UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

SÉRGIO MONTAZZOLLI SILVA

Fast Contextual Text Recognition With Deep Convolutional Neural Networks

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Advisor: Prof. Dr. Claudio R. Jung

Porto Alegre
August 2019

CIP — CATALOGING-IN-PUBLICATION

Montazzolli Silva, Sérgio

Fast Contextual Text Recognition With Deep Convolutional Neural Networks / Sérgio Montazzolli Silva. – Porto Alegre: PPGC da UFRGS, 2019.

118 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2019. Advisor: Claudio R. Jung.

1. Deep Learning. 2. Computer Vision. 3. Text Recognition. 4. License Plate. 5. Bib Recognition. I. Jung, Claudio R.. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Rui Vicente Oppermann
Vice-Reitora: Prof^{a}. Jane Fraga Tutikian
Pró-Reitor de Pós-Graduação: Prof. Celso Giannetti Loureiro Chaves
Diretora do Instituto de Informática: Prof^{a}. Carla Maria Dal Sasso Freitas
Coordenadora do PPGC: Prof^{a}. Luciana Salete Buriol
Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

“Artificial Intelligence
is the new electricity.”
— ANDREW NG

AGRADECIMENTOS

Primeiramente gostaria de agradecer ao professor Claudio Rosito Jung, meu orientador, por todo o suporte dado durante o trabalho. Pelas discussões sempre muito produtivas. Pelas idéias e questionamentos importantíssimos. Por, mesmo dado a grande diferença de experiência e conhecimento, sempre ouvir e acreditar em muitas das minhas idéias. Pelas partidas de futebol e churrascos. Não tenho dúvidas que me foi dado todo o suporte necessário para chegar até aqui, e espero realmente poder replicar esta experiência com futuros alunos e orientandos.

Em segundo, mas não menos importante, quero de agradecer a minha familia: meu pai Gerson, minha mãe Sueli, e minha irmã Julia. Tenho a certeza de que sem eles este trabalho não seria possível de maneira alguma. Acredito que a desigualdade social e de oportunidades é um problema grande no Brasil. Por isso, tenho muita gratidão a eles por me concederem toda estrutura familiar necessária e suporte financeiro nos momentos que precisei. Gostaria que todas as pessoas pudessem ter a sorte que eu tive nesse aspecto. Também quero de agradecer a todos os meus tios, tias, avós e primos, que tiveram um papel importante neste momento.

Agraço a todos os meus amigos e colegas de Porto Alegre que tornaram meus 7 anos de residência na cidade muito mais agradáveis. Em especial, gostaria de agradecer ao Anderson, Ricardo e Gabriela, que foram pessoas com quem compartilhei a maior parte dos meus bons momentos. Agradeço também aos meus colegas do Instituto de Informática com os quais dividi por anos a mesa do RU, compartilhamos o café, e tivemos ótimas conversas e discussões (técnicas ou não).

Agradeço a todos os meus amigos de Londrina que, mesmo longe, sempre me incentivaram, apoiaram, e me proporcionaram bons momentos durante os poucos períodos que pude ficar na cidade. Também em especial, quero agradecer aos meus amigos Edison e Thais, e a minha namorada Vanessa, por sempre terem me opoiado, inclusive nos momentos mais difíceis, e também a minha terapeuta Uérika, cujo o profissionalismo e serviços prestados sempre foram de grande importância.

Infelizmente o espaço é pequeno para nomear todas as pessoas que gostaria. Mas deixo aqui o meu muito obrigado e a minha consideração a todos que de alguma forma partiparam positivamente da minha vida durante os anos de doutorado.

E por fim, agradeço ao Instituto de Informática da UFRGS por me proporcionar toda a infraestrutura necessária para a realização do meu trabalho, e não apenas de doutorado, mas também de mestrado. Sempre pude contar com um corpo docente de qualidade, repleto de professores brilhantes e respeitados nas suas respectivas áreas, e funcionários técnicos e adminstrativos muito competentes e dispostos.

ABSTRACT

In this work we explore Deep Learning techniques to effectively recognize text in images given some context, which we call Contextualized Text Recognition (CTR). CTR arises in many applications, such as Automatic License Plate Recognition (ALPR) and Racing Bib[1] Number Recognition (RBN-R). With the rise of Deep Learning, the results in many computer vision tasks were improved in the past years. Its astonishing recognition capacity allowed the enhancement of existing applications and also the emerging of new challenging ones, such as speech recognition, self-driving cars, black and white image colorization, to name a few. However, this analysis power comes with a price: deep networks typically present a large number of parameters, meaning that a considerable amount of data is needed in order to train such models. To overcome these difficulties in CTR-related tasks where usually there is not much data available, we propose in the first part of this work clever uses of data augmentation, synthetic images and adaptations over the fastest models found in the literature. The results achieved are shown in the context of ALPR, where we demonstrate an approach capable of processing images at around 70 FPS and still achieving state-of-the-art performance. Going further, we noticed that there is a lack of unified datasets in ALPR encompassing license plates from different regions and scenarios. Also, there is no dataset exploring multi-regions and challenging scenarios where the plates are oblique and highly distorted. In the second part of this dissertation, we propose a dataset containing challenging ALPR images, and developed a novel Convolutional Neural Network (CNN) that regresses affine parameters responsible for rectifying license plates, allowing text recognition with high accuracy rates when compared to state-of-the-art methods. Finally, in the last part, we tackled the problem of RBN-R. A novel network was proposed to perform many tasks at once without the need for complex annotations. The network localizes the bib plate, corrects its distortion, and recognize its digits. For the whole approach, the only annotation required is the bib bounding box and the identification number. We obtained state-of-the-art results in the most popular dataset related to this problem.

Keywords: Deep Learning. Computer Vision. Text Recognition. License Plate. Bib Recognition.

[1] Bib is the plate attached to athletes body during competitions.

Reconhecimento Rápido de Texto Contextualizado Utilizando Redes Neurais Convolutivas Profundas

RESUMO

Neste trabalho são exploradas técnicas de Aprendizagem Profunda aplicadas ao reconhecimento de texto em imagens dado um certo contexto, problema aqui chamado de Reconhecimento de Texto Contextualizado (RTC). Como exemplos de aplicações, podemos citar o Reconhecimento Automático de Placas Veiculares (RAPV) e a Identificação de Atletas por Numeração (IAN). Recentemente, muitas tarefas relacionadas à Visão Computacional tiveram seus resultados aprimorados devido ao surgimento de técnicas de Aprendizagem Profunda. A grande capacidade de reconhecimento destas técnicas permitiu o avanço e surgimento de aplicações como Reconhecimento de Fala, Veículos Autônomos, Colorização de Fotos Monocromáticas, entre outras. No entanto, esse poder de análise traz um custo: redes profundas tipicamente apresentam um grande número de parâmetros, necessitando assim de um grande volume de dados durante o treinamento. Para superar este problema em tarefas onde não existem muitos dados disponíveis, na primeira parte deste trabalho, nós propomos o uso cuidadoso de dados aumentados e a adaptação de modelos rápidos encontrados na literatura. Os resultados obtidos são mostrados no contexto de RAPC, onde demonstramos a capacidade da nossa abordagem de obter resultados no estado-da-arte a uma frequência de 70 imagens por segundo. Indo além, nós percebemos que as bases de dados atuais em RAPC não exploram situações desafiadoras, contendo veículos em ângulos oblíquos, placas distorcidas e com a padronização de múltiplos países ou regiões. Então, como uma segunda parte deste trabalho, nós propomos a criação de uma base de dados contendo todas estas situações juntas, e apresentamos uma nova Rede Neural Convolucional para detectas placas ao mesmo tempo em que regride parâmetros para uma transformação afim de correção da distorção. Esse processo retifica a placa, auxiliando o reconhecimento dos caracteres e permitindo a obtenção de resultados estado-da-arte em várias bases de dados. Finalmente, na última parte, tratamos o problema de IAN. Propomos uma nova rede neural para executar várias tarefas de uma vez, sem necessitar de dados com anotações complexas. Basicamente a rede localiza a placa de identificação, corrige sua distorção, e reconhece todos os dígitos. De modo geral, nossa abordagem necessita apenas de duas informações para o treinamento: região da placa e seu número. Foram obtidos resultados no estado-da-arte durante avaliação na principal base de dados relacionada ao problema.

Palavras-chave: Aprendizagem Profunda. Visão Computacional. Reconhecimento de Placas de Carro..

LIST OF ABBREVIATIONS AND ACRONYMS

ALPR	Automatic License Plate Recognition
BNR	Bib Number Recognition
CNN	Convolutional Neural Networks
CRNN	Convolutional Recurrent Neural Network
CTR	Contextualized Text Recognition
FC	Fully Connected
FCN	Fully Convolutional Network
FPS	Frames Per Second
FRV	Frontal/Rear View
HOG	Histogram of Oriented Gradients
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
LBP	Local Binary Patterns
LP	License Plate
mAP	Mean Average Precision
MSER	Maximally Stable Extremal Regions
NN	Neural Network
OCR	Optical Character Recognition
RBN-R	Racing Bib Number Recognition
RNN	Recurrent Neural Network
RPN	Region Proposal Network
SGD	Stochastic Gradient Descent
SSD	Single Shot Detector
SPP	Spatial Pyramid Pooling
ST	Spatial Transformer

STN	Spatial Transformer Networks
STR	Scene Text Detection
SVM	Support Vector Machines
SWT	Stroke Width Transform
YOLO	You Only Look Once

LIST OF FIGURES

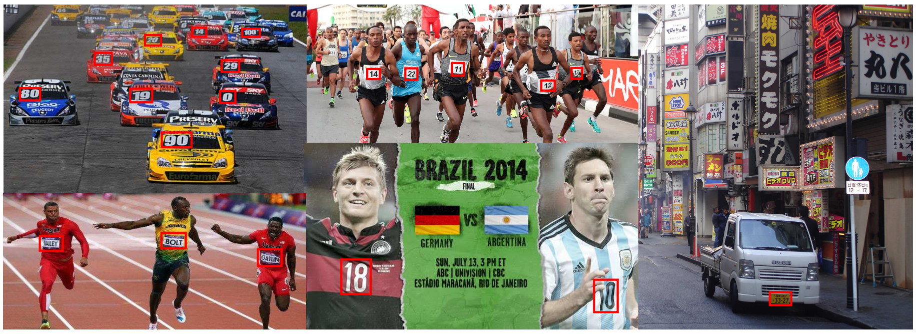
LIST OF TABLES

CONTENTS

1 INTRODUCTION

Text Detection and Recognition, sometimes referred to as Optical Character Recognition (OCR), is a notorious research field in Computer Vision and Machine Learning. Its combination with object detection is the root of many relevant and useful applications today. As examples, we can cite Automatic License Plate Recognition (ALPR), Credit Card Identification, Racing Bib Number Recognition (RBN-R), Automatic Meter Reading, among others. In all of these examples, the text of interest is located inside some object as part of a context, what we call Contextual Text Recognition (CTR). For instance, the license plate of a car in a traffic application must be associated with a physical vehicle, not existing by its own. Thus, detecting every text in the scene is a waste of computational resources and can lead to high false positive rates. Figure 1.1 shows some applications for CTR. By looking at these examples, we can notice that the text that identifies a car or a person usually is not the only text in the scene. The presence of other elements, such as advertising, can introduce errors to retrieve the identification information.

Figure 1.1: Examples of contextualized text detection applications.



The usual pipeline for such applications typically begins with an object detector — in order to find the object where the text is inserted — followed by an OCR algorithm. Until a couple of years ago, state-of-the-art object detection was performed by using a combination of feature extraction and classification methods. As successful feature extractors, we can refer to Histogram of Oriented Gradients (HOG) (DALAL; TRIGGS, 2005), Local Binary Patterns (LBP) (WANG; HE, 1990) and Haar-like Features (VIOLA; JONES, 2001). For classification, common choices were Support Vector Machines (SVM) and Cascade of Boosted Classifiers, among others.

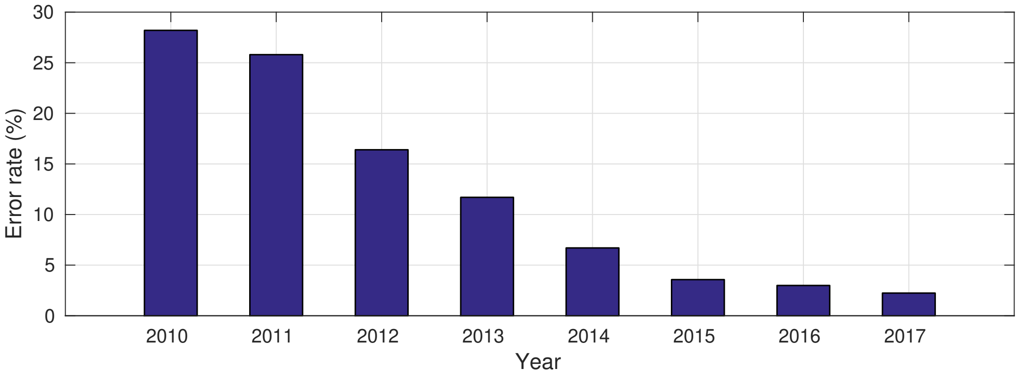
The features in the traditional object recognition pipeline are all hand-crafted, i.e., carefully engineered based on previous human experience. For instance, HOG was designed to work well when the shape of an object is more informative than its texture (e.g., detection of vehicles and pedestrians, or character recognition). On the other hand, LBP and Haar-like are more adequate to handle texture (e.g., human face and text detection). Nowadays, Deep Learning (DL) techniques have dominated this research field. They perform end-to-end object recognition, detection, or both, by learning relevant features and using them to classify the input image. This dominance is also taking place in OCR. Many classical methods, like Maximally Stable Extremal Regions (MSER) (MATAS et al., 2004) and Stroke Width Transform (SWT) (EPSHTEIN; OFEK; WEXLER, 2010), are being replaced by deep networks (JADERBERG et al., 2014; HE et al., 2016b; SHI; BAI; YAO, 2017; WANG et al., 2018; LIU et al., 2019).

1.1 Deep Learning

In the past couple of years, many computer vision tasks related to pattern recognition have been improved using DL. The state-of-the-art in some well-known fields, such as image classification (HE et al., 2016a), image segmentation (CHEN et al., 2016), object detection (REDMON; FARHADI, 2017; REN et al., 2017), pedestrian detection (ANGELOVA et al., 2015), face recognition (TAIGMAN et al., 2014), are now dominated by DL approaches.

To illustrate how effective DL is and its potential, we can refer to the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (RUSSAKOVSKY et al., 2015). ImageNet consists of over one 1 million images and 1,000 classes. Since 2012, when competitors started to use deep networks, the error has been dropping substantially as can be seen in Figure 1.2. The last winner of the ImageNet challenge was the work of (HU; SHEN; SUN, 2018), that achieved impressive 2.25% in the top-5 classification error. In 2018, ImageNet was replaced by a new dataset called Open Images Dataset (KUZNETSOVA et al., 2018), which is even bigger and with other tasks besides object recognition.

Figure 1.2: Top-5 ILSVRC error rate achieved by year.



Historically, the popularity of DL was mainly pushed by the introduction of efficient learning algorithms (e.g., Stochastic Gradient Descent (LECUN et al., 1998) and adaptive optimization algorithms such as AdaDelta (ZEILER, 2012), AdaGrad (DUCHI; HAZAN; SINGER, 2010) and Adam (KINGMA; BA, 2014a)), highly discriminative models (LECUN; BENGIO; HINTON, 2015) (e.g. Convolutional Neural Networks, Deep Belief Networks) and the possibility to hugely accelerate the training and testing process through Graphical Processing Units (GPU). However, despite all of this success, there are some drawbacks that researchers are yet trying to overcome:

- Lack of data: usually deep networks require huge amounts of labeled training data in order to perform well. In many fields, to build a large annotated dataset is a laborious or even impossible task;

- Expensive Hardware: training and (more importantly) executing many state-of-the-art networks demand high-performance GPUs and large amounts of memory to be feasible. Those requirements make it hard to embed DL solutions in small devices (e.g., smart-phones) or tiny computers (e.g., Raspberry Pi);

- Real-time: as DL is hardware demanding, applications requiring fast and accurate response — for instance, pedestrian detection in an autonomous vehicle — becomes very costly.

1.2 Applications of Contextualized Text Recognition

This dissertation is focused on two main applications that involve contextualized text recognition: ALPR and RBN-R. ALPR is an important task in intelligent transportation and surveillance systems, presenting many practical and relevant applications such as automatic traffic law enforcement, detection of stolen vehicles, toll violation and traffic flow control. It has been widely studied for decades, but still is an open problem due to the large variability in image acquisition conditions (illumination, capture angle, distance from the camera, etc.) and license plate (LP) layouts, which vary from one country to another.

A problem similar to ALPR is RBN-R. “Bib” is the plate attached to the body of athletes for identification purposes during competitions. Every competition presents its own bib layout and, unlike the relation between vehicles and license plates, we cannot assume that all persons found in the scene have a bib attached. These characteristics make the RBN-R an even harder problem to solve.

1.3 Research Question and Hypothesis

The question we aim to answer in this research is:

Research Question.

Can we avoid the drawbacks of DL and solve complex CTR problems (i.e., large scale variation, abrupt luminosity changes, partial occlusions, and perspective deformations) fast and accurately?

Despite the short question, the answer involves solving many difficult problems at once. A common CTR task has at least two steps: find the context, and detect/recognize the text within the object. To achieve state-of-the-art results, we used DL in both steps. Thus, the drawbacks mentioned earlier are replicated twice. Given that, we hypothesize that:

Research Hypothesis.

The development of suitable loss functions and structural changes in DL models combined with data augmentation techniques can lead to a fast and accurate performance in CTR tasks even when there is not much available training data.

This hypothesis was confirmed in the context of ALPR, as demonstrated in our two works: Silva and Jung (2017) and Silva and Jung (2018). In the first work, we explored synthetic and augmented data to help enlarge the variability of our small training set of 195 images — which contains only 1 LP per image (max. 3). Both techniques were carefully designed to cover deficient parts of our training set, like size and number of samples per character. Also, we adapted fast DL network models to match the requirements of the task in hand and transferred as much knowledge as possible from other models. The final approach was evaluated in standard scenarios, where the LPs appear mostly frontal in the majority of the images, and with a good illumination condition. For the second work, we moved to a more extreme scenario where most of the LPs were in oblique positions, and without region layout constraints. This new challenge required the development of a novel detection model to manage a planar object (i.e., the LP) in non fronto-parallel conditions. To train such network, we carefully designed an augmentation procedure that altered images through controlled perspective transformations. Furthermore, we generated synthetic data for the OCR model with multiple font types add variability and hence improve the ability of the network to recognize characters of different regions. We again used the same small dataset of 195 images as in the first work.

Supporting our claim for massive use of data augmentation in the training stage is the work of Varol et al. (2017), where the authors generated a huge synthetic dataset to tackle human pose estimation. This is a very challenging task and the generation of real datasets is a particularly laborious process. Furthermore, after the publication of the results presented in this dissertation, Björklund et al. (2019) demonstrated that it is possible to create an ALPR detection system based only on synthetic images. This substantially corroborates our hypothesis.

To tackle the problem of RBN-R, we again explored data augmentation techniques, but also applied structural modifications over existing DL models. The complete description of our approach is presented in the final part of this work. Besides the challenges mentioned earlier in this chapter, we can add that bibs are prone to complicated deformations due to the non-rigid materials used, and they might suffer from abrupt light changes and partial occlusions as a result of the arms movement. To handle such problems, we projected a model that automatically corrects distortions (limited by an affine transformation) without the need of explicitly presenting the expected transformation parameters during the training phase. In fact, the only cue our model has in the training stage is the OCR result (i.e., a string with the bib number), and the proposed method uses this information to find a transformation that maximizes the OCR accuracy. Furthermore, we again faced the problem of relying only on a small dataset, which required the augmentation of the real samples, and the generation of synthetic ones.

1.4 Chapters Organization

This dissertation is organized as follows: Chapter 2 briefly presents the background concepts of object detection and recognition with deep learning models, and data augmentation. Following, Chapter 3 discuss some related works to ours and the tasks we want to tackle (ALPR and RBN-R). Concerning ALPR, Chapters 4 and 5 present two different approaches to solve the problem: the first is focused on real-time processing, and the later is focused on complex unconstrained scenarios. Chapter 6 describes a DL approach to solve the RBN-R problem presenting state-of-the-art results. Finally, our conclusions are presented in Chapter 7.

2 BACKGROUND

This chapter will provide brief explanations about the main concepts and state-of-the-art DL techniques related to our research. As object detection is an important task for CTR, we dedicated the first part of this chapter on reviewing fast CNN-based models and evaluation metrics for object detection. Next, given our interest and necessity of using DL techniques without huge amounts of data, we will discuss Data Augmentation and the generalization improvements it brings to machine learning systems.

2.1 Evaluation Metrics

Before we start showing the main object detection models in the literature, it is important to understand how detections are evaluated. Basically, there are a few metrics that the reader needs to be familiar with:

- **Intersection over Union (IoU)**: in the context of object detection, it is a measure of how a predicted bounding box fits the ground truth annotation. Considering v and g the predicted and the ground truth bounding boxes, respectively, their IoU is given by:

$$IoU(v, g) = \frac{\#(v \cap g)}{\#(v \cup g)}, \tag{2.1}$$

  where $\#(\cdot)$ denotes the area of a region. Note that the IoU always lies inside the interval $[0, 1]$, where 0 means a totally disjoint prediction, and 1 means a perfect fit. In object detection, the threshold value 0.5 is typically used for accepting a candidate detection (EVERINGHAM et al., 2007; LIN et al., 2014), although different values can be considered depending on the task or analysis.

- **Precision and Recall**: classic prediction measures to describe the relevance and quantity of information retrieved. Precision measures the rate of relevant information, given by:

$$p = \frac{\text{true\_positives}}{\text{true\_positives} + \text{false\_positives}}, \tag{2.2}$$

  while recall correspond to the rate of correct information retrieved:

$$r = \frac{\text{true\_positives}}{\text{true\_positives} + \text{false\_negative}}. \tag{2.3}$$

  Usually, these measures are inversely associated: the more recall, the less precision, and vice-versa. A useful way to visualize the performance of a machine learning model is to plot the precision vs. recall curve. Considering that the output of a model is a probability of being and not-being the desired class, one can set a minimum threshold (θ) to accept or not the model's answer. Thus, for each threshold θ we can compute the corresponding precision and recall values, and varying θ leads to a curve

- **Mean Average Precision (mAP)**: all object detection models that will be described in the next section use mAP as an evaluation metric. It is a measure to quantify the quality of a multi-class classifier in a single number, taking into account the average precision per class. Thus, before computing mAP, it is necessary to calculate the Average Precision (AP). Formally, the AP is defined as:

  $$a = \int_0^1 p(r)dr, \tag{2.4}$$

  where $p(r)$ is the precision at a given recall value r, i.e. a is the area under the precision vs. recall curve. The mAP of an N-class classifier is given by:

  $$mAP = \frac{1}{N}\sum_{c=0}^{N} a_c, \tag{2.5}$$

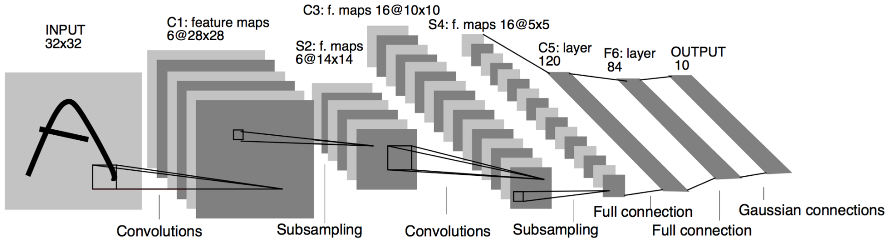
  where a_c is the average precision for class c. In general, if a classifier presents several detections for the same object, only one of them is considered correct, and the remaining are classified as false positives.

2.2 Convolutional Neural Networks

Convolutional Neural Networks were first described by LeCun et al. (1989), in the context of Zip Code Recognition. Basically, they consist of learning a set of convolutional filters for each hidden layer of the neural network. These filters are applied successively to the input data in a process that starts with a high-resolution input image, and ends in a low-resolution feature map or feature vector, depending on the architecture.

To solve tasks related to classification, CNNs commonly end with a feature vector, where each position corresponds to a class. The popular architectures are formed by a set of convolutional layers followed by fully connected layers. A classical example is the LeNet-5, presented in Lecun et al. (1998) and illustrated in Figure 2.1. Note that the first four layers correspond to two convolutional layers and two sub-sampling layers, and they are followed by three fully connected layers. The final fully connected layer creates a feature vector of length 10, i.e., one for each class of hand-written digit the network proposes to recognize.

Figure 2.1: LeNet-5 architecture. Image taken from Lecun et al. (1998)



An active research field is the development of new layers for Neural Networks. Therefore, we can find many different kinds of layers in the literature, where many of them are specifically designed to solve a particular problem. Even so, there are three very important (an popular) kinds of layers that the reader must be familiar with in order to understand CNN architectures:

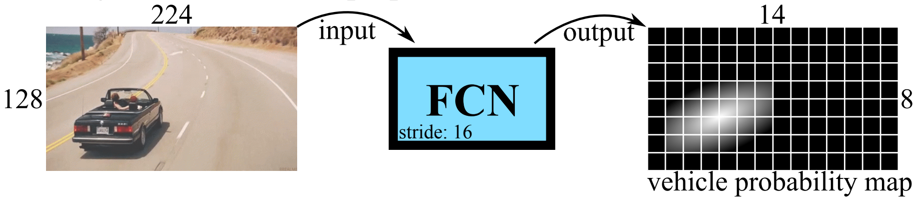
- Convolutional: layers designed to apply multiple convolutional filters in its input. The filters are learned based on data during the training stage;

- Pooling: layers focused on reducing the input resolution. They do not contain any parameter and, consequently, do not need to be learned, representing only architectural artifacts. The most common type of pooling layer is max-pooling, which simply computes the maximum value in a given window;

- Fully Connected: usually, after a sequence of convolutional and max-pooling layers, the data start to be treated as feature vectors — instead of maps. Fully connect layers connect every value in the input vector to every value in the output vector, just like in the Multi-Layer Perceptron.

Another popular kind of CNN architecture is the Fully Convolutional Network (FCN). They are simply CNNs without the fully connected layers. Thus, instead of outputting a feature vector, FCN outputs a feature map. If we consider the object detection task, this is a nice property since feature maps can preserve locations. For instance, considering an FCN designed to detect cars, and an input image containing a vehicle. The activations in the feature map will take place mainly in the region proportionally close the vehicle, as illustrated in Figure 2.2. Note that there is a relation between the input image resolution and the output feature map resolution, which is called network stride. The network stride is the ratio between the input and the output resolution. In other words, it measures how many times the input is reduced after forwarding in an FCN.

Figure 2.2: Example of an FCN that detect cars. The output feature map represent the probabilities of having/not-having a vehicle. Note that the input resolution suffered a downsize, during the feed-forward, proportional to the network stride of 16.



The massive use of CNNs began in the early 2010s, when efficient GPU implementations started to appear. The networks used since then have several parameters in the order of hundreds of millions to billions, and they started to be called deep networks. Training them is hardware demanding and the use of CPU is usually impractical.

2.3 Feature Extraction Sub-Networks (Backbones)

Training a deep CNN with the ImageNet dataset can take minutes or hours in a massive cluster of GPUs, or days on high-end machines (GOYAL et al., 2017; YOU et al., 2017). Taking in mind that the dataset presents a huge amount of annotated images and 1,000 classes, it is assumed that a network that performs well on ImageNet has learned how to extract very informative features, particularly in the first layers. Thus, a common practice – namely Transfer Learning (YOSINSKI et al., 2014) – is to use the initial layers of models trained on ImageNet to compose new models developed for other specific tasks. The part of these new models related to pre-trained models is called backbone.

Below we briefly comment a few widely used backbone models:

- AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012): one of the first networks to augment the number of layers to improve accuracy on ImageNet. The authors successfully created a deep CNN that, at the time, improved substantially the results on the dataset. Structurally, the top layers of AlexNet architecture contain large convolutional filters of size 11×11 and 5×5, and the network has a total of 8 layers: 5 convolutional and three fully connected;

- VGG (SIMONYAN; ZISSERMAN, 2014): short for Visual Geometry Group, VGG is a classical model of deep CNNs. One of the main differences to AlexNet is the total depth. This model replaced the large sized filters used by AlexNet to smaller ones (3×3), allowing to build an even deeper network by increasing the number of layers. The two most used VGG configurations are the VGG-16 and VGG-19, with 16 and 19 layers respectively;

- ResNet (HE et al., 2016a): following the tendency of further increasing the depth, ResNet employed a block of layers called residual blocks, that only change the data coming from the preceding layer or block if needed, otherwise act as an identity function. This allowed for much deeper networks and breaking through results on ImageNet. The model with best results presented by the authors have 110 layers, called ResNet-110;

- MobileNet (HOWARD et al., 2017): devised targeting efficiency concerning size and speed. MobileNets use special kinds of convolutional layers, called depthwise and pairwise convolutions, in which the computation is much faster. The network has 28 layers and was also trained on ImageNet, achieving strong results when compared to VGG-16, for instance. VGG-16 have a total of 138 millions of parameters and achieved 71.5% accuracy on ImageNet, and MobileNet-224 [1] has 4.2 millions of parameters and achieved 70.6% accuracy.

2.4 Object Detection with Convolutional Neural Networks

In order to perform object detection with deep networks, the common sliding-window strategy is onerous due to the number of computations needed over the whole search space. Therefore, many architectural modifications and the use of external pre- and post-processing steps (e.g. object proposal methods and SVMs) were employed in an attempt to overcome this issue. To exemplify, we briefly describe three of the first most popular methods, highlighting their weakness:

- **OverFeat**: despite being relatively old, OverFeat (SERMANET et al., 2013) gave a valuable insight which is used today in state-of-the-art models, like YOLO (see Section 2.4.3). The authors noted that a Fully Connected (FC) layer could be seen as a

[1] In this case, 224 stands for the input size resolution (224×224), and not the number of layers.

1×1 convolutional layer. Thus, an arbitrary sized image can feed any CNN, producing an output feature map proportional to its size. This allows us to efficiently run a kind of sliding-window approach with just a fraction of the computations needed, by reusing most of the information from the previous window, as explained in Figure 2.3. Nevertheless, OverFeat employs multi-scale sliding window detection and uses a second network for regression of object bounding-boxes, which significantly increase the detection time;

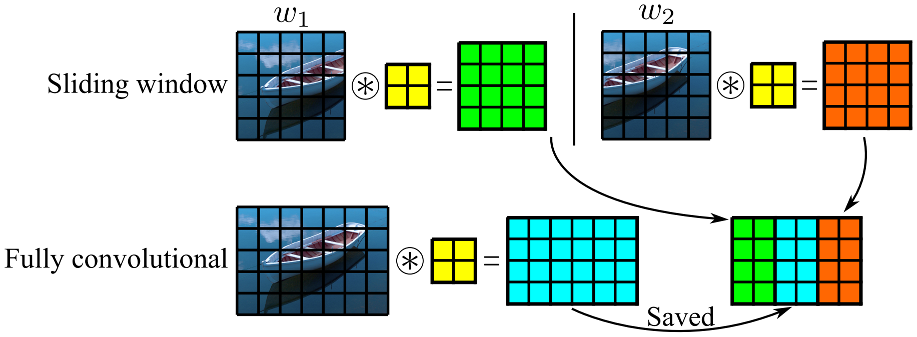
- **R-CNN**: as a predecessor of Fast R-CNN and Faster R-CNN, the *Regions with CNN features* model (GIRSHICK et al., 2014) is a backbone CNN - the authors used AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) - coupled with a trained SVM plugged after the penultimate FC layer. The model includes an object proposal method to search the input image for object candidates. The selected candidates are then fed to the network and the features in its penultimate layer are used as input for the SVM classifier. This strategy is much better than a sliding-window procedure. However, it is still very slow for training and testing. For instance, the training process of an R-CNN for PASCAL-VOC 2007 (EVERINGHAM et al., 2007) can take 3.5 days, and the testing for a single image can take up to 47 seconds [2];

- **SPP-net**: the Spatial Pyramid Pooling (SPP) Network (HE et al., 2015) basically consists of adding an SPP layer after the last convolutional layer of a backbone CNN. Knowing that a convolutional layer can receive an input of arbitrary size, and the fully connected layer must receive an input of a fixed size, the SPP works by converting an arbitrarily sized feature map to a fixed size. Thus, it is suitable to detect objects of different aspect ratios without the need for scaling and warping. Similarly to R-CNN, the authors coupled a backbone network - in this case, the ZF (ZEILER; FERGUS, 2014) - with an object proposal method and an SVM classifier, instead of FC layers. The ZF network is much smaller than AlexNet, and plugging it on the SPP-net scheme made the model several times faster than R-CNN, but still far from real-time.

