5.12, the proposed approach was able to successfully detect and classify LPs of various layouts, including those with few examples in the training set such as LPs issued in the U.S. states of Connecticut and Utah, or LPs of motorcycles in Taiwan. It should be noted that, in some cases, the LP occupies a very small portion of the original image and therefore the vehicle detection stage is essential for the effectiveness of our ALPR system.



(a) FPs predicted by the network (dashed bounding boxes).

(b) Vehicles not predicted by the network (dashed bounding boxes).

Figure 5.11: FP and FN predictions obtained in the vehicle detection stage. As can be seen in (a), the predicted FPs are mostly unlabelled vehicles in the background. In (b), one can see that the vehicles not predicted by the network (i.e., the FNs) are predominantly those occluded or in the background.

Table 5.4: Results attained by the modified Fast-YOLOv2 network in the LP detection and layout classification stage. The recall rates achieved in all datasets when disregarding the vehicles not detected in the previous stage are presented in (a), while the recall rates obtained when considering the entire test set are listed in (b).

(a)

Dataset	Recall (%)
Caltech Cars	99.13 ± 1.19
EnglishLP	100.00 ± 0.00
UCSD-Stills	100.00 ± 0.00
ChineseLP	100.00 ± 0.00
AOLP	99.94 ± 0.08
OpenALPR-EU	98.52 ± 0.51
SSIG-SegPlate	99.83 ± 0.26
UFPR-ALPR	98.67 ± 0.25
Average	**99.51 ± 0.29**

(b)

Dataset	Recall (%)
Caltech Cars	99.13 ± 1.19
EnglishLP	100.00 ± 0.00
UCSD-Stills	100.00 ± 0.00
ChineseLP	99.63 ± 0.34
AOLP	99.85 ± 0.10
OpenALPR-EU	98.52 ± 0.51
SSIG-SegPlate	99.80 ± 0.24
UFPR-ALPR	98.67 ± 0.25
Average	**99.45 ± 0.33**

Some images where our network failed either to detect the LP or to classify the LP layout are shown in Figure 5.13. As can be seen in Figure 5.13a, our network failed to detect the LP in cases where there is a textual block very similar to an LP in the vehicle patch, or even when the LP of another vehicle appears within the patch (a single case in our experiments). This is due to the fact that one vehicle can be almost totally occluded by another. Regarding the errors in which the LP layout was misclassified, they occurred mainly in cases where the LP is considerably similar to LP of other layouts. For example, the left image in Figure 5.13b shows a European LP (which has exactly the same colors and number of characters as standard Chinese LPs) incorrectly classified as Chinese.

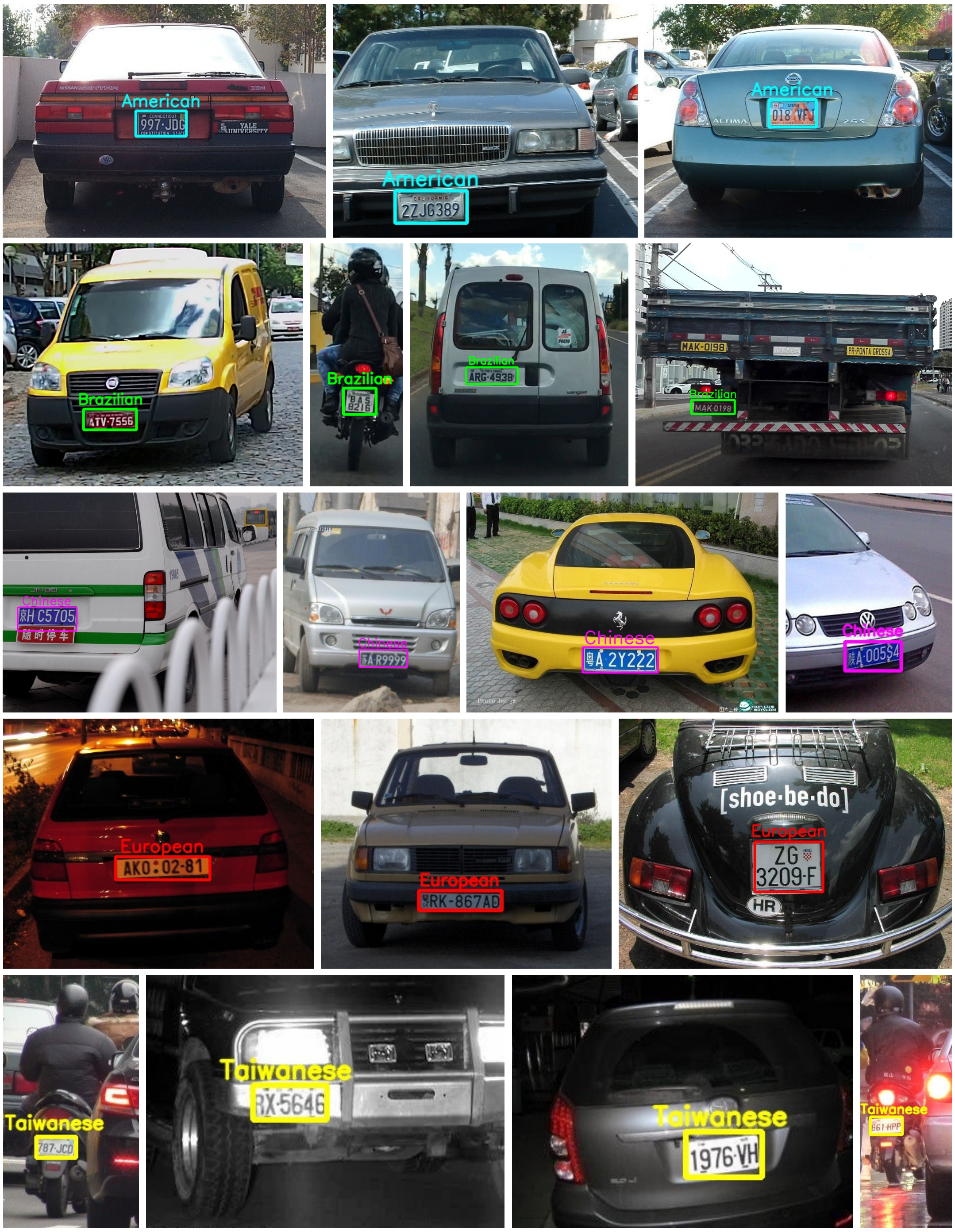
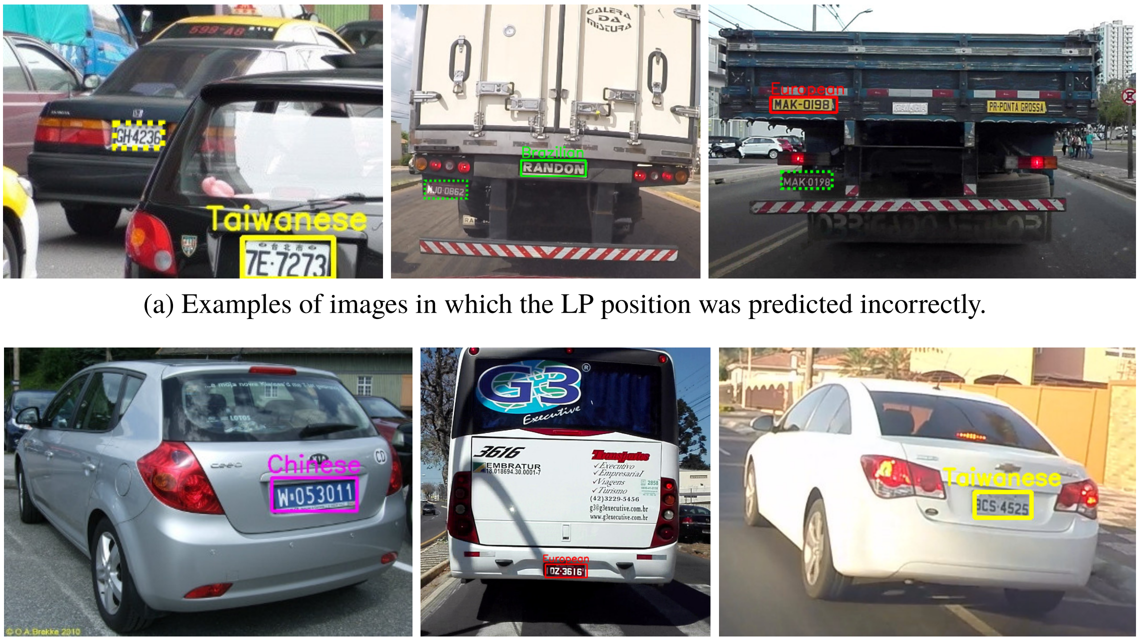


Figure 5.12: LPs correctly detected and classified by the proposed approach. Observe that the modified Fast-YOLOv2 model is robust for this task regardless of vehicle type, lighting conditions, camera distance, and other factors.

