THE UNIVERSITY
of ADELAIDE

Text Detection and Recognition in Natural Scene Images

Hui Li

A thesis submitted for the degree of
DOCTOR OF PHILOSOPHY
School of Computer Science
The University of Adelaide

June 2018

Declaration

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

The author acknowledges that copyright of published works contained within this thesis resides with the copyright holder(s) of those works.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

I acknowledge the support I have received for my research through the provision of an Australian Government Research Training Program Scholarship.

Signed: **Date**:

Acknowledgments

First and foremost, I would like to thank my principle supervisor, Prof. Chunhua Shen, for all his support and help through the years. He introduces me to the world of computer vision and machine learning. During the course of my PhD study, he gave me a lot of constructive advice and in-depth guidance. His dedication to cutting edge research encouraged me a lot, and his rigorous attitude on research work affected my greatly. I would not be able to pursue my PhD degree without his support.

Next, I would like to thank the other members in my advisory term, Dr. Qinfeng Shi and Dr. Qi Wu, for their valuable support and insightful feedback during my study. I would also like to thank my friends and colleagues in the lab at the University of Adelaide. Thanks to everyone there, including but not limited to Qichang Hu, Teng Li, Yao Li, Bohan Zhuang, Zhibin Liao, Yu Chen, Tong He, Yuanzhouhan Cao, Guosheng Lin, Lingqiao Liu and Peng Wang. I have learnt so much from them.

Last but not least, I would like to thank my family for their unconditional love and endless support. I am especially grateful to my husband. He is not only my life partner, but also my soul mate. His smart idea and serious attitude towards work impress me a lot. Thanks to my parents for helping me look after my daughter and supporting me in the pursuit of my academic career. Thanks to my lovely daughter for putting up with mother's late nights and weekends work. They give me the energy to keep going on my research.

Publications

This thesis is based on the content of the following peer-reviewed conference and journal papers:

- Hui Li, Peng Wang, Mingyu You, Chunhua Shen; "Reading Car License Plates Using Deep Neural Networks"; *Image and Vision Computing* (IVC), 2018. Corresponding to Chapter 3
- Hui Li, Peng Wang, Chunhua Shen; "Towards End-to-End Car License Plates Detection and Recognition with Deep Neural Networks"; *IEEE Transactions on Intelligent Transportation Systems* (TITS), 2018. Corresponding to Chapter 4
- Hui Li, Peng Wang, Chunhua Shen; "Towards End-to-end Text Spotting with Convolutional Recurrent Neural Networks"; In *International Conference on Computer Vision* (ICCV), 2017 (spotlight). Corresponding to Chapter 5
- Hui Li, Chunhua Shen, Anton van den Hengel, Qinfeng Shi; "Worst Case Linear Discriminant Analysis as Scalable Semidefinite Feasibility Problems"; *IEEE Transactions on Image processing* (TIP), 2015. This is a separate work that I have done at the beginning of my Ph.D Studying. This work does not relate to text detection and recognition. It is an efficient semidefinite programming approach to worst-case linear discriminant analysis. We will not present this work in this thesis.

Abstract

This thesis addresses the problem of end-to-end text detection and recognition in natural scene images based on deep neural networks. Scene text detection and recognition aim to find regions in an image that are considered as text by human beings, generate a bounding box for each word and output a corresponding sequence of characters. As a useful task in image analysis, scene text detection and recognition attract much attention in computer vision field. In this thesis, we tackle this problem by taking advantage of the success in deep learning techniques.

Car license plates can be viewed as a spacial case of scene text, as they both consist of characters and appear in natural scenes. Nevertheless, they have their respective specificities. During the research progress, we start from car license plate detection and recognition. Then we extend the methods to general scene text, with additional ideas proposed.

For both tasks, we develop two approaches respectively: a stepwise one and an integrated one. Stepwise methods tackle text detection and recognition step by step by respective models; while integrated methods handle both text detection and recognition simultaneously via one model. All approaches are based on the powerful deep Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), considering the tremendous breakthroughs they brought into the computer vision community.

To begin with, a stepwise framework is proposed to tackle text detection and recognition, with its application to car license plates and general scene text respectively. A character CNN classifier is well trained to detect characters from an image in a sliding window manner. The detected characters are then grouped together as license plates or text lines according to some heuristic rules. A sequence labeling based method is proposed to recognize the whole license plate or text line without character level segmentation.

On the basis of the sequence labeling based recognition method, to accelerate the processing speed, an integrated deep neural network is then proposed to address car license plate detection and recognition concurrently. It integrates both CNNs and RNNs in one network, and can be trained end-to-end. Both car license plate bounding boxes and their labels are generated in a single forward evaluation of the network. The whole process involves no heuristic rule, and avoids intermediate procedures like image cropping or feature recalculation, which not only prevents error accumulation, but also reduces computation burden.

Lastly, the unified network is extended to simultaneous general text detection and recognition in natural scene. In contrast to the one for car license plates, some innovations are proposed to accommodate the special characteristics of general text. A varying-size RoI encoding method is proposed to handle the various aspect ratios of

general text. An attention-based sequence-to-sequence learning structure is adopted for word recognition. It is expected that a character-level language model can be learnt in this manner. The whole framework can be trained end-to-end, requiring only images, the ground-truth bounding boxes and text labels. Through end-to-end training, the learned features can be more discriminative, which improves the overall performance. The convolutional features are calculated only once and shared by both detection and recognition, which saves the processing time. The proposed method has achieved state-of-the-art performance on several standard benchmark datasets.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Text, as a kind of physical representation of language, is considered as one of the most influential inventions of humanity. It plays an important role in human life, such as communicating ideas, delivering information, *etc.*. In fact, ancient civilization is interpreted and inherited via the use of text or textual cues. As a rich and precise information carrier, text contained in images can be of great semantic value for the whole image understanding. With the popularization of high-performance mobile devices, such as mobile phones, digital cameras, it becomes much faster and easier to acquire scene images or videos. The collection of massive amount of images drives data analysis and processing. Text detection and recognition in natural scene images, as important analyzing procedures, have attracted increasing attentions in computer vision community.

The objective of text detection and recognition, also referred to as text spotting for simplicity [Jaderberg et al., 2014b], is to mark all areas in an image that are considered as text by humans, in term of bounding boxes, and output corresponding text transcriptions, as presented in Figure 1.1 . Although human beings can read text from images instantaneously if they are familiar with that language, it is not an easy work for machine. Fortunately, there is a rapid development in computer vision and pattern recognition technologies in recent years, which makes it more possible to address these challenging problems. In this thesis, we try to tackle both text detection and recognition by leveraging the advanced deep learning techniques. Deep learning techniques have been applied to a wide range of vision problems

Figure 1.1: Illustration of text detection and recognition in natural scene images, with both text bounding boxes and the corresponding text labels expected.

such as image classification, object detection, image segmentation, and made a lot of breakthrough progresses [Krizhevsky et al., 2012; Simonyan et al., 2013; Girshick et al., 2014; Ren et al., 2015; He et al., 2016a, 2017a]. For instance, the object detection average precision on the PASCAL VOC 2007 dataset has been increased from 29.1% to 78.8% during the last 5 years, thanks to the use of deep Convolutional Neural Network (CNN), and the related technical innovations. However, standard object detection and image recognition methods cannot be adopted directly for text, not only because text has distinct patterns, but also because text by itself is quite variable. Even if under a single language in which text is composed of alphabet of size $\mathscr{A}$ with a maximal length L, there would be up to $\mathscr{A}^{\mathrm{L}}$ different text classes. Thereby, it is necessary to propose distinctive methods for text with its special characteristic into consideration.

To better understand the overall value of text spotting systems, firstly we would like to introduce some background information in detail, including potential appli-

cations, technical challenges and so on.

1.1 Background

1.1.1 Applications

Automatic text detection and recognition offer a mean to read text within images or videos directly. They allow a better understanding of the scene, and result in a number of vision applications.

- **Scene Understanding.** Street and shop signs in natural scenes carry crucial information for localization. Automatic sign recognizer gives robots scene context, and enables them to understand the environment they are involved. It can also be used to help visually impaired person. For example, integrating the scene text spotting algorithm into a speech synthesizer would help blind people understand street signs or make out road instructions.

- **Image Retrieval.** Automatically reading text within images would allow image search and retrieval from a large scale database of digital images or videos. Given a text query, the images that contain the text can be returned instantly. With this technique, humongous images can be indexed by the text within them. Users can find their favorite movie posters from a large database easily by typing the name of the movie.