In the networks mentioned above, training and testing are slow processes even through the use of modern GPUs. Moreover, due to dependence on external pre- and post-processing steps, their training is complex and not end-to-end.

[2] Using an NVIDIA K40 GPU (GIRSHICK, 2015)

Figure 2.3: Simple demonstration of the difference between sliding window and fully convolutional approaches. Considering a 5×7 input image (the boat), a 2×2 filter (yellow grid) and a window size of 5×5, computing the output using an sliding window approach adds 50% more computation than a fully convolutional way.



Given the need for faster and more elegant DL models, some interesting methods arose in the last couple of years. Three of them will be shortly described next.

2.4.1 Faster R-CNN

The Faster R-CNN is the result of successive enhancements in the R-CNN model. One can refer to Fast R-CNN (GIRSHICK, 2015) and R-CNN minus R (LENC; VEDALDI, 2015) for the previous model enhancements. More specifically, the R-CNN has two main problems:

- It is not end-to-end: the training process is complicated and requires three stages. The first stage fine-tunes the chosen backbone CNN (with FC layers) using object proposals. The second stage trains an SVM given the output of the penultimate FC layer of the backbone, hence acting as an object detector. And finally, a bounding-box regressor is trained aiming to improve candidate regions;

- It relies on object proposal methods: with more proposals, more time is needed since it is necessary to forward every proposal through a backbone network, in order to generate features for the SVM.

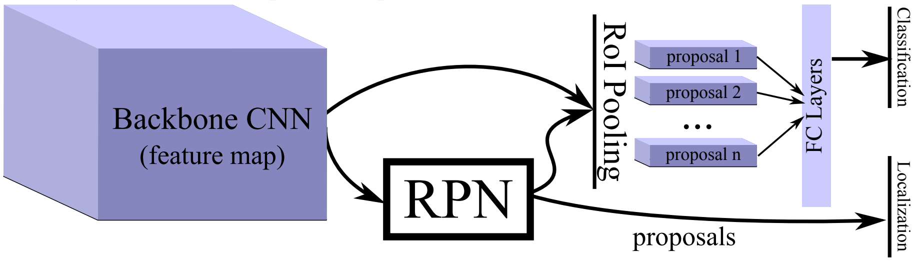
A previous work to Faster R-CNN is the Fast R-CNN. There, the authors proposed some modifications that were adopted for Faster R-CNN. The main modification was the inclusion of a ROI Pooling Layer (a special case of the SPP layer) to reduce computations for each object proposal. In this new approach, the feature maps are computed once for the whole image, and pooled according to the proposed object region. For instance, suppose the object proposal method generated 2000 proposals. In R-CNN model, the feature maps should be computed 2000 times. In Fast and Faster R-CNN, it is computed just once, generating a significant speed up.

Another enhancement from R-CNN to Faster R-CNN is the use of FC layers with Softmax, instead of SVM. The authors demonstrated that there is no significant difference by using one or another. Considering that FC layers can be directly attached to the network, it saves one stage of training, letting the model be more elegant.

The final modification, now presented in Faster R-CNN paper (REN et al., 2017), is the use of a Region Proposal Network (RPN) in replacement to object proposal methods. The RPN is a second network that receives as input the feature map obtained by forwarding the input image through the backbone CNN, and outputs object/non-object probabilities and their bounding boxes.

The bounding box regressor in RPN uses anchors to improve the final detection. To clarify, anchors are predefined bounding boxes with just two parameters: width and height. Its center is assumed to be the same as the feature cell being analyzed. During testing, the model analyses the RPN output and selects the anchor that best fits (Higher probability). the object found, and regresses (x, y) offsets for the anchor center and (s_w, s_h) scales for the anchor dimensions. Hence, the final bounding box is a translated and scaled version of the chosen anchor.

Figure 2.4: Overview of Faster R-CNN: first, an arbitrary sized image is forwarded through a backbone network (the authors mentioned the use of VGG-16). Second, the output feature map is used by the RPN to find proposals. Then, a ROI Pooling layer converts these proposals to a smaller and fixed size feature maps, by cropping and pooling from the backbone feature map. Finally, those small feature maps are passed through a range of FC layers, which outputs the probabilities for each class.



The Faster R-CNN still presents a 2-stage training process, thus it is not fully end-to-end. However, the two networks (CNN and RPN) can be merged after training, allowing 1-stage testing. These performance gains granted a huge speedup over R-CNN, from 1 frame every 47 seconds to 5 frames per second. An overview of the Faster R-CNN model is given in Figure 2.4.

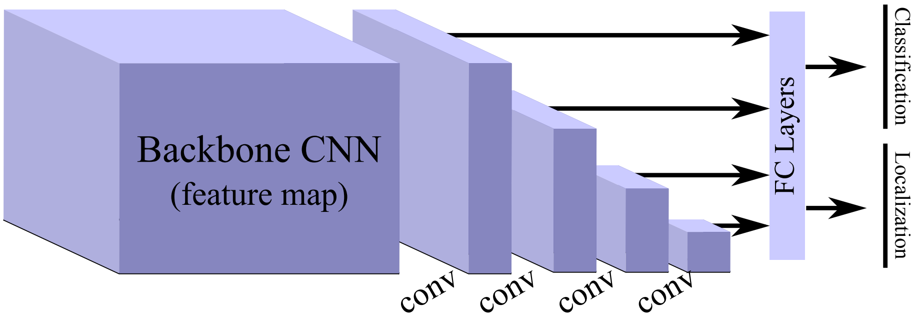
2.4.2 Single Shot Detector (SSD)

The SSD (LIU et al., 2016) is an end-to-end CNN model for object detection. Its elegant architecture performs training and testing in single stage each. Also, this network achieves a frame rate of 59 FPS [3] with comparable results to Faster R-CNN (5 FPS).

In SSD there is only one CNN, which basically is the result of merging a backbone CNN with a classifier and bounding box regressor. This is the main difference from Faster R-CNN, where the bounding boxes are processed by another CNN (the RPN). Therefore, the whole SSD network is trained in the same process, which is beneficial since all the weights - including the weights in the backbone - are fine-tuned to better regress the bounding boxes. Such improvements allowed the authors to significantly shrink the input image, which reduces the forwarding cost and time, while keeping a good detection accuracy. For instance, in the fastest version of SSD the input is a 300×300 pixels image (SSD300).

To precisely output bounding boxes in multiple resolutions, the authors proposed a combination of feature maps from successive convolutional layers with sub-sampling (stride > 1), as illustrated in Figure 2.5. Note that from the backbone CNN output onwards, the feature map resolution starts to decrease at every convolutional layer, and a connection between each feature map to the first FC layer is made.

Figure 2.5: SSD overview: a fixed sized image is forwarded through a backbone CNN. The output feature map starts passing through sequences of convolutional layers with 3×3 filters and stride equals to 2. Hence, those layers output lower resolution feature maps, which are routed to the first FC layer, and to the next convolutional layer. According to the authors, this routing enables the network to correctly regress small and big objects, since each layer process the feature map in a different resolution. In the end, the FC layers regress both class probabilities and bounding boxes for thousands of detections simultaneously.

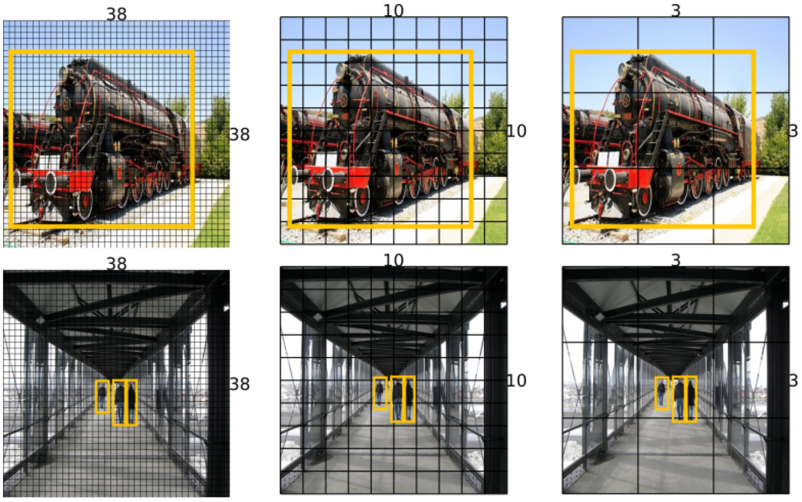


The illustration in Figure 2.6 demonstrates how the SSD model is expected to work. There are two images depicting two different objects (a locomotive and humans).

[3] Using an NVIDIA Titan X GPU.

Due to perspective, the locomotive occupies most of the first image, and the humans occupy just a small portion of the second image. In SSD300, the first feature map after the backbone CNN is of resolution 38×38, the fourth, 10×10, and the sixth, 3×3. Drawing an imaginary grid over the input image with the size of the respective feature map provides an idea of the region where each detector is acting on. We can see that detectors on the first and fourth feature maps lack of enough information to infer the locomotive class and its bounding box. However, the regressors on the sixth layer have a more adequate size for this task, since they accumulate information from other resolutions. The inverse scenario can be seen for the detection of humans in the second picture (and second row). It is impossible for the sixth layer to regress all three humans. This happens because each detector can output just one object. However, all humans fall inside the central detector.

Figure 2.6: Different feature map resolutions for two images containing very different sized objects. The yellow rectangles show the desired detection.



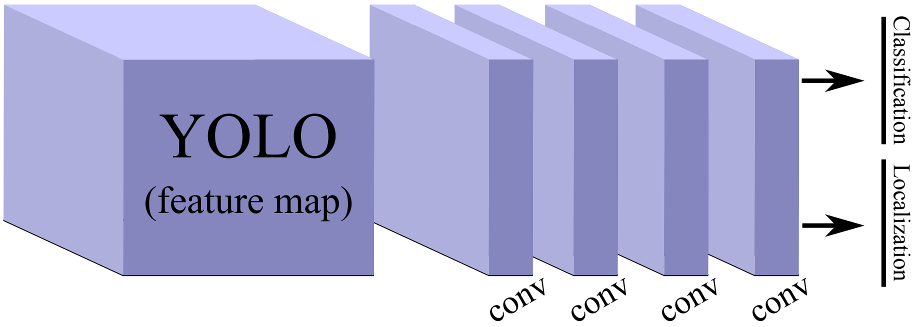
A drawback of SSD is the use of FC layers at the end of the model. This requires the input image to be of a fixed size. We will see next that other models such as YOLO (REDMON et al., 2016) use this restriction for training, but it is not necessary for testing.

2.4.3 You Only Look Once (YOLO)

The YOLO model was first described in Redmon et al. (2016), improved less than a year later, in YOLOv2 (REDMON; FARHADI, 2017), and recently evolved again to version 3 (REDMON; FARHADI, 2018). Although there is a reasonable intersection between these three models, in this section we will describe only version 2, which is faster, intuitive, and contains the major updates among all three models.

In terms of model design, the YOLOv2 is closer to SSD than Faster R-CNN, being capable of performing both training and testing in a single stage each. The model overview is shown in Figure 2.7. The main difference from SSD is that YOLO eliminated the multi-resolution convolutional layers scheme and the FC layers. Also, the authors preferred to not use common backbones available, and trained their model from scratch. They argue that common backbone models, like VGG-16, are slow to compute and do not bring improvements in accuracy.

Figure 2.7: YOLO overview.



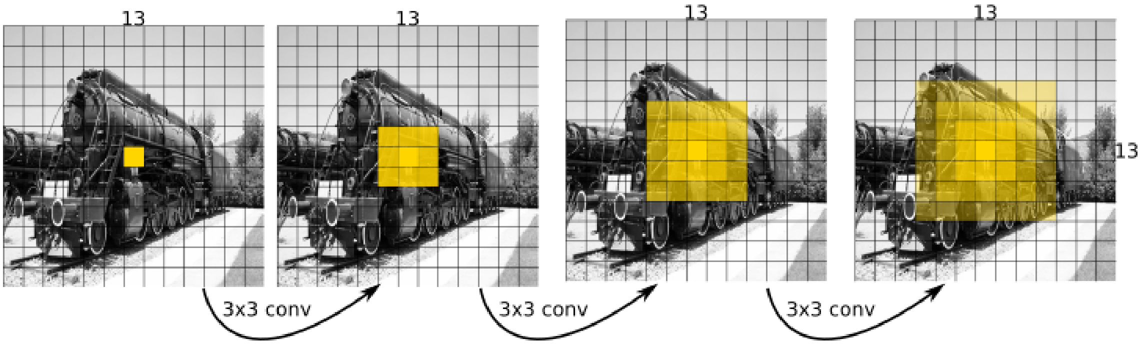
The final layers of YOLOv2 model are a sequence of 3×3 convolutional layers, without sub-sampling or max-pooling. They process an input 13×13 feature map and outputs a same 13×13 resolution map, containing the class probabilities and bounding boxes at each point. This makes the network even more elegant than SSD and, as all the layers are convolutional, allowing arbitrary size images as input.

Comparing to SSD, one can think that a fixed output resolution of 13×13 might not be adequate to detect bigger objects, like the locomotive example in Figure 2.6. However, this set of convolutional layers propagates information through adjacent points. This is illustrated in Figure 2.8. Note that the first layer does not have enough information to detect the whole locomotive, but adjacent information is aggregated as it is propagated through these layers, allowing a single predictor to accurately infer the locomotive bounding box.

Anchor boxes are also used by YOLOv2, but they are not manually defined as in other models. The dimensions of all the annotated bounding boxes in the training set were used by a k-means algorithm - with $(1 - IoU)$ as distance measure - to infer a good set of anchor boxes (k-means centroids). This allowed reducing from 9 to 5 the number of anchors without mAP loss.

Those modifications allowed YOLOv2 to achieve state-of-the-art object detection on PASCAL-VOC 2007 at 40 FPS. Furthermore, a much faster and lightweight version, called Fast-YOLO, was demonstrated achieving 213 FPS in the same scenario, but with a

Figure 2.8: Information propagation example through three layers of same input and output dimensionality, and filter size.



lower mAP.

2.5 Recurrent Layers For Neural Networks

Among many different kinds of layers that are possible to couple with Neural Networks, a recurrent layers is a particular class that interprets the feature map as sequences of temporally aligned data. As such, this kind of layer is useful in areas such as word recognition from images, syllable recognition in speech analysis, and text analysis in Natural Language Processing. In all these tasks, there is a relation between the current information (e.g., a character or phoneme) and the past ones.

To exemplify how recurrent layers work, imagine that we have a feature matrix $A_{M \times N}$. A recurrent layer considers this data as M feature vectors of size N that are temporally aligned from 1 to M. Recurrent layers process each feature vector independently, but using information obtained from the previous feature vector — the output of the m^{th}vector is used to process vector $m + 1$, and so on.

Mathematically, a simple Neural Network layer is represented by a matrix of weights W, a bias vector $\vec{b}$ and an activation function σ. Given an input vector $\vec{x}$ at time t, the layer output $\vec{h}_t$ is:

$$\vec{h}_t = \sigma(W\vec{x}_t + \vec{b}). \tag{2.6}$$

Diferently, recurrent layers also consider the last output into the calculations. For instance, a simple recurrent layer can be modeled as:

$$\vec{h}_t = \sigma(W\vec{x}_t + U\vec{h}_{t-1} + \vec{b}), \tag{2.7}$$

where U is a matrix that combines the last output with the new input. Such junctions are called gates.

Nowadays, two common and more sophisticated types of recurrent layers are widely employed: the Long-Short Term Memory (LSTM) and the Gated Recurrent Unit (GRU).

The LSTM (HOCHREITER; SCHMIDHUBER, 1997) is a block formed by several gates with specific roles:

- Forget gate: receives $\vec{x}_t$ and $\vec{h}_{t-1}$ as input, sum them and pass through a sigmoid function. It is expected that the sigmoid generate values close to zero in areas that must get forgotten;
- Input gate: it is similar to the forget gate with the addition of a hyperbolic tangent function ($\tanh$) at the end. The sigmoid function filters the values that are important to persist, and the $\tanh$ enhances them;
- Output gate: receives $\vec{x}_t$ and $\vec{h}_{t-1}$ and also the input gate output. This information is processed and multiplied together in order to filter relevant features for the output.

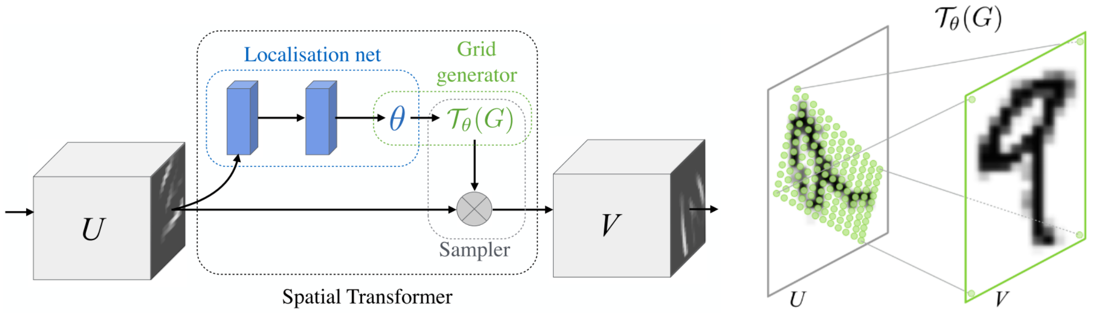
Every gate in an LSTM model has its own set of weights that are applied to the data. Hence, the layer by itself is large in terms of the number of parameters, which also impacts the computational cost. In Cho et al. (2014), the authors proposed the GRU layer, which is a simplified version of the LSTM that does not present the output gate. It reduces the model complexity and also the number of parameters to be trained, allowing the GRU to be faster in the number of computations needed. Originally it was tested in the context of machine translation, achieving superior results when compared directly to LSTM. Later, a wide range of applications appeared, such as video sequences (BALLAS et al., 2015), image segmentation (VISIN et al., 2016) and speech analysis (IRIE et al., 2016; RANA, 2016).

2.6 Spatial Transformer Networks (STN)

The STN (JADERBERG et al., 2015) is a module that can be placed inside a network to perform spatial transformations on a given feature map. It operates by interpolating part of this feature map into a predefined grid, generating an output with the same size of the grid and having the same number of channels of the input feature map. The interpolation uses a bilinear scheme based on an affine (or perspective) transformation whose parameters were inferred by some sequence of layers, called localization network. In other words, the STN can perform a non-trivial (non-rectangular) crop focusing on a specific part of the feature map, giving attention to that region.

The STN scheme is summarized in Figure 2.9a: a localization network is used to analyze the feature map U and regresses the affine parameters θ; then, a bilinear algorithm employs these parameters, along with the predefined grid G, to interpolate U into V using the transformation $\mathcal{T}_\theta$. This last process is exemplified closely in Figure 2.9b. During the network instantiation, every layer in the localization network is initialized with random weights, except the last one. In this layer, the parameters must be set to generate the identity matrix as output, initially predicting the identity transform.

Figure 2.9: Spatial Transformer Networks (Source: Jaderberg et al. (2015))



(a) STN module.

(b) Affine attention.

Tasks such as character recognition manage to substantially benefit from spatial transformations, as rotated characters can potentially drop the OCR accuracy. Moreover, object detection and fine-grained classification can also be performed by an STN due to its attention mechanism: the network can localize the object before performing the classification.

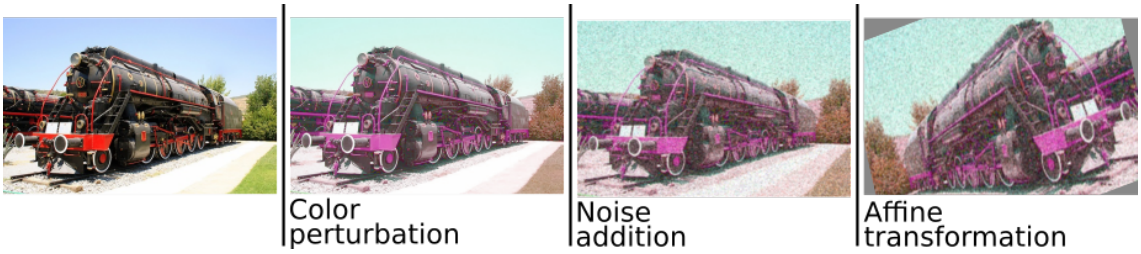
2.7 Data Augmentation

Data Augmentation is a common practice in Machine Learning systems and is present in the training process of all the models described in the previous section. Due to the huge number of parameters, DL techniques tend to overfit if the database used is not big enough. However, creating such big databases can be immensely laborious or, in many cases, impossible. The three most common techniques for augmenting an image dataset (PÉREZ; WANG, 2017) are:

- **Affine Transformations**: consists of simply applying random affine transformations (which include rotation, scaling and translation) to an image, allowing the system to be robust to spatial transformations. The degree of randomness must be controlled depending on the problem and network used;
- **Color Space Perturbation**: perturbing the color space is beneficial to improve the system robustness to illumination changes and small color variations. A common process adopted is to convert an RGB image to the HSV colorspace, and alter intensities by summing a random value for each channel;
- **Noise addition**: adding noise can be helpful to handle images of different qualities. The most common types of artificially injected noise are the Salt & Pepper and Gaussian noise.

In YOLOv2 and SSD, the authors used affine transformations and color space perturbation during the training stage. The inclusion of noise can also be advantageous, as pointed by Nazaré et al. (2017). In Figure 2.10 we show an example of a data augmentation pipeline, where successive changes were made resulting in a much different image.

Figure 2.10: Example of a data augmentation pipeline.



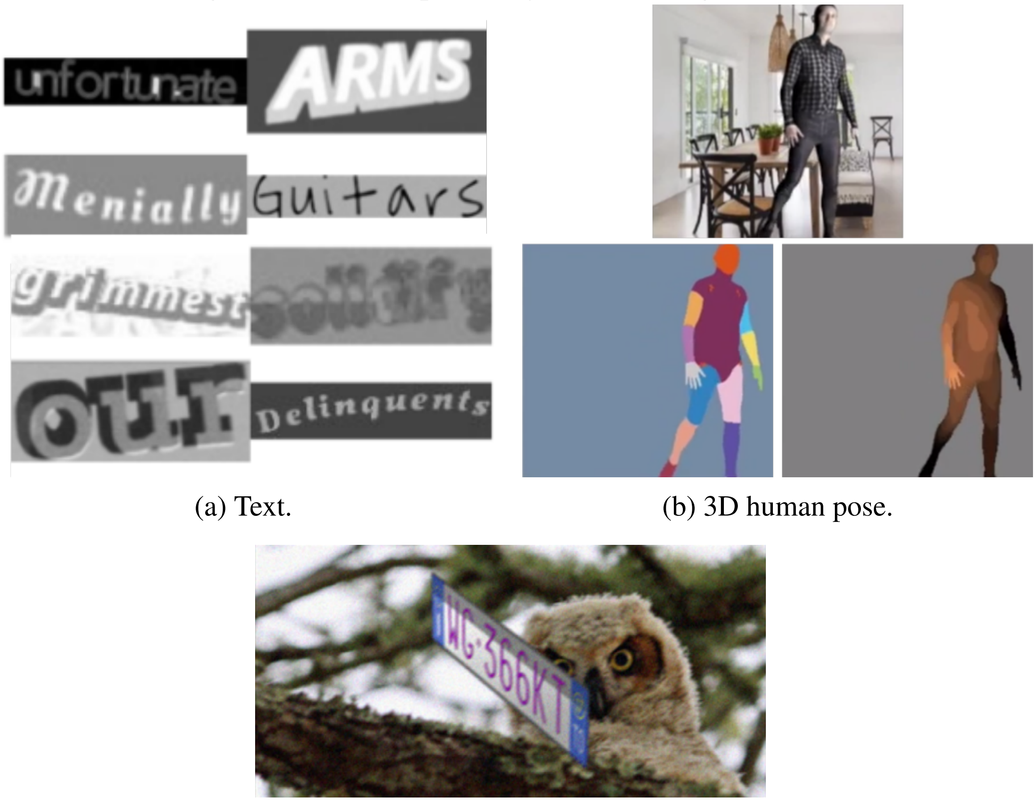
2.7.1 Synthetic Data Generation

The augmentation processes previously described are used to modify images, generating outputs that resemble in some way the original input. Differently, synthetic data generation approaches are focused on creating new artificial data. Depending on the combination of parameters and methods used in the synthetic pipeline, there is no limit for the number of outputs it can generate. This is beneficial when training DL based models for problems where the datasets available are small or nonexistent, and demands high variability. For instance, in Jaderberg et al. (2014) the authors generated totally artificial words (Figure 2.11a) and characters datasets for text recognition, based on spatial transformations and blending modes. They trained a CNN with only these synthetic data, and were able to achieve state-of-the-art accuracy in many text recognition datasets.

In Gupta, Vedaldi and Zisserman (2016), a synthetic text detection dataset was created by inserting text into images after analyzing the scene geometry. Considering that text usually does not appear in areas with strong discontinuity, they used a segmentation algorithm to select homogeneous areas. Furthermore, to determine the text perspective, the image depth map information, originated from a CNN, was used.

Another interesting example is the generation of a dataset for human pose estimation, proposed in Varol et al. (2017). The manual labeling of such data includes 3D pose and depth, which is impractical for humans. The authors then proposed a pipeline where synthetic humans were rendered over arbitrary images from house environments without real humans. Therefore, the 3D pose and depth map were precisely extracted. In Figure 2.11b we show a sample from their dataset along with its ground truth pose and depth images. Regarding ALPR, Björklund et al. (2019) presented an approach trained only with synthetic data. Their work will be addressed in the next chapter.

Figure 2.11: Examples of synthetic data generation.



(a) Text.

(b) 3D human pose.

(c) Licence plates.

3 RELATED WORK

In this chapter, we present a literature review of topics related to our problem. Although there is no baseline work directly tackling the CTR problem, we can find abundant material related to associated tasks. Hence, we organized this chapter by the tasks that we will address during this dissertation.

We start by reviewing works related to the Automatic License Plate Recognition (ALPR) problem in Section 3.1. Despite the vast literature on this subject, we focused our attention on Deep Learning methods. Next, in Section 3.2, we briefly describe a few Racing Bib Number Recognition (RBN-R) approaches, since this problem is far less popular than ALPR. In Section 3.3 we briefly describe another application that involves CTR: automatic meter reading, and finally we present, in Section 3.4, a review of generic purpose OCR works, again focusing on new DL approaches.

3.1 Automatic License Plate Recognition (ALPR)

Automatic License Plate Recognition (ALPR) is the task of finding and recognizing license plates in images. It is commonly broken into four subtasks that form a sequential pipeline: i) vehicle detection; ii) license plate detection; iii) character segmentation and iv) character recognition. For simplicity, we call OCR the combination of the last two subtasks.

In the past decades, many different ALPR systems or related subtasks were proposed using a variety of computer vision and machine learning techniques, as pointed out by the reviews of Anagnostopoulos et al. (2008) and Du et al. (2013). Earlier methods were commonly based on image binarization or gray-scale analysis to find candidate proposals (e.g. LPs and characters), followed by a handcrafted feature extractor, such as Local Binary Patterns (LBPs) or Histograms of Oriented Gradients (HoG), and a classical machine learning classifier, such as Support Vector Machine (SVMs), AdaBoost or Neural Networks. With the rise of DL, the state-of-the-art started moving to another direction, and nowadays many works employ CNNs due to its high accuracy for generic object detection and recognition, as mentioned in the previous chapter.

Knowing that vehicle is a classical type of object present in many popular object detection datasets, such as PASCAL-VOC (EVERINGHAM et al., 2007), ImageNet (RUSSAKOVSKY et al., 2015) and COCO (LIN et al., 2014), we will omit this part of the pipeline to focus on a more specific subtask: License Plate Detection. Furthermore, this section also reviews some approaches that handle the ALPR problem in an end-to-end fashion. It is important to clarify that here that “end-to-end” means treating the ALPR problem from the input image to the final LP number (i.e., the whole pipeline), and not the usual interpretation of “end-to-end” in the context of DL approaches (i.e., a single network that handles all the problems).

3.1.1 License Plate Detection

The success of YOLO networks for fast and accurate object detection and recognition inspired many recent works to use their architecture, targeting real-time performance for LP detection (SILVA; JUNG, 2017; HSU et al., 2017; XIE et al., 2018; LAROCA et al., 2018). A slightly modified version of the YOLO and YOLOv2 networks were used by Hsu et al. (2017). The authors enlarged the networks output granularity to improve the number of detections and set the probabilities for two classes (license plate and background). Furthermore, they trained the networks using a dataset with high illumination variability. Their network achieved a good compromise between precision and recall; however, the paper lacks a detailed evaluation of the extracted bounding boxes. For instance, there is no information about the threshold used when evaluating the Intersection over Union (IoU). Moreover, it is known that YOLO networks struggle to detect small-sized objects, so that further evaluations over scenarios where the car is far from the camera (for instance, in the SSIG Dataset) are needed.

In Xie et al. (2018) a setup of two YOLO-based networks was trained with the goal of detecting rotated LPs. The first network is used to find a region containing the LP, called “attention model” (in a similar manner to frontal-views detection in Silva and Jung (2017) and Chapter 4), and the second network captures a rotated rectangular bounding-box of the LP. Nonetheless, they considered only on-plane rotations, and not more realistic deformations caused by oblique camera views. Also, as they do not present a complete ALPR system, it is difficult to evaluate how well an OCR method would perform on the detected regions.

License plate detectors using sliding window approaches or candidate filtering coupled with CNNs can also be found in the literature (KURPIEL; MINETTO; NASSU, 2017; BULAN et al., 2017; SELMI; HALIMA; ALIMI, 2017). However, they tend to be computationally inefficient as a result of not sharing calculations like in modern metaarchitectures for object detection such as YOLO, SSD and Faster R-CNN.

3.1.2 End-to-end ALPR

The work of Laroca et al. (2018) used five networks to build a complete ALPR system. The first three networks are focused on detecting objects, which in this case is: vehicles, license plates and characters. They represent a considerable part of the ALPR pipeline, remaining only the segmented characters to be recognized, which is performed by two other specialized networks, being one for letters and another for digits. The resulting system is focused only on Brazilian LPs, where the authors reported real-time performance. This approach was not trained to capture oblique and distorted LPs.

Another complete ALPR system was presented by Selmi, Halima and Alimi (2017). The authors used a series of pre-processing approaches based on morphological operators, Gaussian filtering, edge detection, and geometry analysis to find LP candidates and characters. Then, two distinct CNNs were used to (i) classify a set of LP candidates per image into one single positive sample, and (ii) to recognize the segmented characters. There is no vehicle detection and it is assumed that every input images contains one LP. Thus, the system fails to identify multiple vehicles in the scene. Moreover, according to the authors, distorted LPs and poor illumination conditions can compromise the performance.

An interesting approach based on the idea of using just a single network to perform the whole ALPR pipeline was presented by Li, Wang and Shen (2017), based on the Faster R-CNN model architecture. Shortly, a Region Proposal Network is assigned to find candidate LP regions, whose corresponding feature maps are cropped from the backbone by a RoI Pooling layer. Then, these candidates are fed into the final part of the network, which computes the probability of being/not being an LP, and performs OCR through a Recurrent Neural Network. Despite promising, the evaluation presented by the authors shows a lack of performance in most challenging scenarios containing oblique LPs.

Region proposals along with rectification were used by Dong et al. (2017). In their paper, the authors described an ALPR system divided into to stages: (i) a LP detector based on R-CNN and Region Proposal Networks, which recovers the four corner points of a LP; and (ii) a recognition scheme that performs LP rectification based on the regressed corners, and character segmentation and recognition with an ensemble of spatial transformer networks and CNN recognizers. Their method is focused on Chinese license plates, and the dataset used (containing $18k$ images) and implementation were not made publicly available.

Recently, the massive use of synthetic data was explored by Björklund et al. (2019). The authors proposed a complete ALPR system trained only with synthetic license plates (an example of a license plate generated by their method was shown earlier in Figure 2.11c of the last chapter). They argue that geometric features are essential features for the network to learn, and by artificially generating data it is possible to create license plates in any geometric view. Two networks were designed and trained using the artificially generated samples: the first detects the license plate by outputting its four corner points, and the second is responsible for detecting and recognizing the characters.