It is important to note that it is still possible to correctly recognize the characters in some cases where our network has failed at this stage. For example, in the right image in Figure 5.13a, the detected region contains exactly the same text as the ground truth (i.e., the LP). Moreover, a Brazilian LP classified as European (e.g., the middle image in Figure 5.13b) can still be correctly recognized in the next stage since the only post-processing rule we apply to European LPs is that they have between 5 and 8 characters (see Section 4.2.3).

Additionally, we could employ post-processing methods in the next stage in cases where more than one LP is detected, for example, evaluate on each detected LP if the number of detected/recognized characters is within the range defined for the predicted layout, or consider



(a) Examples of images in which the LP position was predicted incorrectly.

(b) Examples of images in which the position of the LP was predicted correctly, but not the layout. In the left image, the LP is European. In the middle and right ones, the LPs are Brazilian.

Figure 5.13: Some images in which our network failed either to detect the LP or to classify the LP layout.

only the LP in which the characters' confidence is greater. However, since the actual LP can be detected with very low confidence values (i.e., ≤ 0.1), many false negatives would have to be analyzed, increasing the overall computational cost of the system.

As mentioned earlier, in this stage we disabled the color-related data augmentation of the Darknet framework [69]. In this way, we eliminated more than half of the layout classification errors obtained when the model was trained using images with changed colors (this evaluation was performed in the validation set). We believe this is due to the fact that the network leverages color information (which may be distorted with some data augmentation approaches) for layout classification, as well as other characteristics such as the position of the characters and symbols on the LP.

5.3.3 License Plate Recognition (Overall Evaluation)

As in the vehicle detection stage, once the YOLOv2 model was trained, we first evaluated (in the validation set) different confidence threshold values in the validation set in order to miss as few characters as possible, while avoiding high FP rates. We adopted a 0.5 confidence threshold for all LPs except European ones, where a higher threshold (i.e., 0.65) was adopted since European LPs can have up to 8 characters and several FPs were predicted on LPs with fewer characters when using a lower confidence threshold.

It is important to note that we considered the '1' and 'I' characters as a single class in the assessments performed in the SSIG-SegPlate and UFPR-ALPR datasets, as those characters are identical but occupy different positions on Brazilian LPs. The same procedure was done in [33].

For each dataset, we compared the proposed ALPR system with state-of-the-art methods that were evaluated using the same protocol as the one described in Section 5.2. In addition, our results are compared with those obtained by two commercial systems[5]: *Sighthound* [18] and *OpenALPR*[6] [102]. According to the authors, both systems are robust for the detection and recognition of LPs of different layouts. It is important to emphasize that although the commercial systems were not tuned specifically for the datasets employed in our experiments, they are trained in much larger private datasets, which is a great advantage, especially in deep learning approaches. In addition, OpenALPR contains specialized solutions for LPs from different regions (e.g., China, Europe, among others) and the user must enter the correct region before using its API, that is, it requires prior knowledge regarding the LP layout. Sighthound, on the other hand, uses a single model/approach for LPs from different countries/regions, as well as the proposed system.

The results obtained in all datasets by the proposed ALPR system, previous works, and commercial systems are shown in Table 5.5. In the average of five runs, the proposed system correctly recognized 96.8% of the LPs, outperforming both previous works and commercial systems in the ChineseLP, OpenALPR-EU, SSIG-SegPlate and UFPR-ALPR datasets. In the other datasets, the proposed approach obtained competitive results to the best result attained by the baselines.

Table 5.5: Recognition rates (%) obtained by the proposed system, previous works, and commercial systems in all datasets used in our experiments. To the best of our knowledge, in the literature, only algorithms for LP detection and character segmentation were evaluated in the Caltech Cars, UCSD-Stills and ChineseLP datasets. Therefore, our approach is compared only with the commercial systems in these datasets.

Dataset	[87]	[95]	[33]	[13]	[30]	Sighthound	OpenALPR	Proposed*	Proposed
Caltech Cars	–	–	–	–	–	95.7 ± 2.7	**99.1 ± 1.2**	96.1 ± 1.8	98.7 ± 1.2
EnglishLP	**97.0**	–	–	–	–	92.5 ± 3.7	78.6 ± 3.6	95.5 ± 2.4	95.7 ± 2.3
UCSD-Stills	–	–	–	–	–	**98.3**	**98.3**	97.3 ± 1.9	98.0 ± 1.4
ChineseLP	–	–	–	–	–	90.4 ± 2.4	92.6 ± 1.9	95.4 ± 1.1	**97.5 ± 0.9**
AOLP	–	**99.8****	–	–	–	87.1 ± 0.8	–	98.4 ± 0.7	99.2 ± 0.4
OpenALPR-EU	–	–	93.5	–	–	92.6	90.7	95.7 ± 1.9	**96.9 ± 1.1**
SSIG-SegPlate	–	–	88.6	88.8	85.5	82.8	92.0	96.9 ± 0.5	**98.2 ± 0.5**
UFPR-ALPR	–	–	–	–	64.9	62.3	82.2	82.5 ± 1.1	**90.0 ± 0.7**
Average	–	–	–	–	–	87.7 ± 2.4	90.5 ± 2.3	94.7 ± 1.4	**96.8 ± 1.0**

* The proposed ALPR system assuming that all LP layouts were classified as undefined (i.e., without layout classification).

** The LP patches for the LP recognition stage were cropped directly from the ground truth in [95].

The proposed system attained results similar to those obtained by OpenALPR in the Caltech Cars dataset (98.7% against 99.1%, which represents a difference of less than one LP per run, on average, as are only 46 testing images), even though our system does not require prior knowledge. Regarding the EnglishLP dataset, our system performed better than the best baseline [87] in 2 of the 5 runs. Although we used the same number of images for testing, in [87] the dataset was divided only once and the images used for testing were not specified. In the UCSD-Stills dataset, both commercial systems reached a recognition rate of 98.3% while our system achieved 98% on average (with a standard deviation of 1.4%). Lastly, in the AOLP dataset, the proposed approach obtained similar results to those reported by Zhuang et al. [95], even though in their work the LP patches used as input in the LP recognition stage were cropped directly from the ground truth; in other words, they did not take into account vehicles or LPs not detected in the earlier stages, nor background noise in the LP patches due to less accurate LP detections.