- **Text Translation.** Traditionally, users have to type the word to be translated into machine translation applications. However, this can be quite slow or sometimes even impossible, e.g., a European tourist travels to Asia and wants to translate Chinese or Korean text. It would be convenient if a scene text spotter is integrated into the translator. Users only need to take a photo including the words or phrases they wants to know. The words will then be localized and recognized automatically by the text spotter and translated to users.

- **Intelligent Transportation System.** As a spacial case of scene text, car license plates contain critical information and are often used as identifications for cars. Automatic car license plate detection and recognition are essential parts of intelligent transportation systems. Applications include electronic payment systems, traffic surveillance for vehicles violating traffic laws. They can also be used to help policemen to find stolen cars or cars with criminals.

1.1.2 Challenges

As a similar task, text recognition from scanned document (usually be named as Optical Character Recognition (OCR)) is usually regarded as a well solved problem. The recognition rate has achieved higher than 99%, and many systems are already used in real applications [Weinman et al., 2009; Chen and Yuille, 2004]. However, text detection and recognition in natural scene images are still challenging problems and far less developed. For example, the best-reported end-to-end detection and recognition performance evaluated by F-measure is only 40.07% on COCO Text Dataset [Veit et al., 2016]. The difficulties lie in the following aspects:

- **Diversity of text patterns.** In contrast to text in scanned document images, which is usually with a single color, font and uniform arrangement, text appeared in natural scene images can be of diverse fonts and colors, as compared in Figure 1.2. Some text can be in a very small size compared with others in the same image, with low quality or even multiple orientations (vertical, curved, *etc.*).

- **Highly complicated background.** In natural environment, image background can be really complicated, with a variety of scene, textual, color, *etc.*. Some background may have similar structures and appearances as text, such as windows, leaves, bricks, and often cause false alarms in detection.

- **Capturing conditions.** When capturing images in the wild, the resulting

Chapter 1

Introduction

1.1 Overview

Image understanding is effortless and instantaneous for humans but remains a fundamental challenge for machine vision. Enabling machines to understand images as we humans do is one of the ultimate goals in the field of computer vision. By appropriately defining the classes, image understanding can be used to determine the presence of objects in an image (object recognition), locate the position of an object in an image (object detection), classify the category of an object in an image (object classification), to name just a few.

Image understanding has attracted a lot of attention for a long time in the computer vision community. A crucial problem in the image understanding research is what is a good feature representation for machines to understand images. Historically, many previous works have been focused on the design of meaningful feature representations. Tremendous efforts have been made in seeking good feature representations for image understanding in the last twenty years. After the introduction of the scale-invariant features (Lowe, 1999), better performance was achieved by new feature representations(e.g. local binary patterns (Ahonen et al., 2004), histogram of gradients (Dalal and Triggs, 2005), region covariance features (Tuzel et al., 2006), aggregated channel features (Dollár et al., 2010), etc.). Although many significant research progresses have been made along this line, their performances on different applications of image understanding are still far from being satisfactory. The main drawback for these representations is that they often make strong assumptions on the input image, e.g. the images are well aligned and the illumination of images is stable. However, for the images captured from real environments, these assumptions are often violated and hence these methods are usually too fragile for real-world applications.

Recently, benefiting from the collection of large-scale datasets and the development of computing power, a breakthrough was made by deep convolutional neural

1

Figure 1.2: The comparison between text in document images (left) and in natural scene images (right).

images may be affected by a lot of factors, such as uneven lighting, reflection, blur, distortion, low resolution, or partial occlusion, which make scene text detection and recognition even challenging.

- **Multilingual environment.** In modern society, multiple cultures live together and it is quite common that an image contains multi-lingual text. The basic characters in different language are really distinct from each other, e.g., Latin, Chinese, Japanese, Arabic, which makes text recognition even more difficult. A robust text detection and recognition method should be able to handle various scripts without fundamental changes in algorithm.

Some challenging examples of images captured in natural scenes, with text should be read out by a text spotting system, are presented in Figure 1.3.

1.1.3 Tasks and Methodologies

As we stated, text spotting in natural images basically includes two tasks: text detection and word recognition. Text detection aims to generate candidate bounding boxes from natural scene images that correspond to words, while word recognition attempts to recognize the character string in each candidate bounding box.

Figure 1.3: Some challenging examples for text spotting, where text is found in a variety of scenarios, with different fonts, sizes, colors, illumination conditions, *etc.*. However, it is a useful problem to be solved, e.g., for scene understanding, localization, *etc.*.

There are heaps of methods proposed for text detection and recognition tasks. According to the overall architecture, these methods can be roughly categorized as: stepwise and integrated [Ye and Doermann, 2015], as presented in Figure 1.4. The stepwise methodology implements text detection and word recognition by totally separated modules, with a feed-forward pipeline, while the integrated methodology, by contrast, settles both text detection and recognition concurrently. In the following, we would like to present some examples so as to better understand those methodologies.

The stepwise methodology divides text spotting into two separated tasks, i.e., detection and recognition, and tackles them step by step. Text detection is carried out firstly. The detected bounding boxes are cropped out from the original image and then recognized by another model. Sometimes the result of word recognition

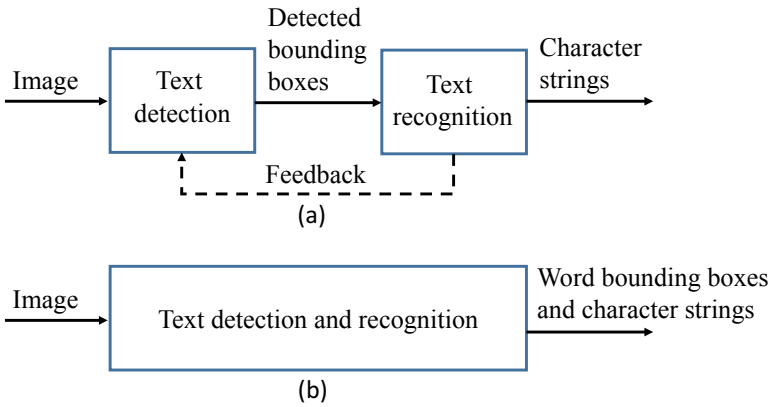


Figure 1.4: Two commonly used architectures for text spotting. (a) Stepwise methodology. (b) Integrated methodology.

can be used to further improve the detection performance, e.g., to remove false positives. Some methods start from characters, which seem to have exclusive outlines. Character grouping and word splitting are needed to obtain the word regions. Some others coarsely generate bounding boxes corresponding to word regions. A verification step is needed then to check those regions as text or non-text. For instance, in [Alsharif and Pineau, 2014], a stepwise pipeline including character detection, character grouping, text line separation and word recognition was proposed. Maximally Stable Extremal Regions (MSERs) were adopted firstly to get character regions, which were then clustered into text lines base on DB-SCAN. Line-to-word hybrid Hidden Markov Model (HMM)/Maxout was trained to segment lines into words and do word inference. Word Detection Maxout was followed for word verification. PhotoOCR [Bissacco et al., 2013] is also a stepwise method. It combined the outputs of three different text detection approaches firstly to get a high detection recall. Recognition began with over-segmentation of text line to identify candidate character regions, and was achieved by combining character classifiers and language model likelihoods. Jaderberg et al. [2016] used a combination of complementary proposal generation techniques for word region proposal, a random forest classifier for word/no-word classification. Word recognition was achieved by training a deep CNN over a huge dictionary of 90k words. The whole pipeline is presented in Fig-

ure 1.5 for reference. Liao et al. [2017] combined "TextBoxes" and "CRNN" for word detection and recognition respectively, where "TextBoxes" is a 28-layer fully convolutional network which can be trained to produce word level bounding boxes directly from input images, while "CRNN" [Shi et al., 2015] is a combination of CNN, RNN (Recurrent Neural Network) and CTC (Connectionist Temporal Classification) loss for image-based sequence recognition task. There are also some methods that solely focus on text detection [Tian et al., 2015; Busta et al., 2015; Zhu and Zanibbi, 2016] or word recognition [Su and Lu, 2014; Jaderberg et al., 2014a; Shi et al., 2015], which may be combined together for text spotting in a stepwise manner. We will review them in detail in Chapter 2.