Commercial systems are also good reference points to the state-of-the-art. Although the companies usually provide only partial (or no) information about their product architecture, we still can use them as black boxes to evaluate the final output. In this dissertation, we considered the following three commercial systems:

- Sighthound: few details were given in Masood et al. (2017). The system uses CNNs throughout the entire pipeline for detection and recognition tasks, and it is trained with huge private databases composed by around 20,000 vehicle images and 25,000 cropped LPs;

- OpenALPR: there is no detailed information available, but the system massively uses deep learning and it is an official NVIDIA partner in the Metropolis platform[1];

- Amazon Rekognition: it is a more general purpose AI engine. It performs many kinds of visual analysis, and in its text detection and recognition module, the company informs that it can be used for LP recognition as well.

Despite the existence of several academic and commercial systems for ALPR, most of them focus on camera setups that capture mostly frontal LPs, not being suited for unconstrained scenarios with highly distorted LPs. Also, most of these methods are designed for a specific LP region (e.g., Brazilian, European, etc.), which limits their application. Later in this work, we present a method designed to detect, unwarp and recognize LPs in a wide variety of camera setups, and that is generic enough to capture LPs from different regions. We also show that our generic approach can be fine-tuned to a specific scenario (e.g., only Brazilian LP models) to yield even better results.

[1] NVIDIA platform for video analysis in smart cities (<https://www.nvidia.com/en-us/autonomous-machines/intelligent-video-analytics-platform/>).

3.2 Racing Bib Number Recognition

Racing Bib Number (RBN) is the identification attached to an athlete's body during a competition. It usually consists of a card with the ID printed inside, and its material should be resistant to sweat and constant deformations. The identification is typically a number, but in some restricted competitions it can be a name. Recognizing RBNs is a challenging task in CTR due to many factors, such as:

- Camera movement;
- Deformations (a result of the athlete's movement or the location where the card was pinned in his/her body);
- Oblique angles;
- Illumination variation;
- Presence of irrelevant text near the ID (such as sponsors or clothing details).

To the best of our knowledge, there are only three approaches in the literature that tackle this problem directly, as it does not attract much attention compared to ALPR. Moreover, the only existing publicly available dataset [2] specific for RBN (to our knwoledge) is small, imposing additional challenges to train robust classifiers.

The first work on RBN-R was presented in Ben-ami, Basha and Avidan (2012). The authors proposed a multi-staged pipeline that starts by detecting the athlete's face. Then, a rough estimate of the torso area was generated based on a scaling and translation transformation of the face region. Given that in most cases the Bib is attached to the torso, this estimation greatly reduces the search area for the number ID. Next, a modified version of the Stroke Width Transform (EPSHTEIN; OFEK; WEXLER, 2010) was used to detect text regions. The resulting detections, if more than one, were filtered based on a size assumption and average confidence. Finally, the Tesseract OCR engine (SMITH, 2007) was used to recognize the segmented characters.

More recently, Shivakumara et al. (2017) presented an approach based on HoG+SVM, GrabCut segmentation (ROTHER; KOLMOGOROV; BLAKE, 2004) and Conditional Random Fields. First, a HoG+SVM classifier was used to detect upper body parts in the image (i.e., a rectangular area containing the face and shoulders). Then, using GrabCut and a larger area, the athlete (foreground) is segmented and its parts separated using

[2] Available at <http://people.csail.mit.edu/talidekel/RBNR.html> .

Pictorial Structural Models. The torso part, which is where the bib is usually pinned in, is selected and a pyramidal HoG+SVM is used to select text areas. Finally, a set of morphological operators were used to segment the characters, and the resulting binary image sent to Tesseract for recognition.

In Kamlesh et al. (2017), the authors adapted existing text spotting/recognition methods in the context of RBN-R. More precisely, they adapted the TextBoxes (LIAO et al., 2017) network to carry out bib number detection, and re-trained the CRNN model to perform number recognition (explanation about TextBoxes and CRNN is in Section 3.4)). Despite the good results, the authors only compared to variations of their work and a baseline CNN. Moreover, their dataset was not published until the publication of this dissertation.

3.3 Other Applications

Another example of CTR is the Automatic Meter Reading (AMR) problem. AMR aims to recognize information related to consumption of water, electricity or gas, given an image from a local meter. This is useful, for instance, for fraud monitoring or billing.

An AMR approach for electricity meters was presented in Laroca et al. (2019). Similarly to their previous work on ALPR, the authors used two networks based on the YOLO architecture: one to detect the counter region in the image (i.e., the context), and another to detect and recognize digits. The evaluation was performed in a dataset with 2,000 images and achieved an impressive 97.30% of accuracy.

Another application for AMR concerns gas meter. In this context, Gallo, Zamberletti and Noce (2015) used a neural network to perform meter localization, and a mixture of MSER and HoG+SVM to detect and recognize digits. For water meter, Gao et al. (2018) presented an approach convenient for mobile devices. The meter detection, which usually is the most onerous part in terms of computation, used handcraft features along with a boosted classifier for fast performance. A CNN feature extractor coupled with a variation of LSTM that computes recurrently in both horizontal directions was employed to recognize digits. The authors conclude that this bidirectional LSTM was able to better model dependencies between digits.

3.4 Optical Character Recognition (OCR)

Optical Character Recognition (OCR) is a classical task in computer vision, where the first reading machine dates from the 50s (EIKVIL, 1993). As there is a vast literature on this subject with many different approaches, we suggest the reading of Ye and Doermann (2015) for further information. In recent years, due to the advance of DL techniques, the research attention in this field moved to Scene Text Recognition (STR). This is a particularly complex task due to the amount of background area, large variety of font-types and, many times, absence of high contrast between text and background. The STR task is commonly divided into two subtasks: text detection and word spotting.

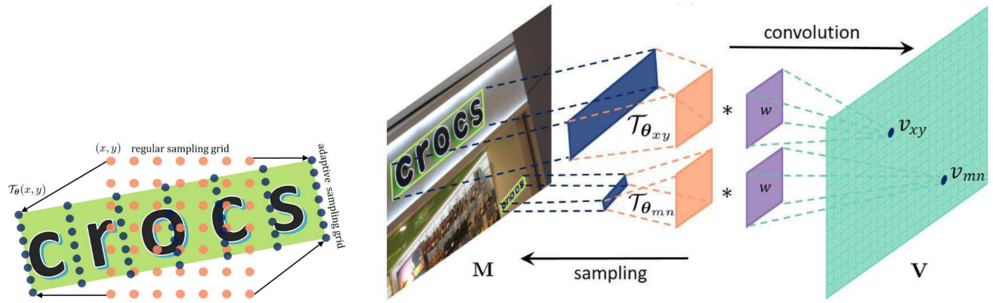
3.4.1 Text Detection

Starting with text detection, He et al. (2016b) proposed an enhancement of the classical MSER algorithm (MATAS et al., 2002) coupled with a so-called Text-CNN. The network was used to eliminate false-positives coming from the MSER-enhanced algorithm. It is interesting to note that the network achieved a high discriminative text/non-text accuracy by using many different annotated information during the training stage, like character label and region mask at the character level. The authors argue that this joint training produced a more robust text/non-text classifier than training with single binary text/non-text labels.

Liao et al. (2017) presented a network inspired by the SSD architecture, which they called TextBoxes. Originally, SSD was developed for object detection, meaning, among many things, that its anchor boxes cover a variety of aspect ratios. However, when applied to text detection, the failure cases occur mainly due to the wider aspect ratio of text. Hence, the authors proposed a new output for the network that manages wider aspect ratios, improving text detection.

Wang et al. (2018) presented an approach to detect text in a variety of geometric positions. It is basically a composition of three CNNs, namely Instance Transformation Network (ITN). The first network is a backbone (e.g., VGG or ResNet) that computes the feature map M from an input image. The second network uses M to infer affine transformations that warp squared grids of fixed size, centered in the cell being analyzed, into the text area (see Figure 3.1a). These new warped grids are used to sample from M and feed a third network that performs classification. An overview of ITN is shown

Figure 3.1: Figures taken from Wang et al. (2018) presenting the warping grid process (a) and an overview of ITN (b).


(x, y) regular sampling grid
$\mathcal{T}_{\theta}(x, y)$
adaptive sampling grid
crocs
convolution
$\mathcal{T}_{\theta_{xy}}$
$\mathcal{T}_{\theta_{mn}}$
*
w
v_{xy}
v_{mn}
M
sampling
V

(a)

(b)

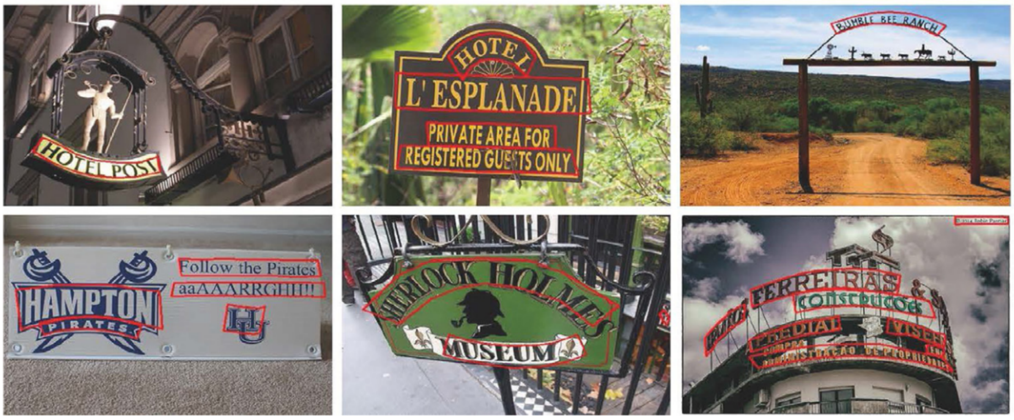
in Figure 3.1b. The authors reported state-of-the-art results in the classical ICDAR 2015 dataset (KARATZAS et al., 2015).

In (DENG et al., 2019), a simple but efficient approach was presented. The authors presented a single architecture that can capture word candidates, and, through a called Corner-based Region Proposal Network, find the corners of that word to retrieve a precise bounding box.

The detection of highly distorted text (e.g., curved) is a new trend, and was recently tackled by Liu et al. (2019) and Baek et al. (2019). In the work of Liu et al. (2019), the authors presented an architecture that recovers a polygon of 14 points, where half of these points correspond to the top part of the text line, and the other half to the bottom part. To train such network, the authors also presented a new dataset with 1,500 annotated images, where the annotations follow the same protocol of the architecture output: 14 points surrounding the text area. A few results of this approach are shown in Figure 3.2.

In the work of Baek et al. (2019), the authors also demonstrated a method for detection of curved text that analyses an input image to character level. To this end,

Figure 3.2: Image taken from Liu et al. (2019) showing a few outputs from their curved text detection approach.



they developed a novel network which outputs two distributions: one to represent the characters, and another to represent the space between characters. Together, they provide enough information to post-processing procedures that infer character deformation and their relation to neighbor characters, which allows to precise draw a polygon around text regions.

3.4.2 Word Spotting

Word spotting techniques are used after text detection to isolate and recognize words. A successful use of CNNs for word spotting can be found in (JADERBERG et al., 2014), where the authors proposed a CNN model with three different kinds of encoding for the output layer:

- Dictionary encoding: it outputs one among 90k different words (in this case, classes) in a lexicon;
- Character sequence encoding: the layer outputs the sequence of characters (given a maximum word length) and "null" when it does not apply;
- N-gram encoding: the output layer is composed of $10k$ different English N-grams. As a word can contain multiple N-grams, high probabilities ideally must be assigned only to the N-grams composing the word.

The use of recurrent networks along with CNNs was also explored in text recognition by Shi, Bai and Yao (2017). The authors used Long-Short Term Memory (LSTM) layers at the end of a sequence of convolutional layers to handle character sequences. The network was named das Convolutional Recurrent Neural Network (CRNN) One advantage of this approach is that the training stage is simplified, needing only words as labels, instead of complicated annotation like segmented characters. We investigate a similar approach in the context of RBN-R in Chapter 6.

Despite being accurate for text recognition, the approaches based on lexicons or N-grams are not directly applicable to our target tasks (e.g., ALPR or RBN-R) since the number of combinations to build a lexicon is immense. However, the character sequence encoding used by Shi, Bai and Yao (2017), Jaderberg et al. (2014) or Jaderberg et al. (2015) may be suitable if not applied directly and submitted first to a retraining stage. In fact, we tried to apply these methods directly to images containing license plates or bibs, and noticed that a network trained for word recognition intrinsically learns to consider the information of the characters around. This is not desirable for license plates or bib numbers as this information is usually not valid.

3.4.3 Joint Detection and Spotting

Following the path of simple annotation, we can refer to the work of Jaderberg et al. (2015), where the authors presented a CNN-based detector and recognizer for numbers trained with only digit sequence strings as an annotation. Their approach, called Spatial Transformer Networks, was used by many authors referenced throughout this dissertation, and also inspired our ALPR and RBN-R approaches, presented in Chapters 5 and 6. Basically, they propose the use of networks as layers inside the main network. These networks regress spatial transformation parameters (for instance, affine parameters) and use them to transform the input feature map. Tests performed in the Street View House Numbers (WANG et al., 2011) dataset demonstrated the ability to learn digit sequence detection without any spatial annotation when plugged into a recognition network. Its worth to mention that the STN was able to perform such detection in small patches and containing a single piece of text, so it does not mean that it can handle the detection of multiple text instances on entire images.

3.5 Chapter Conclusions

This chapter presented existing works for ALPR, RBN-R, AMR and OCR. We can note that the ALPR, AMR and RBN-R problems are closely related. Solutions to these problems share the common characteristic of first detecting the license plate, meter or bib (i.e., the contextual information), and then finding the identification string or number through some OCR method. In all cases, the use of OCR methods directly to the full input image might generate many false positives that are not present in the corresponding context, so that using contextual information might be crucial.

Although ALPR is a well known and studied problem, there is a lack of approaches that correctly recognizes license plates in more complex scenarios, with less constrains, and at a reasonable computational cost. For instance, scenarios where the vehicles are oblique to the camera are still challenging.

Considering the RBN-R problem, the few existing approaches rely on old detection and recognition methods, like HoG+SVM for object detection and the Tesseract engine for OCR. With the rise of Deep Learning, state-of-the-art detectors and OCR approaches started to use deep CNNs, achieving impressive results. Hence, there is a need for a more modern look at this problem, taking care of the challenges described in Section 3.2.

Finally, OCR (its broad context that encompasses detection and recognition) is one of the oldest problems in computer vision. However, the focus of modern approaches are on word and text detection/recognition, and not characters. This line of research limits its use to recognize identification strings, since they often are not words but a sequence of characters not following comprehensive semantic rules. In the next chapters, we present our approaches for ALPR and RBN-R.

4 REAL-TIME ALPR FOR MOSTLY FRONTAL CAPTURES

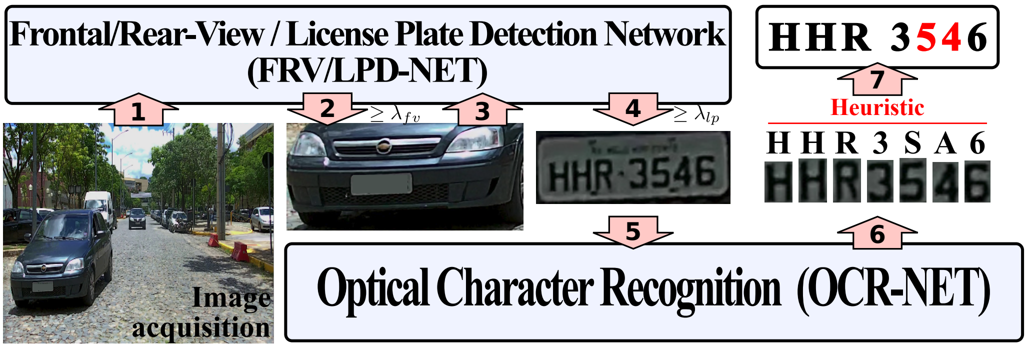
Our first attempt towards real-time CTR was in the context of ALPR, in which the text to be detected (related to the license plate) is contextually related to a vehicle. This chapter describes a real-time ALPR approach suited for mostly frontal views, as in applications such as parking or toll monitoring. Part of this approach was published in a conference paper (SILVA; JUNG, 2017), which was extended and submitted to a journal.

4.1 Overview

Although vehicle detection might be an optional task for some ALPR methods, it is essential in our CNN-based approach. Taking into account that we are using DL techniques known to be computationally expensive, avoiding small objects becomes even more important when one seeks for a real-time application. Shrinking the image is not an alternative, because the plate may become excessively small to the point of being undetectable (in the SSIG Dataset (GONÇALVES et al., 2016), focused on Brazilian plates and used in this work, the average size of the vehicles LPs is 0.26% of the full image area). Nonetheless, vehicles are larger objects and therefore less affected by downsampling, so our strategy is to hierarchically detect the vehicle and then the license plate. Figure 4.1 presents an overview of the proposed method, each step is briefly described next, and then detailed in the following sections.

Another decision concerning real-time performance is the chosen network model. We need to use a DL model capable of performing detection and recognition in a very

Figure 4.1: Proposed system pipeline: a frontal-rear view detector is applied to the input image (1 and 2) and a crop of each detection is re-submitted to the same network in order to locate LPs (3 and 4). Next, the detected LPs are scanned by a character recognition network (5 and 6) and the characters found are sorted left-to-right to compose the system output (7).



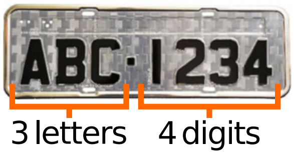
short time. In order to accomplish this goal, we choose to use YOLO-based networks (Section 2.4.3), which are the fastest networks reported in the literature as far as we know. In our tests, the Fast-YOLO network (a reduced version of YOLO) performed a 20-class detection and classification in an 416×416 image in less than 5.5ms[1], or around 180 frames per second (FPS). The precision reported (mAP) for this network was 57.1% on PASCAL-VOC 2007 database. This is not much when compared to other deeper networks like YOLO (76.8%) and Faster-RCNN (76.4%). However, YOLO has $3\times$ more parameters than Fast-YOLO, which makes it slower and data-hungry for training. This is not convenient when aiming a real-time application with just a small set of training samples.

As our intention is to tackle a two-class problem (car frontal/rear views and license plates) instead of twenty, we hypothesized that refining the Fast-YOLO network would be fast and accurate enough to detect both classes in a sequential manner: first the car frontal/rear views are detected, and then its corresponding bounding boxes are fed to the same CNN to extract the license plate, as described in Section 4.3.

For the detection and recognition of characters, we built a new network inspired on the YOLO architecture, with fundamental technical differences to accommodate 35 classes (0-9, A-Z except the letter O, which is detected jointly with the number 0) and to output a feature map having the same aspect ratio of the LP (width is three times larger than height). The details of this network are presented on Section 4.4.

Optionally, a post-processing step is performed in order to swap class of highly confused letters to digits and digits to letters. This is useful in the context of Brazilian LPs since they are formed by exactly three letters followed by four digits, as illustrated in Figure 4.2.

Figure 4.2: Brazilian LP layout: 3 letters followed by 4 digits.



4.2 Frontal/Rear-View Extraction

The farther the vehicle is from the camera, the smaller its LP will appear in the image. As pointed by Zhu et al. (2016) and Redmon et al. (2016), deep learning techniques

[1] Hardware used: Intel Xeon E5620 2.4GHz processor, 12GB of RAM, and NVIDIA TITAN X GPU.

struggle to detect small objects. As a rule of thumb, larger input images are required to allow the detection of smaller objects, which increases the computational cost (HUANG et al., 2017). Thus, directly locating the LP may not be a good strategy when targeting a wide range of scenarios. For the sake of curiosity, we trained a Fast-YOLO network to directly detect the LPs in the Brazilian SSIG Database (GONÇALVES et al., 2016) (where the LPs represent typically just a small portion of the image), and the result was a low recall rate of 82% (which tends to be further reduced after adding the character recognition errors).

Although "vehicle" (or subtypes such as "car", "bus", etc.) is a typical class in object detection/recognition datasets (EVERINGHAM et al., 2007; RUSSAKOVSKY et al., 2015; LIN et al., 2014), only roughly frontal and rear views are adequate in the context of ALPR, so that the corresponding LP is visible and readable. However, ALPR datasets typically present annotation only for the LP region, not for the vehicle (GONÇALVES et al., 2016; HSU; CHEN; CHUNG, 2013; OPENALPR, 2014). To cope with these issues, we propose a simple method to extract the frontal/rear portion of the cars only based on the annotated LP. This is a semi-automatic way of generating new annotations to aggregate information to the datasets in hands. We start by manually annotating frontal/rear views (FRV) regions around the LP in a small set of 40 images from the SSIG Database (where the LP region is also known). Then, we assume that both regions are related by an anisotropic scaling and translation. More precisely, let us consider a 4-dimensional vector $\vec{p} = (p_x, p_y, p_w, p_h)$ encoding an annotated LP, where (p_x, p_y) are the center coordinates and (p_w, p_h) are the width and height. We assume that the FRV bounding box is given by an affine transformation

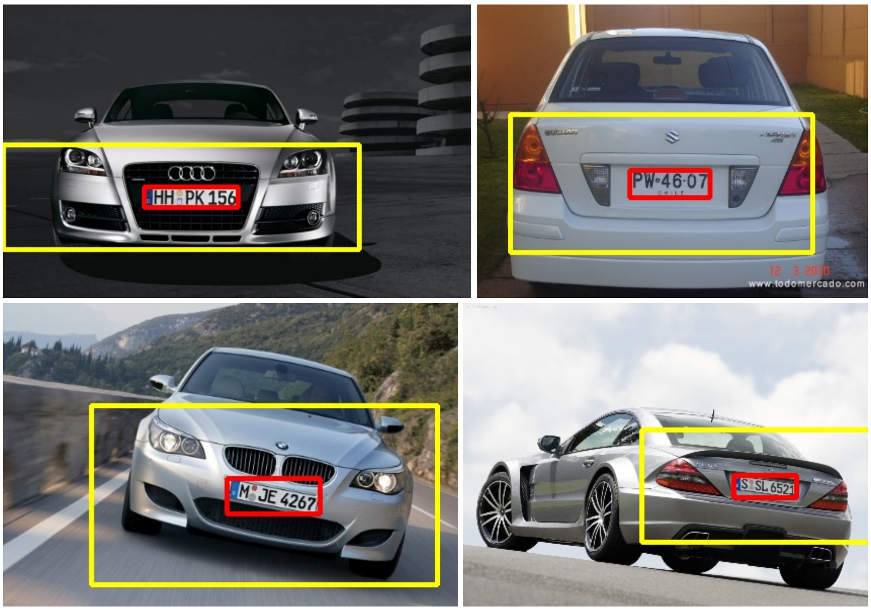
$$\vec{v} = \vec{f}(\vec{p}, \vec{\alpha}) = \begin{bmatrix} p_x + \alpha_x p_w \\ p_y + \alpha_y p_h \\ \alpha_w p_w \\ \alpha_h p_h \end{bmatrix}, \tag{4.1}$$

where $\vec{\alpha} = (\alpha_x, \alpha_y, \alpha_w, \alpha_h)$ contains the scaling and translation parameters. The estimated mapping from LPs and FRVs is encoded by

$$\vec{\alpha}_* = \underset{\vec{\alpha}}{\text{argmax}} \sum_{i=1}^{N_{lp}} IoU\left(f(\vec{p}_i, \vec{\alpha}), \vec{v}_i\right), \tag{4.2}$$

where N_{lp} is the number of images with annotated LP and FRV bounding boxes. We used the Gradient Descent with a learning-rate of 0.01 as the optimization algorithm. Figure 4.3 shows a few examples of estimated FRV bounding boxes on images that were not present in the training set, indicating that the FRV can be roughly extracted using the proposed approach. It is worth mentioning that related approaches can be found in Psyllos, Anagnostopoulos and Kayafas (2011) and Yang et al. (2013), where LP regions where used to estimate car frontal-views in order to recognize their model/manufacturer.

Figure 4.3: Example of Vehicle Frontal/Rear bounding box estimation (yellow rectangle) based on the annotated LP region (in red).



4.3 Frontal/Rear-View and License Plate Detection CNN

In this work, the car FRV and its LP are detected using a single classifier arranged in a cascaded manner (see Figure 4.1): the first layer (arrows 1 and 2) detects the FRV from the input image, and the second layer (arrows 3 and 4) extracts the LP from the detected FRV image.

To achieve a good compromise between accuracy rates and running times, our classifier is based on the Fast-YOLO network architecture. We hypothesized that such network customized for 2 classes only could have the capacity to accommodate both tasks in a single network when executed in a cascaded way. Besides, if we trained two networks separately for each class, the top layers would probably be similar, as they are feature extractors from the same backbone. The proposed network is shown in Table 4.1, and the only modification to the original architecture was made in layer 15, where we reduced the number of filters from 125 to 35 in order to output 2 classes instead of 20 (please refer to Redmon et al. (2016) for more information about the detection layer).

Table 4.1: Frontal/Rear-View and License Plate Detection Network (FRV/LPD-NET): basically a Fast-YOLO network adjusted to output 2 object classes (FRV and LP).

n^o	Layer	Filters	Size	Input	Output
1	conv	16	3 × 3 / 1	416 × 416 × 3	416 × 416 × 16
2	max		2 × 2 / 2	416 × 416 × 16	208 × 208 × 16
3	conv	32	3 × 3 / 1	208 × 208 × 16	208 × 208 × 32
4	max		2 × 2 / 2	208 × 208 × 32	104 × 104 × 32
5	conv	64	3 × 3 / 1	104 × 104 × 32	104 × 104 × 64
6	max		2 × 2 / 2	104 × 104 × 64	52 × 52 × 64
7	conv	128	3 × 3 / 1	52 × 52 × 64	52 × 52 × 128
8	max		2 × 2 / 2	52 × 52 × 128	26 × 26 × 128
9	conv	256	3 × 3 / 1	26 × 26 × 128	26 × 26 × 256
10	max		2 × 2 / 2	26 × 26 × 256	13 × 13 × 256
11	conv	512	3 × 3 / 1	13 × 13 × 256	13 × 13 × 512
12	max		2 × 2 / 1	13 × 13 × 512	13 × 13 × 512
13	conv	1024	3 × 3 / 1	13 × 13 × 512	13 × 13 × 1024
14	conv	1024	3 × 3 / 1	13 × 13 × 1024	13 × 13 × 1024
15	conv	35	1 × 1 / 1	13 × 13 × 1024	13 × 13 × 35
16	detection				

4.3.1 Training

The weights from layers 1 to 14 were transfered (YOSINSKI et al., 2014) from a pre-trained YOLO backbone [2] and refined using samples containing FRV and LP images. More precisely, the training data is provided as a combination of two subsets: (i) images from vehicles having annotated FRVs, where the FRVs were obtained with the weak labeling process described previously; and (ii) cropped FRV regions with their respective annotated LP regions. In both cases, the images were augmented (see Section 4.5) and rescaled to match the network input size of 416×416.

[2] Pre-trained YOLO models for Darknet framework are available at <https://pjreddie.com/darknet/yolo/ >.

4.3.2 Test

The full license plate detection process consists of two passes of the network, as follows:

- The first pass involves the whole image and retrieves only detected FRV: any LP found is discarded;
- Each detected FRV is cropped and fed to the same network; the resulting FRVs – if there is any – are ignored and the LP detection with highest probability is kept.

Note that if we have N LPs in the scene, the network is expected to run $N + 1$ times: one to detect the N frontal or rear views, and N to detect LPs (one for each detected vehicle). Nonetheless, in several ALPR applications, such as parking or toll monitoring, there is only one vehicle at a time.

The advantages of having the same network for two different tasks are two-fold:

- Simplified training process: both classes (FRV and LP) can be trained together in a single step;
- Less physical storage: despite using high-end hardware to achieve real-time performance, it might still be possible to run in mobile devices with far less FPS, but still reasonably fast. Therefore, as DL networks usually consume a considerable amount of memory to be stored, using a single network can bring a huge advantage in dynamic and permanent memory consumption.

4.4 Character detection and recognition

Although each country/region around the world presents specific regulations concerning the layout of LPs, they share common properties: the LP is rectangular, width larger than height, it is formed by letters and numbers distributed horizontally (except for motorcycles), and the font types and sizes do not present a significant variability. If the LP region is correctly detected, a generic OCR algorithm might be used. In this work, however, we present a new character segmentation and recognition algorithm that accounts for the characteristics typically present in license plates.

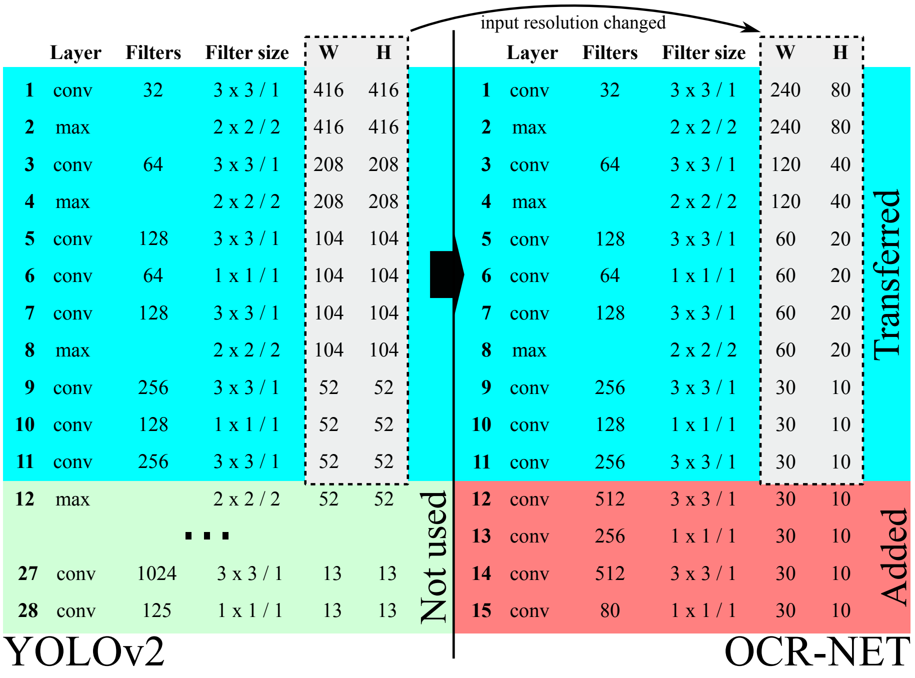
As in the LP detection stage, our OCR network is inspired in the YOLO architecture. However, we made more substantial structural changes. The network input is now an horizontally elongated rectangular region that mimics the expected aspect ratio of a LP set as $3:1$ (based on Brazilian LPs). Although the actual LP aspect ratio might change from region to region, YOLO-based architectures present a good detection performance for varying aspect ratios of the same object (in this case, characters), as demonstrated in Redmon and Farhadi (2017). More precisely, we chose to use 240×80 images as input, which outputs a 30×10 feature map due to the network stride of 2^3. This input size was empirically set to achieve a good compromise between speed and accuracy. Just for the record, we also tried different input $\rightarrow$ output sizes, such as $144 \times 48 \rightarrow 18 \times 6$, $192 \times 64 \rightarrow 24 \times 8$ and $288 \times 96 \rightarrow 36 \times 12$. The first 2 networks performed worse than the chosen one, and the last one performed equally well, but being slower.

To accommodate a smaller input than the original proposal (240×80 against 416×416), other layers had to be modified to keep the architecture consistent. First, we need to cut down the number of max pooling layers from five to three, in order lower the network stride and keep the fine output granularity by avoiding many dimensionality reductions. In this way, there are enough space in the feature map to avoid overlap between characters. Second, to maintain the network depth equal to Fast-YOLO and yet being allowed to use as much transfer learning as possible, we used the first eleven layers of original network (YOLO for PASCALVOC), stopping on the twelfth layer, since it contains the fourth max pooling in that network. If we had applied the same idea of capturing all layers before the fourth max pooling to Fast-YOLO, we would end up using just seven layers, reducing the network depth. Lastly, four more layers were added and trained from scratch to improve non-linearity.