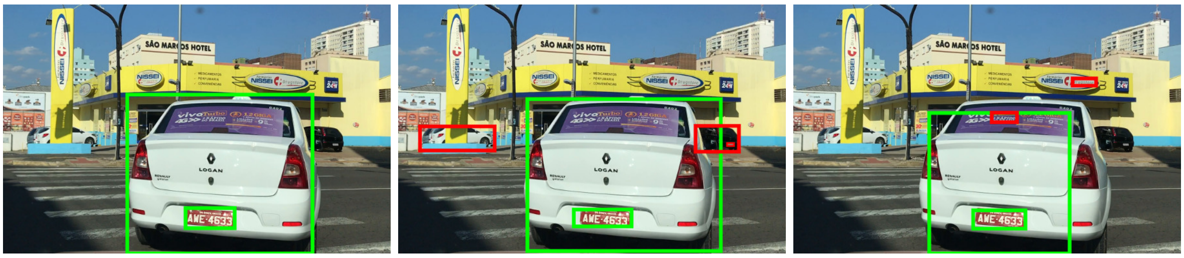
[5] OpenALPR and Sighthound systems have APIs available at `https://www.openalpr.com/cloud-api.html` and `https://www.sighthound.com/products/cloud`, respectively. The results presented here were obtained in January 2019.

[6] Although OpenALPR has an open source version, the commercial version uses different algorithms for OCR trained with larger datasets to improve accuracy [102].

To evaluate the impact of classifying the LP layout prior to LP recognition, we also report in Table 5.5 the results obtained when assuming that all LP layouts were classified as undefined and that a generic approach (i.e., without heuristic rules) was employed in the LP recognition stage. The mean recognition rate was improved by 2.1%. We consider this strategy (layout classification + heuristic rules) essential for accomplishing outstanding results in datasets that contain LPs with fixed positions for letters and digits (e.g., Brazilian and Chinese LPs), as the recognition rates attained in the ChineseLP, SSIG-SegPlate and UFPR-ALPR datasets were improved by 3.6%, on average.

The robustness of our ALPR system is remarkable since it achieved recognition rates higher than 95% in all datasets except the proposed one. The commercial systems, on the other hand, achieved similar results only in the Caltech Cars and UCSD-Stills datasets, which contain exclusively American LPs, and performed poorly (i.e., recognition rates below 85%) in at least two datasets. This suggests that the commercial systems are not so well trained for LPs of other layouts.

Furthermore, the commercial systems often predicted FPs that our system ignored, for example, two or three LPs in the same vehicle, phone numbers on storefronts, or even illegible LPs on very distant vehicles. This directly affects the overall cost of those systems since many FPs have to be processed in the LP recognition stage. In Figure 5.14, as an example, we compare the detections obtained by the proposed approach and commercial systems in the same image.



(a) Proposed (b) Sighthound [18] (c) OpenALPR [102]

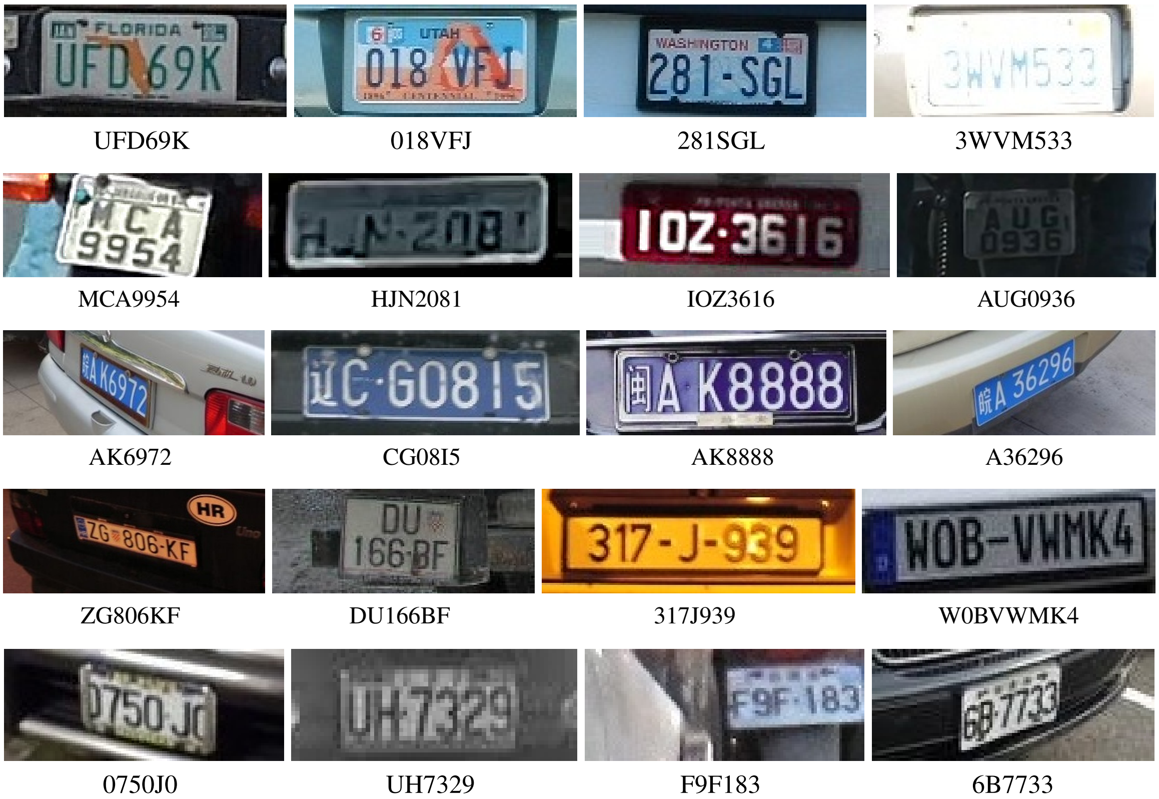
Figure 5.14: Comparison of the detections obtained by the proposed approach and commercial systems in the same image (slightly cropped for display purposes). As can be seen, the commercial systems often predicted FPs (red bounding boxes) that our system ignored. Remark that the LPs of the vehicles in the background are not legible.

Although OpenALPR achieved better results than Sighthound (on average across all datasets), the latter system can be seen as more robust than the former since it does not require prior knowledge regarding the LP layout. In addition, OpenALPR does not support Taiwanese LPs. In this sense, we tried to employ OpenALPR solutions designed for LPs from other countries (including China) in the experiments performed in the AOLP dataset (which contains only Taiwanese LPs), however, very low detection and recognition rates were obtained.

In Table 5.5, it can also be observed that the UFPR-ALPR dataset is the most challenging dataset among those employed in this work, as neither our approach nor the baselines were able to achieve recognition rates above 90% in its test set. This is due to the fact that we eliminated many of the constraints found in other datasets by using different non-static cameras to capture images from different types of vehicles (cars, motorcycles, buses and trucks) with complex backgrounds and under different lighting conditions, as detailed in Section 4.1.

According to our experiments, a great improvement in our system lies on classifying the LP layout prior to LP recognition, so that we can employ layout-specific approaches for the recognition task. Moreover, we trained every network using images from several datasets, which were collected under different conditions, as well as many other images created artificially. In this way, we prevented overfitting in certain scenarios.

Figure 5.15 shows some examples of LPs that were correctly recognized by the proposed approach. As can be seen, our system can generalize well and correctly recognize LPs of different layouts, even when the images were captured under challenging conditions. It is noteworthy that in this work, as in [33], the exact same networks were applied to all datasets; in other words, no specific training procedure was used to tune the networks for a given dataset or layout class.