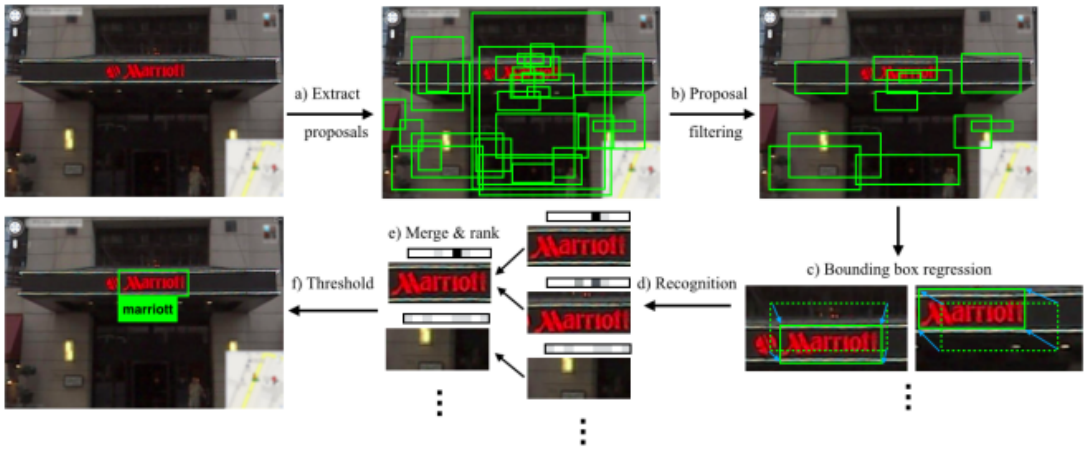


Figure 1.5: A stepwise methods for end-to-end text spotting used in [Jaderberg et al., 2016].

With the integrated methodology, both text detection and recognition are jointly addressed. Some methods share the character level classification responses, as each character has discriminative features from the background as well as from each other. For instance, as shown in Figure 1.6, Wang et al. [2011] proposed a word spotting method based on a given lexicon. A multi-scale sliding window based method was employed firstly to localize character candidates, using random ferns classifiers. Then Pictorial Structures (PS) formulation was adopted that takes the locations and scores of detected characters as inputs to find an optimal configuration of a

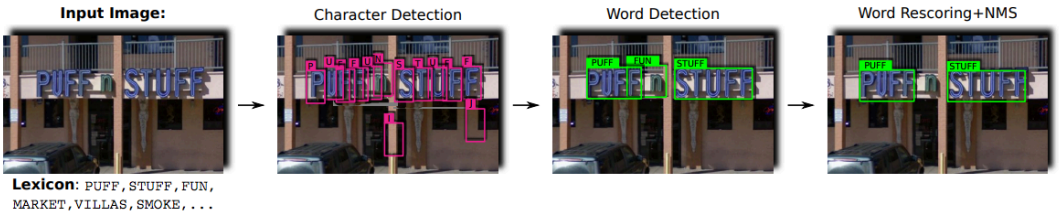


Figure 1.6: An integrated methods for end-to-end word spotting used in [Wang et al., 2011], with character level classification responses shared for both detection and recognition.

particular word from a given lexicon, and outputs both word bounding boxes as well as word labels simultaneously. Wang et al. [2012] integrated the character responses with character spacings and a defined lexicon using a beam search algorithm to localize and recognize words, where the character responses were obtained by a CNN classifier with unsupervised feature learning. Neumann and Matas [2013a] proposed an efficient algorithm to select the optimal sequence of characters from a directed graph, which was constructed with character classification scores, character intervals and language priors. The sequence of regions and their labels induced by the optimal path were the outputs. Most recently, Busta et al. [2017] proposed an end-to-end trainable network to address both word level detection and recognition concurrently, with a jointly optimized multi-task loss. Bartz et al. [2018] proposed a semi-supervised neural networks for scene text detection and recognition, where only ground-truth labels are needed to train the network. It is argued that the detection of text can be learned by the network itself.

Different methodologies have their own merits and demerits. The stepwise methods are easy to design and implement as each module only needs to deal with a single task, but they may include repeat calculation and introduce error accumulation. The integrated methods can avoid the challenging segmentation step or handle it in company with word recognition. The joint multi-task optimization may lead to better performance. However, the model is difficult to design. How to coordinate each part in the model is a main issue.

In this thesis, we would like to address the problem of end-to-end text detection and recognition by leveraging the advanced deep learning techniques, considering their powerful feature learning ability. We also base on those methodologies, but propose new frameworks and methods to get better performance. In particular, we focus on two tasks in this thesis, namely car license plate detection and recognition and general text detection and recognition. Car license plate can be viewed as a spatial case of general text, but has its own specificity. For example, car license plate mostly has obvious borders and appears in the front or rear of a vehicle. Characters in the license plate are usually randomly selected from a given set. In addition, the algorithm should be able to distinguish car license plates from other text in the background. In this work, A stepwise method and an integrated method are proposed respectively for car license plate detection and recognition. Then we extend these methods to general scene text spotting. General scene text also has its own particularities. For instance, there are usually several words flocking together in an image, in comparison with independent car license plates. The characters in a word may appear according to some rules. Considering these characteristics, additional innovations are proposed in both frameworks so as to better deal with general text.

1.2 Main Contributions

The main contributions of this thesis include a set of new frameworks proposed for car license plate and general text detection and recognition. The frameworks are built upon the state-of-the-art deep learning techniques, including deep CNNs and RNNs. More specifically, the contributions are as follows:

- Based on the stepwise methodology, we propose a cascade framework for car license plate detection and recognition. Firstly, a 37-class convolutional neural network (CNN) is trained to detect characters in an image via a sliding window manner. It can lead to a higher recall compared with a binary text/non-text classifier. The detected characters are then grouped into license plates based on

heuristic rules. Another plate/non-plate CNN classifier is designed to eliminate false positives. As to the license plate recognition, we regard the character string reading as a sequence labeling problem. Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM) is trained to recognize the sequential features extracted from the whole license plate via CNNs. The main advantage of this approach is that it is segmentation free. By exploring context information and avoiding errors caused by segmentation, this method performs better than conventional methods and achieves state-of-the-art recognition accuracy. To the best of our knowledge, this is the first work that recognizes license plates without character segmentation. Then we extend the method to general scene text detection and recognition. Considering the words clustering property of general text, we group the detected characters as text lines. An end-to-end trainable text line recognition network is proposed, based on the sequence labeling idea, which can tackle word recognition, word splitting and false positive removal in the same time. With this method, we achieved No. 1 on ICDAR2015 Challenge 2 "Focused Scene Text" for the task of "End-to-End Scene Text Recognition" at the time of submission.

- Inspired by the recent advances on object detection, we propose a unified deep neural network which can localize car license plates and recognize the letters simultaneously in a single forward pass. This is an integrated method. In contrast to previous stepwise approaches, this method jointly settles both detection and recognition by a single network. It not only avoids intermediate error accumulation, but also accelerates the processing speed. Compared to previous integrated methods, our network can be trained end-to-end, without a separate character detection needed. With convolutional features shared by both detection and recognition, the resulting system has fewer parameters and is more efficient. Moreover, with the end-to-end training and the joint optimization of both detection and recognition losses, the extracted features are more

discriminate, and lead to better performance.

- Considering the characteristics of general scene text, an end-to-end trainable text spotting network is proposed for text detection and recognition in natural scene images. Different from the above framework for car license plates, we adopt a sequence to sequence learning structure with an attention based RNN decoder for word recognition. In this way a character level language model can be learned implicitly. In addition, compared with general objects or car license plates, text bounding boxes usually have a significant diversity of aspect ratios. A novel feature encoding method is proposed which can keep the aspect ratio of the original word image and avoid distortion. A curriculum learning strategy is designed to train the model with gradually more complex training data. To our best knowledge, this is the first successful attempt to integrate text detection and recognition into a single end-to-end trainable network.

1.3 Thesis Outline

The structure of the thesis is organized as follows.

In Chapter 2, we firstly introduce some background information on deep learning techniques, including CNNs for local feature learning and RNNs for sequence feature learning, which are two powerful tools employed in our work. We also investigate the latest development of deep learning in related computer vision tasks, such as object detection, image caption, *etc.*. Some of these methods give us great inspirations on our text detection and recognition work. Then we give a brief introduction about existing methods on our target tasks, i.e., car license plate detection and recognition, and general scene text detection and recognition, respectively.

In Chapter 3, we present the stepwise method for text detection and recognition, with its application on both car license plates and general scene text. For each task, two independent modules are designed respectively for detection and recognition,

including a sliding window based method for character detection and a sequence labeling based method for sequence recognition. Some novel ideas are introduced to improve the detection and recognition performance.