Since the characters are distributed horizontally along the LP, the granularity of the output layer should also be finer along the horizontal direction to better detect and recognize the characters. More precisely, we almost tripled the horizontal granularity compared to original YOLOv2 models (13×13), letting the final network output as 30×10. This slight decrease in vertical size output did not affect the network performance, since the LPs have no vertical sequence of objects to detect. The final architecture of the proposed network and the comparison to YOLOv2 is provided in Figure 4.4, and the loss function remained the same as YOLO, presented in Redmon et al. (2016).

As a final note, in YOLO models the number of filters in the last layer corresponds to the number of anchors, multiplied by the number of classes we want to detect plus five — related to the offset, scale, and probability parameters of an anchor. Since the size of characters does not vary much for LPs, we opted to use only 2 anchors to better

Figure 4.4: Optical Character Recognition Network (OCR-NET): All layers from 1 to 11 were transferred from YOLOv2 network. The input image is a 240×80 gray scaled LP patch with replicated channels, and the output is a 30×10 feature map ensuring enough horizontal granularity for all characters.



	Layer	Filters	Filter size	W	H
1	conv	32	3 x 3 / 1	416	416
2	max		2 x 2 / 2	416	416
3	conv	64	3 x 3 / 1	208	208
4	max		2 x 2 / 2	208	208
5	conv	128	3 x 3 / 1	104	104
6	conv	64	1 x 1 / 1	104	104
7	conv	128	3 x 3 / 1	104	104
8	max		2 x 2 / 2	104	104
9	conv	256	3 x 3 / 1	52	52
10	conv	128	1 x 1 / 1	52	52
11	conv	256	3 x 3 / 1	52	52
12	max		2 x 2 / 2	52	52
		...			
27	conv	1024	3 x 3 / 1	13	13
28	conv	125	1 x 1 / 1	13	13

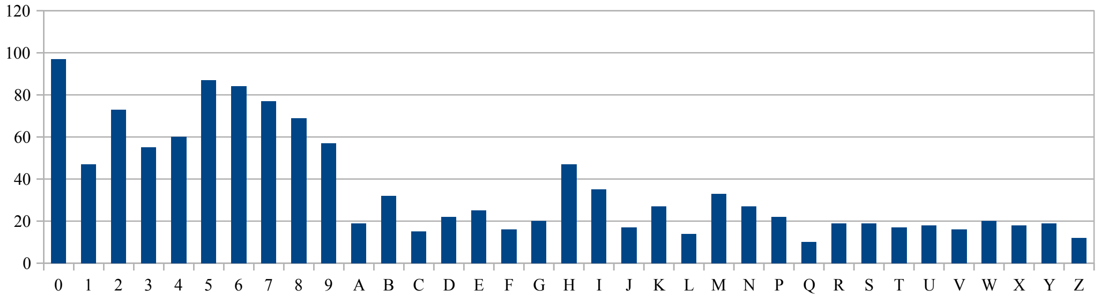
	Layer	Filters	Filter size	W	H
1	conv	32	3 x 3 / 1	240	80
2	max		2 x 2 / 2	240	80
3	conv	64	3 x 3 / 1	120	40
4	max		2 x 2 / 2	120	40
5	conv	128	3 x 3 / 1	60	20
6	conv	64	1 x 1 / 1	60	20
7	conv	128	3 x 3 / 1	60	20
8	max		2 x 2 / 2	60	20
9	conv	256	3 x 3 / 1	30	10
10	conv	128	1 x 1 / 1	30	10
11	conv	256	3 x 3 / 1	30	10
12	conv	512	3 x 3 / 1	30	10
13	conv	256	1 x 1 / 1	30	10
14	conv	512	3 x 3 / 1	30	10
15	conv	80	1 x 1 / 1	30	10

fit wide and thin characters, such as “W” and “I”. The final number of parameters are $2 \times (35 + 5) = 80$, where 35 accounts for the number characters we want to detect.

4.4.1 Training

The training procedure regarding transfer learning was similar to FRV/LPD-NET presented in Section 4.3.1. However, the amount of annotated characters in our database was very small and imbalanced, as shown in Figure 4.5. For instance, there are 14 characters with less than 20 samples each. Hence, we strongly relied on synthetic and augmented data, as explained next.

Figure 4.5: Histogram of characters occurrence in the training database.



4.5 Synthetic and Augmented Training Data

In this work, we created a new dataset for training our networks that consists of a mixture of 195 images obtained from other three publicly available databases: AOLP database (HSU; CHEN; CHUNG, 2013), law enforcement subset (50 images used); SSIG (GONÇALVES et al., 2016), training subset (40 images used); and Cars Dataset (KRAUSE et al., 2013) (105 images used). As each of these databases represents a different scenario, this selection has a positive bias by enhancing the variability. Originally, the LP annotation from those images, when existent, was composed by a rectangular (2-point) bounding box. Although this representation looks adequate for roughly horizontally aligned LPs, when the LP is oblique or rotated, this representation ends up capturing surrounding areas. Thus, a quadrilateral (4-point bounding box) seems to be a more suitable representation. Therefore, this dataset manually replaced the 2-point representation by the 4-point representation, allowing to capture different distortions and rotations, as illustrated in the second row of Figure 4.6a, where a rectification turns out to be possible due to the 4-point annotation. We took advantage of this to greatly augment the dataset for training. Moreover, the dataset also contains 4-point annotations for the characters. The first row of Figure 4.6a shows a sample and its annotations.

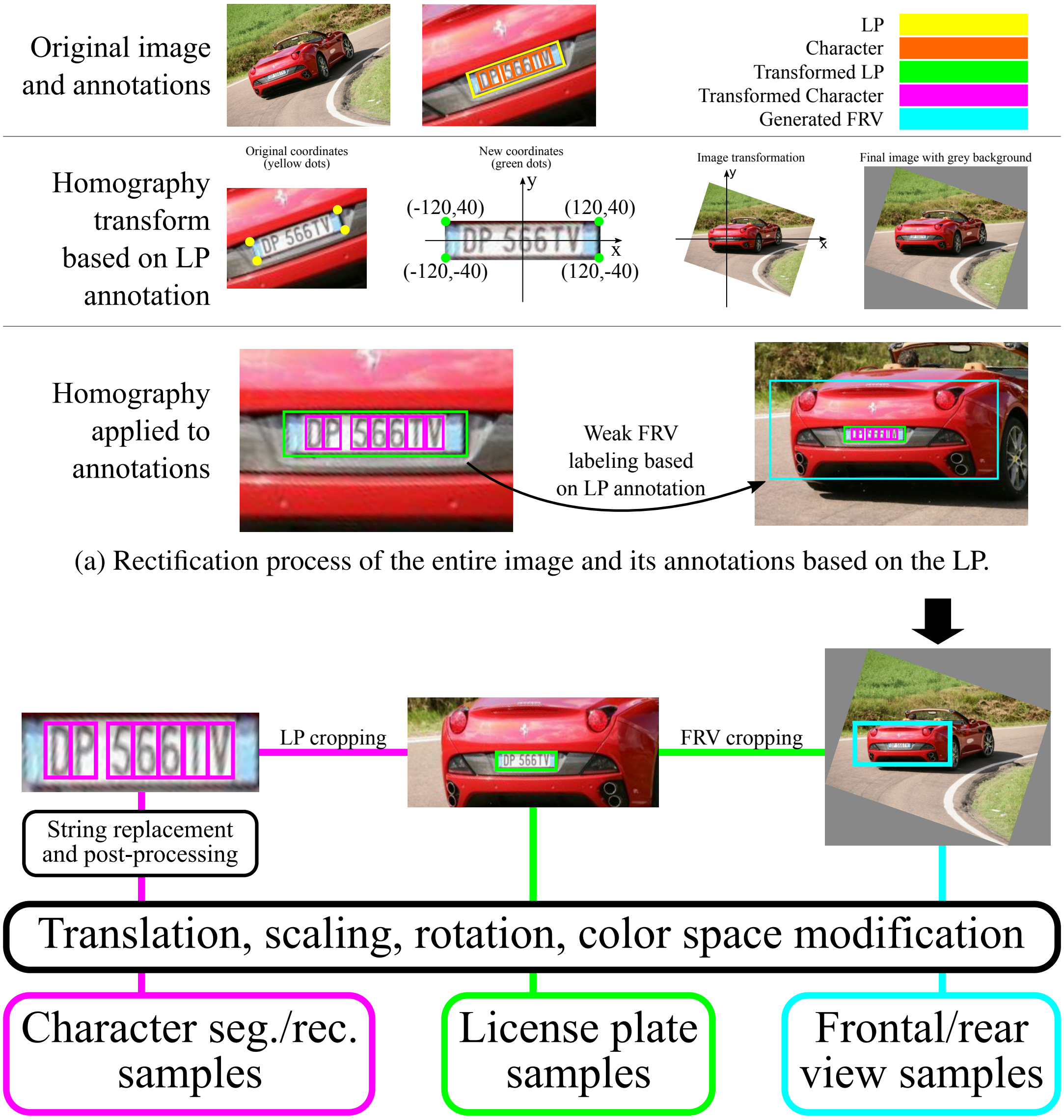
Rectifying the entire image based on its LP is the first step to generate samples for FRV, LP and character detection. For each image, we fixed the new LP size and shape to be a 240×80 rectangle centered in the origin (see the second row of Figure 4.6a). With the 4-annotated LP points from the dataset, and the desired new points, we generated an homography transform matrix used to interpolate the image and its annotations. The empty regions were filled with grey — as it becomes zero when scaling the pixel values between $(-1, 1)$. Finally, an FRV annotation was generated using the new rectified LP coordinates.

A detailed description of how data is generated for each detection part after this rectification process is described below. Nonetheless, a summarized overview is depicted in Figure 4.6b.

4.5.1 Frontal/Rear Views and License Plates Samples

The whole transformed image and its FRV annotation is used as input to train the FRV detector, and the cropped FRV region with its LP annotation is used to train the

Figure 4.6: Augmentation pipeline.


Original image and annotations
LP
Character
Transformed LP
Transformed Character
Generated FRV
Homography transform based on LP annotation
Original coordinates (yellow dots)
New coordinates (green dots)
y
(-120,40)
(120,40)
(-120,-40)
(120,-40)
x
Image transformation
Final image with grey background
Homography applied to annotations
Weak FRV labeling based on LP annotation
LP cropping
FRV cropping
String replacement and post-processing
Translation, scaling, rotation, color space modification
Character seg./rec. samples
License plate samples
Frontal/rear view samples

(a) Rectification process of the entire image and its annotations based on the LP.

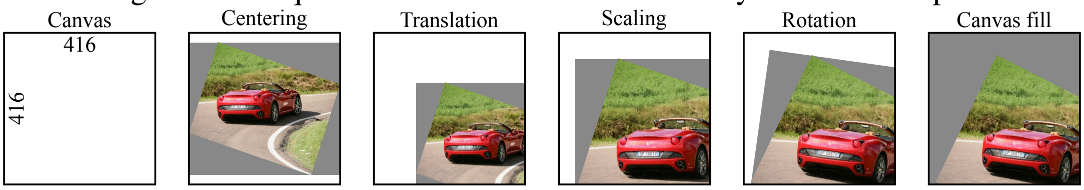
(b) Brief overview of the data generation pipeline after the homography transformation.

LP detector. As explained in Section 4.3.1, during the training stage these samples are merged together to compose the same set, since there is a single network for both tasks.

The network input is fixed in 416×416, thus every sample must be of this resolution. The first step in the augmentation is to create a canvas and resize the image to fit its center. Them, a random translation, scaling and rotation operation (altogether in a single affine matrix) is performed inside the canvas area. This process is depicted in Figure 4.7. It is important to note that we used small angles to avoid aggressive perturbations that do not bring any improvement since the test scenarios only involve frontal (or rear) LP images. Actually, the performance suffers degradation when using high angle values as end up forcing the network to the accommodate unusable patterns.

In the final step, we transferred the sample from the RGB domain to HSV, per-

Figure 4.7: Sequence of tasks to create a randomly modified sample.



formed a slight and fixed modification in its colorspace, and brought it back to RGB.

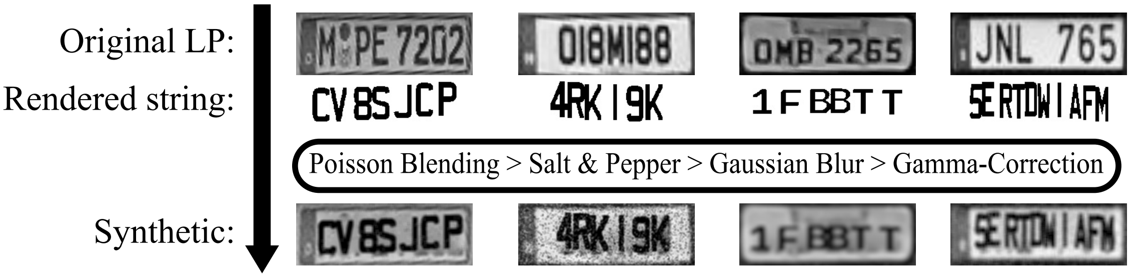
4.5.2 Characters Samples

To train the OCR network, we mixed real augmented data to synthetically generated ones. The real augmented samples are simply a crop of the LPs passed through a set of affine transformations and colorspace modification procedures, similar to FRV and LP samples described above.

The synthetic samples, however, are generated by a more sophisticated process. After the rectification, we end up with well aligned LP and characters. Thus, it is easier to extract the text area and replace its content by a new synthetic sequence of characters. To avoid having a plain synthetic text pasted over a real image (which could bias the network), we mixed real and synthetic data using Poisson Blending (PÉREZ; GANGNET; BLAKE, 2003), which is a traditional method in image composition used for blending contents from two different images.

The steps for generating a synthetic LP are shown in Figure 4.8. First, we generate a random sequence of characters of length varying from 5 to 10 (since LPs from different regions might have a different number of characters). The used font is randomly chosen from two types: European and Brazilian (which are our target scenarios), but other fonts might be easily included into the pipeline. The new string then is re-scaled to match the text area size, and blended with the real image containing the original LP using Poisson Blending. As illustrated in Figure 4.8, this step propagates the information from the real image onto the synthetic text, capturing information such as illumination and texture. Finally, we apply random amounts of Salt & Pepper noise, Gaussian blur (since images captured by video cameras might present some motion blur artifacts) and γ-correction (to modify the illumination). The output samples are also submitted to the same affine transformations of the real augmented samples.

Figure 4.8: Synthetic LPs generated.



4.6 Post-Processing

As previously mentioned in this chapter, several pairs of characters present visual similarity and tend to generate ambiguities. In some countries/regions, the LPs present specific combinations of digits and letters. For instance, Brazilian LPs are formed by exactly three letters followed by four numbers. In those cases, a set of rules can be used to disambiguate and correct classification errors. Other LP regions (e.g. in Europe) do not present a pre-defined order of characters/digits, so that the same disambiguation procedure cannot be applied.

In this work, we present a set of rules designed for Brazilian LPs only as a proof of concept. We used two heuristic rules to filter the results: i) from all characters detected by the network (we set the threshold close to zero for maximum recall), only the seven most probable are kept. ii) from left to right, the first three characters are assumed to be letters, and the following four, digits. This assumption is used to swap letters by numbers and vice-versa, depending on the character position. In summary, if a letter is recognized in the LP block related to digits, it is swapped by the digit that presented the largest occurrence in the confusion matrix obtained with the validation data. A similar process is applied when a digit is recognized in the LP block related to letters. The specific swaps are given by:

- Swap rules for the first 3 positions (letters): $5 \Rightarrow S$, $7 \Rightarrow Z$, $1 \Rightarrow I$, $8 \Rightarrow B$, $2 \Rightarrow Z$, $4 \Rightarrow A$ and $6 \Rightarrow G$;

- Swap rules for the last 4 positions (numbers): $Q \Rightarrow 0$, $D \Rightarrow 0$, $Z \Rightarrow 7$, $S \Rightarrow 5$, $J \Rightarrow 1$, $I \Rightarrow 1$, $A \Rightarrow 4$ and $B \Rightarrow 8$.

4.7 Experimental Results

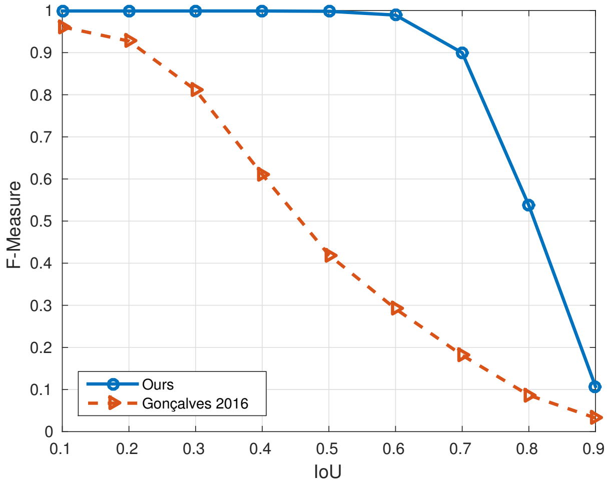
In this section we evaluate the proposed method in two steps. First, we performed individual assessments in each ALPR subtask to verify their performance and set values for fixed (default) parameters. Then, we evaluated the full ALPR system and compared to state-of-the-art academic methods and commercial systems.

To train our LPD and OCR networks, the following parametrization was used: 10k iterations on Stochastic Gradient Descent algorithm, with mini-batch size of 64, and learning rate of 10^{-3} for the first $1k$ iterations, and 10^{-4} after them. The validation set was used to select thresholds to filter the outputs. Basically, all segments found by adopting a threshold λ, were submitted to a Non-Maximum Suppression algorithm in order to filter out overlapped detections (considering an IoU of 0.45). We set $\lambda_{FRV} = 0.2$ and $\lambda_{chars} = 0.6$ for FRV and European characters detection, respectively. For LP and Brazilian characters thresholding is not necessary, since we can assume one LP per FRV, and seven characters per LP. Thus, in those cases, we only select the detections with highest probabilities.

The LP detection was evaluated in terms of Average Precision (AP). Using the typical IoU threshold of 0.5 for validating a detection, the average LPD precision for the European and Brazilian datasets were 90.94% and 96.09%, respectively. Using a higher threshold 0.6 (which provides a tighter detection rule), the AP values are 88.28% and 86.73%, respectively. It is important to note that these values do not provide an upper bound to the full ALPR system: a candidate LP detection presenting an IoU lower that the threshold might be considered a miss in terms of LPD, but it might generate a correct final LP recognition result if all the characters are still inside that region and correctly recognized.

For the Optical Character Recognition Network (OCR-NET), we retrieve always the seven most probable detections for Brazilian LPs (as described in Section 4.6). For the European scenario, we set the detection threshold to 0.6 based on experiments, and all detections above this threshold are kept. The input for OCR-NET is the cropped region related to the LP detected by FRV/LPD-NET. This leads to 783 (out of 804) Brazilian and 106 (out of 108) European LPs to be recognized. Considering the Brazilian scenario, our network (OCR-NET) achieved more than 99% of recall and precision (for an IoU of 0.5) to segment the characters, with just a few position mistakes. We compared the segmentation to the method presented in GonÇalves, Menotti and Schwartz (2016) in terms of F-measure (which combines precision and recall) for different IoU acceptance thresholds. The results are shown in Figure 4.9. In particular, for an IoU of 0.5, our network achieved an F-measure of 99.82% against 41.96% of GonÇalves, Menotti and Schwartz (2016), showing that it successfully segmented most of the characters.

Figure 4.9: Character Detection performance in the SSIG-test dataset: F-Measure comparison with the segmentation method presented in (GONÇALVES; MENOTTI; SCHWARTZ, 2016).



Regarding the proposed full ALPR pipeline, most existing approaches present their results on specific databases, so that a comprehensive evaluation is jeopardized. In this section, we compared our results with Sighthound (MASOOD et al., 2017)[3] and Amazon Rekognition [4]. For the Brazilian dataset SSIG, we have also compared with our previous conference paper (SILVA; JUNG, 2017) and the recent work of Laroca et al. (2018). It is important to mention that Sighthound was trained using large private datasets (over 20,000 images). On the other hand, the proposed approach was trained using only 195 real images, and strongly relied on synthetic images and data augmentation to generalize the network.

The final results are presented in Table 4.2. In the Brazilian scenario, our system successfully recognized 89.15% of the license plates present in the test set, resulting in a relative improvement of 9.4% when compared to Sighthound. We also performed better than the work of Laroca et al. (2018), using only two CNNs against five, and our previous approach (in this case by a large margin). Moreover, the same system can also work for European LPs. In our tests we closely outperformed Sighthound system in the OpenALPR European benchmark. Examples of ALPR results produced by our system are depicted in

[3] State-of-the-art commercial system, which has a cloud service available at <https://www.sighthound.com/products/cloud.>. The results presented here were collected on March, 2018.

[4] The results presented here were collected on March, 2018.

Table 4.2: Evaluation of the full ALPR pipeline considering the percentage of fully correct recognized LPs.

ALPR	**SSIG (Brazilian)**	**OpenALPR (European)**
Ours	**89.15%**	**85.19%**
Sighthound*	81.46%	83.33%
Amazon Rekognition*	31.21%	69.44%
Laroca et al (2018)	85.45%	–
Silva and Jung (2017)	63.18%	–

*Data collected on February, 2018.

Figure 4.10.

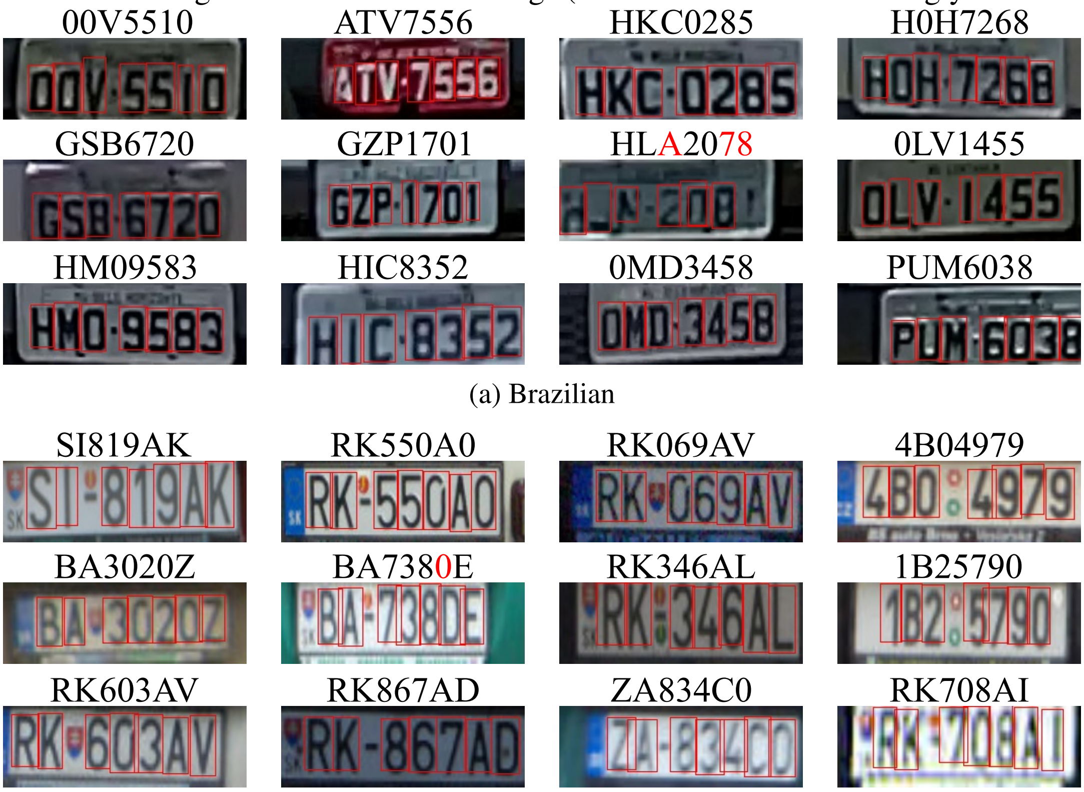
Our final evaluation is regarding execution time. In Table 4.3 we show the average time needed for each network to process an input, and the total time for the whole system (assuming that a single vehicle is being processed). Our ALPR system was implemented using the Darknet framework [5], and it runs at 76 FPS using a high-end GPU, and it can achieve around 9 FPS with a cheaper mid-end GPU, which is also feasible in several applications such as parking and toll monitoring systems.

Table 4.3: Execution time: networks feed forward times for high and mid-end GPUs.

Network	Time (FPS) High-end GPU (NVIDIA TITAN X)	 Mid-end GPU (GeForce GT 750M)
FRV/LPD-NET	5.4ms (185)	47.2ms (21)
OCR-NET	2.2ms (448)	20.1ms (47)
Total (2× FRV/LPD-NET + OCR-NET)	**13.0ms (76)**	**114.5ms (9)**

[5] Darknet website: <https://pjreddie.com/darknet/>.

Figure 4.10: Examples of detected and recognized LPs. The images are the detected LPs using FRV/LPD-NET, and the segmented characters are represented by the red rectangles. The final LP string is shown above each image (characters in red were wrongly classified).



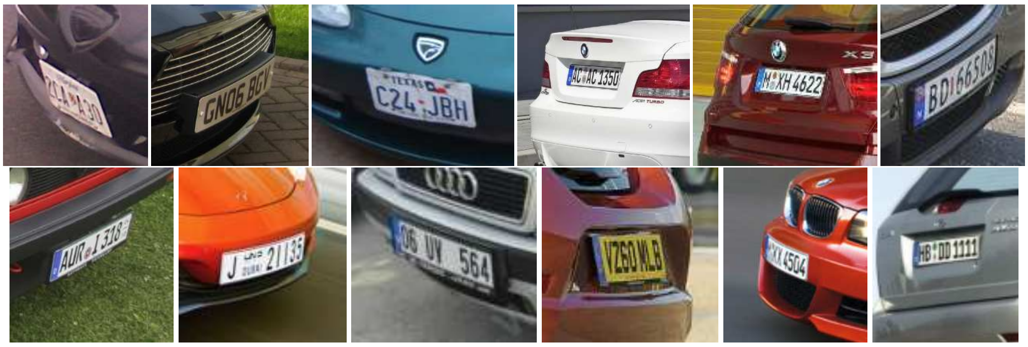
(a) Brazilian

(b) European

5 ALPR IN UNCONSTRAINED SCENARIOS

Chapter 4 presented an ALPR approach focused on mostly frontal/rear views. In more generic capture situations, such as a mobile camera held by a law/traffic enforcement agent, the images could produce very oblique views of the vehicles and the LPs, such as the ones shown in Figure 5.1. In these situations, the (rectangular) bounding box representation for the LP is not adequate, since it might either crop part of the text in the LP or encompass a considerable background region. This chapter presents an ALPR approach suited for unconstrained capture scenarios, and a paper that reflects most of the chapter (SILVA; JUNG, 2018) was published at ECCV 2018.

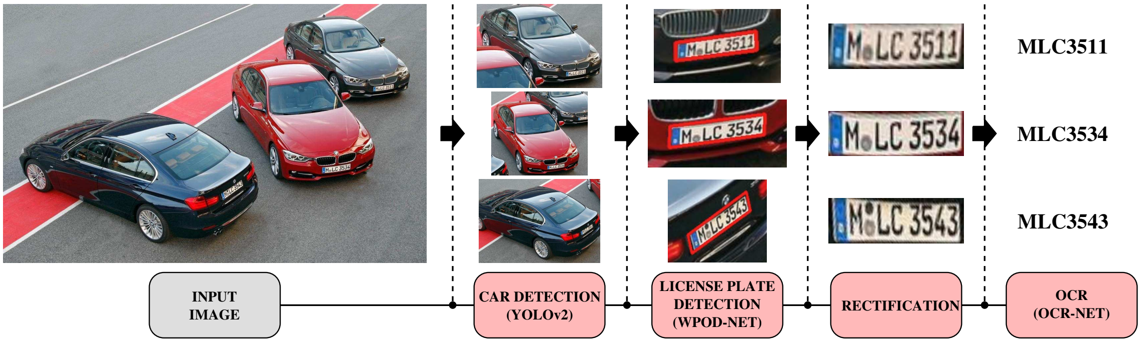
Figure 5.1: Examples of challenging oblique license plates present in the proposed evaluation dataset.



5.1 Overview

The proposed approach is composed of three main steps: vehicle detection, LP detection and OCR, as illustrated in Figure 5.2. Given an input image, the first step is to find vehicles in the scene. Within each detected vehicle, our proposed Warped Planar Object Detection Network (WPOD-NET) searches for LPs (even distorted and oblique ones), and returns a set of quadrilaterals whose corners approximately match the LP corners. A Non-Maximal Suppression algorithm is applied to filter out overlapping detections, and the final detections are unwarped to be a fixed aspect ratio rectangle (3:1), emulating a frontal view. These positive detections are fed to an OCR Network for final character recognition.

Figure 5.2: System pipeline.



5.2 Vehicle Detection

Recall that Chapter 4 focused on mostly frontal/rear views, and a weak labeling procedure was applied to train a vehicle detector in such views. For unconstrained scenarios, however, a generic vehicle detector can be used directly. In fact, vehicles are one of the popular objects present in many classical detection and recognition datasets, such as PASCAL-VOC (EVERINGHAM et al., 2010), ImageNet (RUSSAKOVSKY et al., 2015), and COCO (LIN et al., 2014). Hence, we decided not to train a detector from scratch, and instead chose an existing model to perform vehicle detection.

On the one hand, a high recall rate is desired, since any miss-detected vehicle having a visible LP leads directly to an overall LP miss-detection. On the other hand, high precision is also desirable to keep running times low, since each falsely detected vehicle must be verified by WPOD-NET. Based on these considerations, we decided to use the YOLOv2 object detection network due to its fast execution (around 70 FPS) and good precision and recall compromise (76.8% mAP over the PASCAL-VOC dataset). We did not perform any change or refinement to YOLOv2, just used its PASCAL-VOC pre-trained model as a black box, and merged the classes related to cars and buses, ignoring the others.

The positive detections are resized before being fed to WPOD-NET. As a rule of thumb, larger input resolutions allow the detection of smaller objects but increase the computational cost (HUANG et al., 2017). In roughly frontal/rear views, the ratio between the LP and the vehicle bounding box (BB) areas is relatively high. On the other hand, this ratio tends to be much smaller for oblique/lateral views, since the vehicle BB tends to be larger and more elongated. Hence, oblique views should be resized focusing on having a better granularity than frontal views, to keep the LP region recognizable. Figure 5.3 shows examples of oblique and frontal views, along with the LP dimensions relative to the whole vehicle image crop.