UFD69K 018VFJ 281SGL 3WVM533

MCA9954 HJN2081 IOZ3616 AUG0936

AK6972 CG08I5 AK8888 A36296

ZG806KF DU166BF 317J939 W0BVWMK4

0750J0 UH7329 F9F183 6B7733

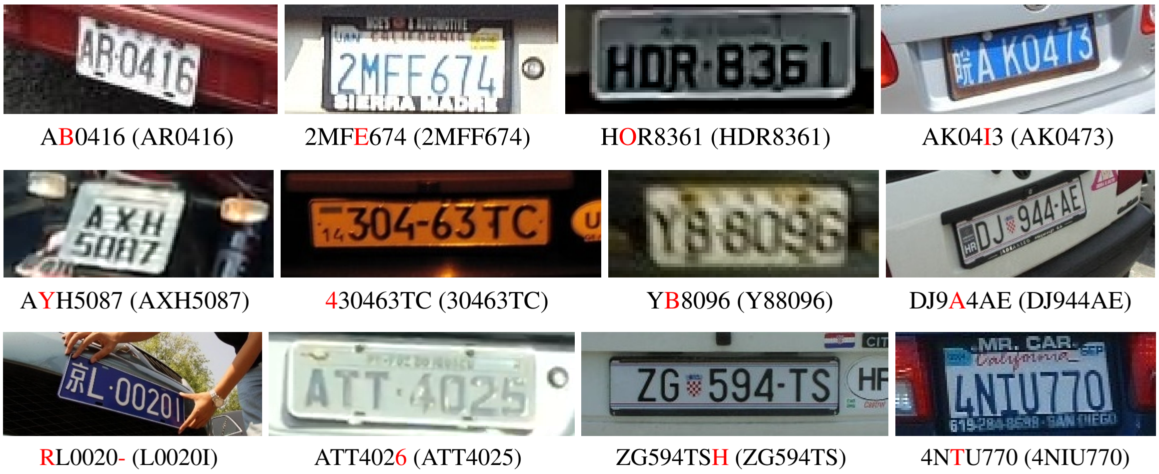
Figure 5.15: Examples of LPs that were correctly recognized by the proposed ALPR system. In the rows, LPs of different layout classes are shown. From top to bottom: American, Brazilian, Chinese, European and Taiwanese LPs.

Our system had no difficulty recognizing LPs of any specific layout, even those with less training examples (e.g., red Brazilian LPs or those with two rows of characters). According to our experiments, this is due to the negative images used when training the CR-NET model, as well as the images generated employing the data augmentation technique proposed in [13].

Some LPs in which our system failed to correctly detect/recognize all characters are shown in Figure 5.16. As one may see, the errors occurred mainly in challenging LP images, where even humans can make mistakes. Note that, in some cases, one character might become very similar to another due to the inclination of the LP, the LP frame, shadows, blur, etc.

Although highly unlikely (i.e., $\approx 0.05\%$), in some cases the network predicted a character in the outer region of the LP frame (i.e., an FP). In this sense, we evaluated adding a smaller margin to the LP patch before feeding it into the recognition network, however, better overall results were not achieved. This experiment was carried out in the validation set.

In Table 5.6, we report the time required for each network in our system to process an input. As in [11, 30], the reported time is the average time spent processing all inputs in each stage, assuming that the network weights are already loaded and that there is a single vehicle in the scene. It is remarkable that although a deep CNN model (i.e., YOLOv2) is used for vehicle detection, our system is still able to process images at 73 FPS on a high-end GPU. This is sufficient for real-time usage, as commercial cameras generally record videos at 30 FPS.



AB0416 (AR0416) 2MFE674 (2MFF674) HOR8361 (HDR8361) AK04I3 (AK0473)

AYH5087 (AXH5087) 430463TC (30463TC) YB8096 (Y88096) DJ9A4AE (DJ944AE)

RL0020- (L0020I) ATT4026 (ATT4025) ZG594TSH (ZG594TS) 4NTU770 (4NIU770)

Figure 5.16: Examples of LPs that were incorrectly recognized by the proposed ALPR system. The ground truth is shown in parentheses.

Table 5.6: The time required for each network in our system to process an input on an NVIDIA Titan Xp GPU.

ALPR Stage	Model	Time (ms)	FPS
Vehicle Detection	YOLOv2	8.5382	117
LP Detection and Layout Classification	Fast-YOLOv2	3.0854	324
LP Recognition	CR-NET	1.9935	502
End-to-end	**-**	**13.6171**	**73**

It should be noted that practically all images from the datasets used in our experiments contain only one labeled vehicle. However, to perform a more realistic analysis of the execution time, we listed in Table 5.7 the time required for the proposed system to process images assuming that there is a certain number of vehicles in every image (note that vehicle detection is performed only once, regardless of the number of vehicles in the image). According to the results, our system is able to process more than 30 FPS even when there are 4 vehicles in the scene. This information is relevant since some ALPR approaches, including the one proposed in our previous work [30], can only process frames in real time if there is at most one vehicle in the scene.

Table 5.7: Execution times considering that there is a certain number of vehicles in every image. Remark that our approach is able to process more than 30 FPS even when there are 4 vehicles in the image.

# Vehicles	Time (ms)	FPS
1	13.6171	73
2	18.6960	53
3	23.7749	42
4	28.8538	35
5	33.9327	29

The proposed approach achieved an outstanding trade-off between accuracy and speed, unlike others recently proposed in the literature. For example, the methods proposed in [11, 13] are capable of processing more images per second than our system but reached poor recognition rates (i.e., below 65%) in at least one dataset in which they were evaluated. On the other hand, impressive results were achieved on different scenarios in [14, 17, 33]. However, the methods presented in these works are computationally expensive and cannot be applied in real time. The Sighthound and OpenALPR commercial systems do not report the execution time.

It is important to highlight the number of experiments carried out to develop the proposed ALPR system. More than 50 models were evaluated (with different input sizes, number of filters and layers) and combined in several ways. It takes about two days and a half to **train** our networks on an NVIDIA Titan Xp GPU (a single run). In the testing phase, unlike most works in the literature (which report the results achieved in a single run), we reported in each stage the average result of 5 runs of our approach to avoid an overestimation or bias in the random division of the images into the training, validation and test subsets.

6 Conclusions

In this work, we presented an efficient and layout-independent ALPR system using the state-of-the-art YOLO object detection CNNs. First, the YOLOv2 model was employed for vehicle detection. Second, the Fast-YOLOv2 model was used for LP detection and layout classification. Finally, we detected and recognized all LP characters simultaneously using CR-NET, avoiding the challenging character segmentation task. We performed several data augmentation tricks and modifications to each network to achieve the best speed/accuracy trade-off at each stage.

We also introduced a public dataset for ALPR that includes 4,500 fully annotated images (with over 30,000 LP characters) from 150 vehicles in real-world scenarios where both the vehicle and the camera (inside another vehicle) are moving. Compared to the SSIG-SegPlate dataset [32], which is the public dataset of Brazilian LPs best known and most frequently used in the ALPR context, our dataset has more than twice the images and contains a larger variety in different aspects. Furthermore, we manually labeled the position of the vehicles, LPs and characters, as well as their classes, in all datasets used in this work since they have no annotations or contain labels only for part of the ALPR pipeline. It should be noted that the labeling process took a considerable amount of time since there are several bounding boxes to be labeled on each image (precisely, we manually labeled 38,351 bounding boxes on 6,239 images). These annotations are also publicly available to the research community, assisting the development and evaluation of new ALPR approaches as well as the fair comparison among published works.

The current bottleneck of ALPR systems is the LP recognition stage. In this sense, we proposed a unified approach for LP detection and layout classification in order to improve the recognition results through post-processing rules. This strategy was essential for accomplishing outstanding results since, depending on the LP layout class, we avoided errors in characters that are often misclassified and also in the number of predicted characters to be considered.

Our system achieved an average recognition rate of 96.8% across eight public datasets used in the experiments, outperforming both previous works and commercial systems in the ChineseLP, OpenALPR-EU, SSIG-SegPlate and UFPR-ALPR datasets. In the other datasets, the proposed approach obtained competitive results to the best result attained by the baselines. The robustness of our ALPR system is remarkable since, unlike some baselines, we did not apply any specific training procedure to tune the networks for a given dataset or layout class. Instead, we use heuristic rules in cases where the LP layout is classified with a high confidence value.

The results demonstrated that the UFPR-ALPR dataset is the most challenging dataset among those employed in this work, as neither our approach nor the baselines were able to achieve recognition rates above 95% in its test set. OpenALPR [102], which leverage prior knowledge regarding the LP layout, obtained the best result (i.e., 82.22%) among the baselines. The proposed system performed considerably better, with a recognition rate of 89.96%.