In Chapter 4, an integrated method is described for car license plate detection and recognition. We integrate CNNs and RNNs in an end-to-end trainable deep neural network, and show that the convolutional features can be shared for both detection and recognition. Via the joint optimization of multi-loss, both detection and recognition performance can be improved. The computational speed can be accelerated as well.

Then in Chapter 5, we extend the unified network to process general text detection and recognition in natural scene images. Some new ideas are proposed to tackle the characteristics of general text, including a novel feature extraction and encoding approach, the sequence to sequence learning based word recognition method, as well as a curriculum learning strategy.

In Chapter 6, we conclude the thesis and discuss some potential research directions.

Chapter 2

Literature review

In this thesis, we are interested in leveraging deep learning techniques to solve the tasks, given the tremendous breakthroughs they brought to the computer vision community. In this chapter, we start with a review about basic concepts on deep learning and their related applications on object detection, image recognition, *etc.*, which are closely related to our tasks. Then we provide related works of this thesis' target problems, i.e., car license plate detection and recognition, and general scene text detection and recognition, respectively. We will introduce a range of different methods for each task.

2.1 Basic Concepts on Deep Learning and Related Applications

Feature representation is one of the key issues in many computer vision problems, such as image classification, object detection. A good feature representation should be able to capture all relevant and discriminative information needed for that task, for example, the key points of human faces. It should be invariant to appearance changes (e.g., front or side faces, various lighting conditions), and be discriminative against other objects.

Previous work usually depends on designing handcrafted features, such as Scale Invariant Feature Transform (SIFT) [Lowe, 2004], Histogram of Oriented Gradients (HOG) [Dalal and Triggs, 2005], Local Binary Patterns (LBP) [Ojala et al., 1996]. These

features are often incomplete and not robust. Moreover, repetitive work has to be done several times manually for each task. It is expected that machine learning algorithms could learn features automatically so that the entire learning process could be simplified and more tasks could be solved. Initially, most machine learning techniques exploited shallow-structured architectures, such as Support Vector Machines (SVMs) [Cortes and Vapnik, 1995], Gaussian Mixture Models (GMM) [Permuter et al., 2003], Conditional Random Fields (CRFs) [Lafferty et al., 2001]. These shallow architectures have shown effectiveness on a number of problems. However, they contain at most two layers of nonlinear transformations, which limits their power of representation and causes difficulties when dealing with complicated applications. With the emergence of large scale datasets, faster and parallel computers, deep learning techniques are developed. They provide ways of automated and hierarchical feature learning, and bring tremendous breakthroughs on several applications.

Deep learning uses multiple layers of nonlinear processing units for feature extraction and transformation. It learns in supervised and/or unsupervised manners, where algorithms make use of multiple levels of representations to gradually transform data into different levels of abstraction. Through layers of transformations, higher level features are derived from the lower level ones, which leads to a hierarchical representation of the object [Deng and Yu, 2014].

Such kind of design is largely motivated from animal brains. They are highly interconnected networks with billions of neurons and trillions of interconnections among them, and present a strong ability of learning and information processing. A number of algorithm architectures are proposed to emulate this structure, and the most successful one is Neural Networks (NNs) [Rosenblatt, 1958].

2.1.1 Neural Networks

In this section, we give a basic description about some well-known neural network architectures, which will be used later in our work.

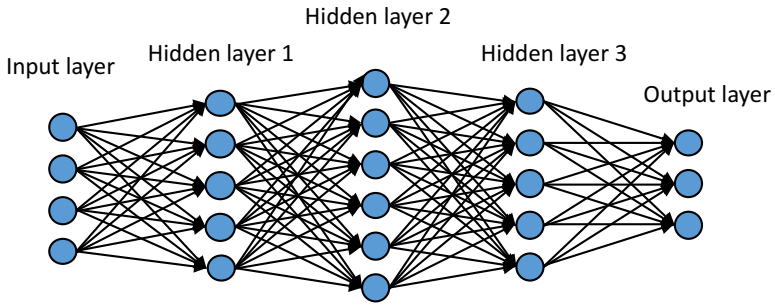


Figure 2.1: The architecture of a multi-layer perceptron. Each neuron in hidden layers are fully connected to all neurons in the previous layer, and neurons in one layer are completely independent with each other and do not share any connections.

Multi-Layer Perceptrons As presented in Figure 2.1, a Multi-Layer Perceptron (MLP) is a feed forward neural network that employs a series of hidden layers to map input vectors to output with a non-linear function. Each hidden layer consists of a set of *neurons*. Each neuron receives input signals from the previous layer and generates output by a weighted sum of inputs followed by a non-linear activation function. Mathematically, the output of a neuron can be presented as:

$$g(\mathbf{x}) = \sigma(\sum_{k=0}^{K} w_k x_k + b) = \sigma(\mathbf{w}^T \mathbf{x} + b) \tag{2.1}$$

where $\mathbf{x}$ is the input vector, σ is the non-linear function (i.e., *sigmoid* or *tanh*), $\mathbf{w}$ and b are the weights and bias to be learned. In this way the information is propagated forward until the last layer, which is known as the *output* layer. The values of the output layer can be class probabilities for classification problems or relative coordinates for regression problems. A loss function is usually defined to calculate the error between output values and the ground-truths. By minimizing the loss function, the weights and biases are optimized iteratively until the loss reaches a local optimal point.

As we can see from Figure 2.1, in MLP, each neuron in hidden layers is fully connected to all neurons in the previous layer, and neurons in one layer are completely independent of each other and do not share any connections. This fully connected structure cannot scale well to large images [Karpathy, 2017]. For example, given an

image of size $300 \times 300 \times 3$, a single fully connected neuron in the first hidden layer of MLP would have $300 \times 300 \times 3 = 270,000$ weights. This huge number of parameters are difficult to be trained and would easily lead to overfitting. Moreover, there are usually some correlations between neighboring inputs that are not taken used of in MLP.

Convolutional Neural Networks Convolutional Neural Networks (CNNs) [LeCun et al., 1998] are an extension of MLP which mitigate the aforementioned drawbacks by exploiting the strong spatial correlation, and are designed specifically for processing images. The neurons of a CNN in one layer are connected to only a small region of the previous layer, called a *receptive field*. Thus CNNs can exploit the spatial locality by enforcing a local connectivity pattern between neurons of adjacent layers. By stacking a number of such layers, the receptive fields are gradually increased. This allows the network to first extract representations of small parts of the input, then assemble representations of larger areas [Karpathy, 2017; Karpathy]. In addition, each filter is replicated across the entire visual field. The replicated neurons share the same parameters, which dramatically reduces the total number of parameters to be learned, lowers the memory requirement and allows CNNs to have a better generalization ability. Moreover, weights sharing allows features to be extracted regardless of their positions in the visual field, which constitutes the property of *translation invariance*.

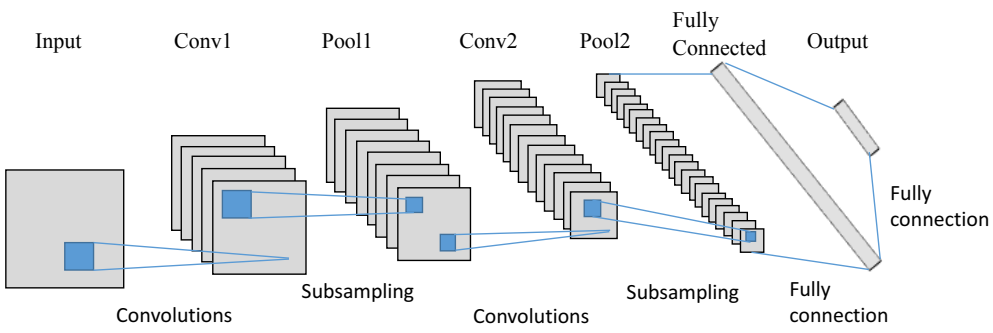


Figure 2.2: An example architecture of CNN, which consists of convolutional layers, pooling layers, fully connected layers and a loss layer.

A typical CNN is composed of convolutional layers, pooling layers, fully connected layers and a loss layer, as shown in Figure 2.2.