Although 3D pose estimation methods such as (ZHOU et al., 2017) might be used to determine the resize scale, this work presents a simple and fast procedure based on the aspect ratio of the vehicle BB. When it is close to one (1:1), a smaller dimension can be used, and it must be increased as the aspect ratio gets larger. More precisely, the resizing factor f_{sc} is given by

$$f_{sc} = \frac{\min\left(D_{min}\frac{\max(W_v,H_v)}{\min(W_v,H_v)}, D_{max}\right)}{\min(W_v, H_v)}, \tag{5.1}$$

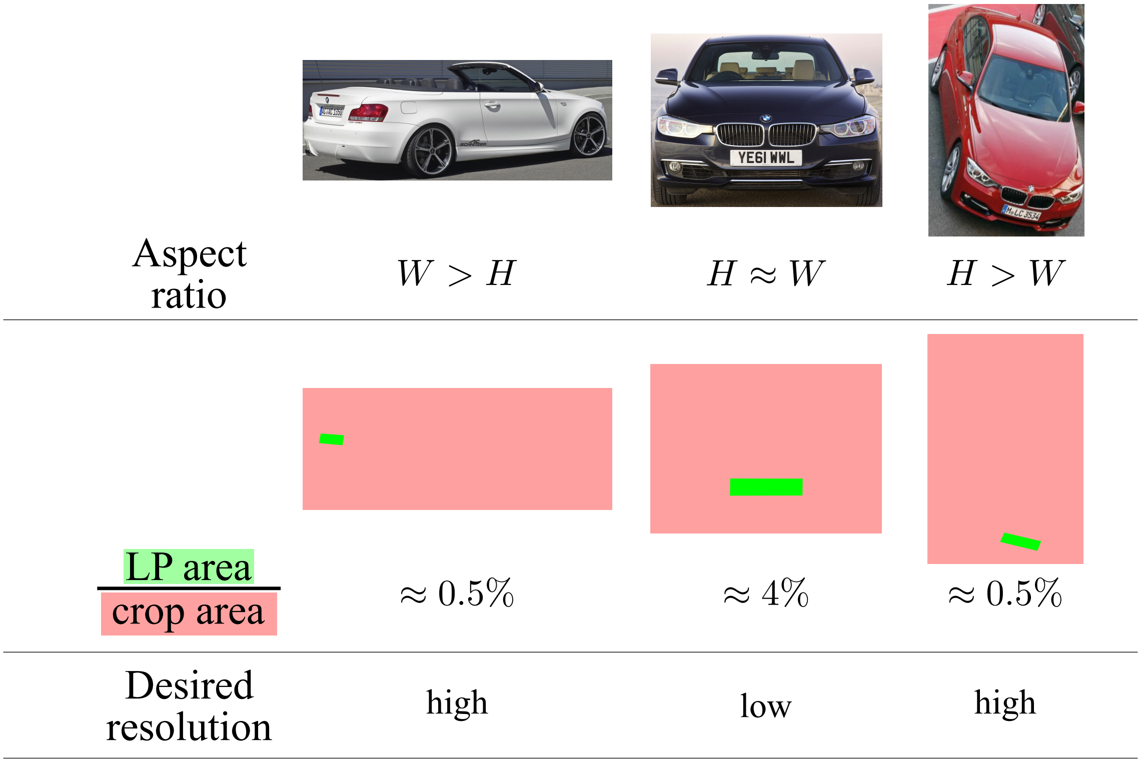
where W_v and H_v are the height and width of the vehicle BB, respectively. Note that $D_{min} \leq f_{sc}\min(W_v, H_v) \leq D_{max}$, so that D_{min} and D_{max} delimit the range for the smallest dimension of the resized BB. Based on experiments and trying to keep a good compromise between accuracy and running times, we selected $D_{min} = 288$ and $D_{max} = 608$ (these values are multiples of the stride $2^4 = 16$ of the network, as it will be discussed next). A drawback of this solution is when the car is not appearing entirely in the scene, like in close captures. In this situation, the LP might be much larger than in a normal capture, needing a re-calibration of the parameters, for instance a reduction of D_{min}.

5.3 License Plate Detection and Unwarping

License plates are intrinsically rectangular and planar objects, which are attached to vehicles for identification purposes. In order to take advantage of its shape, we proposed a novel CNN called Warped Planar Object Detection Network. This network learns to detect LPs in a variety of different distortions, as well as the coefficients of an affine transformation that “unwarps” the distorted LP into a rectangular shape resembling a frontal view. Although a planar perspective projection could be learned instead of the affine transform, it presents two extra DOFs (6 vs. 8), which typically implies in larger training datasets. Furthermore, the division involved in the planar perspective might lead to numerical instabilities when training the network.

WPOD-NET was developed using insights from YOLO, SSD and Spatial Transformer Networks (STN) (JADERBERG et al., 2015). On the one hand, YOLO and SSD perform fast object bounding box detection and recognition, but they do not take spatial transformations into account, generating only rectangular bounding boxes for every de-

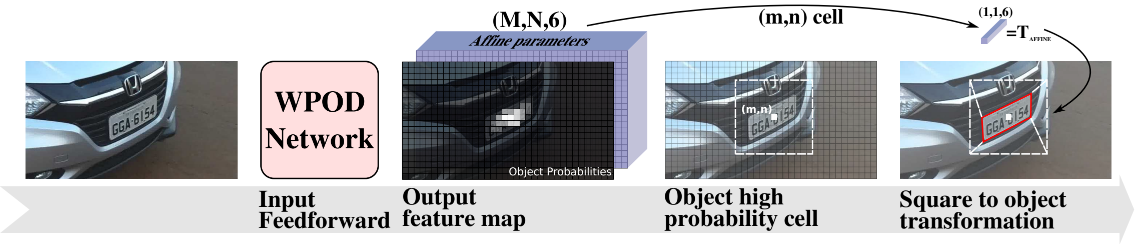
Figure 5.3: Examples of vehicles in different capture angles. In oblique views (first and third pictures), the LP area related to the whole crop tends to be small. In mostly frontal captures (middle), LP proportional area tends to be larger. Also, note that the aspect ratio of the vehicle crop region relates to the capture angle.



tection. On the other hand, STN can be used for detecting non-rectangular regions, but cannot be used in a fully convolutional fashion as in YOLO or SSD, since it performs a single spatial transformation over the entire input. As the scene may contain multiple LPs laid on different planes, this behavior is not desired.

The detection process using WPOD-NET is illustrated in Figure 5.4. Initially, the network is fed by the resized output of the vehicle detection module. The feedforwarding results in an 8-channel feature map that encodes the two probabilities (object/non-object) and the six affine transformation parameters. To extract the warped LP, let us first consider an imaginary square of fixed size around the center of a cell (m, n). If the object probability for this cell is above a given detection threshold, the regressed parameters are

Figure 5.4: Fully convolutional detection of planar objects. The input is the BB corresponding to a detected vehicle, but we show here just a cropped region around th LP for visualization purposes.



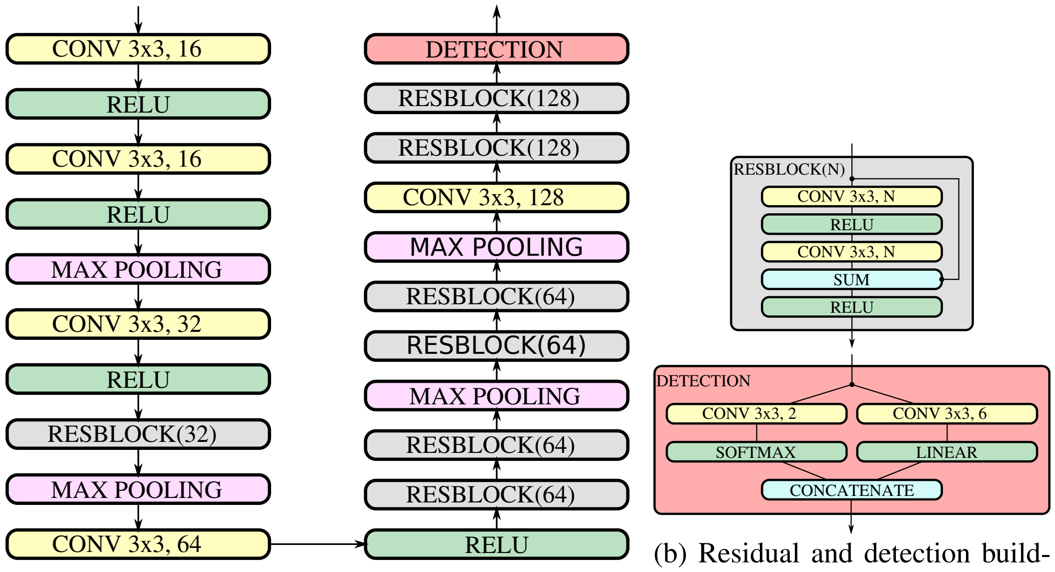
used to build an affine matrix that transforms the fictional square into an LP region. Thus, we can easily unwarp the LP into a horizontally and vertically aligned object.

5.3.1 Network architecture

In the previous approach focused on frontal/rear views, we used an adaptation of the YOLO model, which is fast and presented good results. For the task of learning the affine transformation of distorted LPs, however, our tests with YOLO-based backbones failed to either converge during the training process, or caused the loss function not to decrease enough, leading to a poor inference of the affine transformation. We have also tried other fast backbones, such as Mobilenet, and faced the same difficulties. Based on these observations, we have explored Residual Networks (HE et al., 2016a) to infer complex non-linear patterns, as explained next.

The proposed architecture presents a total of 21 convolutional layers, where 14 are inside residual blocks. The size of all convolutional filters is fixed to 3×3. ReLU activations are used throughout the entire network, except in the detection block. There are four max-pooling layers of size 2×2 and stride two that reduces the input dimensionality by a factor of 16. Finally, the detection block presents two parallel convolutional layers: (i) one for inferring the probability, activated by a softmax function, and (ii) another for regressing the affine parameters, without activation (or, equivalently, using the identity $\vec{F}(\vec{x}) = \vec{x}$ as the activation function).

Figure 5.5: Detailed WPOD-NET architecture.



(a) Network Layers.

(b) Residual and detection building blocks.

5.3.2 Loss function

Let $\vec{p_i} = [x_i, y_i]^T$, for $i = 1, \cdots, 4$, denote the four corners (top-left, top-right, bottom-right and bottom-left, respectively) of an annotated LP. Also, let $\vec{q_1} = [-0.5, -0.5]^T$, $\vec{q_2} = [0.5, -0.5]^T$, $\vec{q_3} = [0.5, 0.5]^T$, $\vec{q_4} = [-0.5, 0.5]^T$ denote the corresponding vertices of a canonical unit square centered at the origin.

For an input image with height H and width W, and network stride given by $N_s = 2^4$ (recall that there are four max pooling layers), the network output feature map consists of an $M \times N \times 8$ volume, where $M = H/N_s$ and $N = W/N_s$. For each point cell (m, n) in the feature map, there are eight values to be estimated: the first two values (v_1 and v_2) are the object/non-object probabilities, and the last six values (v_3 to v_8) are used to build the local affine transformation T_{mn} given by:

$$\vec{T}_{mn}(\vec{q}) = \begin{bmatrix} \max(v_3, 0) & v_4 \\ v_5 & \max(v_6, 0) \end{bmatrix} \vec{q} + \begin{bmatrix} v_7 \\ v_8 \end{bmatrix}, \tag{5.2}$$

where the max function used for v_3 and v_6 was adopted to ensure that the diagonal is positive (avoiding undesired mirroring or excessive rotations).

To transfer the annotation domain (pixels) to the network output domain (feature map size), the points $\vec{p_i}$ are re-scaled by the inverse of the network stride, and re-centered according to each point (m, n) in the feature map. This is accomplished by applying a normalization function:

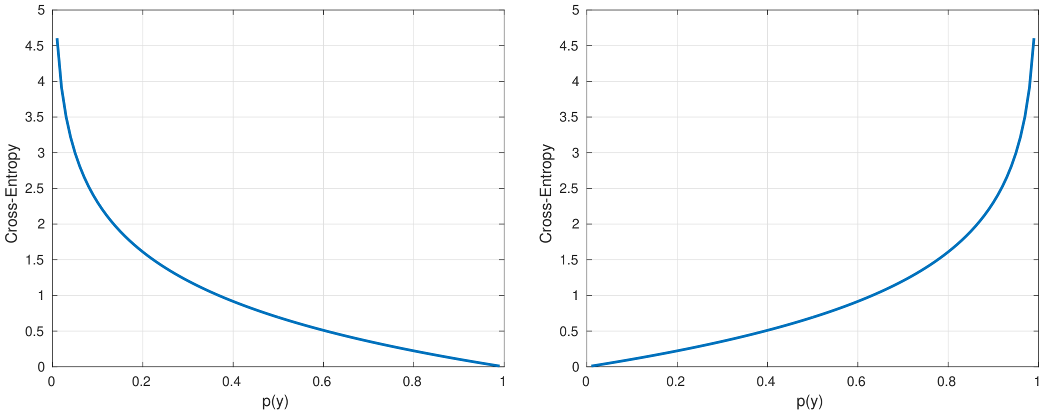
$$\vec{A}_{mn}(\vec{p}) = \frac{1}{\alpha}\left(\frac{1}{N_s}\vec{p} - \begin{bmatrix} n \\ m \end{bmatrix}\right), \tag{5.3}$$

where α is a scaling constant that represents the side of the fictional square. We set $\alpha = 7.75$, which is the mean point between the maximum and minimum LP dimensions in the augmented training data divided by the network stride.

Assuming that there is an object (LP) at cell (m, n), the first part of the loss function considers the error between a warped version of the canonical square and the normalized annotated points of the LP, given by

$$f_{affine}(m, n) = \sum_{i=1}^{4} \|\vec{T}_{mn}(\vec{q_i}) - \vec{A}_{mn}(\vec{p_i})\|_1. \tag{5.4}$$

Figure 5.6: Binary cross-entropy values.



(a) When $y = 1$.

(b) When $y = 0$.

We opted to use the L1-norm , like SSD and Faster R-CNN, instead of a quadratic function, like YOLO. In empirical tests, the average IoU was slightly better.

The second part of the loss function handles the probability of having/not having an object at (m, n). For this we used the binary cross-entropy, which is similar to the SSD confidence loss (LIU et al., 2016), and basically is the sum of two log-loss functions [1]:

$$\text{cross\_entropy}(y, p(y)) = -\left(y \log(p(y)) + (1 - y) \log(1 - p(y))\right) \tag{5.5}$$

where y is the true probability $\in \{0, 1\}$ and $p(y)$ is the predicted probability. Figure 5.6 depicts the values assumed by the cross-entropy w.r.t. y. We can note that, when $p(y)$ and y are close, the entropy tends to zero. When they are very different (i.e. $p(y) \approx 0$ or $p(y) \approx 1$ when $y = 1$ and $y = 0$, respectively), then the entropy grows asymptotically to infinity with $\log(p(y))$ or $\log(1 - p(y))$.

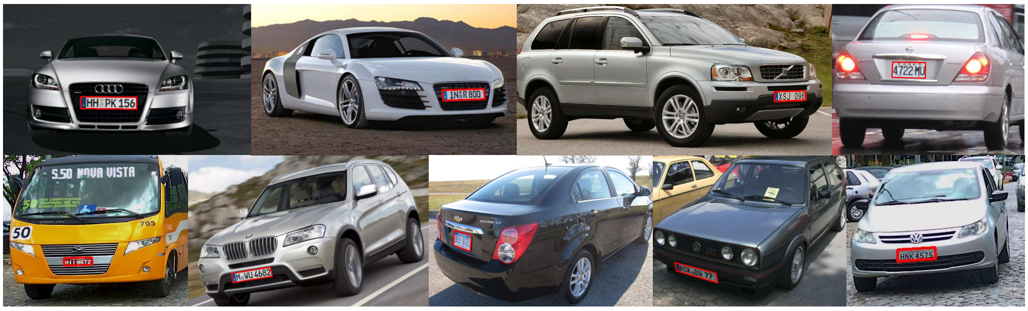
Since v_1 and v_2 are related through the *softmax* function, where $v_1 + v_2 = 1$, our loss considers the value of v_1, which is the object probability:

$$f_{probs}(m, n) = \text{cross\_entropy}(\mathbb{I}_{obj}, v_1), \tag{5.6}$$

where $\mathbb{I}_{obj}$ is the object indicator function that returns 1 if there is an object at point (m, n) or 0 otherwise. An object is considered inside a point (m, n) if its rectangular bounding box presents an IoU larger than a threshold γ_{obj} (set empirically to 0.3) w.r.t. another

[1] $\text{logloss}(y, p) = -y \log(p)$.

Figure 5.7: Examples of the annotated LPs in the training dataset.



bounding box of the same size and centered at (m, n).

The final loss function is then obtained by adding the partial terms defined in Eqs. (5.4) and (5.6):

$$loss = \sum_{m=1}^{M} \sum_{n=1}^{N} [\mathbb{I}_{obj} f_{affine}(m, n) + f_{probs}(m, n)]. \tag{5.7}$$

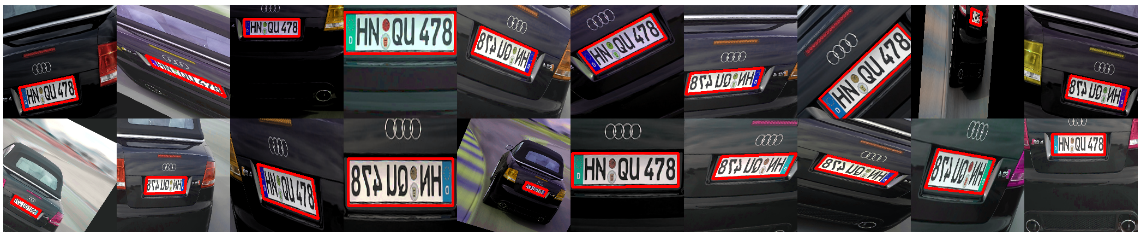
5.3.3 Training Data

For training the proposed WPOD-NET, we created a dataset with 195 images, being 105 from the Cars Dataset (KRAUSE et al., 2013), 40 from the SSIG Dataset (training subset), and 50 from the AOLP Dataset – LE subset (HSU; CHEN; CHUNG, 2013). For each image, we manually annotated the four corners of the LP in the picture (sometimes more than one). The selected images from the Cars Dataset are mostly European, but there are many US LPs as well as LPs from other countries. Images from SSIG and AOLP contain Brazilian and Chinese LPs, respectively. A few annotated samples are shown in Figure 5.7.

Given the reduced number of annotated images in the training dataset, and the fact that we need to train our model from scratch, the use of data augmentation is crucial. It takes as input an image from our training dataset, and performs the following sequence of modifications:

- Rectification: the entire image is rectified based on the LP annotation, assuming that the LP lies on a plane;
- Aspect-ratio: the LP aspect-ratio is randomly set in the interval $[2, 4]$ to accommodate sizes from different regions;
- Centering: the LP center becomes the image center;

Figure 5.8: Different augmentations for the same sample. The red quadrilateral represents the transformed LP annotation.



- Scaling: the LP is scaled so its width matches a value between $40px$ and $208px$ (set experimentally based on the readability of the LPs). This range is used to define the value of α used in Eq. (5.3);
- Rotation: a 3D rotation with randomly chosen angles is performed, to account for a wide range of camera setups;
- Mirroring: 50% chance [2];
- Translation: random translation to move the LP from the center of the image, limited to a square of 208×208 pixels around the center;
- Cropping: considering the LP center before the translation, we crop a 208×208 region around it;
- Colorspace: slight modifications in the HSV colorspace;
- Annotation: the locations of the four points related to LP corners are adjusted by applying the same spatial transformations used to augment the input image.

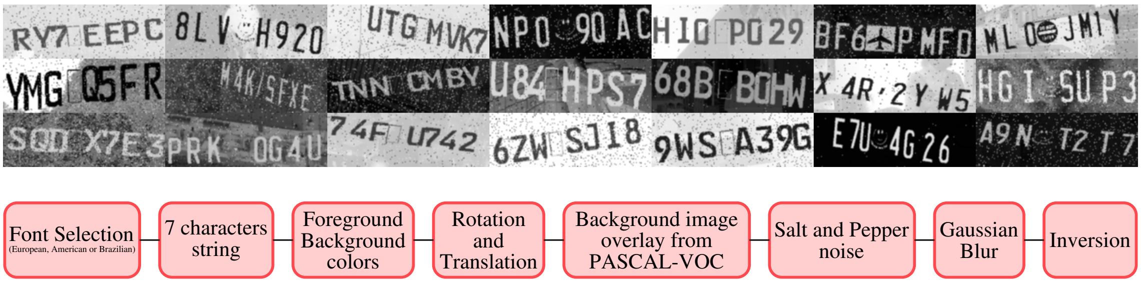
From the chosen set of transformations mentioned above, a great variety of augmented test images with very distinct visual characteristics can be obtained from a single manually labeled sample. For example, Figure 5.8 shows 20 different augmentation samples obtained from the same image.

5.3.4 Training Protocol

We initialized the network weights using Glorot's method (GLOROT; BENGIO, 2010), and trained for $100k$ mini-batches iterations of size 32 using the ADAM opti-

[2] Mirroring might sound counter-intuitive since the characters will be mirrored too. However, at this point, we are not performing OCR yet, so it is more important to have richer information about the vehicles and the surrounding region of the LP than the characters themselves.

Figure 5.9: Artificially created LP samples with the proposed generation pipeline (bottom).


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

mizer (KINGMA; BA, 2014b). The learning rate was set to 0.001 and its β_1 and β_2 parameters were set to 0.9 and 0.999, respectively. The mini-batches were created by randomly choosing and augmenting samples from the training set, being tensors of size $32 \times 208 \times 208 \times 3$. After the feed-forwarding, the network outputs a $32 \times 13 \times 13 \times 8$ tensor for each mini-batch. Note that this is just the training set up, and it does not mean that the network input must be a fixed 208×208 image. In fact, the input image can be of any size multiple of 16 (stride). However, the width for any LP inside the input image must be higher than 40 and lower than 208 pixels (see “Scaling” in the data augmentation process).

5.4 Character Detection and Recognition

Since the output of WPOD-NET is a rectified version of the LP emulating a frontal view, character detection and recognition is performed using the network presented in Section 4.4. However, the training dataset was considerably enlarged in this part of the work by using synthetic and augmented data to cope with LP characteristics of different regions around the world (Europe, the United States and Brazil)[3].

The artificially created data consist of pasting a string of seven characters onto a textured background, and then performing random transformations such as rotations, translations, noise contamination and blur. Some generated samples and a short overview of the pipeline for synthetic data generation are shown in Figure 5.9. As shown in Section 5.5.2, the use of synthetic data helped to greatly improve the network generalization, so that the exact same network performs well for LPs of different regions around the world.

[3] Although we also used Chinese LPs, we could not find information in English about the font type used by this country in order to include in the artificial data generation.

5.5 Experimental Results

This section covers the experimental analysis of our full ALPR system, as well as comparisons with other state-of-the-art methods and commercial systems. Unfortunately, most academic ALPR papers focus on specific scenarios (e.g., single country or region, environment conditions, camera position, etc.). As a result, there are many scattered datasets available in the literature, each one evaluated by a subset of methods. Moreover, many papers are focused only on LP detection or character segmentation, which limits even more the end-to-end comparison possibilities. In particular, the LP detection procedure proposed in this chapter generates a quadrilateral region as output, while typical LP annotations are provided as rectangular BBs. As such, the experimental analysis in this chapter will focus on the final LP recognition result. Next, we describe the chosen datasets used in the experimental validation. We also present, at the end of the section, an evaluation considering the scenario where our system was tuned to deal with LPs from a single country (Brazil).

5.5.1 Evaluation Datasets

One of our goals is to develop a technique able to perform well in a variety of unconstrained scenarios, but that should also work well in controlled ones (such as mostly frontal views). We chose four datasets available online, namely OpenALPR (BR and EU), SSIG and AOLP (RP), which cover many different situations, as summarized in the first part of Table 5.1. We consider three distinct variables: LP angle (frontal and oblique), distance from vehicles to the camera (close, intermediate and far), and the region where the pictures were taken.

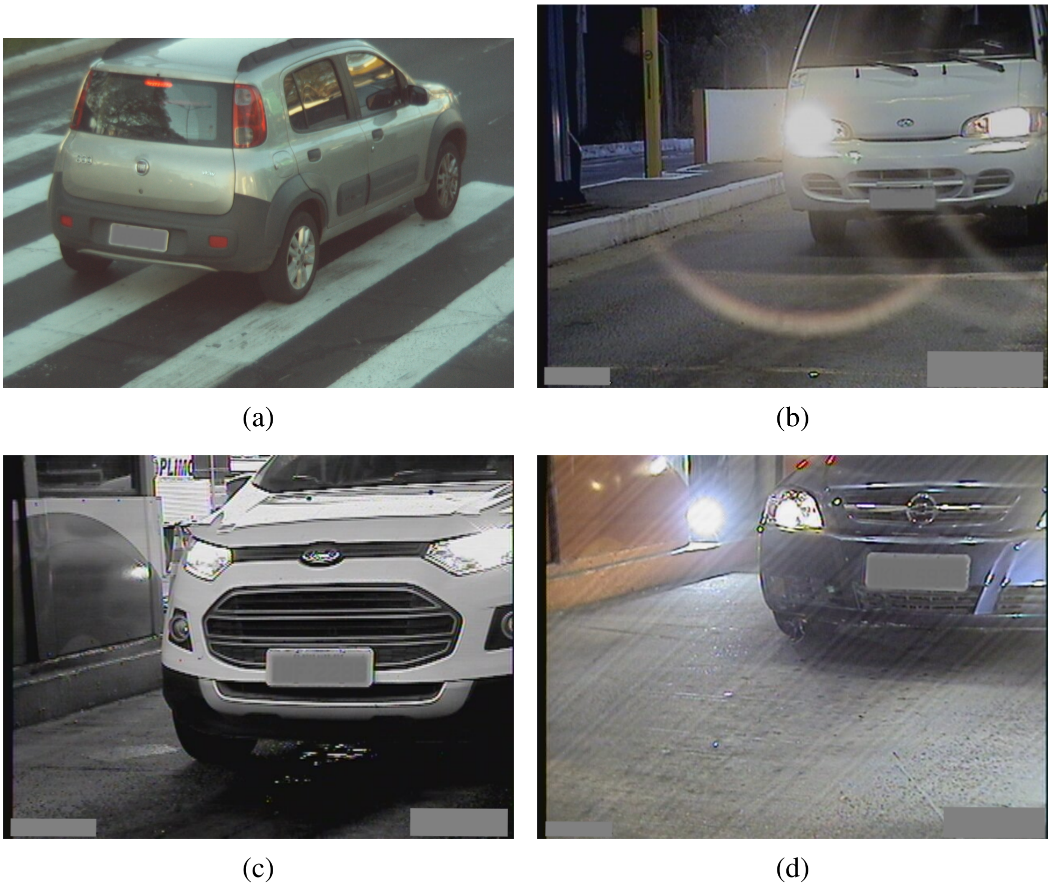
Table 5.1: Evaluation databases.

Database (subset)	**LP angle**	**Vehicle Dist.**	**#images**	**Region**
OpenALPR (EU)	mostly frontal	close	108	Europe
OpenALPR (BR)	mostly frontal	close	114	Brazil
SSIG (test-set)	mostly frontal	medium,far	804	Brazil
AOLP (Road Patrol)	frontal + oblique	close	611	Chinese
Proposed (CD-HARD)	mostly oblique	close,medium,far	102	Various

The more challenging dataset currently used in terms of LP distortion is the AOLP Road Patrol (RP) subset, and it tries to simulate the case where a camera is installed in a patrolling vehicle or hand-held by a person. In terms of distance from the camera to the vehicles, the SSIG dataset appears to be the most challenging one. It is composed of high-resolution images, allowing that LPs from distant vehicles might still be readable. None of them presents LPs from multiple (simultaneous) vehicles at once.

Although all these databases together cover numerous situations, to the best of our knowledge there is a lack of more general purpose dataset with challenging images in the literature. Thus, an additional contribution of this work is the manual annotation of a new set of 102 images (named as CD-HARD [4] [5]) selected from the Cars Dataset, covering a variety of challenging situations. We selected only images that pose difficulties to be detected, and where the LPs are still readable for humans. In fact, some of these images (crops around the LP region) are shown in Figure 5.1, which was used to motivate the problem tackled in this chapter.

Figure 5.10: Samples from the private dataset. For legal reasons the license plates were covered.



(a)

(b)

(c)

(d)

Finally, we also evaluated our approach using a private dataset with over $50,000$ images. It contains only Brazilian vehicles with pictures taken from security cameras

[4] Acronym for Cars Dataset Hard.

[5] Available at <http://www.inf.ufrgs.br/~crjung/alpr-datasets/>.

and toll plazas spread along the country. Although this dataset does not present many oblique views as the CD-HARD dataset does, it depicts a wide variety of real-life capture situations, such as varying illumination conditions, LPs with worn-out characters, motion blur, etc. A few samples from this dataset were shown in Figure 5.10 (LP characters were artificially blurred). The majority of the images (92%) present dimensions of 640×480 and 1280×945, which are not large considering modern cameras.

5.5.2 Comparison with Competitive Approaches

Here we present the result produced by the proposed technique and competitive approaches using the datasets describe in Table 5.1. Our approach presents three networks in the pipeline, for which we empirically set the following acceptance thresholds: 0.5 for vehicle (YOLOv2) and LP (WPOD-NET) detection, and 0.4 for character detection and recognition (OCR-NET). Also, it is worth noticing that characters "I" and "1" are identical for most Brazilian LPs. Hence, they were considered as a single class in the evaluation of the OpenALPR BR and SSIG datasets. No other heuristic or post-processing was applied to the results produced by the OCR module, except when explicitly mentioned.

We evaluate the system in terms of the percentage of correctly recognized LPs. An LP is considered correct if all characters were correctly recognized, and no (wrong) additional characters were detected. It is important to note that the exact same networks were applied to all datasets: no specific training procedure was used to tune the networks for a given type of LP (e.g., European or Chinese). The only slight modification performed in the pipeline was for the AOLP Road Patrol dataset. In this dataset, the vehicles are very close to the camera (and the vehicle detector fails in several cases), so that we applied the LP detector (WPOD-NET) directly to the input images, bypassing the vehicle detection step.

To show the benefits of including fully synthetic data in the OCR-NET training procedure, we evaluated our system using two sets training data: (i) real augmented data plus artificially generated ones; and (ii) only real augmented data. These two versions are denoted by "Ours" and "Ours (no artf.)", respectively, in Table 5.2. As can be observed, the addition of fully synthetic data improved the accuracy in all tested datasets (with a gain $\approx 5\%$ for the AOLP RP dataset). Moreover, to highlight the improvements of rectifying the detection bounding box provided by WPOD-NET, we also present the recognition results using a regular non-rectified bounding box in "Ours (Unrect.)". As

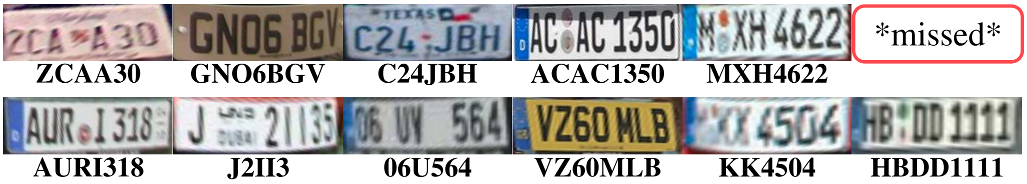
Table 5.2: Full ALPR results for all 5 datasets.

	OpenALPR EU	OpenALPR BR	SSIG Test	AOLP RP	Proposed CD-HARD	Average
Ours	93.52%	91.23%	88.56%	**98.36%**	**75.00%**	**89.33%**
Ours (no artf.)	92.59%	88.60%	84.58%	93.29%	73.08%	86.43%
Ours (Unrect.)	94.44%	90.35%	87.81%	84.61%	57.69%	82.98%
Ours (Chapter 4)	85.19%	90.35%	**89.15%**	69.56%	46.29%	76.10%
Commercial systems						
OpenALPR*	**96.30%**	85.96%	87.44%	69.72%	67.31%	81.35%
Sighthound*	83.33%	**94.73%**	81.46%	83.47%	45.19%	77.64%
Amazon Rekog.*	69.44%	83.33%	31.21%	68.25%	30.77%	56.60%
Literature						
Laroca et al. (2018)	-	-	85.45%	-	-	-
Li et al. (2016)	-	-	-	88.38%	-	-
Li et al. (2017)	-	-	-	83.63%	-	-
Hsu et al. (2013)	-	-	-	85.70%**	-	-

*Data collected on February, 2018.

**In Hsu et al. the authors gave an estimative, and not the real evaluation.

Figure 5.11: Detection results of WPOD-NET: unwarped LPs and final ALPR result produced by OCR-NET for the images shown in Figure 5.1.



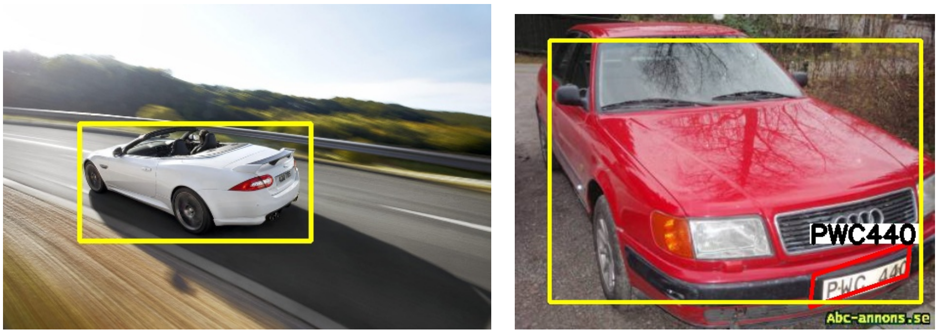
expected, the results do not vary much for the mostly frontal datasets (and results were even slightly better for ALPR-EU). However, using unrectified LPs lead to huge performance drops in more challenging datasets containing oblique views, such as AOLP-RP and CD-HARD. Finally, for the sake of comparison, we showed the results from Chapter 4 to all datasets. It is important to note that the approach from Chapter 4 was focused on real-time performance and trained specifically for Brazilian and European LPs.