We also carried out experiments to measure the execution time. Compared to previous works, our system achieved an impressive trade-off between accuracy and speed. Specifically, even though the proposed approach achieved recognition rates above 95% in all datasets except the proposed one, it is able to process images in real time even when there are 4 vehicles in the scene.

6.1 Future Work

As future work, we intend to employ other object detection systems such as SSD [81] and Tiny-SSD [129] for ALPR, aiming to design a system that achieves higher recognition rates or processes images with a lower computational cost. We also want to explore the vehicle's make and model in the ALPR pipeline as the proposed dataset provides such information. As stated in [10], it is possible to further improve the system performance using post-processing approaches considering that there is a database of registered LPs and vehicle models.

In addition, we plan to correct the alignment of the detected LPs and also rectify them in order to achieve even better results in the LP recognition stage. Some methods have been employed for these tasks in the literature [26, 33, 130, 131], generally improving the accuracy of LP recognition. We intend to evaluate the effect of different approaches in our ALPR system from both the speed and accuracy points of view, as such approaches can be computationally expensive.

We want to create a large-scale ALPR dataset with Mercosur LPs. Mercosur (*Mercado Común del Sur*, i.e. Southern Common Market in Castilian) is an economic and political bloc comprising Argentina, Brazil, Paraguay and Uruguay[1]. These countries are adopting a new standard of LPs (see Figure 6.1) for newly purchased vehicles, inspired by the integrated system adopted several years ago by European Union countries [132]. Therefore, a large dataset would enable the community to apply, develop and adapt various ALPR systems for this new layout.

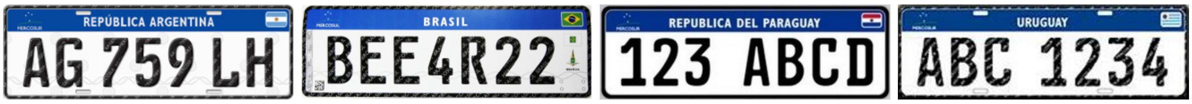


Figure 6.1: The new standard of Mercosur LPs. Observe that the letters and digits might be in any position. Above is shown the initial pattern that each country adopted.

Finally, we intend to conduct additional experiments in our next studies, which includes: (i) using for training all available datasets except one, which would be used for testing. In this way, we can truly assess the robustness of our ALPR system and also identify drawbacks such as low detection rates in certain scenarios and low recognition rates in a specific LP layout; (ii) fully labeling some public datasets that have annotations only for LP detection [27, 90], which would allow the entire ALPR pipeline to be evaluated in different scenarios without acquiring new images; and (iii) comparing our approach with other commercial systems such as *Plate Recognizer*[2] and *Meerkat*[3]. As these systems have trial versions through APIs, they can be evaluated with relatively little effort. It is worth noting that large private datasets are used to train these systems and outperforming them demonstrates the robustness of the proposed system.

[1] Venezuela is currently suspended and Bolivia is in the process of becoming a member.

[2] The Plate Recognizer API is available at `https://platerecognizer.com/`.

[3] The Merkat API is available at `https://www.meerkat.com.br/`.

6.2 Publications

The works published during the master's degree (or shortly after) are listed below [13, 30, 72, 133–139]. The publications related to the dissertation are marked with asterisks [*]. Although some papers are not directly related to ALPR, deep learning approaches (including YOLO-based models) were employed in all of them, contributing to the results obtained in this work.

- [*] R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Gonçalves, W. R. Schwartz, D. Menotti, "A robust real-time automatic license plate recognition based on the YOLO detector," in *International Joint Conference on Neural Networks (IJCNN)*, July 2018, pp. 1–10.

- [*] R. Laroca, L. A. Zanlorensi, G. R. Gonçalves, E. Todt, W. R. Schwartz, D. Menotti, "An efficient and layout-independent automatic license plate recognition system based on the YOLO detector," *IET Intelligent Transport Systems*, vol. 15, no. 4, pp. 483-503, 2021.

- [*] R. Laroca, V. Barroso, M. A. Diniz, G. R. Gonçalves, W. R. Schwartz, D. Menotti, "Convolutional neural networks for automatic meter reading," *Journal of Electronic Imaging*, vol. 28, no. 1, p. 013023, 2019.

- [*] G. R. Gonçalves, M. A. Diniz, R. Laroca, D. Menotti, W. R. Schwartz, "Real-time automatic license plate recognition through deep multi-task networks," in *Conference on Graphics, Patterns and Images (SIBGRAPI)*, Oct 2018, pp. 110-117.

- [*] G. R. Gonçalves, M. A. Diniz, R. Laroca, D. Menotti, W. R. Schwartz, "Multi-task learning for low-resolution license plate recognition," in *Iberoamerican Congress on Pattern Recognition (CIARP)*, Oct 2019, pp. 251-261.

- E. Severo, R. Laroca, C. S. Bezerra, L. A. Zanlorensi, D. Weingaertner, G. Moreira, D. Menotti, "A benchmark for iris location and a deep learning detector evaluation," in *International Joint Conference on Neural Networks (IJCNN)*, July 2018, pp. 1–7.

- D. R. Lucio, R. Laroca, E. Severo, A. S. Britto Jr., D. Menotti, "Fully convolutional networks and generative adversarial networks applied to sclera segmentation," in *IEEE International Conference on Biometrics: Theory, Applications, and Systems (BTAS)*, Oct 2018, pp. 1-7.

- C. S. Bezerra, R. Laroca, D. R. Lucio, E. Severo, L. F. Oliveira, A. S. Britto Jr., D. Menotti, "Robust iris segmentation based on fully convolutional networks and generative adversarial networks," in *Conference on Graphics, Patterns and Images (SIBGRAPI)*, Oct 2018, pp. 281-288.

- L. A. Zanlorensi, E. Luz, R. Laroca, A. S. Britto Jr., L. S. Oliveira, D. Menotti, "The impact of preprocessing on deep representations for iris recognition on unconstrained environments," in *Conference on Graphics, Patterns and Images (SIBGRAPI)*, Oct 2018, pp. 289-296.

References

[1] R. A. Lotufo, A. D. Morgan, and A. S. Johnson, "Automatic number-plate recognition," in *IEE Colloquium on Image Analysis for Transport Applications*, Feb 1990, pp. 1–6.

[2] K. Kanayama, Y. Fujikawa, K. Fujimoto, and M. Horino, "Development of vehicle-license number recognition system using real-time image processing and its application to travel-time measurement," in *IEEE Vehicular Technology Conference*, May 1991, pp. 798–804.

[3] K. Miyamoto, K. Nagano, M. Tamagawa, I. Fujita, and M. Yamamoto, "Vehicle license-plate recognition by image analysis," in *International Conference on Industrial Electronics, Control and Instrumentation (IECON)*, vol. 3, Oct 1991, pp. 1734–1738.

[4] 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 Transactions on Intelligent Transportation Systems*, vol. 9, no. 3, pp. 377–391, 2008.

[5] S. Kranthi, K. Pranathi, and A. Srisaila, "Automatic number plate recognition," *International Journal of Advancements in Technology*, vol. 2, no. 3, pp. 408–422, 2011.

[6] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, "Automatic license plate recognition (ALPR): A state-of-the-art review," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 2, pp. 311–325, Feb 2013.

[7] C. Patel, D. Shah, and A. Patel, "Automatic number plate recognition system (ANPR): A survey," *International Journal of Computer Applications*, vol. 69, no. 9, pp. 21–33, May 2013.

[8] O. Bulan, V. Kozitsky, P. Ramesh, and M. Shreve, "Segmentation- and annotation-free license plate recognition with deep localization and failure identification," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 9, pp. 2351–2363, 2017.