- **Convolutional layer**. The convolutional layer is the core building block of a CNN. The weights of convolutional layers are a set of learnable filters, which correspond to the receptive field. Each filter takes as an input the original image or previous feature maps $\mathbf{z}_i \in \mathbb{R}^{H \times W \times C}$, where H and W are the height and width, C is the depth of the input volume, and outputs new feature maps $\mathbf{z}_{i+1} \in \mathbb{R}^{H' \times W' \times C'}$ according to the following calculation:

$$\mathbf{z}_{i+1}^{c} = \mathrm{h}_i(\mathbf{w}_{ic} * \mathbf{z}_i + \mathbf{b}_{ic}), \forall c \in [1, \ldots, C'] \tag{2.2}$$

  where $\mathbf{w}_{ic}$ and $\mathbf{b}_{ic}$ denote the c-th filter kernel and bias respectively, $\mathbf{z}_{i+1}^{c}$ is the c-th output channel of $\mathbf{z}_{i+1}$, $*$ means the convolution operator. h_i is an element-wise non-linear activation function, which is most often a Sigmoid $\mathrm{h}_i(\mathbf{z}) = (1 + e^{-\mathbf{z}})^{-1}$ or a Rectified Linear Unit (ReLU) $\mathrm{h}_i(\mathbf{z}) = \max\{0, \mathbf{z}\}$. We slide the filter across the width and height of the input volume and produce a 2-dimensional activation map that gives the responses of that filter at every spatial position. This convolutional operation shows two characteristics, i.e., *local connectivity* and *parameter sharing*. In other words, each neuron is only connected to a small part of neurons in the previous layer (filter kernel size), and neurons on the same channel of feature map are actually from the same filter. With these specificities, CNNs have far less parameters than MLP, which enables easier training and faster convergence. It shows that the filters learnt in convolutional layers capture feature information such as edge, texture on lower layers, and entire patterns on higher layers [Zeiler and Fergus, 2014], which are essential for object recognition.

- **Pooling layer**. Pooling layers are periodically inserted between successive convolutional layers in CNNs. They are kinds of non-linear down-samplings.

The most commonly used ones are *average pooling*, which takes the average value of the pooling region as output, and *max pooling*, which takes the maximum value of the pooling region as output. The pooling operation is used to reduce the size of feature maps so as to cut down the size of network parameters and computations. Besides, it can pick up the most salient information in the pooling region, which provides another form of *translation invariance*.

- **Fully-connected layer**. Fully-connected layers in a CNN are the same as MLP. Neurons in a fully-connected layer have full connections to all activations in the previous layer. The output of a fully-connected layer is a matrix multiplication followed by a bias offset.

- **Loss layer**. Loss layer specifies the way of penalizing the error between the predicted output and the ground-truth. It is defined according to different tasks to be solved. For instance, Softmax loss and Sigmoid cross-entropy loss are used for classification, while Euclidean loss and Smooth L1 loss are used for regressing .

Some work tried to analyze and visualize CNNs, so as to better understand their internal mechanism, exploit the limitations and improve their performance. For example, in [Zeiler and Fergus, 2014], Zeiler and Fergus introduced a novel visualization technique called *Deconvolutional Network* that gave insights into the function of intermediate feature layers. It revealed many intuitively desirable properties such as compositionality, the increase of invariance and class discrimination with the adding of layers. Mahendran and Vedaldi [2015], by contrast, proposed to reconstruct an image from its CNN representation with an image prior. Their method can invert representations such as HOG more accurately than recent alternatives while being applicable to CNNs. It also showed that several layers in CNNs retained photographically accurate information about the image, with different degrees of geometric and photometric invariance. Similarly, Dosovitskiy and Brox [2016] studied image representations by inverting them with an up-convolutional neural network. The results

showed that the colors and the rough contours of an image can be reconstructed from activations in higher network layers and even from the predicted class probabilities. In addition, work in [Donahue et al., 2014; Azizpour et al., 2016] demonstrated that the earlier layers in a CNN contain more generic features (e.g., edge, contour) that could be useful to many other tasks, while the later layers become progressively more specific to the details of the classes contained in the original dataset. These work provides us ways to better understand CNNs, so as to build suitable model architectures for our task.

Recurrent Neural Networks Unlike aforementioned feed-forward NNs, Recurrent Neural Network (RNN) is another type of NN which involves connections between units and forms a directed cycle. This architecture provides a powerful mechanism to exhibit dynamic temporal behavior and exploit past contextual information, and makes RNNs applicable to tasks such as speech recognition, handwriting recognition or other sequential related problems.

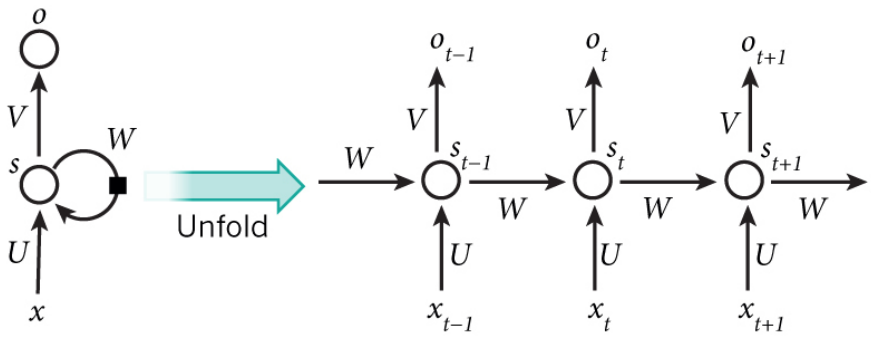


Figure 2.3: A typical architecture of RNN. The loop inside allows to exploit dynamic temporal information.

Figure 2.3 shows a typical architecture of RNN. It has a loop inside which allows information persisting. By unfolding the network, we simply see that the network captures temporal information. Given a sequence of input $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t\}$, we

have the output as:

$$\begin{aligned} \mathbf{h}_t &= \mathrm{g}(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h) \\ \mathbf{y}_t &= \mathbf{W}_{yh}\mathbf{h}_t + \mathbf{b}_y \end{aligned} \tag{2.3}$$

where $\mathbf{h}_t$ is known as the *hidden state* of current time step. It not only relates to current input $\mathbf{x}_t$, but also retains information from past inputs that are passed by $\mathbf{h}_{t-1}$. g is an elementwise non-linear function such as Sigmoid. $\mathbf{y}_t$ is the output vector at current time step.

Unfortunately, the range of contextual information that RNNs can maintain is quite limited, since the influence of a given input on network hidden states and then on the outputs would be vanished or exploded during cycling around the network recurrent connections [Hochreiter and Schmidhuber, 1997; Pascanu et al., 2013]. To solve this problem, a lot of variants have been proposed, where the more popular ones are Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU).

- **Long Short-Term Memory (LSTM)**

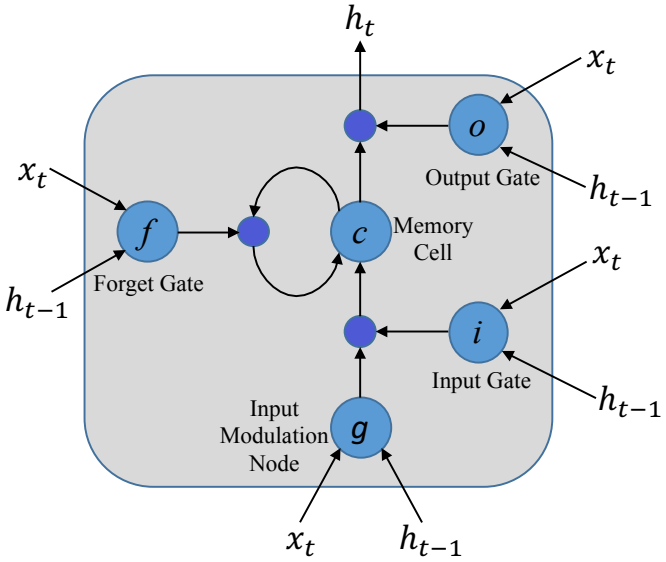


Figure 2.4: A memory block of LSTM, with a memory cell and three gates.

LSTM was introduced by Hochreiter and Schmidhuber [1997]. It contains some

memory blocks which can store information for a long period of time. Figure 2.4 depicts the structure of a typical LSTM memory block. It includes a memory cell with recurrent connection to itself, and three multiplicative gates (input gate, forget gate and output gate) to control the information flow. The gates allow the cell to store and access information over a long period of time. For example, as long as the input gate remains closed (i.e., the activation is close to zero), the activation of the cell will not be overwritten by the new inputs arriving in the network. Similarly, the cell's activation is only available to the rest of the network when the output gate is open. The cell's recurrent connection is switched on and off by the forget gate. The computation at time t can be mathematically written as follows:

$$\begin{aligned} g_t &= \phi(W_{gx}x_t + W_{gh}h_{t-1} + b_g) \\ i_t &= \sigma(W_{ix}x_t + W_{ih}h_{t-1} + b_i) \\ f_t &= \sigma(W_{fx}x_t + W_{fh}h_{t-1} + b_f) \\ o_t &= \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot \phi(c_t) \end{aligned} \tag{2.4}$$

where x_t is the input and h_t is the hidden state; i_t,f_t,o_t are the activations of the input gate, forget gate and output gate respectively; σ is the Sigmoid function; g_t is the input modulation node which takes activations from the rest of the hidden layer at previous time step h_{t-1} and current input x_t with a *tanh* function ϕ; c_t is the memory cell state, which is controlled by the forget gate f_t and input gate i_t, to trade off the influences between previous memory cell and current input; The current hidden state h_t is controlled by the output gate o_t, which decides how much of the cell state is transfered to hidden state; W is the weighting parameters of LSTM; $\odot$ represents an element-wise multiplication.