Table 5.2 also shows the results from competitive (commercial and academic) systems. They indicate that the method proposed in this chapter achieved recognition rates comparable to best systems in databases representing more controlled scenarios, where the LPs are mostly frontal (OpenALPR EU and BR, and SSIG). More precisely, it was the second best method in both OpenALPR datasets (BR and EU), as well as in SSIG (being inferior only to the method proposed in the previous chapter). In the challenging scenarios (AOLP RP and the proposed CD-HARD dataset), however, our system outperformed all compared approaches by a significant margin (over 7% accuracy gain when compared to the second best result).

It is worth mentioning that the works of Li et al. (LI; SHEN, 2016; LI; WANG;

SHEN, 2017), Hsu et al. (HSU; CHEN; CHUNG, 2013) and Laroca et al. (LAROCA et al., 2018) are focused on a single region or dataset. By outperforming them, we demonstrate the strong generalization capacity of the presented method. It is also important to note that the full LP recognition rate for the most challenging datasets (AOLP-RP and CD-HARD), shown in Table 5.2, was higher than directly applying the OCR module to the annotated rectangular LP bounding boxes (79.21% for AOLP-RP and 53.85% for CD-HARD). This gain is due to the unwarping performed by WPOD-NET, which greatly helps the OCR task when the LP is strongly distorted. To illustrate this behavior, we show in Figure 5.11 the detected and unwarped LPs for the images in Figure 5.1, as well as the final recognition result produced by OCR-NET. Also, a few outputs of the whole system are shown in Figures 5.12 and 5.13. In Figure 5.12, an example of a miss detection and a bad alignment generated by WPOD-NET are shown. Despite the bad alignment in Figure 5.12b, the OCR-NET was still able to recover the correct characters.

Figure 5.12: WPOD-NET miss-detection and miss-alignment.

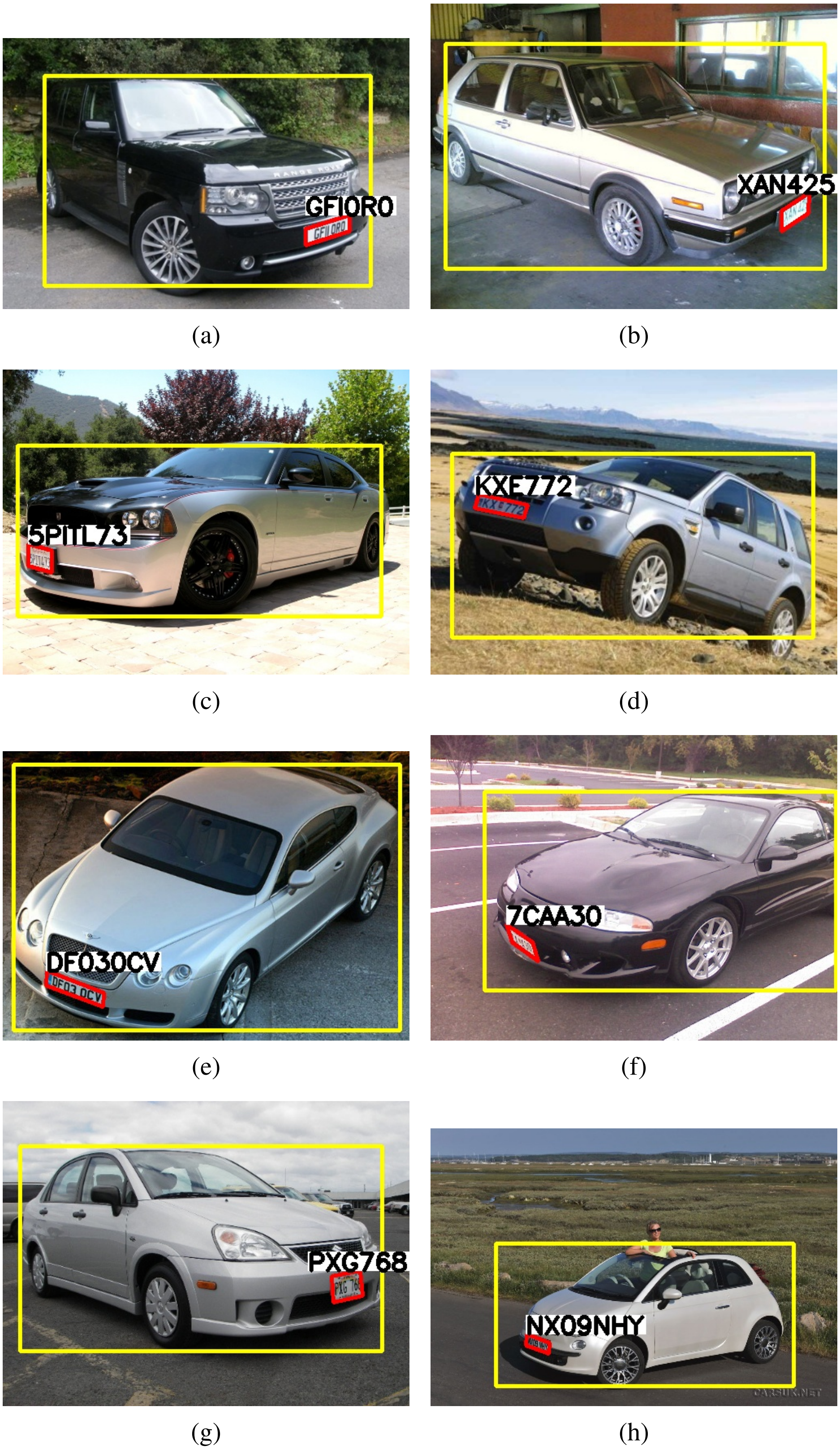


(a) Miss-detection.

(b) Miss-alignment.

The proposed CNN (WPOD-NET) was implemented in the TensorFlow framework, while the initial YOLOv2 vehicle detection and OCR-NET were performed using the DarkNet framework. A Python wrapper was used to integrate the two frameworks. The hardware used for our experiments was an Intel Xeon processor, with 12Gb of RAM and an NVIDIA Titan X GPU. With that configuration, we were able to run the full ALPR system with an average of 5 FPS (considering all datasets). This time is highly dependent on the number of vehicles detected in the input image. Hence, increasing the vehicle detection threshold reduces the number of detections, culminating in a higher FPS but lower recall rates.

Figure 5.13: ALPR results of the proposed method for some images of the CD-HARD dataset. The yellow rectangles represent the car detection from YOLOv2 network, the red polygons are the LP detection from WPOD-NET, and the strings above the LPs are output of the OCR-NET given a rectified LP.



(a) (b) (c) (d) (e) (f) (g) (h)

5.5.3 Accuracy vs. License Plate Size

In both ALPR methods proposed in this dissertation, the input to OCR-NET is a 240×80 image representing the LP. However, the actual size of a detected LP is highly dependent on the image. For a given detection provided by WPOD-NET, we first unwarp it based on the estimated affine parameters and then resize it to the desired resolution (240×80) using bilinear interpolation, which might generate distortions (particularly for small LPs). The goal of this section is to evaluate how the initial LP size (before interpolation) affects the quality of the OCR.

From each image of the CD-HARD dataset, we generated another set of images with sizes varying from 30% to 100% of their original size. By reducing the image resolution, this controlled down-sampling causes loss of information. The modified dataset was used to evaluate the system by measuring the OCR accuracy as a function of the original area (in pixels) of each detected license plate. Figure 5.14a shows a histogram of the recall rate for different LP sizes. We can note that the license plates whose areas fall between $0 - 2,000$ pixels were poorly recognized, license plates from $2,000 - 5,000$ have a small performance decrease and, from $5,000$ and so on, the performance stabilizes.

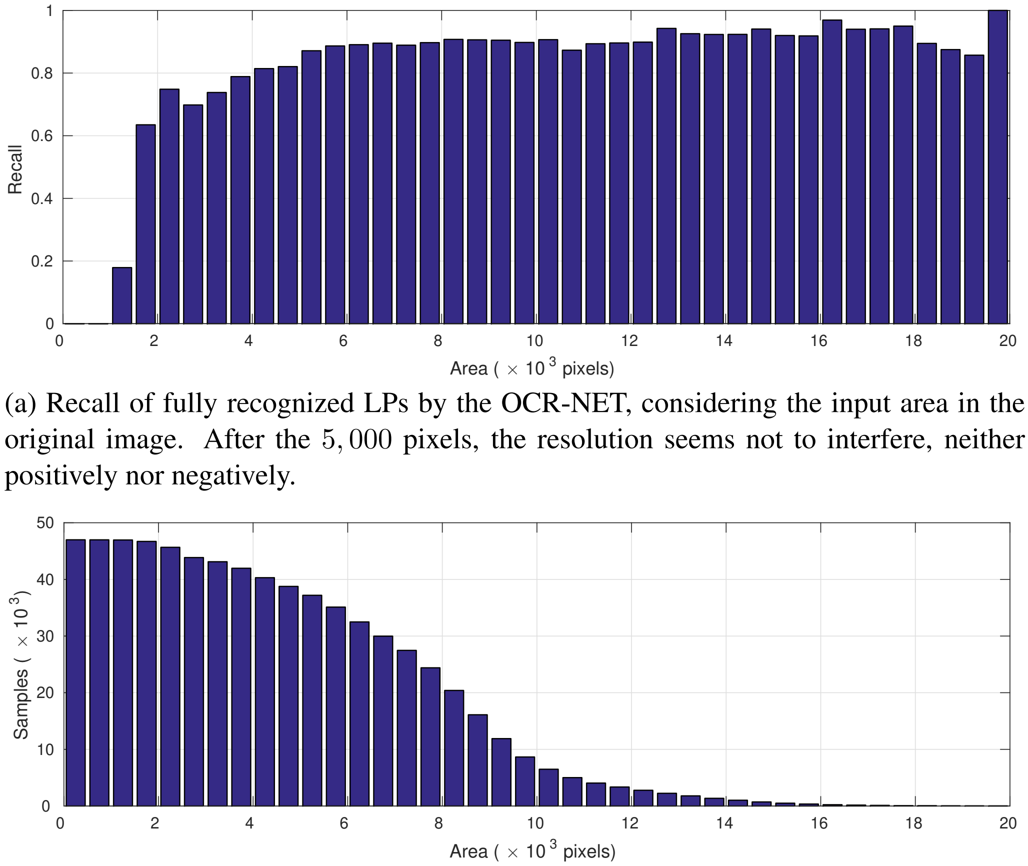
The number of samples per area bin used to calculate the recall rate are shown in Figure 5.14b. As we only performed size reduction, larger LP areas had progressively fewer samples. Therefore, the confidence for areas close to $20,000$ pixels is weak, meaning that the high recall obtained in such areas was computed with just a few samples. For instance, only 6 samples were used to calculate the last bin in Figure 5.14a.

5.5.4 Brazilian Dataset Evaluation

Now we evaluate our system using a private dataset of Brazilian license plates. This dataset presents around $50k$ images with the annotated license plate number (but no localization), and a few samples of this dataset were shown previously in Figure 5.10. When there is additional knowledge on the structure of the LP (as in this database), our OCR approach can be fine-tuned to yield improved accuracy. For Brazilian LPs, the following rules can be applied:

1. Fixed number of characters: since Brazilian LPs present exactly seven characters, we lowered the OCR detection threshold (to 0.1) and retrieve the seven most prob-

Figure 5.14



(a) Recall of fully recognized LPs by the OCR-NET, considering the input area in the original image. After the $5,000$ pixels, the resolution seems not to interfere, neither positively nor negatively.

(b) Number of samples per area bin that were used to calculate the recall rate in (a).

able characters, after non-maxima suppression;

2. Digit-to-letter and letter-to-digit replacement: as shown previously in Figure 4.2, the license plates in Brazil have 3 letters followed by 4 digits. Considering that our OCR returns a character that can be a digit or a letter, by taking into account the position of the character we can infer if it should be a letter or a digit. Based on a confusion matrix obtained from a test set (shown in Figure 5.16), we devised two heuristics to exchange digits for letters and letters for digits when it is needed due to character position;

3. Due to the nature of the dataset, each image is expected to present at least one LP. However, in several images the vehicle is only partially visible, so that bounding box produced by the vehicle detector might be highly distorted (an example is shown in Figure 5.15). In such cases, applying WPOD-NET to the vehicle detection leads to a missed LP. In this experiment, a vehicle is detected but no LP is detected within the vehicle crop, WPOD-NET is applied to the full image.

Figure 5.15: Bad LP detection example: applying WPOD-NET to the the vehicle bounding box (green rectangle) returns no LP. The LP is correctly detected when apllying WPOD-NET to the full image.

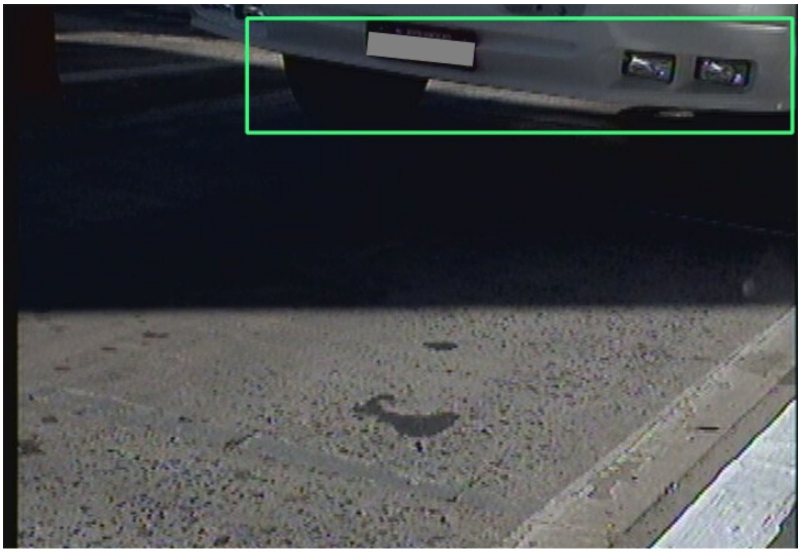
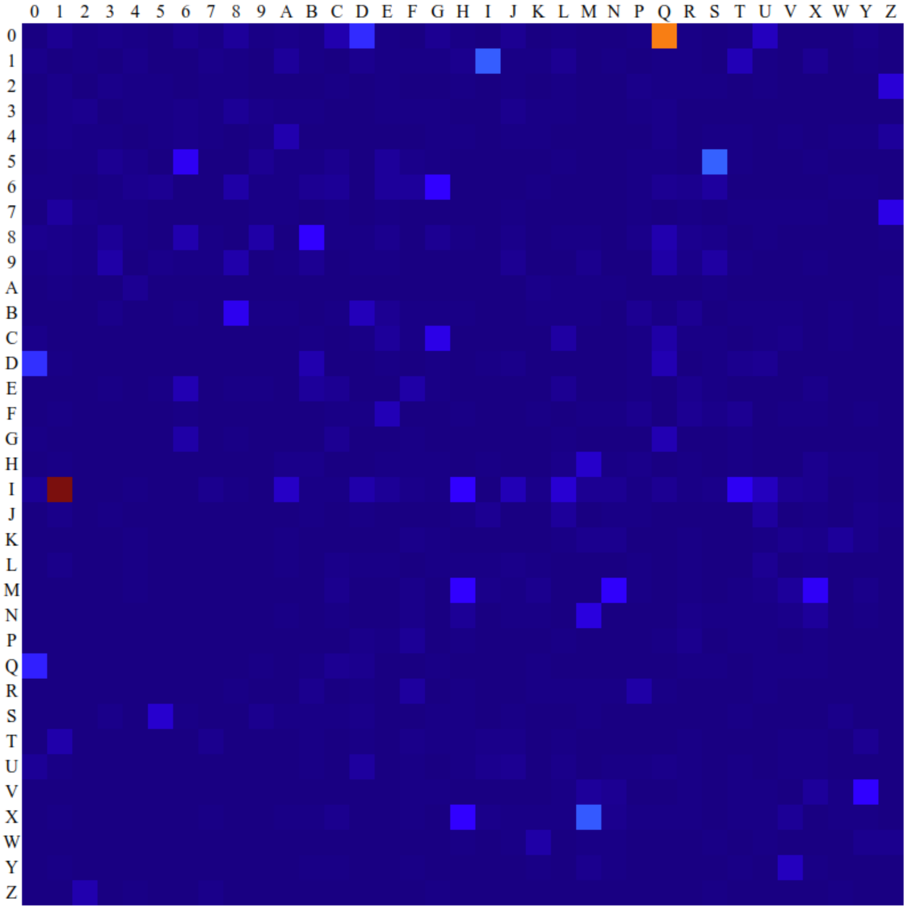


Figure 5.16: Visualization of the normalized confusion matrix per character.



The results using our approach combined with the heuristics described above are shown in Table 5.3. Considering the size of the dataset ($\approx 50k$ images) and the fact that we did not fine-tune any model to this scenario, the results demonstrates a strong generalization capacity of our approach. When analyzing the overall performance, almost 88% of all license plates were correctly recovered (recall). Among all detections, in 89.53% of them all seven characters were correctly recognized (precision). Furthermore, the accuracy of OCR-NET in a per-character evaluation within all detected LPs (i.e., the percentage of correctly recognized characters) was 97.39%.

We also carried a more careful analysis considering a tolerance of n of incorrect characters to consider a given recognition result correct, noting that $n = 0$ relates to validating a result only when all characters were correctly recognized. These results are shown on the second part of Table 5.3, and it can be noticed that the accuracy increases to 96.74% by allowing a tolerance of a single character ($n = 1$), which is close to the per-character accuracy.

Table 5.3: Results for the Brazilian scenario. The precision by character tolerance means the final precision if we ignore $0 - 7$ mistaken characters. On the one hand, if we ignore zero characters, it corresponds to the system accuracy, and on the other hand, by ignoring seven characters, we have 100% accuracy as the license plates have no more than this.

Accuracy:	89.53%	Recall:	87.87%	Characters	97.39%
Accuracy by allowing errors in 0 to 7 characters:					
0:	89.53%	3:	98.84%	6:	99.80%
1:	96.74%	4:	99.16%	7:	100.00%
2:	98.25%	5:	99.46%		

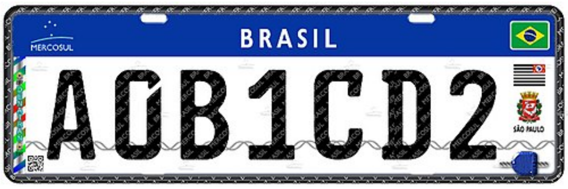
5.5.5 Mercosur License Plates

In this section, we present some considerations related to the new license plate model adopted by the countries in Mercosur. This model also consists of 7 characters, but now there are four letters and three digits in the format LLLDLDD, where L stands for “letter” and D stands for “digit”[6]. It uses a new font type, where the digit 1 and letter I are now different from each other. The predominant colors for the most common model are white and blue with black striped characters (an example is shown in Figure 5.17). This new LP model is still being implemented in Brazil, and to the moment of this publication there is no evaluation dataset available in the literature.

To evaluate how well the proposed method would generalize to the new Mercosur LP model, we ran our approach on about a dozen of images captured from the internet or

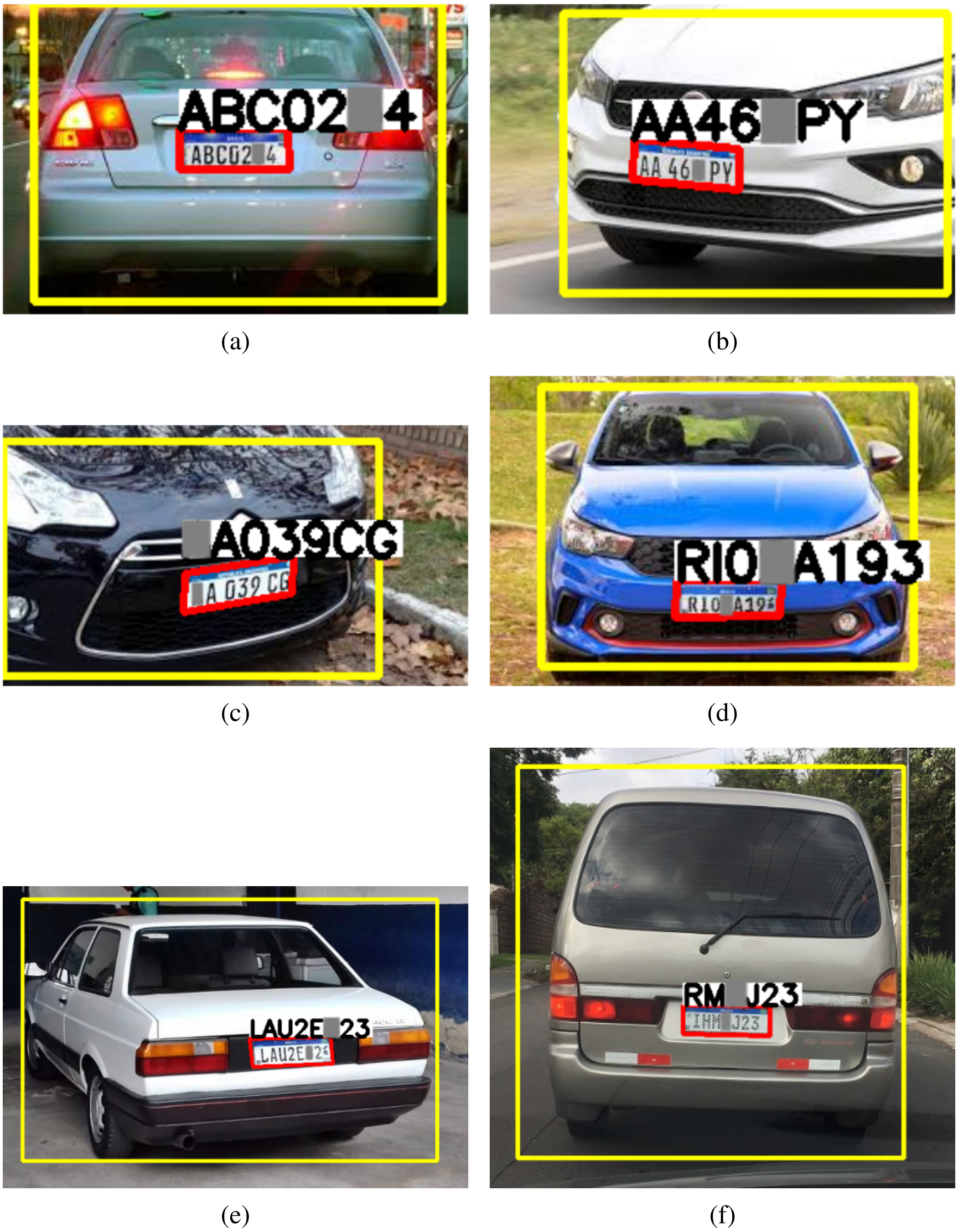
[6] <http://www.denatran.gov.br/images/Resolucoes/Resolucao7412018.pdf>

Figure 5.17: Brazilian license plate model implemented in 2019.



taken on the street. A few of them are shown in Figure 5.18 along with the system output, where a single character on each image was deliberately blurred to preserve anonymity. In general, our approach recognized most of the tested license plates without any adaptation on WPOD-NET or OCR-NET. However, we noticed that the font used in some Mercosur LPs is “striped”, in which a Mercosur logo is embedded in the font, as shown in Figure 5.18(f), being particularly hard to read under some lighting conditions. In those cases, OCR-NET did not produce good results. Moreover, it also false-detected a few numbers “3” at the end of the license plate, as shown in Figure 5.18(e). We associate this to the adornments present at the right end of some license plates, which caused the confusion.

Figure 5.18: Recognition results for a few images of the new Mercosur license plate model.



(a) (b) (c) (d) (e) (f)

6 RACING BIB NUMBER RECOGNITION

In every marathon or urban racing, there is a plate attached to each athlete, called bib, that contains their identification number. These events can involve thousands of people, like the Boston Marathon, in the USA, or the Saint Silvester Road Race, in Brazil. In 2018, they united 30,000 athletes each [1,2]. During these events, a huge amount of data is generated by professional photographers, video makers, and spectators. The computational process of indexing such data by bib number is the subject of study in this chapter.

Besides drawing much less attention than ALPR, Racing Bib Number Recognition (RBN-R) is a challenging problem related to contextualized text recognition. In Figure 6.1 we show a few samples of our training dataset that present problems that do not exist (or are uncommon) in ALPR, such as:

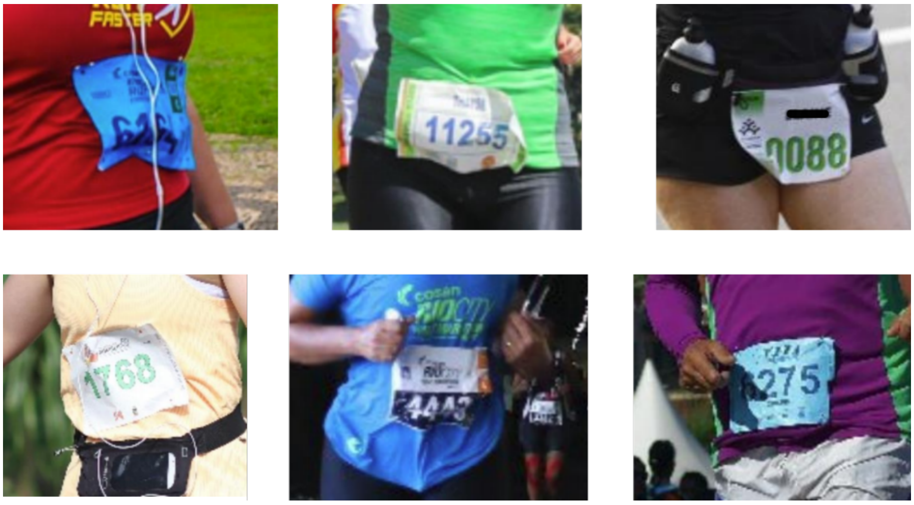
- Deformation: due to its soft and flaccid material, the bib fits the athlete body and becomes subject to deformations during movement;
- Sweat: perspiration causes changes in color, sometimes adding transparency to colored regions;
- Layout: every event presents its own layout, hardening the task of separating the identification number from the background and other adornments;
- Location: bibs are usually pinned in the torso region, most likely in the chest or belly, and a few athletes pin on their thigh;
- Athlete vs. Spectators: not every person in a picture has a bib. In fact, a common scenario presents athletes surrounded by spectators during some parts of the event.

State-of-the-art techniques assume that the bib is attached to the upper body part. They typically start by detecting the torso and then extract the bib number, which relates to the notion of contextual text already described before. However, we argue that using such a broad context (pedestrian/torso) for RBN-R might not be the best idea, since there are typically several spectators (without bibs) in the input image. This approach would increase the search space and could also lead to more false-positives, since spectators often wear T-shirts that present letters and numbers in their stamps.

[1] Saint Silvester: <https://bit.ly/2kZeBQJ>.

[2] Boston Marathon: <https://www.baa.org/races/boston-marathon/results/participation>.

Figure 6.1: Examples of difficult to recognize bib images.



In the ALPR approaches presented in the previous chapters, we used three "contextual stages": vehicle (or frontal/rear views) detection, license plate detection, and OCR. A direct analogy to RBN-N would imply in pedestrian (or torso) detection, bib detection, and OCR. However, the approach we present in this chapter considers only two stages: bib detection and OCR, based on the previous discussion. Taking into account that a bib occupies a large part of the torso, we hypothesized that, when detecting the bibs, the network intrinsically learns features related to the pedestrian/torso. At the end of Section 6.1.1.5, we show examples that corroborate with our hypothesis.

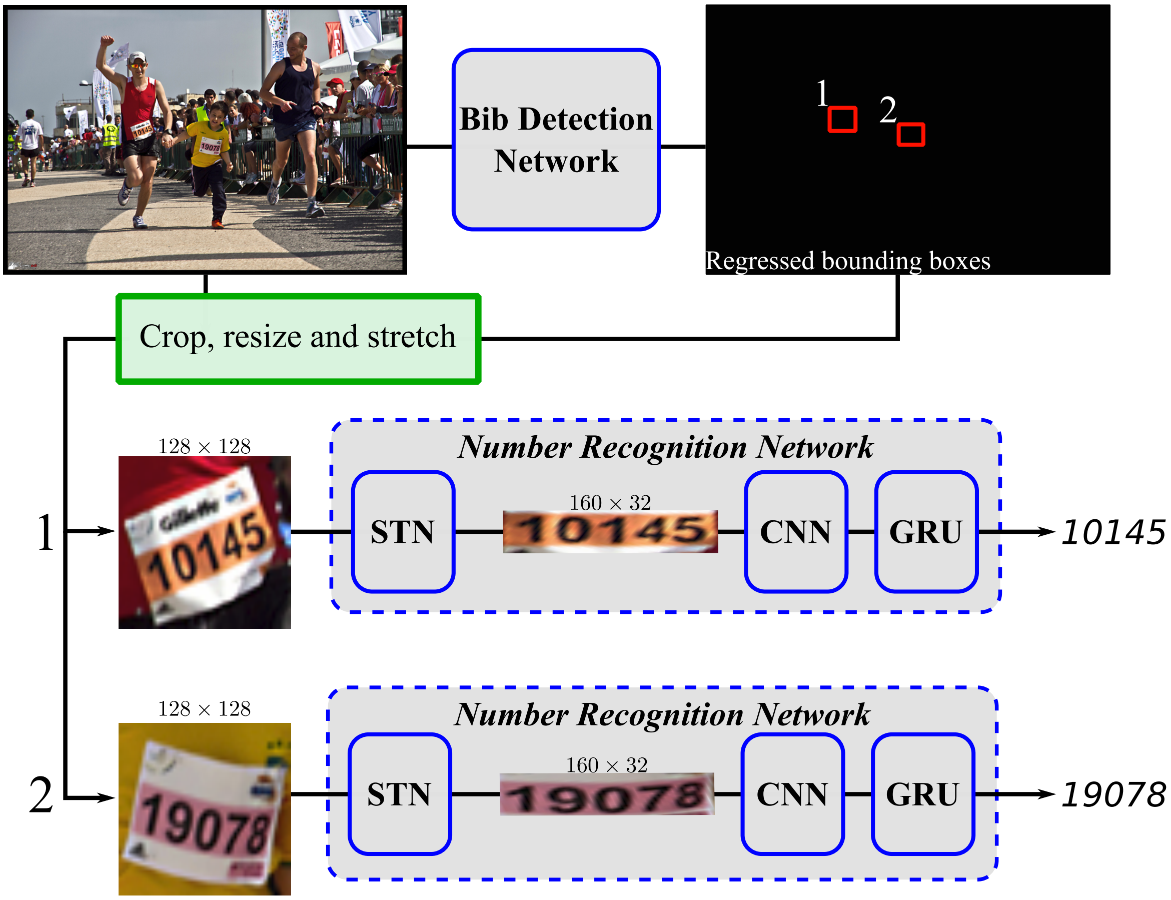
In this chapter, we present in details our proposed solution to the problem of RBN-R, and describe the datasets used for training and testing it. Results and comparisons with other approaches are presented and discussed at the end.

6.1 The Proposed Approach

Our approach consists of a two-stage pipeline: bib detection and number recognition, both performed by deep CNNs. The first network resembles YOLO in the way the detection is performed. However, the network backbone and input granularity were changed. The second network uses STN and GRU (Gated Recurrent Units) layers to infer the number without needing complex annotations during training. The system overview is presented in Figure 6.2.

In the previous chapter, the concept of STN was explored in the detection stage by taking advantage of the fact that license plates present four distinct corners that can be used to learn the affine translation. For bibs, however, recovering the corners is often impossible due to the many factors mentioned earlier. Thus, we decided to explore STN in the recognition stage and let the network learn the affine transformation from the data, without using corner annotations.

Figure 6.2: RBN-R proposed pipeline.



6.1.1 Bib Detection

Given an input image, we opted to detect the bib directly, instead of looking for any human body part like the face (BEN-AMI; BASHA; AVIDAN, 2012) or torso (SHIVAKUMARA et al., 2017). This choice leads us to the problem of detecting small objects, commented at the beginning of this dissertation. There is a considerable variation in bib size when the athlete is close or far from the camera. Actually, even when it is close, the bib does not occupy the majority of the scene, as often happens to vehicles. This is because the focus or the main context of a picture is not the bib, but the athlete.