[9] G.-S. Hsu, S.-D. Zeng, C.-W. Chiu, and S.-L. Chung, "A comparison study on motorcycle license plate detection," in *IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, June 2015, pp. 1–6.

[10] G. R. Gonçalves, D. Menotti, and W. R. Schwartz, "License plate recognition based on temporal redundancy," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, Nov 2016, pp. 2577–2582.

[11] S. M. Silva and C. R. Jung, "Real-time brazilian license plate detection and recognition using deep convolutional neural networks," in *Conference on Graphics, Patterns and Images (SIBGRAPI)*, Oct 2017, pp. 55–62.

[12] J. Špaňhel, J. Sochor, R. Juránek, A. Herout, L. Maršík, and P. Zemčík, "Holistic recognition of low quality license plates by CNN using track annotated data," in *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Aug 2017, pp. 1–6.

[13] 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 *Conference on Graphics, Patterns and Images (SIBGRAPI)*, Oct 2018, pp. 110–117.

[14] H. Li, P. Wang, M. You, and C. Shen, "Reading car license plates using deep neural networks," *Image and Vision Computing*, vol. 72, pp. 14–23, 2018.

[15] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2009, pp. 248–255.

[16] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[17] H. Li, P. Wang, and C. Shen, "Toward end-to-end car license plate detection and recognition with deep neural networks," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11, 2018.

[18] S. Z. Masood, G. Shu, A. Dehghan, and E. G. Ortiz, "License plate detection and recognition using deeply learned convolutional neural networks," *arXiv preprint*, vol. arXiv:1703.07330, 2017. [Online]. Available: http://arxiv.org/abs/1703.07330

[19] A. M. Al-Ghaili, S. Mashohor, A. R. Ramli, and A. Ismail, "Vertical-edge-based car-license-plate detection method," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 1, pp. 26–38, Jan 2013.

[20] M. R. Asif, Q. Chun, S. Hussain, and M. S. Fareed, "Multiple licence plate detection for Chinese vehicles in dense traffic scenarios," *IET Intelligent Transport Systems*, vol. 10, no. 8, pp. 535–544, 2016.

[21] M. R. Asif, Q. Chun, S. Hussain, M. S. Fareed, and S. Khan, "Multinational vehicle license plate detection in complex backgrounds," *Journal of Visual Communication and Image Representation*, vol. 46, pp. 176–186, 2017.

[22] M. S. Al-Shemarry, Y. Li, and S. Abdulla, "Ensemble of adaboost cascades of 3L-LBPs classifiers for license plates detection with low quality images," *Expert Systems with Applications*, vol. 92, pp. 216–235, 2018.

[23] J. A. Guggenheim and J. M. Silversmith, "Confederate license plates at the constitutional crossroads: Vanity plates, special registration organization plates, bumper stickers, viewpoints, vulgarity, and the first amendment," *U. Miami L. Rev.*, vol. 54, p. 563, 2000.

[24] J. Tian, R. Wang, G. Wang, J. Liu, and Y. Xia, "A two-stage character segmentation method for chinese license plate," *Computers & Electrical Engineering*, vol. 46, pp. 539–553, 2015.

[25] C. Gou, K. Wang, Y. Yao, and Z. Li, "Vehicle license plate recognition based on extremal regions and restricted boltzmann machines," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1096–1107, April 2016.

[26] M. Dong, D. He, C. Luo, D. Liu, and W. Zeng, "A CNN-based approach for automatic license plate recognition in the wild," in *British Machine Vision Conference (BMVC)*, September 2017, pp. 1–12.

[27] Y. Yuan, W. Zou, Y. Zhao, X. Wang, X. Hu, and N. Komodakis, “A robust and efficient approach to license plate detection,” *IEEE Transactions on Image Processing*, vol. 26, no. 3, pp. 1102–1114, March 2017.

[28] F. Abtahi, Z. Zhu, and A. M. Burry, “A deep reinforcement learning approach to character segmentation of license plate images,” in *IAPR International Conference on Machine Vision Applications*, May 2015, pp. 539–542.

[29] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 779–788.

[30] R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Gonçalves, W. R. Schwartz, and D. Menotti, “A robust real-time automatic license plate recognition based on the YOLO detector,” in *International Joint Conference on Neural Networks (IJCNN)*, July 2018, pp. 1–10.

[31] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[32] G. R. Gonçalves, S. P. G. da Silva, D. Menotti, and W. R. Schwartz, “Benchmark for license plate character segmentation,” *Journal of Electronic Imaging*, vol. 25, no. 5, p. 053034, 2016.

[33] S. M. Silva and C. R. Jung, “License plate detection and recognition in unconstrained scenarios,” in *European Conference on Computer Vision (ECCV)*, Sept 2018, pp. 593–609.

[34] J. Salamon and J. P. Bello, “Deep convolutional neural networks and data augmentation for environmental sound classification,” *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, March 2017.

[35] G. S. Hsu, J. C. Chen, and Y. Z. Chung, “Application-oriented license plate recognition,” *IEEE Transactions on Vehicular Technology*, vol. 62, no. 2, pp. 552–561, Feb 2013.

[36] J. Redmon and A. Farhadi, “YOLO9000: Better, faster, stronger,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 6517–6525.

[37] ——, “YOLOv3: An incremental improvement,” *CoRR*, vol. abs/1804.02767, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[38] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (VOC) challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun 2010.

[39] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common objects in context,” in *European Conference on Computer Vision (ECCV)*, 2014, pp. 740–755.

[40] G. S. Hsu, A. Ambikapathi, S. L. Chung, and C. P. Su, “Robust license plate detection in the wild,” in *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Aug 2017, pp. 1–6.

[41] L. Xie, T. Ahmad, L. Jin, Y. Liu, and S. Zhang, “A new CNN-based method for multi-directional car license plate detection,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 2, pp. 507–517, Feb 2018.

[42] G. Ning, Z. Zhang, C. Huang, X. Ren, H. Wang, C. Cai, and Z. He, “Spatially supervised recurrent convolutional neural networks for visual object tracking,” in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–4.

[43] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer, “SqueezeDet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving,” in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, July 2017, pp. 446–454.

[44] S. Tripathi, G. Dane, B. Kang, V. Bhaskaran, and T. Nguyen, “LCDet: Low-complexity fully-convolutional neural networks for object detection in embedded systems,” in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 411–420.

[45] W. Zhou, H. Li, Y. Lu, and Q. Tian, “Principal visual word discovery for automatic license plate detection,” *IEEE Transactions on Image Processing*, vol. 21, no. 9, pp. 4269–4279, Sept 2012.

[46] OpenALPR Software Solutions, “OpenALPR-EU dataset,” https://github.com/openalpr/benchmarks/tree/master/endtoend/eu, 2016.

[47] B. A. Krinski, D. V. Ruiz, G. Z. Machado, and E. Todt, “Masking salient object detection, a mask region-based convolutional neural network analysis for segmentation of salient objects,” in *Latin American Robotics Symposium (LARS)*, 2019, pp. 55–60.

[48] J. Davis and M. Goadrich, “The relationship between precision-recall and ROC curves,” in *International Conference on Machine Learning (ICML)*, 2006, pp. 233–240.

[49] F. Giunchiglia, P. Shvaiko, and M. Yatskevich, “S-Match: an algorithm and an implementation of semantic matching,” in *The Semantic Web: Research and Applications*. Springer Berlin Heidelberg, 2004, pp. 61–75.

[50] H.-H. Do, S. Melnik, and E. Rahm, “Comparison of schema matching evaluations,” in *Web, Web-Services, and Database Systems*. Springer Berlin Heidelberg, 2003, pp. 221–237.

[51] D. M. W. Powers, “What the F-measure doesn’t measure: Features, flaws, fallacies and fixes,” *arXiv preprint*, vol. arXiv:1503.06410, 2015. [Online]. Available: http://arxiv.org/abs/1503.06410

[52] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[53] J. Redmon, “Computers can see. now what?” TEDxGateway, 2018. [Online]. Available: https://www.youtube.com/watch?v=XS2UWYuh5u0

[54] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, Aug 2013.