Compared to a vanilla RNN which overwrites its content at each time step, LSTM is able to decide whether to keep the current cell memory via the introduced gates. Intuitively, if LSTM detects an import information at an early stage, it can be retained in the cell memory for a long time, which enables LSTM to learn long-term dependencies.

- **Gated Recurrent Unit (GRU)**

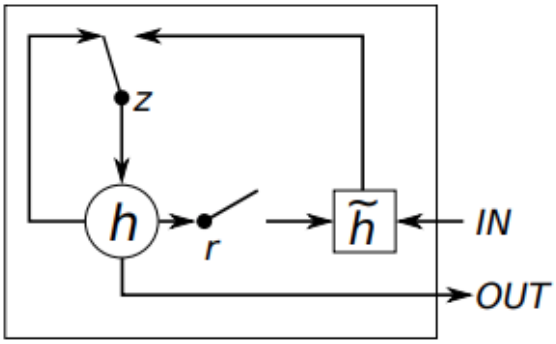


Figure 2.5: The structure of GRU [Chung et al., 2014]. r and z are the reset and update gates, h and $\widetilde{h}$ are the hidden state and the new candidate state.

GRU was introduced recently by Cho et al. [2014] and became increasingly popular. It is relatively simpler than LSTM. The forget gate and input gate are merged into a single update gate, and the output gate is replaced by a reset gate. Its structure is shown in Figure 2.5, and the mathematically representation is:

$$\begin{aligned} h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h}_t \\ z_t &= \sigma(W_{zx} x_t + W_{zh} h_{t-1} + b_z) \\ r_t &= \sigma(W_{rx} x_t + W_{rh} h_{t-1} + b_r) \\ \widetilde{h}_t &= \tanh(W_{hx} x_t + W_{hh}(r_t \odot h_{t-1}) + b_h) \end{aligned} \tag{2.5}$$

where the hidden state h_t is a linear interpolation between the previous hidden state h_{t-1} and the new candidate state $\widetilde{h}_t$, z_t is the update gate which decides how much the unit updates its activation, r_t denotes the reset gate. When the

reset gate is off (i.e., r_t is close to 0), it effectively makes the unit act as if it is only reading the input signal and forgets the previous state.

GRU has fewer parameters than LSTM and thus may train a bit faster or needs less data to generalize. In many tasks, both architectures can yield comparable performance [Chung et al., 2014; Józefowicz et al., 2015].

In [Hermans and Schrauwen, 2013], Michiel and Benjamin studied a character level language modeling and showed that RNNs are well-suitable for capturing temporal hierarchies. RNNs embed different timescales directly in the structure, and are able to model long-term dependencies. Karpathy et al. [2016] studied the performance improvements of RNNs in language modeling task compared to other finite-horizon models both qualitatively and quantitatively. Their experiments revealed the existence of interpretable cells that kept track of the long-range dependencies such as line lengths, quotes and brackets.

Another method of persisting long term memory is the use of *attention* mechanisms [Bahdanau et al., 2015; Chorowski et al., 2015]. Attention mechanisms select or weight the signals produced by a trained feature extraction at potentially all time steps in the input sequence. The basic idea is to let RNNs pick information to look at at every time step from a larger collection of information. For example, in the image caption task, the attention model will pick a part of the image to look at at each time step for every word outputting. Recent work usually integrates LSTM or GRU with attention models to boost the performance furthermore [Xu et al., 2015; Shi et al., 2016].

For many tasks, it is useful to have access to both past and future context. For instance, in text recognition, it would be helpful to recognize the current letter by knowing the context on both the left and the right. Bidirectional RNNs (BRNNs) were proposed [Schuster and Paliwal, 1997] , which are able to access context in both directions along the input sequence. BRNNs have two separated hidden layers, one of which processes the input sequence forward, while the other one processes it back-

ward. Both hidden layers are connected to the same output layer, providing it with access to both the past and the future context at every point in the sequence. BRNNs show better performance than standard RNNs in several sequence learning tasks, such as speech processing [Graves et al., 2013], handwriting recognition [Graves et al., 2009b], *etc.*. Combining BRNNs and LSTM gives BLSTMs, which will be used in our work.

2.1.2 Training Neural Networks

Backpropation is the most commonly used method to train NNs. For each example, we compute the prediction and its associated loss. The losses are summed up or averaged as the final error. Then we use the backpropagation algorithm to propagate the error from output layer to input layer and calculate the gradients to all learnable parameters in the network using the chain rule [Nielsen, 2015]. Once all the gradients are computed, the parameters are updated by a chosen optimization algorithm, such as *Stochastic Gradient Descent* (SGD), with a parameter known as *learning rate*. We then iterate the predication (i.e., forward pass), the backpropagation of errors (i.e., backward pass) and the optimization until the loss hopping to a local minimum low enough to ensure good predictions.

Though powerful, deep neural networks are difficult to train because of the huge number of parameters to be learnt. To overcome the overfitting problem and improve the generalization ability, many techniques are proposed. Here we briefly introduce some popular ones.

- **Dropout** [Srivastava et al., 2014]. Fully connected layers occupy most of the parameters and are prone to overfit. One method to reduce overfitting is Dropout. In the training stage, each neuron is either dropped out of the net with a probability $1 - p$ or kept with a probability p, so that a reduced network is left. In this sense, neurons are randomly dropped and the weights associated with them are also removed. At test time, no dropout is adopted and all

neurons are used for prediction.

- **Maxout** [Goodfellow et al., 2013]. The Maxout neuron picks the maximum value within a group of linear pieces as its activation. This nonlinearity is a generalization to the rectified nonlinearity and has the ability to approximate any form of activation functions. It is particular well suitable for training with Dropout. Dropout can be thought of as a form of model averaging. Maxout, nevertheless, can exploit this model averaging behavior much better, as the approximation is more accurate with Maxout units than with *tanh* units.

- **Batch Normalization** [Ioffe and Szegedy, 2015]. As proposed in [Ioffe and Szegedy, 2015], this method makes normalization as a part of the model architecture and performs normalization for each mini-batch. Batch normalization allows us to use much larger learning rates and be less careful about initialization. It will result in a faster learning and a higher overall accuracy.

2.1.3 NN based Applications

Deep neural network based methods have demonstrated impressive progress in many tasks in computer vision community, such as image classification, object detection, semantic segmentation, *etc.*. Here we review some fundamental and thrilling ones that relate to our tasks.

Image Classification Image classification is the task of assigning an input image one label selected from a set of categories. Despite its simplicity, this is one of the core problems in computer vision, as it is embedded in many other seemingly distinct tasks (e.g., object detection, segmentation). Although image classification is relatively trivial for human, it is not an easy work for a computer vision algorithm. The challenges include viewpoint variation, scale variation, deformation, occlusion, *etc.*. A good image classification model is expected to be invariant to all these variations, while retain sensitivity to inter-class variations.

As a breakthrough in computer vision, Krizhevsky et al. [2012] proposed AlexNet

to classify 1.2 million images in ILSVRC 2010 dataset into 1000 different classes. They made use of techniques like non-saturating neurons and Dropout, and achieved Top-1 and Top-5 test error rates of 37.5% and 17.0%, which are about 8% higher than the previous best results. Having more than a million images provides opportunities for novel algorithms and model innovations. Some popular CNN models, such as VGGNet [Simonyan and Zisserman, 2015], GoogleNet [Szegedy et al., 2015], and ResNet [He et al., 2016a] were then proposed one by one and demonstrated superior performance in image classification. The rich features learnt through ImageNet data can also be used as supervised pre-training for other tasks. For instance, Zeiler and Fergus [2014] showed that the ImageNet trained model can generalize well to other datasets, such as Caltech-101 [Fei-Fei et al., 2006] and Caltech-256 [Griffin et al., 2006]. In addition, Huang *et al.* [Huang et al., 2016] proposed densely connected convolutional networks which connected each layer to every other layer in a feed-forward fashion. With this design, they alleviated the vanishing-gradient problem, strengthened feature propagation, encouraged feature reuse, and substantially reduced the number of parameters. The model achieved state-of-the-art results on several benchmarks.