We know that, for networks of the depth, there is an inverse relationship between the number of parameters from a network, and how fast the feed-forward is. Thus, more parameters mean a slow feed-forward. Considering our goal of fast detection, and the need for a higher input/output granularity to keep track of small objects, we targeted a backbone model with as few parameters as possible.

By reminding this work so far, the reader might be inclined to think that we, once again, opted to use FAST-YOLO. FAST-YOLO works very well to detect larger objects, like vehicles, but bibs are generally much smaller in pictures. For instance, a 48×48 bib in a 1200×800 picture can still be readable. Thus, we opted to use an even smaller and faster network, the MobileNet (HOWARD et al., 2017), which was designed, as the name suggests, to run in smartphones and embedded devices.

MobileNet does not perform as well as other state-of-the-art networks on the ImageNet dataset. For instance, it achieved 72% (mAP) against 80.6% from ResNet. However, MobileNet presents a very good ratio between the number of parameters, depth and performance. Thus, we once more hypothesized that by lowering the number of classes to detect — from $1,000$ to 1 (a single bib) — the network could accommodate nicely.

We created the network architecture shown in Table 6.1, where the first 23 layers were directly transferred from MobileNet. They consist of sequences of convolution and depthwise convolution layers. Details about the Depthwise Convolution layer can be found in MobileNet paper. Shortly, a depthwise convolution uses a single 1-channel filter per input channel. This operation is much faster than traditional convolution and can be fastly performed even by a CPU. The last two layers were created and trained from scratch, and are responsible for regressing the object/non-object probabilities (layer 24) and bounding-boxes (layer 25).

6.1.1.1 Our Regression scheme

The regression scheme used for bib detection is inspired in SSD and YOLOv2 models, and uses the concept of anchor boxes. First we describe a generic multi-class solution that explores C classes and A anchor boxes, where each anchor presents one parameter for objectness, C parameters for class probabilities, and 4 parameters for bounding box regression. We then focus on the single-class problem ($C = 1$) related to the bib only, and experimentally defined $A = 3$ (3 anchor boxes) based on our training dataset. For this reason, layer 24 outputs a feature map of 6 channels (3 anchors $\times$ 2 probabilities) and layer 25 outputs a feature map of 12 channels (3 anchors $\times$ 4 bounding box parameters).

It is important to note that changing the number of classes and anchors does not impact the network architecture, only the final two layers. The use of more classes could

Table 6.1: Bib Detection (BD) Network architecture.

	#	Layer	Size	Stride	Filters	Input	Output Function
MobileNet Backbone	1	Convolution	3×3	2	32	input image	ReLU
	2	Depthwise	3×3	1	32	previous layer	ReLU
	3	Convolution	1×1	1	64	previous layer	ReLU
	4	Depthwise	3×3	2	64	previous layer	ReLU
	5	Convolution	1×1	1	128	previous layer	ReLU
	6	Depthwise	3×3	1	128	previous layer	ReLU
	7	Convolution	1×1	1	128	previous layer	ReLU
	8	Depthwise	3×3	2	128	previous layer	ReLU
	9	Convolution	1×1	1	256	previous layer	ReLU
	10	Depthwise	3×3	1	256	previous layer	ReLU
	11	Convolution	1×1	1	256	previous layer	ReLU
	12	Depthwise	3×3	2	256	previous layer	ReLU
	13	Convolution	1×1	1	512	previous layer	ReLU
	14	Depthwise	3×3	1	512	previous layer	ReLU
	15	Convolution	1×1	1	512	previous layer	ReLU
	16	Depthwise	3×3	1	512	previous layer	ReLU
	17	Convolution	1×1	1	512	previous layer	ReLU
	18	Depthwise	3×3	1	512	previous layer	ReLU
	19	Convolution	1×1	1	512	previous layer	ReLU
	20	Depthwise	3×3	1	512	previous layer	ReLU
	21	Convolution	1×1	1	512	previous layer	ReLU
	22	Depthwise	3×3	1	512	previous layer	ReLU
	23	Convolution	1×1	1	512	previous layer	ReLU
	24	Convolution	3×3	1	6	layer 23	Sigmoid
	25	Convolution	3×3	1	12	layer 23	Linear

be explored for the detection of complementary cues related to the athlete's identity, such as the face and body, and then be used to devise a multimodal identification scheme. However, this dissertation focuses only on bib detection and recognition.

6.1.1.2 Objectness and class probabilities

In layer 24, the outputs are fed to a sigmoid function, returning values between 0 and 1 and used as probabilities for objectness and classes. For each anchor, there is a single objectness probability that tells if an arbitrary object exists or not in that cell, plus an independent probability parameter for each class (yielding $C + 1$ parameters for a C-class problem). The choice for the sigmoid function was mainly motivated by the findings of Redmon and Farhadi (2018), who argued that using softmax for the anchors instead of sigmoid does not necessarily lead to good accuracy. Moreover, employing softmax with anchors poses some implementation difficulties as it needs to be applied per anchor, and not point-wise in the whole output.

Hereafter, we will call the output of layer 24 the volume P with dimensions $M \times N \times A(C+1)$, where A is the number of anchors. For each spatial cell P_{mn} there is a vector of length $A(C+1)$, called $\vec{p}_{mn}$[3], containing all probabilities for that region. We can say that $\vec{p}$ is a concatenation of A vectors $\vec{p_a}$, where $a \in \{1, ..., A\}$. Each vector $\vec{p_a}$ presents $C+1$ parameters corresponding to objectness plus class probabilities. Hence, we can write $\vec{p_a} = [p_{a_{obj}}, p_{a_1}, ..., p_{a_C}]^T$, where $p_{a_{obj}}$ stands for the objectness probability, and p_{a_1} to p_{a_C} to the class probabilities. Finally, we have $\vec{p} = \text{concat}(\vec{p_1}, ..., \vec{p_A})$, where concat stands for the concatenation operator.

6.1.1.3 Bounding-box regression

The output of layer 25 is a volume B of size $M \times N \times 4A$. An arbitrary cell B_{mn} of this volume is represented by a vector of length $4A$, containing A encoded bounding-boxes, each one presenting 4 parameters. We call this vector $\vec{b} = \text{concat}(\vec{b_1}, ..., \vec{b_A})$, where $\vec{b_a} = [b_{y_a}, b_{x_a}, b_{h_a}, b_{w_a}]^T$, for $a \in \{1, ..., A\}$. Thus, for each anchor[4], the bounding box in image coordinates is encoded by

$$\begin{aligned} y_a &= (m - b_{y_a} H_a), \\ x_a &= (n - b_{x_a} W_a), \\ h_a &= e^{b_{h_a}} H_a, \\ w_a &= e^{b_{w_a}} W_a, \end{aligned} \tag{6.1}$$

where the point (x_a, y_a) is the final anchor box center point with dimension $w_a \times h_a$ relative to the spatial resolution $M \times N$ of the feature map.

We used the same method of YOLOv2 (REDMON; FARHADI, 2017) to obtain the constants H_a and W_a related to the anchor boxes. In a nutshell, it consists of k-means clustering with IoU as distance measure applied over all bounding boxes in the training data. As mentioned before, we set $A = 3$ based on experiments with our datasets, and the obtained values for the three anchors were $(2.0, 1.7)$, $(3.1, 2.6)$ and $(1.3, 1.1)$.

[3] We omit the indices mn for simplicity.

[4] We omit the indices mn of a_{mn} a in cell (m, n) for notation simplicity.

6.1.1.4 Loss function

A loss function for the problem of classification and detection must handle both tasks at once. For clarity purposes, we will break it into probabilities and bounding box regression, and then merge the pieces together.

Similarly to the ALPR approach described in the previous chapter, we also used the binary cross-entropy function (Equation 5.5) to evaluate each output parameter computed by layer 24. Taking cell (m, n) as a reference, this choice leads to:

$$loss_{prob}(m, n) = \sum_{a=1}^{A} \sum_{c=1}^{C} \text{cross\_entropy}(prob^{*}(a, c), prob(a, c)), \tag{6.2}$$

where $prob^{*}()$ and $prob()$ denote the expected and the inferred probabilities for anchor a and class c, respectively.

In the bounding box regression problem, we used the L^1 norm to estimate the error in regressed parameters, which was successfully used in SSD (LIU et al., 2016). It is given by

$$loss_{reg}(m, n) = \sum_{a=1}^{A} \left(|y_a^{*} - y_a| + |x_a^{*} - x_a| + |h_a^{*} - h_a| + |w_a^{*} - w_a|\right), \tag{6.3}$$

where (y_a, x_a, h_a, w_a) are the inferred bounding box parameters, and $(y_a^{*}, x_a^{*}, h_a^{*}, w_a^{*})$ the corresponding ground-truth values.

The final loss is a weighted sum of the probability and regression losses per cell, averaged by the number of cells in the feature map, given by

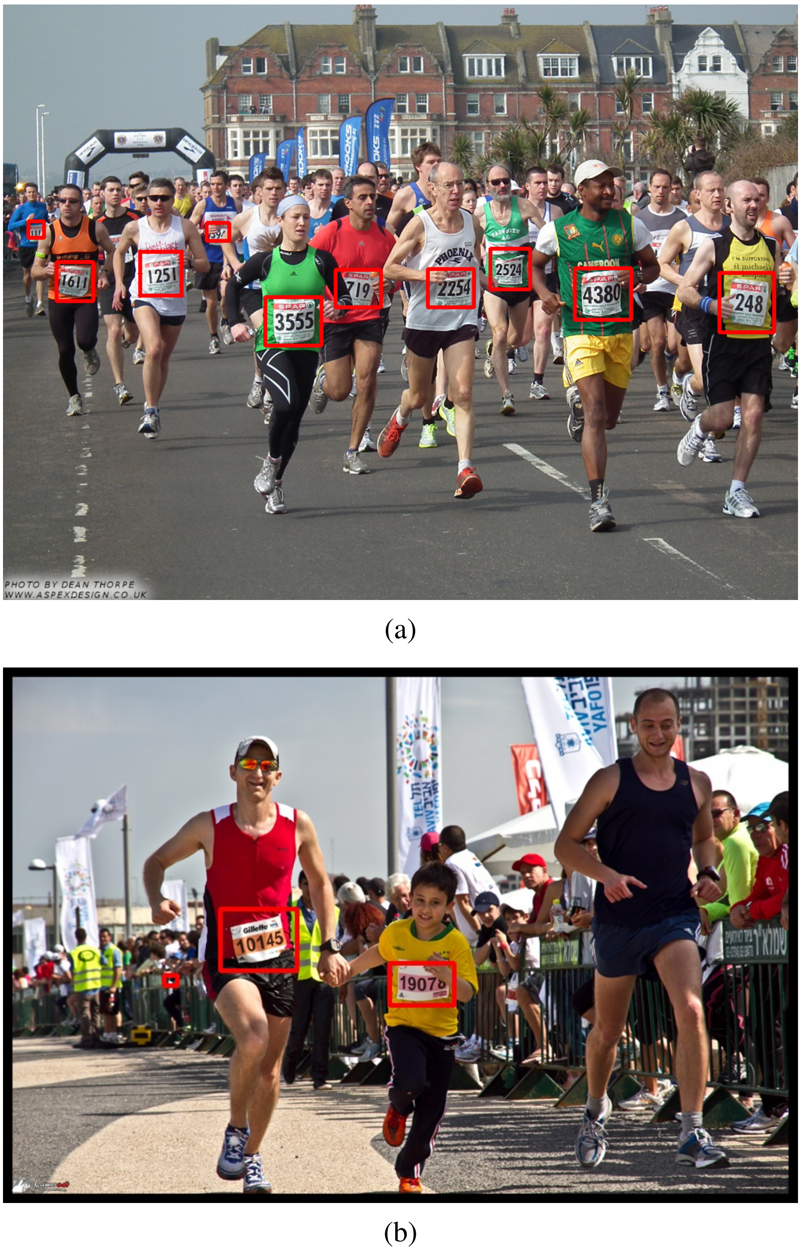
$$loss = \frac{1}{MN} \left(\sum_{m}^{M} \sum_{n}^{N} loss_{prob}(m, n) + \gamma loss_{reg}(m, n) \right), \tag{6.4}$$

where γ controls the regression weight. Since the probability parameters seems to converge faster than the regression parameters, we set $\gamma = 2$ to enhance the regression loss.

6.1.1.5 Visual Analysis

In Figure 6.3, we show two examples of bib detection using the proposed approach in images of the test set. Figure 6.3(a) shows mostly athletes in the scene, and the proposed method was able to successfully detect most (or all) bibs. Figure 6.3(b) shows a few athletes and several spectators (without bibs), along with textual information (on the vertical banners). Despite these challenges, all real bibs were correctly retrieved along with a single and small false positive in the spectators. That false detection could easily be removed by a simple post-processing step based on the region size.

Figure 6.3: Examples of bib detection on images taken from test set. Automatically detected bibs are shown in red.



(a)

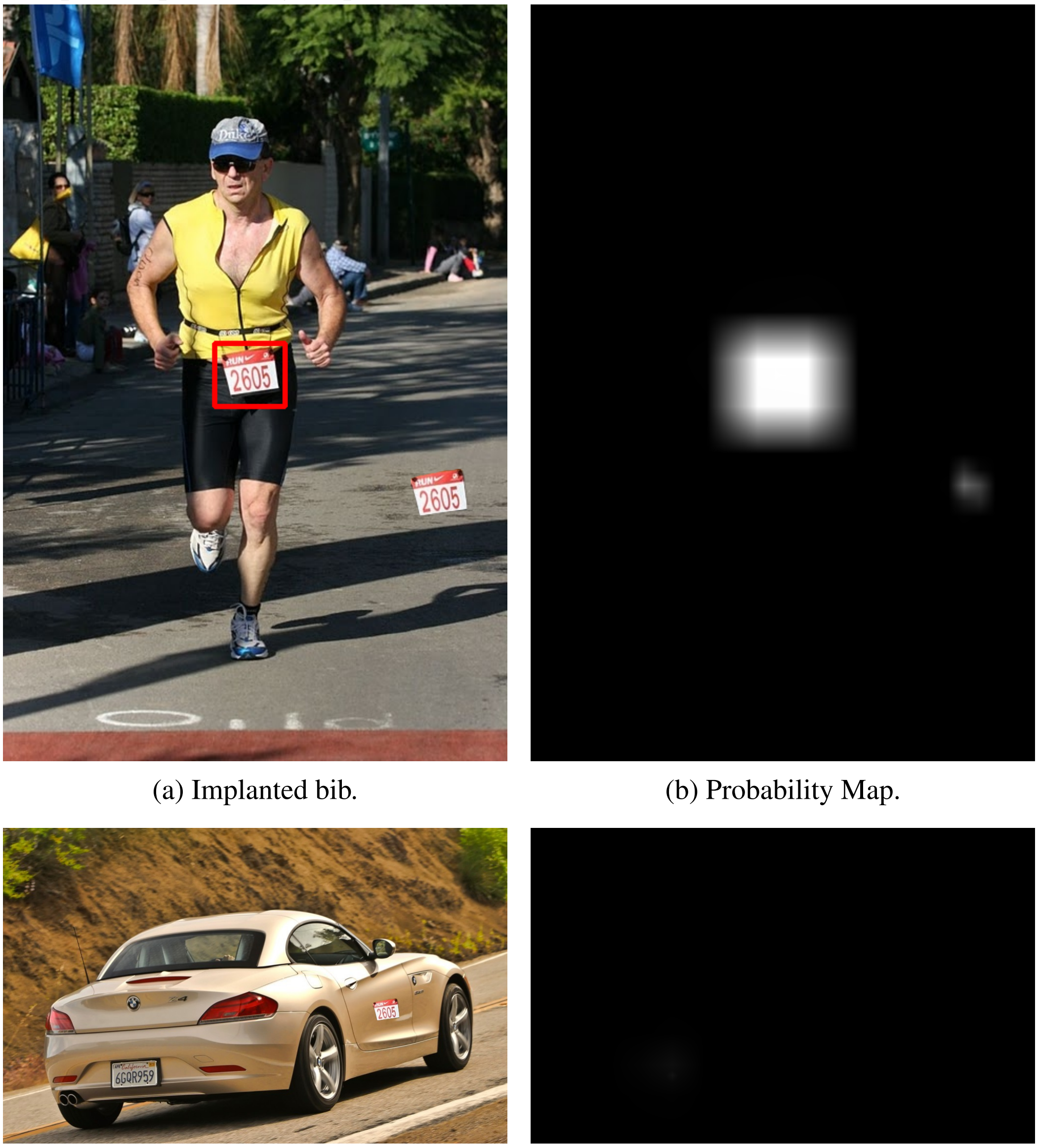
(b)

To demonstrate that our network can often understand a broader context beyond the bib and differentiate it from other similar objects, like a license plate, we created a simple test that consists of using an image editor software [5] to synthetically implant a bib (that was previously detected by our network) to a different place of the same image, and see if the bib can still be detected without the context. Also, we implanted this same

[5] GNU Image Manipulation Program (Gimp): <https://www.gimp.org/>.

bib over the passenger's door of a car in another picture, and check if our network would detect this contextless bib or even the license plate. The test images and output of our BD network are shown in Figure 6.4, and we can observe that the implanted bibs were not detected indeed. Moreover, Figures 6.4b and 6.4d shows the respective probability maps for bibs. Figure 6.4b shows a strong activation region around the actual bib and some activation close to the implanted bib (but not enough to break the 50% threshold). In Figure 6.4d, neither the vehicle LP not the implanted bib showed response in the activation maps, corroborating the hypothesis that a broader context related to the athlete's body was learned by the network.

Figure 6.4: Detected images where a bib was implanted on non usual places (a and c) and their respective probability maps (b and d).



(a) Implanted bib.

(b) Probability Map.

(c) Implanted bib.

(d) Probability map.

6.1.2 Number recognition

The works of Ben-ami, Basha and Avidan (2012) and Shivakumara et al. (2017) detect first the torso[6], and then perform the character segmentation and recognition using binarization techniques. The disadvantage of this approach is that the torso usually contains other textual information besides the bib, like T-shirt drawings, sponsorship logos, event name, etc. Furthermore, considering the spectators, one can see that not all torsos will be from athletes. This may lead to unnecessary computation and a higher number of false positives. Besides, Shivakumara et al. (2017) do not perform any kind of rotation correction, and Ben-ami, Basha and Avidan (2012) correct the rotation in a per-character basis. If we consider that the numbers are aligned and inside the same bib, a single rotation over all characters might be more effective as characters with a different rotation angle are probably not part of the bib. Although this may not be true when the bib deformation is excessive, such assumption holds for the majority of the situations in the datasets used for evaluation.

We partially solved these problems using the BD Network, which detects the bib instead of the whole torso, and hence eliminates a substantial part of misleading information not related to the characters in the bib. For the recognition step, we developed a novel network that crops only the digits of the bib, corrects the bib rotation, and recognizes its characters, everything at the same time in a single pass. As an additional and very important advantage, this network only needs the full number inside the bib as an annotation instead of complex annotation like per-character 2 or 4-point bounding boxes and labels.

We used a Spatial Transformer Network (STN) to crop out unwanted parts of the input image and give focus to the bib number region only, and at the same time correcting its rotation and some deformation in the process. This step represents the first part of the network. In the second part, the unwarped and focused patch is fed to the rest of the CNN, which presents a recurrent layer at the end. It is important to note that although there is no expected order in the digits of a bib number, they all share common features such as background and foreground colors, size, and font-type. Therefore, we hypothesized that this information could be passed from one character to another, helping the overall recognition process.

Our choice to explore sequential information in our network was a GRU layer.

[6] Actually Ben-ami, Basha and Avidan (2012) detects the face first, but they infer the torso region based on the face region.

Although the popular Long Short-Term Memory (LSTM) layer could be used, our choice for GRU is due to the following factors:

- GRU layers have fewer parameters per unit than LSTM, which makes the training less complex and facilitates the convergence and generalization when there is not much data available, making a perfect link with the goals of this work. Moreover, Chung et al. (2014) demonstrates that GRU, besides less complex, can outperform LSTM in various tasks.

- In theory, LSTM should have an advantage when the task requires keeping track of long sequences, or "old memories". This characteristic is useful in tasks such as text and speech processing, but not in our scenario where there are only five steps to keep track in the worst case (see Section 6.1.2.1).

The network architecture and the customized loss function developed for this purpose are described next.

6.1.2.1 Network Architecture

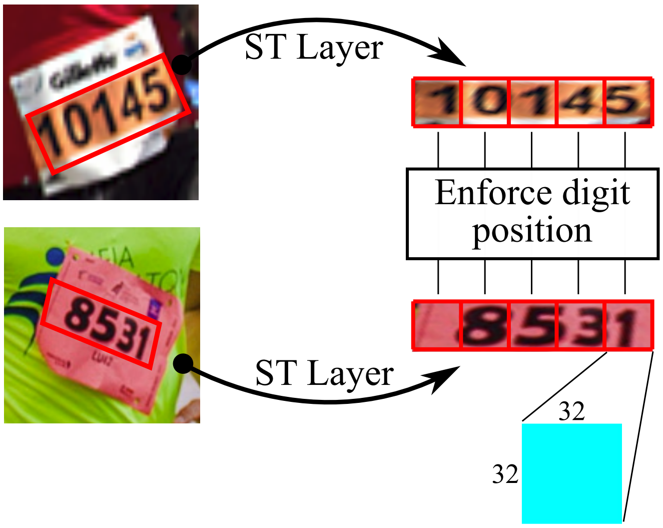
The CNN architecture for number recognition (NR-NET) is more complex than the one for bib detection. There are several tasks that must be accomplished at once with only weak annotation.

Initially, the first task is to isolate the number from the rest of the bib. This can be performed by an affine transformation if set correctly. Therefore, our network starts with a backbone (MobileNet) followed by a Spatial Transformer (ST) layer. To recall, an ST layer converts an input volume of size $H \times W \times Ch$ into a new volume of size $H' \times W' \times Ch$ by using an affine transformation that was also provided as input. Seeing the ST layer as a function, we can say that $F_{ST} : H \times W \times Ch \longrightarrow H' \times W' \times Ch$.

Ideally, given our scenario, the F_{ST} should find a transformation that outputs a horizontally aligned crop of the bib number. In our approach, the parameters of this transformation are trained implicitly by maximizing the success rate of the final OCR result. As a consequence, if the ST layer fails to frame the number, the recognition will also fail. This poses a constraint that allows the network to learn localization without complex annotation. In this sense, our approach is similar to the one presented in Jaderberg et al. (2015) for house number recognition, but with several differences in the architecture and objective function.

Our ST layer is characterized by a function $F_{ST} : 128 \times 128 \times 3 \longrightarrow 32 \times 160 \times 3$, so that an RGB input image of size 128×128 is converted into a 32×160 patch. We set the output size assuming that the ST layer is able to find a single transformation that places each digit inside a fixed square of size 32×32. By setting to 5 the maximum number of digits (which can accommodate races with up to 100,000 participants), and assuming they will be horizontally spread in the ST output patch, we obtain the 32×160 output resolution with all digits ideally cropped and horizontally aligned. This process is depicted in Figure 6.5. On the one hand, we are not adding any information by enlarging the number region (from $128 \longrightarrow 160$ pixels in the best case). On the other hand, we are framing each digit in a 32×32 square. This digit size was defined based on previous successful research on character recognition using Neural Networks (LECUN et al., 1998; JADERBERG et al., 2014).

Figure 6.5: Desired behavior of the learned affine transformation. Note that the transformation should ideally frame each digit inside a 32×32 division.



The final part of our network is fed with a 32×160 image patch from the initial part and, after a sequence of convolutional layers and a single GRU layer, outputs a 5×11 matrix. Each row of this matrix represents a digit (maximum of 5 digits), and the columns are the class probabilities for the ten possible digits $(0 - 9)$, plus one when there is no digit in that position. This “no digit” class is crucial to deal with smaller numbers, noting that not significant zeroes are typically not shown in racing bibs.

The full network architecture is described in Table 6.2. The long sequences of layers from MobileNet were omitted, but the reader can easily find this information by exploring the MobileNet model (HOWARD et al., 2017).

Table 6.2: Number Recognition (NR) Network architecture.

Layer	Input Size	Output Size	Stride	Filter Size	Input	Output Function
Input Layer	-	$128 \times 128 \times 3$	-	-	-	-
Avg. Pooling	$128 \times 128 \times 3$	$64 \times 64 \times 3$	2	2	IL	-
		...				
		MobileNet Backbone				
		...				
Flatten	$2 \times 2 \times 512$	2048	-	-	PL	-
Fully Connected	2048	512	-	-	PL	ReLU
Fully Connected	512	512	-	-	PL	ReLU
Fully Connected	512	256	-	-	PL	ReLU
Fully Connected	256	128	-	-	PL	ReLU
Fully Con.	128	6	-	-	PL	ReLU
Spatial Trans.	$(6, 128 \times 128 \times 3)$	$32 \times 160 \times 3$	-	-	(PL,IL)	-
		...				
		Another MobileNet Backbone				
		...				
Convolution	$2 \times 10 \times 256$	$1 \times 5 \times 256$	2	2	PL	ReLU
Convolution	$1 \times 5 \times 256$	$1 \times 5 \times 256$	1	1	PL	ReLU
Convolution	$1 \times 5 \times 256$	$1 \times 5 \times 256$	1	1	PL	ReLU
GRU	$1 \times 5 \times 256$	$1 \times 5 \times 256$	-	-	PL	-
Convolution	$1 \times 5 \times 256$	$1 \times 5 \times 11$	1	1	PL	Softmax
Reshape	$1 \times 5 \times 11$	5×11	-	-	PL	-

(IL: Input Layer, PL: Previous layer)

6.1.2.2 Loss Function and Training

Although we expect the ST layer to geometrically rectify the bib to a frontal view, this is not its main goal. We want to find an adequate affine transformation such that the rectified image leads to correct recognition of numbers in the bib. Hence, the geometric transformation should be learned by using only classification loss function. To formulate our loss, let us denote by $M_{5\times11}$ the output matrix from NR-NET, where M_{ij} is the probability of class j for digit i. Similarly, let us denote by $M^*_{5\times11}$ the ground truth annotation, where $M^*_{ij} = 1$ if j is the correct class for digit i and zero otherwise. The proposed loss function is the entropy computed only in positions having a digit or background:

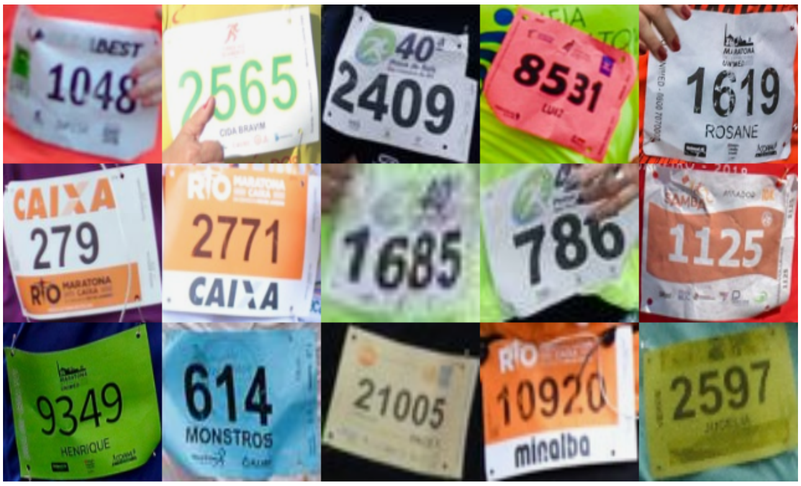
$$loss(M, M^*) = -\sum_{i=1}^{5}\sum_{j=1}^{11} M^*_{ij} \log(M_{ij}), \tag{6.5}$$

which means that M_{ij} is expected to increase when $M^*_{ij} = 1$ (i.e., at the correct class j for each digit i), but not explicitly enforced to decreased when $M^*_{ij} = 0$. If it was enforced, the negative samples would dominate the loss function, as 50 out of 55 positions in matrix

M are expected to be 0. Since the loss function is fed with a softmax activation function from the previous layer, the sum of responses across different classes must be one. Hence, if the response for the winning class is increased, the remaining responses must decrease. It is important to note that we also tested the widely used cross-entropy loss function between M^*_{ij} and M_{ij}, but results were inferior to the proposed loss.

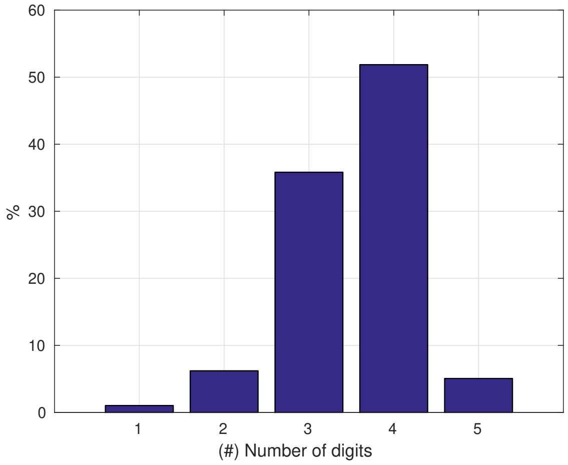
In compliance with the STN formulation, the weights of the fully connected layer just before the spatial transformer layer were initialized to generate the identity transformation. To train the network, we used the Adam solver (KINGMA; BA, 2014b) with a learning rate of 10^{-3} and batch size of 128, running this configuration for $100k$ iterations[7]. Moreover, the first half of the layers from each backbone were frozen or prevented from learning. A private dataset was used for training, and data augmentation was used to scale, rotate, translate, and modify the color space from images before being fed to the network. The data augmentation parametrization is similar to the one presented in Chapter 5 for ALPR, also including off-plane rotations. The main difference is related to scaling: in the bib recognition case, this variation needs to be much less aggressive to couple with the detection step, keeping the final size close to the original. Our dataset presents around $2,000$ annotated bib crops, with an average bib size of 60×72 pixels. A few samples of this dataset is shown in Figure 6.6. Moreover, Figure 6.7 presents a histogram of the number of digits per annotated bib, and it can be observed that the majority (52%) of bibs present four digits.

Figure 6.6: Samples of the training dataset.



[7] Training took around 12 hours using an NVIDIA Titan X GPU.

Figure 6.7: Normalized histogram (in %) of the number of digits per bib in our training dataset.



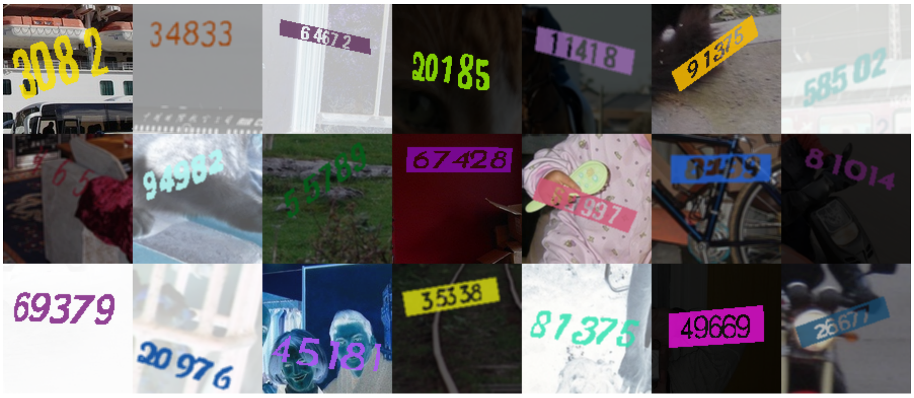
6.1.2.3 Synthetic Data

One problem observed with our architecture during testing is the difficulty to handle five-digit numbers, as they are considerably less frequent in the training dataset. In fact, one can imagine that every race will (most likely) have an athlete wearing the number 1 racing bib, but the probability of having the numbers 100, $1,000$ and $10,000$ becomes progressively lower as they rely upon the number of registrations to the event. Therefore, building a dataset for RBN-R with such variability is a very challenging and laborious process. To overcome this issue, we opted to generate synthetic data to help improving variability and, ultimately, alleviate overfitting of the GRU layer.

We created a synthetic dataset similar to the one presented in Section 5.4 in the context of license plates. Basically, we rendered numbers uniformly distributed from 1 to 99,999 over a crop of a random image from the PASCAL-VOC dataset. The rendered numbers are arbitrarily colored and rotated by an angle between -20 and 20 degrees, and there is a chance of including a rectangular solid background aligned with the bin number, as are several race templates. Figure 6.8 shows a few examples of generated images, all of them with 128×128 pixels, which is the input size of NR-NET.