[55] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *International Conference on Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.

[56] L. Deng and D. Yu, “Deep learning: Methods and applications,” *Foundations and Trends in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[57] M. A. Ponti, L. S. F. Ribeiro, T. S. Nazare, T. Bui, and J. Collomosse, “Everything you wanted to know about deep learning for computer vision but were afraid to ask,” in *Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*, 2017, pp. 17–41.

[58] S. Tejani, “Machines that can see: Convolutional neural networks,” 2016. [Online]. Available: https://shafeentejani.github.io/2016-12-20/convolutional-neural-nets/

[59] A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities improve neural network acoustic models,” in *International Conference on Machine Learning (ICML)*, 2013.

[60] V. Dumoulin and F. Visin, “A guide to convolution arithmetic for deep learning,” *arXiv preprint*, vol. arXiv:1603.07285, 2016. [Online]. Available: http://arxiv.org/abs/1603.07285

[61] Y. Boureau, F. Bach, Y. LeCun, and J. Ponce, “Learning mid-level features for recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2559–2566.

[62] Y.-L. Boureau, J. Ponce, and Y. LeCun, “A theoretical analysis of feature pooling in visual recognition,” in *International Conference on Machine Learning*, 2010, pp. 111–118.

[63] M. Lin, Q. Chen, and S. Yan, “Network in network,” in *International Conference on Learning Representations (ICLR)*, 2014, pp. 1–10.

[64] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[65] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International Conference on Machine Learning (ICML)*, 2015, pp. 448–456.

[66] T. Cooijmans, N. Ballas, C. Laurent, and A. C. Courville, “Recurrent batch normalization,” in *International Conference on Learning Representations (ICLR)*, 2017, pp. 1–13.

[67] T. Salimans and D. P. Kingma, “Weight normalization: A simple reparameterization to accelerate training of deep neural networks,” in *Conference on Neural Information Processing Systems (NIPS)*, 2016, pp. 901–909.

[68] D. V. Ruiz, B. A. Krinski, and E. Todt, “ANDA: A novel data augmentation technique applied to salient object detection,” in *International Conference on Advanced Robotics*, Dec 2019, pp. 487–492.

[69] J. Redmon, “Darknet: Open source neural networks in C,” http://pjreddie.com/darknet/, 2013-2019.

[70] B. Alexe, T. Deselaers, and V. Ferrari, “Measuring the objectness of image windows,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2189–2202, Nov 2012.

[71] J. Hui, “Real-time object detection with YOLO, YOLOv2 and now YOLOv3,” 2018. [Online]. Available: https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088/

[72] R. Laroca, V. Barroso, M. A. Diniz, G. R. Gonçalves, W. R. Schwartz, and D. Menotti, “Convolutional neural networks for automatic meter reading,” *Journal of Electronic Imaging*, vol. 28, no. 1, p. 013023, 2019.

[73] A. Kathuria, “What’s new in YOLOv3?” 2018. [Online]. Available: https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b

[74] I. Krasin *et al.*, “Openimages: A public dataset for large-scale multi-label and multi-class image classification.” https://storage.googleapis.com/openimages/web/index.html, 2017.

[75] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[76] D. Han, J. Kim, and J. Kim, “Deep pyramidal residual networks,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 6307–6315.

[77] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 936–944.

[78] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” in *IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 2999–3007.

[79] J. Huang *et al.*, “Speed/accuracy trade-offs for modern convolutional object detectors,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3296–3297.

[80] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta, “Beyond skip connections: Top-down modulation for object detection,” *arXiv preprint*, vol. arXiv:1612.06851, 2016. [Online]. Available: http://arxiv.org/abs/1612.06851

[81] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single shot multibox detector,” in *European Conference on Computer Vision (ECCV)*, 2016, pp. 21–37.

[82] C. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, “DSSD : Deconvolutional single shot detector,” *arXiv preprint*, vol. arXiv:1701.06659, 2017. [Online]. Available: http://arxiv.org/abs/1701.06659

[83] P. R. Mendes Júnior, J. M. R. Neves, A. I. Tavares, and D. Menotti, “Towards an automatic vehicle access control system: License plate location,” in *IEEE International Conference on Systems, Man, and Cybernetics*, Oct 2011, pp. 2916–2921.

[84] R. F. Prates, G. Camara-Chavez, W. R. Schwartz, and D. Menotti, “Brazilian license plate detection using histogram of oriented gradients and sliding windows,” *International Journal of Computer Science and Information Technology*, vol. 5, pp. 39–52, 2013.

[85] A. H. Ashtari, M. J. Nordin, and M. Fathy, "An iranian license plate recognition system based on color features," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 4, pp. 1690–1705, Aug 2014.

[86] M. S. Sarfraz, A. Shahzad, M. A. Elahi, M. Fraz, I. Zafar, and E. A. Edirisinghe, "Real-time automatic license plate recognition for CCTV forensic applications," *Journal of Real-Time Image Processing*, vol. 8, no. 3, pp. 285–295, Sep 2013.

[87] R. Panahi and I. Gholampour, "Accurate detection and recognition of dirty vehicle plate numbers for high-speed applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 4, pp. 767–779, April 2017.

[88] S. Azam and M. M. Islam, "Automatic license plate detection in hazardous condition," *Journal of Visual Communication and Image Representation*, vol. 36, pp. 172–186, 2016.

[89] M. A. Rafique, W. Pedrycz, and M. Jeon, "Vehicle license plate detection using region-based convolutional neural networks," *Soft Computing*, June 2017.

[90] F. D. Kurpiel, R. Minetto, and B. T. Nassu, "Convolutional neural networks for license plate detection in images," in *IEEE International Conference on Image Processing (ICIP)*, Sept 2017, pp. 3395–3399.

[91] R. Polishetty, M. Roopaei, and P. Rad, "A next-generation secure cloud-based deep learning license plate recognition for smart cities," in *IEEE International Conference on Machine Learning and Applications (ICMLA)*, Dec 2016, pp. 286–293.

[92] Z. Selmi, M. B. Halima, and A. M. Alimi, "Deep learning system for automatic license plate detection and recognition," in *IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 01, Nov 2017, pp. 1132–1138.

[93] D. Menotti, G. Chiachia, A. X. Falcão, and V. J. O. Neto, "Vehicle license plate recognition with random convolutional networks," in *Conference on Graphics, Patterns and Images (SIBGRAPI)*, Aug 2014, pp. 298–303.

[94] Y. Yang, D. Li, and Z. Duan, "Chinese vehicle license plate recognition using kernel-based extreme learning machine with deep convolutional features," *IET Intelligent Transport Systems*, vol. 12, no. 3, pp. 213–219, 2018.

[95] J. Zhuang, S. Hou, Z. Wang, and Z.-J. Zha, "Towards human-level license plate recognition," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 314–329.

[96] 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 Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, April 2018.

[97] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 2818–2826.

[98] Q. Lu, W. Zhou, L. Fang, and H. Li, "Robust blur kernel estimation for license plate images from fast moving vehicles," *IEEE Transactions on Image Processing*, vol. 25, no. 5, pp. 2311–2323, May 2016.

[99] J. Fang, Y. Yuan, W. Ji, P. Tang, and Y. Zhao, “Licence plate images deblurring with binarization threshold,” in *IEEE International Conference on Imaging Systems and Techniques (IST)*, Sept 2015, pp. 1–6.

[100] C. Song and X. Lin, “Blurred license plate recognition based on single snapshot from drive recorder,” in *IEEE International Conference on Communications (ICC)*, June 2015, pp. 7108–7113.