Image classification techniques are quite essential to our text recognition task, since a good text recognition model is based on a strong character classifier. The improvement on CNN architecture design and CNN feature understanding give us a great inspiration on our model design and training.

Object Detection Object detection is another fundamental task in computer vision community. Before the era of CNNs, the best performance model is obtained by building ensemble systems based on multiple low-level features, e.g., [Everingham et al., 2010; Felzenszwalb et al., 2010].

A breakthrough work is introduced by R-CNN [Girshick et al., 2014], which improves mean Average Precision (mAP) by more than 30% relative to the previous best result on PASCAL VOC 2012 dataset, thanks to the applying of high capacity CNNs.

Selective search is employed firstly to generate category-independent region proposals. Then CNN features are extracted from each region for object classification. The work also shows that when labeled training data is scarce, supervised pre-training on an auxiliary task, followed by domain-specific fine-tuning, will yield a significant performance boost. However, in R-CNN, the proposed regions need to be resized to a fixed size so as to be processed by CNNs and obtain fixed length feature vectors for classification. SPP-net [He et al., 2014] is then proposed which eliminates this requirement. It is a spatial pyramid pooling method that can generate a fixed-length representation for each region regardless of its size/scale. Using SPP-net, people only need to calculate the feature maps from the entire image once, and then pool features in arbitrary regions to generate fixed-length representations. This method avoids repeatedly computing the convolutional features, while achieves comparable detection results as R-CNN. The following improvement work are Fast R-CNN [Girshick, 2015] and Faster R-CNN [Ren et al., 2015], which not only improve training and test speed, but also increase detection accuracy. Fast R-CNN takes as input an entire image and a set of object proposals. Region of Interest (RoI) pooling layer is proposed to extract a fixed-length feature vector for each object proposal. The training is a single stage process, using a multi-task loss, and the parameters are updated through all network layers. Fast R-CNN can train a very deep detection network 9 faster than R-CNN and 3 faster than SPP-net, and achieves an mAP on PASCAL VOC 2012 of 66% (vs. 62% for R-CNN). Furthermore, to overcome the bottleneck of separate region proposal computation in R-CNN, SPP-net and Fast R-CNN, Faster R-CNN introduces a Region Proposal Network (RPN) which shares the full-image convolutional features with the detection network, and thus enables nearly cost-free region proposals. It is 10 times faster than Fast R-CNN. RPN can be trained end-to-end and results in high-quality region proposals. With only 300 proposals, Faster R-CNN achieves an mAP of 67% on PASCAL VOC 2012, compared with 2000 proposals used in Fast R-CNN. Our end-to-end text detection and recognition method

is inspired by Faster R-CNN.

Instead of adopting object proposals, another research methodology is to regress object bounding boxes directly from images, with a single shot detection model, like YOLO [Redmon et al., 2016], SSD [Liu et al., 2016]. YOLO re-frames object detection as a single regression problem. A single neural network is employed to predict bounding boxes and class probabilities directly from full images in one evaluation. The base YOLO model can process images in real-time at 45 frames per second (fps), while Faster R-CNN has a frame rate of 7fps. However, YOLO makes more localization errors. SSD makes improvement on YOLO. It combines predictions from multiple level feature maps with different resolutions to naturally handle objects of various sizes. The network can also be trained end-to-end and leads to a higher accuracy. It achieves 59fps with mAP 74.3% on VOC 2007, vs. Faster R-CNN 7fps with mAP 73.2% or YOLO 45fps with mAP 63.4%.

Object detection is more related to our text detection task, since words can be regarded as a special kind of objects, with discriminative textual information from the background. Hence, the development on object detection provides us with reference on text detection framework.

Vision-to-language Vision-to-language problems, such as image captioning [Vinyals et al., 2015; Xu et al., 2015], Visual Question Answering (VQA) [Antol et al., 2015; Lu et al., 2016] attract much attention in computer vision community in recent years. They take a further step to understand the input image by generating a sequence of words in the form of sentences. RNNs are usually adopted here to address these tasks, since they are a kind of sequence prediction problems and RNNs are good at processing sequential problems. For example, Vinyals et al. [2015] proposed a generative model to generate natural sentences describing an image. It is an end-to-end deep neural network consisting of a vision CNN followed by a language generating RNN. CNN feature of an image is fed to RNN at the initial time step as a holistic feature. RNN generates a word at each time step based on previously generated words

and the holistic image feature. The model is trained to maximize the likelihood of the target description sentence given the training image. Xu et al. [2015] introduced attention mechanisms in RNN decoders, so that the model can automatically learn to fix its gaze on salient objects while generating the corresponding words in the output sequence. Johnson et al. [2016] introduced a dense captioning task, which required a computer vision system to both localize and describe salient regions in images by natural language. A fully Convolutional Localization Network (FCLN) was proposed that processes an image with a single forward pass, requiring no external regions proposals, and can be trained end-to-end with a single round of optimization.

Proposed by [Antol et al., 2015], VQA aims to provide an accurate natural language answer, when given an image and a natural language question about the image. In [Antol et al., 2015], the authors developed a 2-channel vision (image) + language (question) model to address this problem, where a two-layer LSTM was used to encode the questions and the last FC layer of VGGNet was used to encode the images. Both the question and image features were fused via element-wise multiplication. The results were then passed through a fully connected layer followed by a Softmax layer to obtain a distribution over answers. Shih et al. [2016] presented an image-region selection mechanism that learned to select image regions relevant to questions. It is focus on learning "where to look". In addition, in [Lu et al., 2016], a co-attention model was proposed that jointly reasons about image and question attentions. Besides "where to look" or visual attention, it also models "what words to listen to" or question attention.

Word is also a kind of language, with a series of characters and a language model embedded. The techniques used to deal with vision-to-language tasks are worth referring to when tackling scene text recognition, such as the attention mechanism, RNN decoding process, *etc.*. The attention mechanism can help us focus on "the character to be recognized", while RNN decoding process can be adopted to generate the character sequence. The framework of dense captioning also gives us a great

inspiration on our end-to-end scene text detection and recognition task. We can regard the text region as a special object. A similar framework can be employed to localize text regions and describe the regions simultaneously.

Other Related Applications In addition to the aforementioned applications, deep neural networks are also used in other tasks such as multi-label classification [Wang et al., 2016], speech recognition [Graves et al., 2006], handwriting recognition [Graves et al., 2009a], instance level segmentation [He et al., 2017a], human pose estimation [Poirson et al.], *etc.*, with a lot of interesting designs and promising results. For instance, Wang et al. [2016] showed that the label co-occurrence information can be modeled by LSTM in multi-label classification. Graves et al. [2006] referred to the task of speech recognition as a temporal classification problem, and used RNNs for Connectionist Temporal Classification (CTC). The basic idea is to interpret the network outputs as a probability distribution over all possible labels, and the objective function is defined as to maximize the probabilities of the correct labellings. Since the objective function is differentiable, the network can then be trained with standard backpropagation through time. RNN and CTC were then adopted in handwriting recognition and produced much better results than previous HMM-based method [Graves et al., 2009a]. The advantage of this design is that it avoids the hard task of sequence segmentation and contains long-range interdependencies that would benefit the recognition. This idea can also be applied to scene text recognition. He *et al.* [He et al., 2017a] proposed a Mask R-CNN framework that efficiently detects objects in an image while simultaneously generates a high-quality segmentation mask for each instance. They demonstrated that the multi-task loss can benefit both detection and segmentation results. Poirson *et al.* [Poirson et al.] extended SSD and proposed a unified deep neural network for simultaneously object detection and pose estimation. These frameworks and experimental designs are also worth referring to when addressing the end-to-end text detection and recognition task.

2.2 Related Work on Car License Plate Detection and Recognition

As a special case of text in natural scene, car license plates have their own characteristics, such as the borderlines, the unique color, *etc.*. A lot of methods have been proposed exclusively for car license plate detection and recognition. In this section, we give a brief overview about previous work on license plate detection and recognition respectively.

2.2.1 Related Work on License Plate Detection

License plate detection aims to localize the license plates in an image in the form of bounding boxes. Existing algorithms can be roughly classified into four categories [Du et al., 2013; Zhou et al., 2012; Anagnostopoulos et al., 2006a]: edge-based, color-based, texture-based, and character-based.