While the NR-NET is focused on recognizing the number from a small patch, it does not handle the context where the number is inserted, as mentioned in the network for bib detection. Thus, randomly cropping image patches to use as a background is not an issue.

Figure 6.8: Synthesized data for bib recognition.



6.2 Experimental Results

6.2.1 Evaluation dataset and metrics

To evaluate our experiments, we used the publicly available dataset proposed by Ben-ami, Basha and Avidan (2012). It is composed of three subsets containing 92, 67 and 58 images respectively, in a total of 217 images. Each set corresponds to a different race, and bib sizes in the dataset vary from 13×28 to 120×182 pixels. In order to keep conformity with their evaluation protocol, we also used the same metrics, which are the precision, recall and F-score over the number of bibs correctly retrieved, i.e., bibs where all digits were correctly recognized. The F-score is given by:

$$F = 2\frac{pr}{r+p}, \tag{6.6}$$

which is the harmonic mean between the precision p and recall r.

6.2.2 Individual Analysis

We begin the evaluation by detailing the results obtained by our approach over each subset of the test set, which is shown in Table 6.3. We considered two scenarios for NR-NET: trained with and without synthetic data. In the synthetic data case, we used 70% of real augmented images, and 30% of synthetic images. This was empirically set based on the convergence of the network in the training set. Using less or more than 30% seemed to lead to a larger asymptotic loss value.

The most important metric, which is also relevant when evaluating license plates, is the recall rate — or the percentage of correctly recovered bibs. By analyzing such metric, we noticed that our approach had a poor performance in subset 2 when using the network trained without synthetic data. Furthermore, the precision over subset 3 was also low in both scenarios.

Table 6.3: Final RBNR results per subset.

Subset	Recall	Precision	F-Score
No synthetic data			
#1	0.88	0.76	0.81
#2	0.50	0.30	0.38
#3	0.84	0.50	0.62
With synthetic data			
#1	**0.90**	**0.78**	**0.83**
#2	**0.64**	**0.39**	**0.49**
#3	**0.87**	**0.52**	**0.65**

The errors in subset 2, when using only real data, were caused by the fact that this subset presents many 5-digit numbers. As mentioned earlier, this is a drawback of our training dataset, which motivated us to include synthesized data among real ones. Essentially, the GRU layer learned to ignore not frequent digits in the most significant position and considered them as a background.

In subset 3, the reason for the low precision rate was not related to our method, but to the annotations instead. Some images in this subset contain multiple athletes, and not all of them were annotated in the ground-truth data provided by Ben-ami, Basha and Avidan (2012). For instance, Figure 6.3a shows an example of an image from subset 3, along with bibs detected by our approach. Although there are eight clearly visible bibs, only four of them were annotated. From these eight visible bibs, our method was able to correctly recognize seven. However, the four bibs correctly recognized but not present in the annotation were considered as errors, deteriorating the precision rate.

The issue related to 5-digit bibs in subset 2 motivated us to perform a new analysis by computing the recall rate separated by the number of bib digits in the dataset. These results are shown in Table 6.4, with and without using synthetic training data. There is a clear advantage of using synthetic data in all scenarios, but the recognition of 5-digit numbers was the one that benefited the most.

Table 6.4: Recall rate per number of bib digits. Bibs with 1 or 2 digits were omitted, as they relate to a total of only three samples in the entire dataset.

Digits per number:	3	4	5
Without Synthetic	77.55%	87.27%	50.68%
With Synthetic	**81.63%**	**89.70%**	**65.75%**

6.2.3 Comparative Analysis

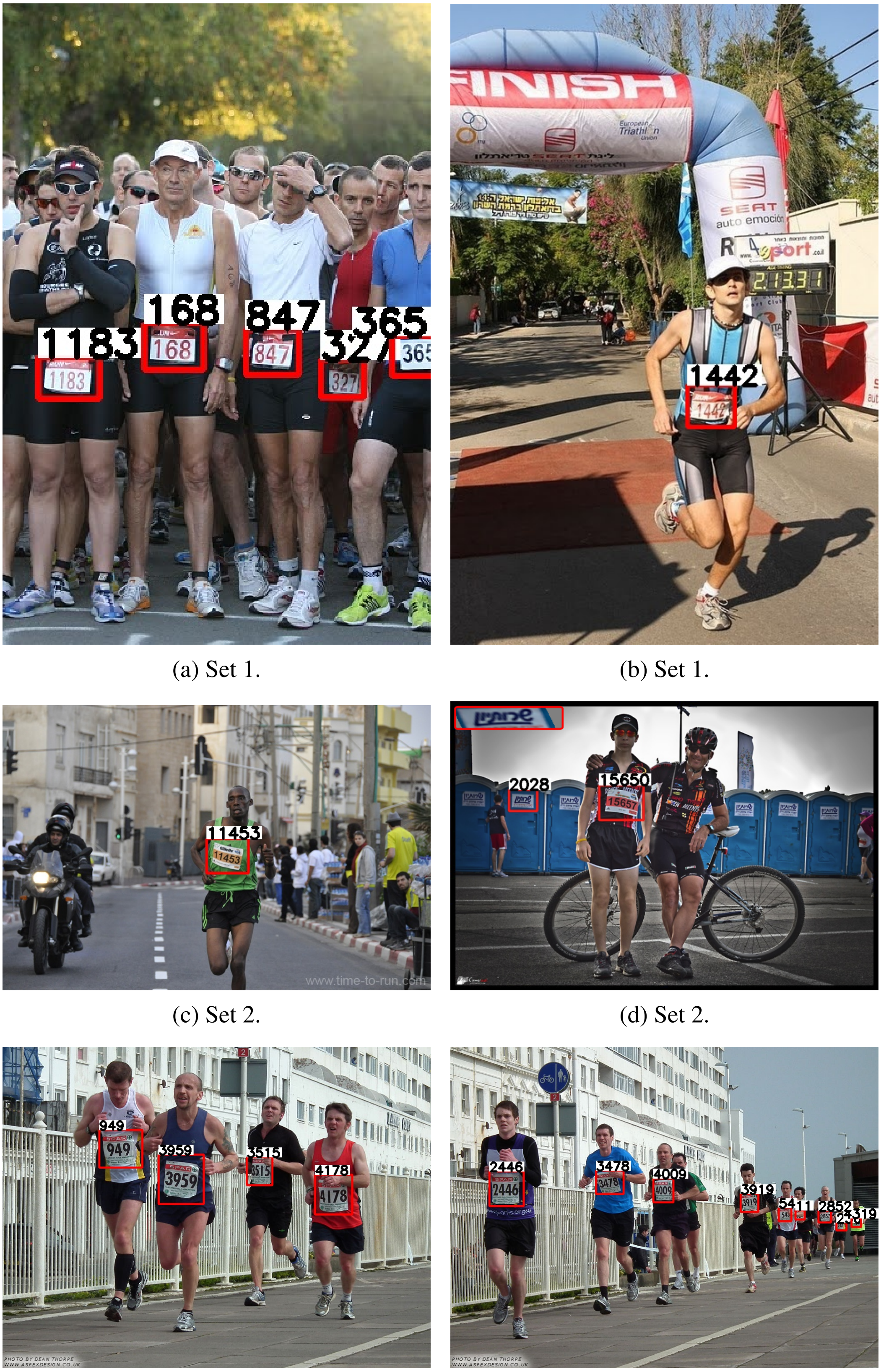
We compared the proposed approach to the works of Ben-ami, Basha and Avidan (2012) — our baseline — and Shivakumara et al. (2017) in terms of the weighted average of recall, precision and F-score from all subsets. The weights reflects the relative size of each subset, i.e. $w_1 = \frac{92}{217}, w_2 = \frac{67}{217}, w_3 = \frac{58}{217}$. The results are presented in Table 6.5, were the competitors accuracy were extracted from the paper of Shivakumara et al. (2017). These results show that our results without using synthetic data for training are already better than competitive approaches in both precision and recall rates (and, consequently, in the F-score). When using synthetic data for training we boost our recall rate in subset 2, and get even better overall results. Considering only the precision metric, the low rate among all methods – including ours – reinforces the annotation problems previously described. Still, by achieving the highest precision, we can say that our method could avoid more false positives than the others.

Table 6.5: Overall comparison.

	Recall	Precision	F-Score
Ours (no synthetic)	76.48%	53.50%	62.35%
Ours (with synthetic)	**81.17%**	**59.01%**	**67.69%**
Ben-ami, Basha and Avidan (2012)	69.00%	39.00%	50.00%
Shivakumara et al. (2017)	72.00%	41.00%	52.00%

A few results from the entire pipeline are shown in Figure 6.9. We can observe that these images present several artifacts that might compromise the final result, such as spectators (without bibs), textual information (e.g., sponsor advertising) and WC cabin plate. Our approach was able to correctly detect the racing bib and recognize the corresponding bib number in most cases, and the only (in these images) visual false positive in the detection stage was the WC cabin plate in Figure 6.9d. For the sake of curiosity, in the top-left corner of Figure 6.9d we show the output of the ST layer in NR-NET when forwarding the WC false-positive. Note that, in this case, the transformation focused on the central text, and slanted the image to the left in an attempt to achieve vertical alignment. It is important to mention that approaches based on torso/pedestrian detection could potentially detect false bibs in images with several spectators (e.g. Figure 6.9c), which is not the case of the proposed method. Finally, regarding system performance, considering a single bib per picture, our approach runs at 3 FPS in a CPU (Intel Core I7 8600). Each additional bib sums 0.03s in the final time.

Figure 6.9: Final results over a few samples from each subset.



(a) Set 1.

(b) Set 1.

(c) Set 2.

(d) Set 2.

(e) Set 3.

(f) Set 3.

7 CONCLUSIONS AND FUTURE WORK

This dissertation presented different approaches based on deep learning to tackle the problem called Contextual Text Recognition (CTR), in which the goal is to extract a string of text attached to a contextualized object. Examples of CTR are Automatic License Plate Recognition (ALPR), in which the license plate text is attached to the physical license plate, which in turn is related to a single vehicle; and Racing Bib Number Recognition (RBN-R), in which the number identifying the racer is attached to a bib, which in turn is attached to a person.

In this dissertation we presented enough information to demonstrate that the clever usage of augmentation techniques, and the design of careful network structures, allows to solve Contextualized Text Recognition problems through Deep Learning techniques, without being excessively slow or data hungry. Although CTR problems share the same common issues, they also differ in terms of the variability of the font shapes and sizes, and also of the contextual object.

We designed two different approaches to solve the ALPR problem, being one focused on real-time performance, and the other focused on accuracy and less constrained capture scenarios. The first, which was presented in Chapter 4, used two YOLO-based models, trained using a small dataset with augmentation, along with synthetic license plates based on the Brazilian and European layouts. As a result, we achieved more than real-time performance (70 FPS) over datasets of frontal and rear images, obtaining state-of-the-art accuracy.

For the second approach, we considered complex scenarios in ALPR which are far less studied than frontal/rear ones. For that end, we created a new dataset containing mostly oblique vehicles, which is not frequent even among challenging datasets. If we compare to the detection of faces-in-the-wild, this complexity usually demands huge quantities of data during training in order to obtain a good set of weights without overfitting. Following our dissertation hypothesis, we once again used a clever augmentation scheme, this time based also on off-plane rotations. Moreover, we designed a new network to regress affine transformations that fits canonical squares into the license plate regions. The affine transformation was also used to rectify the license plate, facilitating the work of the OCR. The resulting system achieved state-of-the-art results in the tested datasets, and also a better or at least competitive results to commercial systems.

Finally, in Chapter 6 we presented a third CTR approach focused on the task of

RBN-R. Comparing directly to ALPR, this task can be even more challenging due to factors like sweat, non-rigid materials (subject to movement), different layouts and constant occlusions. We demonstrate that a pipeline of two carefully designed networks, along with synthetic and augmented data, can again be sufficient to achieve state-of-the-art accuracy results. In particular, due to the lack of annotated datasets for RBN-R, the loss function of the recognition network was designed to work only with the bib number, instead of laborious annotation schemes that capture bounding boxes of each digit.

The approaches presented have many common factors related to the use of synthetic and augmented data, and architectural modifications over CNNs. Despite feature engineering is now in disuse due to the self learning filters of neural networks, there is still a substantial room for the development of data generation methods and architectural artifacts to make them learn faster and efficiently.

7.1 Future Work

Regarding ALPR, we plan to make adjustments to the approach presented in Chapter 5 in order to reduce computation without suffering loss of accuracy, and consequently, improving FPS. Moreover, by using the private dataset of Brazilian license plates, we want to expand the vehicle detection network by also classifying brand and model, since we have over 50,000 annotated images with this information.

Another usage for the approach presented in Chapter 5 that can be explored relates to camera calibration and pose estimation. Once it detects the four corners of distorted license plates and allows correction through affine transformation, we can also use such information coupled with real world data (e.g. license plate official size) to infer camera parameters, as done in techniques that explore planar patterns such as (ZHANG, 2000). In particular, the knowledge of external parameters can be explored to check the vehicle speed and also its direction when temporal information is available.

In the context of RBN-R, there are several possibilities to improve the overall result. Regarding OCR, the work of Baek et al. (2019) presents some useful insights that can be applied to character segmentation, as the text deformation is usually high. Also, we intend to bring Connectionist Temporal Classification (CTC) (GRAVES et al., 2006) to help improving character recognition when there is no annotation of the character position, only the string. This technique is suitable for recognizing characters as it learns to detect in-between character spaces. Similarly, the CTC can be applied to license plates as well, which is also a posterior research effort. We believe that our RBN-R method could be applied to similar tasks, like a soccer player or racing car identification, through simple fine-tuning of the networks.

Videos were not addressed in this work, and extending the current approaches in both ALPR and RBN-R is another future work. We understand that substantial improvements can be obtained by addressing temporal coherence, when applicable. In the case of ALPR, computationally efficient algorithms can be developed by lowering the search space using information from previous frames (if one knows the velocity of a vehicle at a given frame, we can infer its location in the next frame to reduce the search space). Furthermore, corrections over wrongly recognized characters can still be performed. For instance, when the vehicle is approaching the camera, the detection and OCR confidence tends to increase since the LP region is larger in image coordinates, as shown in Fig. 5.14. Thus, the detection size is also a piece of important information to be explored in a temporal coherence algorithm. Finally, in RBN-R, identifying the athletes by their appearance may help recover missing digits or detections. In this direction, works in the field of person re-identification, such as Li et al. (2014), can be employed to boost results.

REFERENCES

ANAGNOSTOPOULOS, C.-N. et al. License Plate Recognition From Still Images and Video Sequences: A Survey. **IEEE Transactions on Intelligent Transportation Systems**, v. 9, n. 3, p. 377–391, sep 2008. ISSN 1524-9050. Available from Internet: <http://ieeexplore.ieee.org/document/4518951/>.

ANGELOVA, A. et al. Real-time pedestrian detection with deep network cascades. In: **Proceedings of BMVC 2015**. [S.l.: s.n.], 2015.

BAEK, Y. et al. Character region awareness for text detection. **arXiv preprint arXiv:1904.01941**, 2019.

BALLAS, N. et al. Delving Deeper into Convolutional Networks for Learning Video Representations. p. 1–11, nov 2015. Available from Internet: <http://arxiv.org/abs/1511.06432>.

BEN-AMI, I.; BASHA, T.; AVIDAN, S. Racing Bib Numbers Recognition. In: **Procedings of the British Machine Vision Conference 2012**. British Machine Vision Association, 2012. p. 19.1–19.10. ISBN 1-901725-46-4. Available from Internet: <http://www.bmva.org/bmvc/2012/BMVC/paper019/index.html>.

BJÖRKLUND, T. et al. Robust license plate recognition using neural networks trained on synthetic images. **Pattern Recognition**, v. 93, p. 134–146, 2019. ISSN 00313203.

BULAN, O. et al. Segmentation- and Annotation-Free License Plate Recognition With Deep Localization and Failure Identification. **IEEE Transactions on Intelligent Transportation Systems**, v. 18, n. 9, p. 2351–2363, sep 2017. ISSN 1524-9050.

CHEN, L.-C. et al. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. **arXiv preprint arXiv:1606.00915**, 2016.

CHO, K. et al. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. 2014. Available from Internet: <http://arxiv.org/abs/1409.1259>.

CHUNG, J. et al. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. dec 2014. Available from Internet: <http://arxiv.org/abs/1412.3555>.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: **2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)**. [S.l.: s.n.], 2005. v. 1, p. 886–893 vol. 1. ISSN 1063-6919.

DENG, L. et al. Detecting multi-oriented text with corner-based region proposals. **Neurocomputing**, v. 334, p. 134–142, 2019. ISSN 18728286.

DONG, M. et al. A CNN-Based Approach for Automatic License Plate Recognition in the Wild. In: **British Machine Vision Conference**. London, UK: [s.n.], 2017.

DU, S. et al. Automatic License Plate Recognition (ALPR): A State-of-the-Art Review. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 23, n. 2, p. 311–325, feb 2013. ISSN 1051-8215. Available from Internet: <http://ieeexplore.ieee.org/document/6213519/>.

DUCHI, J.; HAZAN, E.; SINGER, Y. **Adaptive Subgradient Methods for Online Learning and Stochastic Optimization**. [S.l.], 2010.

EIKVIL, L. **OCR - Optical Character Recognition**. 1993.

EPSHTEIN, B.; OFEK, E.; WEXLER, Y. Detecting text in natural scenes with stroke width transform. In: **2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition**. [S.l.]: IEEE, 2010. p. 2963–2970. ISBN 978-1-4244-6984-0.

EVERINGHAM, M. et al. **The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results**. 2007. Http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

EVERINGHAM, M. et al. The pascal visual object classes (voc) challenge. **International Journal of Computer Vision**, v. 88, n. 2, p. 303–338, jun. 2010.

GALLO, I.; ZAMBERLETTI, A.; NOCE, L. Robust Angle Invariant GAS Meter Reading. In: **2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA)**. IEEE, 2015. p. 1–7. ISBN 978-1-4673-6795-0. Available from Internet: <http://ieeexplore.ieee.org/document/7371300/>.

GAO, Y. et al. Automatic Watermeter Digit Recognition on Mobile Devices. In: **Communications in Computer and Information Science**. [S.l.: s.n.], 2018. v. 819, p. 87–95. ISBN 9789811085291.

GIRSHICK, R. Fast r-cnn. In: **2015 IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2015. p. 1440–1448.

GIRSHICK, R. et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: **Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition**. Washington, DC, USA: IEEE Computer Society, 2014. (CVPR '14), p. 580–587. ISBN 978-1-4799-5118-5.

GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In: **In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics**. [S.l.: s.n.], 2010.

GONÇALVES, G. R.; MENOTTI, D.; SCHWARTZ, W. R. License plate recognition based on temporal redundancy. In: **2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)**. [S.l.: s.n.], 2016. p. 2577–2582.

GONÇALVES, G. R. et al. Benchmark for license plate character segmentation. **Journal of Electronic Imaging**, v. 25, n. 5, p. 1–5, 2016.

GOYAL, P. et al. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. **Arxiv**, jun 2017. ISSN 10495258. Available from Internet: <http://arxiv.org/abs/1706.02677>.

GRAVES, A. et al. Connectionist temporal classification. In: **Proceedings of the 23rd international conference on Machine learning - ICML '06**. New York, New York, USA: ACM Press, 2006. p. 369–376. ISBN 1595933832. ISSN 10987576. Available from Internet: <http://portal.acm.org/citation.cfm?doid=1143844.1143891>.

GUPTA, A.; VEDALDI, A.; ZISSERMAN, A. Synthetic data for text localisation in natural images. In: **IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2016.

HE, K. et al. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 37, n. 9, p. 1904–1916, 2015. ISSN 01628828.

HE, K. et al. Deep residual learning for image recognition. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 770–778.

HE, T. et al. Text-attentional convolutional neural network for scene text detection. **IEEE Transactions on Image Processing**, v. 25, n. 6, p. 2529–2541, June 2016. ISSN 1057-7149.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural Computation**, v. 9, n. 8, p. 1735–1780, 1997. Available from Internet: <https://doi.org/10.1162/neco.1997.9.8.1735>.

HOWARD, A. G. et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. **CoRR**, abs/1704.04861, 2017.

HSU, G.-S. et al. Robust license plate detection in the wild. In: **2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)**. IEEE, 2017. p. 1–6. ISBN 978-1-5386-2939-0. Available from Internet: <http://ieeexplore.ieee.org/document/8078493/>.

HSU, G.-S.; CHEN, J.-C.; CHUNG, Y.-Z. Application-Oriented License Plate Recognition. **IEEE Transactions on Vehicular Technology**, v. 62, n. 2, p. 552–561, feb 2013. ISSN 0018-9545.

HU, J.; SHEN, L.; SUN, G. Squeeze-and-Excitation Networks. In: **2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition**. IEEE, 2018. p. 7132–7141. ISBN 978-1-5386-6420-9. ISSN 10636919. Available from Internet: <https://ieeexplore.ieee.org/document/8578843/>.

HUANG, J. et al. Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors. In: **2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. IEEE, 2017. p. 3296–3297. ISBN 978-1-5386-0457-1. Available from Internet: <http://ieeexplore.ieee.org/document/8099834/>.

IRIE, K. et al. LSTM, GRU, Highway and a Bit of Attention: An Empirical Overview for Language Modeling in Speech Recognition. In: **Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH**. [S.l.: s.n.], 2016. v. 08-12-Sept, p. 3519–3523. ISSN 19909772.

JADERBERG, M. et al. Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition. **NIPS, Conference on Neural Information Processing Systems**, p. 1–10, 2014. ISSN 1550-5499.

JADERBERG, M. et al. Spatial transformer networks. In: CORTES, C. et al. (Ed.). **Advances in Neural Information Processing Systems 28**. Curran Associates, Inc., 2015. p. 2017–2025. Available from Internet: <http://papers.nips.cc/paper/5854-spatial-transformer-networks.pdf>.

KAMLESH et al. Person Re-identification with End-to-End Scene Text Recognition. In: . [s.n.], 2017. p. 363–374. ISBN 9789811073052. Available from Internet: <http://link.springer.com/10.1007/978-981-10-7305-2{_}.>

KARATZAS, D. et al. Icdar 2015 competition on robust reading. In: **2015 13th International Conference on Document Analysis and Recognition (ICDAR)**. [S.l.: s.n.], 2015. p. 1156–1160.

KINGMA, D.; BA, J. Adam: A method for stochastic optimization. **International Conference on Learning Representations (ICLR)**, 2014.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **CoRR**, abs/1412.6980, 2014.

KRAUSE, J. et al. 3d object representations for fine-grained categorization. In: **4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)**. Sydney, Australia: [s.n.], 2013.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: **Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1**. USA: Curran Associates Inc., 2012. (NIPS'12), p. 1097–1105.

KURPIEL, F. D.; MINETTO, R.; NASSU, B. T. Convolutional neural networks for license plate detection in images. In: **2017 IEEE International Conference on Image Processing (ICIP)**. IEEE, 2017. p. 3395–3399. ISBN 978-1-5090-2175-8. Available from Internet: <http://ieeexplore.ieee.org/document/8296912/>.

KUZNETSOVA, A. et al. The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. nov 2018. Available from Internet: <http://arxiv.org/abs/1811.00982>.

LAROCA, R. et al. Convolutional neural networks for automatic meter reading. **Journal of Electronic Imaging**, v. 28, n. 01, p. 1, 2019.

LAROCA, R. et al. A robust real-time automatic license plate recognition based on the YOLO detector. **CoRR**, abs/1802.09567, 2018.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 5 2015. ISSN 0028-0836.

LECUN, Y. et al. Backpropagation Applied to Handwritten Zip Code Recognition. **Neural Computation**, v. 1, n. 4, p. 541–551, dec 1989. ISSN 0899-7667.

LECUN, Y. et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2324, 1998. ISSN 00189219.

LECUN, Y. et al. Efficient backprop. In: ____. **Neural Networks: Tricks of the Trade**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. p. 9–50. ISBN 978-3-540-49430-0.

LENC, K.; VEDALDI, A. R-CNN minus R. In: **Procedings of the British Machine Vision Conference 2015**. [S.l.]: British Machine Vision Association, 2015. p. 5.1–5.12. ISBN 1-901725-53-7.

LI, H.; SHEN, C. Reading Car License Plates Using Deep Convolutional Neural Networks and LSTMs. jan 2016. Available from Internet: <http://arxiv.org/abs/1601.05610>.

LI, H.; WANG, P.; SHEN, C. Towards end-to-end car license plates detection and recognition with deep neural networks. **CoRR**, abs/1709.08828, 2017. Available from Internet: <http://arxiv.org/abs/1709.08828>.

LI, W. et al. DeepReID: Deep Filter Pairing Neural Network for Person Re-identification. In: **2014 IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.]: IEEE, 2014. p. 152–159. ISBN 978-1-4799-5118-5. ISSN 10636919.

LIAO, M. et al. TextBoxes: A Fast Text Detector with a Single Deep Neural Network. In: **AAAI**. San Francisco: [s.n.], 2017.

LIN, T.-Y. et al. Microsoft coco: Common objects in context. In: ____. **Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V**. [S.l.]: Springer International Publishing, 2014. p. 740–755. ISBN 978-3-319-10602-1.

LIU, W. et al. Ssd: Single shot multibox detector. In: ____. **Computer Vision – ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I**. [S.l.]: Springer International Publishing, 2016. p. 21–37.

LIU, Y. et al. Curved scene text detection via transverse and longitudinal sequence connection. **Pattern Recognition**, Elsevier Ltd, v. 90, p. 337–345, 2019. ISSN 00313203.

MASOOD, S. Z. et al. License Plate Detection and Recognition Using Deeply Learned Convolutional Neural Networks. 2017. Available from Internet: <http://arxiv.org/abs/1703.07330>.

MATAS, J. et al. Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. In: **Procedings of the British Machine Vision Conference 2002**. [S.l.]: British Machine Vision Association, 2002. p. 36.1–36.10. ISBN 1-901725-19-7.

MATAS, J. et al. Robust wide-baseline stereo from maximally stable extremal regions. **Image and Vision Computing**, v. 22, n. 10, p. 761 – 767, 2004. ISSN 0262-8856. British Machine Vision Computing 2002.

NAZARÉ, T. S. et al. Deep convolutional neural networks and noisy images. November 2017.

OPENALPR. **Benchmark**. 2014. <https://github.com/openalpr/benchmarks>. [Online; accessed 09-April-2018].

PÉREZ, L.; WANG, J. M. The effectiveness of data augmentation in image classification using deep learning. In: . [S.l.: s.n.], 2017.

PÉREZ, P.; GANGNET, M.; BLAKE, A. Poisson image editing. In: **ACM SIGGRAPH 2003 Papers**. New York, NY, USA: ACM, 2003. (SIGGRAPH '03), p. 313–318. ISBN 1-58113-709-5. Available from Internet: <http://doi.acm.org/10.1145/1201775.882269>.

PSYLLOS, A.; ANAGNOSTOPOULOS, C.; KAYAFAS, E. Vehicle model recognition from frontal view image measurements. **Computer Standards & Interfaces**, Elsevier B.V., v. 33, n. 2, p. 142–151, feb 2011. ISSN 09205489.

RANA, R. Gated Recurrent Unit (GRU) for Emotion Classification from Noisy Speech. p. 1–9, dec 2016. Available from Internet: <http://arxiv.org/abs/1612.07778>.

REDMON, J. et al. You Only Look Once: Unified, Real-Time Object Detection. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.]: IEEE, 2016. p. 779–788. ISBN 978-1-4673-8851-1.

REDMON, J.; FARHADI, A. Yolo9000: better, faster, stronger. p. 7263–7271, 2017.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. **arXiv**, 2018.

REN, S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 39, n. 6, p. 1137–1149, June 2017. ISSN 0162-8828.

ROTHER, C.; KOLMOGOROV, V.; BLAKE, A. "GrabCut". **ACM Transactions on Graphics**, v. 23, n. 3, p. 309, aug 2004. ISSN 07300301.

RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. **International Journal of Computer Vision (IJCV)**, v. 115, n. 3, p. 211–252, 2015.

SELMI, Z.; HALIMA, M. B.; ALIMI, A. M. Deep Learning System for Automatic License Plate Detection and Recognition. In: **2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)**. IEEE, 2017. p. 1132–1138. ISBN 978-1-5386-3586-5. Available from Internet: <http://ieeexplore.ieee.org/document/8270118/>.

SERMANET, P. et al. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. **arXiv preprint arXiv**, p. 1312.6229, 2013.

SHI, B.; BAI, X.; YAO, C. 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**, v. 39, n. 11, p. 2298–2304, nov 2017. ISSN 0162-8828.

SHIVAKUMARA, P. et al. A new multi-modal approach to bib number/text detection and recognition in Marathon images. **Pattern Recognition**, Elsevier, v. 61, p. 479–491, jan 2017.

SILVA, S. M.; JUNG, C. R. Real-time brazilian license plate detection and recognition using deep convolutional neural networks. In: **2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)**. [S.l.: s.n.], 2017. p. 55–62.

SILVA, S. M.; JUNG, C. R. License plate detection and recognition in unconstrained scenarios. In: **2018 European Conference on Computer Vision (ECCV)**. [S.l.: s.n.], 2018. p. 580–596.

SIMONYAN, K.; ZISSERMAN, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. p. 1–14, sep 2014. Available from Internet: <http://arxiv.org/abs/1409.1556>.

SMITH, R. An overview of the tesseract ocr engine. In: **Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)**. [S.l.: s.n.], 2007. v. 2, p. 629–633. ISSN 1520-5363.

TAIGMAN, Y. et al. Deepface: Closing the gap to human-level performance in face verification. In: **2014 IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2014. p. 1701–1708. ISSN 1063-6919.

VAROL, G. et al. Learning from synthetic humans. In: **2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2017. p. 4627–4635. ISSN 1063-6919.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In: **Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001**. [S.l.: s.n.], 2001. v. 1, p. I–511–I–518 vol.1. ISSN 1063-6919.

VISIN, F. et al. ReSeg: A Recurrent Neural Network-Based Model for Semantic Segmentation. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)**. IEEE, 2016. p. 426–433. ISBN 978-1-5090-1437-8. ISSN 21607516. Available from Internet: <http://ieeexplore.ieee.org/document/7789550/>.

WANG, F. et al. Geometry-Aware Scene Text Detection with Instance Transformation Network. In: **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. Salt Lake City: [s.n.], 2018. p. 1381–1389.

WANG, L.; HE, D.-C. Texture classification using texture spectrum. **Pattern Recognition**, v. 23, n. 8, p. 905 – 910, 1990. ISSN 0031-3203.

WANG, T. et al. Reading digits in natural images with unsupervised feature learning. **NIPS**, p. 1–9, 2011. Available from Internet: <http://research.google.com/pubs/archive/37648.pdf>.

XIE, L. et al. A New CNN-Based Method for Multi-Directional Car License Plate Detection. **IEEE Transactions on Intelligent Transportation Systems**, v. 19, n. 2, p. 507–517, feb 2018. ISSN 1524-9050.

YANG, H. et al. An Efficient Method for Vehicle Model Identification via Logo Recognition. In: **2013 International Conference on Computational and Information Sciences**. [S.l.]: IEEE, 2013. p. 1080–1083. ISBN 978-0-7695-5004-6.

YE, Q.; DOERMANN, D. Text detection and recognition in imagery: A survey. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 37, n. 7, p. 1480–1500, July 2015. ISSN 0162-8828.

YOSINSKI, J. et al. How transferable are features in deep neural networks? In: **Proceedings of the 27th International Conference on Neural Information Processing Systems**. Cambridge, MA, USA: MIT Press, 2014. (NIPS'14), p. 3320–3328.

YOU, Y. et al. ImageNet Training in Minutes. 2017. Available from Internet: <http://arxiv.org/abs/1709.05011>.

ZEILER, M. D. ADADELTA: an adaptive learning rate method. **CoRR**, abs/1212.5701, 2012.

ZEILER, M. D.; FERGUS, R. Visualizing and Understanding Convolutional Networks. In: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. [S.l.: s.n.], 2014. v. 8689 LNCS, n. PART 1, p. 818–833. ISBN 9783319105895.

ZHANG, Z. A flexible new technique for camera calibration. **IEEE Transactions on pattern analysis and machine intelligence**, v. 22, p. 1330–1334, 2000.

ZHOU, X. et al. Sparse representation for 3d shape estimation: A convex relaxation approach. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 39, n. 8, p. 1648–1661, 2017.

ZHU, Z. et al. Traffic-sign detection and classification in the wild. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 2110–2118.