[101] P. Svoboda, M. Hradiš, L. Maršík, and P. Zemcík, “CNN for license plate motion deblurring,” in *IEEE International Conference on Image Processing (ICIP)*, Sept 2016, pp. 3832–3836.

[102] OpenALPR Software Solutions, “OpenALPR Library,” http://www.openalpr.com/, 2019.

[103] G. J. Hsu and C. Chiu, “A comparison study on real-time tracking motorcycle license plates,” in *IEEE Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*, July 2016, pp. 1–5.

[104] A. G. Hochuli, L. S. Oliveira, A. S. Britto Jr., and R. Sabourin, “Segmentation-free approaches for handwritten numeral string recognition,” in *International Joint Conference on Neural Networks (IJCNN)*, July 2018, pp. 1–8.

[105] ——, “Handwritten digit segmentation: Is it still necessary?” *Pattern Recognition*, vol. 78, pp. 1–11, 2018.

[106] AlexeyAB, “YOLOv3 and YOLOv2 for Windows and Linux,” 2017-2019. [Online]. Available: https://github.com/AlexeyAB/darknet

[107] G. Hsu, J. Chen, and Y. Chung, “AOLP database,” http://aolpr.ntust.edu.tw/lab/, 2013.

[108] M. Weber, “Caltech Cars dataset,” http://www.vision.caltech.edu/Image_Datasets/cars_markus/cars_markus.tar, 1999.

[109] V. Srebrić, “EnglishLP database,” http://www.zemris.fer.hr/projects/LicensePlates/english/baza_slika.zip, 2003.

[110] L. Dlagnekov and S. Belongie, “UCSD/Calit2 car license plate, make and model database,” http://vision.ucsd.edu/belongie-grp/research/carRec/car_data.html, 2005.

[111] B. Shi, X. Bai, and C. Yao, “An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2298–2304, Nov 2017.

[112] Y. Liu, H. Huang, J. Cao, and T. Huang, “Convolutional neural networks-based intelligent recognition of Chinese license plates,” *Soft Computing*, vol. 22, no. 7, pp. 2403–2419, Apr 2018.

[113] J. Wang, H. Huang, X. Qian, J. Cao, and Y. Dai, “Sequence recognition of chinese license plates,” *Neurocomputing*, vol. 317, pp. 149–158, 2018.

[114] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.

[115] L. Dlagnekov, “Video-based car surveillance: License plate, make, and model recognition,” Master’s thesis, University of California, San Diego, 2005.

[116] G. R. Gonçalves, S. P. G. da Silva, D. Menotti, and W. R. Schwartz, "SSIG-SegPlate database," http://www.ssig.dcc.ufmg.br/ssig-segplate-database/, 2016.

[117] Z. Xu, W. Yang, A. Meng, N. Lu, H. Huang, C. Ying, and L. Huang, "Towards end-to-end license plate detection and recognition: A large dataset and baseline," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 261–277.

[118] M. Molina-Moreno, I. González-Díaz, and F. D. de María, "Efficient scale-adaptive license plate detection system," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–13, 2018.

[119] H. Xiang, Y. Yuan, Y. Zhao, and Z. Fu, "License plate detection based on fully convolutional networks," *Journal of Electronic Imaging*, vol. 26, pp. 1–7, 2017.

[120] H. Xiang, Y. Zhao, Y. Yuan, G. Zhang, and X. Hu, "Lightweight fully convolutional network for license plate detection," *Optik*, vol. 178, pp. 1185–1194, 2019.

[121] X. Zhang, N. Gu, H. Ye, and C. Lin, "Vehicle license plate detection and recognition using deep neural networks and generative adversarial networks," *Journal of Electronic Imaging*, vol. 27, pp. 1–11, 2018.

[122] V. Srebrić, "Postupci za poboljšanje kontrasta sivih slika," Sept 2003, University of Zagreb. Original document in Croatian.

[123] M. Wafy and A. M. M. Madbouly, "Efficient method for vehicle license plate identification based on learning a morphological feature," *IET Intelligent Transport Systems*, vol. 10, no. 6, pp. 389–395, 2016.

[124] K. Lin, H. Tang, and T. S. Huang, "Robust license plate detection using image saliency," in *IEEE International Conference on Image Processing (ICIP)*, Sep. 2010, pp. 3945–3948.

[125] R. Qian, B. Zhang, Y. Yue, Z. Wang, and F. Coenen, "Robust chinese traffic sign detection and recognition with deep convolutional neural network," in *International Conference on Natural Computation (ICNC)*, Aug 2015, pp. 791–796.

[126] J. Tian, G. Wang, J. Liu, and Y. Xia, "License plate detection in an open environment by density-based boundary clustering," *Journal of Electronic Imaging*, vol. 26, pp. 1–11, 2017.

[127] Y. Wu and J. Li, "License plate recognition using deep fcn," in *Cognitive Systems and Signal Processing*, 2017, pp. 225–234.

[128] L. Yang, P. Luo, C. C. Loy, and X. Tang, "A large-scale car dataset for fine-grained categorization and verification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 3973–3981.

[129] A. Wong, M. J. Shafiee, F. Li, and B. Chwyl, "Tiny SSD: A tiny single-shot detection deep convolutional neural network for real-time embedded object detection," *arXiv preprint*, vol. arXiv:1802.06488, 2018. [Online]. Available: http://arxiv.org/abs/1802.06488

[130] M. K. Saini and S. Saini, "Multiwavelet transform based license plate detection," *Journal of Visual Communication and Image Representation*, vol. 44, pp. 128–138, 2017.

[131] I. Türkyılmaz and K. Kaçan, "License plate recognition system using artificial neural networks," *ETRI Journal*, vol. 39, no. 2, pp. 163–172, 2017.

[132] Brazil Monitor, "New mercosur vehicles license plate come into effect in sep. 2018," 2018. [Online]. Available: http://www.brazilmonitor.com/index.php/2018/04/27/new-mercosur-vehicles-license-plate-come-into-effect-in-sep-2018/

[133] E. Severo, R. Laroca, C. S. Bezerra, L. A. Zanlorensi, D. Weingaertner, G. Moreira, and D. Menotti, "A benchmark for iris location and a deep learning detector evaluation," in *International Joint Conference on Neural Networks (IJCNN)*, July 2018, pp. 1–7.

[134] D. R. Lucio, R. Laroca, E. Severo, A. S. Britto Jr., and D. Menotti, "Fully convolutional networks and generative adversarial networks applied to sclera segmentation," in *IEEE International Conference on Biometrics Theory, Applications and Systems (BTAS)*, Oct 2018, pp. 1–7.

[135] C. S. Bezerra, R. Laroca, D. R. Lucio, E. Severo, L. F. Oliveira, A. S. Britto Jr., and D. Menotti, "Robust iris segmentation based on fully convolutional networks and generative adversarial networks," in *Conference on Graphics, Patterns and Images (SIBGRAPI)*, Oct 2018, pp. 281–288.

[136] L. A. Zanlorensi, E. Luz, R. Laroca, A. S. Britto Jr., L. S. Oliveira, and D. Menotti, "The impact of preprocessing on deep representations for iris recognition on unconstrained environments," in *Conference on Graphics, Patterns and Images (SIBGRAPI)*, Oct 2018, pp. 289–296.

[137] G. R. Gonçalves, M. A. Diniz, R. Laroca, D. Menotti, and W. R. Schwartz, "Multi-task learning for low-resolution license plate recognition," in *Iberoamerican Congress on Pattern Recognition (CIARP)*, Oct 2019, pp. 251–261.

[138] I. O. Oliveira, R. Laroca, D. Menotti, K. V. O. Fonseca, and R. Minetto, "Vehicle re-identification: exploring feature fusion using multi-stream convolutional networks," *arXiv preprint*, vol. arXiv:1911.05541, pp. 1–10, 2019.

[139] 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," *IET Intelligent Transport Systems*, vol. 15, no. 4, pp. 483–503, 2021.