Edge-based approaches try to find regions with higher edge density than elsewhere in the image as license plates. Considering the property that the brightness change in license plate region is more remarkable and more frequent than elsewhere, Tan et al. [2013] used an edge detector and some morphological operations to find candidate license plates. In [Hsu et al., 2013], a Sobel detector was employed firstly to extract vertical edges in the input image. Expectation Maximization (EM) was applied for edge clustering which extracts the regions with dense sets of edges and with shapes similar to plates as the candidate license plates. In [Chen and Luo, 2012], a license plate localization method based on an improved Prewitt arithmetic operator was proposed. The exact location was then determined by horizontal and vertical projections. Edge-based methods are fast in computation, but they cannot be applied to complex images as they are too sensitive to unwanted edges. Furthermore, it is difficult to find license plates with edge information if they are blurry.

Color-based approaches are based on the observation that color of the license

plate is usually distinct from that of the background. In [Ashtari et al., 2014], a plate detection method was developed by analyzing the target color pixels. A color-geometric template was utilized to localize Iranian license plates via strip search. HSI color model was adopted in [Deb and Jo, 2008] to detect candidate license plate regions, which were later verified by position histogram. Jia et al. [2007] firstly segmented the image into different regions according to their colors via mean-shift algorithm. License plates were then distinguished based on features including rectangularity, aspect ratio and edge density. Color-based methods can be used to detect inclined or deformed license plates. However, they cannot distinguish other objects in the image with similar color and size as the license plates. Moreover, they are very sensitive to various illumination changes in natural scene images.

Texture-based approaches attempt to detect license plates according to the uneven pixel intensity distribution in plate regions. For example, Zhang et al. [2006] proposed a license plate detection method using both global statistical features and local Haar-like features. Classifiers based on global features can exclude more than 70% of the background area, while classifiers based on local features are robust to brightness, color, size and position of license plates. In [Anagnostopoulos et al., 2006b; Giannoukos et al., 2010], Sliding Concentric Window (SCW) algorithm was developed to identify license plates based on the local irregularity property of license plate images. Operator Context Scanning (OCS) algorithm was proposed in [Giannoukos et al., 2010] to accelerate the detection speed. In [Yu et al., 2015], wavelet transform was applied to get the horizontal and vertical details of an image. Empirical Mode Decomposition (EMD) analysis was then employed to deal with the projection data and locate the desired wave crest which indicates the position of a license plate. Texture-based methods use more discriminative characteristics than edge or color, but result in a higher computational complexity.

Character-based approaches are based on the fact that license plates are composed of a string of characters, which have more specific information compared to

the background. In [Ho et al., 2009], a two-stage method was presented for license plate detection. An Adaboost classifier was used across the gray-scale images via window scanning for character detection in the first stage. An SVM based filter was utilized in the second stage to remove false positives. Lin et al. [2010] detected license plates based on image saliency. This method firstly segmented out characters in the image with a high recall using an intensity saliency map, then applied a sliding window on these characters to compute some saliency related features and detect license plates. Zhou et al. [2012] formulated license plate detection as a visual matching problem. Principal Visual Word (PVW) was generated for each character which contained geometric clues such as orientation, characteristic scale and relative position, and used for plate extraction. Li et al. [2013] applied MSER at beginning to extract candidate characters in images. A CRF model was then constructed to represent the relationship among license plate characters. License plates were finally localized through the belief propagation inference on CRF. Llorca et al. [2016] made use of a combination of the MSER and Stroke Width Transform (SWT) to detect isolate character regions. The license plates were finally bordered using the probabilistic Hough transform. Character-based methods are more reliable and can lead to a high recall. However, the performance is easy to be affected by the general text in the image background.

Our stepwise method on license plate detection is a character-based approach. We use CNNs to distinguish characters from cluttered background. The strong classification capability of CNNs guarantees the character detection performance. To distinguish license plates from general text in the image, another plate/non-plate CNN classifier is designed, which eliminates those hard false positives effectively.

2.2.2 Related Work on License Plate Recognition

Previous work on license plate recognition typically needs to segment characters in the license plate firstly, and then recognizes each segmented character using OCR

techniques.

Existing approaches on license plate segmentation can mainly be divided into two categories: projection-based and Connected Component (CC)-based. Since characters and background have obviously different colors in a license plate, in theory, they should have opposite binary values in the binarized image. **Projection-based approaches** exploit the histograms of vertical and horizontal binary pixel projections for character segmentation [Guo and Liu, 2008; Qiao et al., 2010]. The license plate image is binarized firstly, projected horizontally to determine the top and bottom boundaries of the characters, and then vertically to separate each character. This method can be easily influenced by the rotation of license plate. **CC-based methods** label connected pixels in the binary license plate into components based on 4 or 8 neighborhood connectivity [Anagnostopoulos et al., 2006b; Giannoukos et al., 2010; Jiao et al., 2009]. In [Gou et al., 2016], Extremal Regions (ER) was employed to segment characters from coarsely detected license plates and to refine plate location. In [Hsu et al., 2013], MSER was adopted for character segmentation. Hou et al. [2015] proposed to segment license plate character based on SWT, which can process rotated license plates. CC-based methods can extract characters from rotated license plate, but cannot segment characters correctly if they are joined together or broken apart. Zheng et al. [2013] combined both methods to enhance the segmentation accuracy. It is worth noting that both methods are implemented on binary images. Hence the binarization result has a significant influence on the segmentation performance. Neither of them can result in good segmentation if background pixels are wrongly classified as foreground pixels.

License plate recognition can be sorted as a kind of image classification problem, where the segmented characters need to be categorized according to the characters used in that country. Existing algorithms on character classification include template matching based and learning based methods.

Template matching based methods recognize each character by measuring the

similarity between characters and templates. The most similar template is regarded as the target [Rasheed et al., 2012]. Several similarity measuring methods are proposed, like Mahalanobis distance, Hausdorff distance, Hamming distance, *etc.* [Du et al., 2013]. Template matching methods are simple and straightforward, but they can only be used to recognize characters of single font, fixed size, no rotation and broken.

Learning based methods are more robust. They learn information from more discriminative features, such as image density [Jiao et al., 2009; Giannoukos et al., 2010], LBP [Hsu et al., 2013], *etc.*. They can deal with characters of various fonts, illuminations, or rotations. The common used learning methods include SVMs [Wen et al., 2011], NNs [Jiao et al., 2009; Giannoukos et al., 2010], HMM [Llorens et al., 2005], and so on. Some researchers even integrate multiple features [Wen et al., 2011], or combine multiple classifiers [Sharma et al., 2014] to improve the recognition accuracy.

Our method on license plate recognition is a learning based method, where CNNs and RNNs are trained for character string recognition. It is worth noting that our plate recognition method is segmentation free. We treat the characters in license plate as an unsegmented sequence, and solve the problem from the viewpoint of sequence labeling. By avoiding the challenging task of character segmentation, our method can achieve promising results.

2.3 Related Work on Text Spotting in Natural Scene Images

This section focuses on a review of related work specially designed for text spotting in natural scene images. Text spotting essentially includes two tasks: text detection and word recognition. In this section, we present a range of different methods that solely focus on text detection or word recognition. The existing work on end-to-end text spotting has already been introduced before in Chapter 1.1.3. Comprehensive surveys for text detection and recognition can also refer to [Ye and Doermann, 2015;

Zhu et al., 2016].

2.3.1 Related work on Text Detection

Text detection aims to localize text in images and generate bounding boxes for words. Existing approaches can be roughly classified into three categories: character based, text-line based and word based methods.

Character based methods firstly find characters in images, and then group them into words. They can be further divided into sliding window based [Wang et al., 2011; Jaderberg et al., 2014c; Tian et al., 2015; Zhu and Zanibbi, 2016] and Connected Components (CC) based [Huang et al., 2013; Neumann and Matas, 2013b; Busta et al., 2015] methods. Sliding window based approaches use a trained character classifier to detect characters across the image in a multi-scale sliding window fashion. The positive ones are then grouped together into text regions with morphological operations, CRF or other graph methods. For instance, Kim et al. [2003] adopted SVMs in a sliding window manner to classify the pixel located at the center of the window into text or non-text, via analyzing its textural properties, and then applied a continuously adaptive mean shift algorithm to the texture classification results to obtain text chips. In [Tian et al., 2015], a sliding window based cascade boosting approach was proposed for character candidate detection, with six simple hand-craft features adopted to accelerate the feature extraction process. A novel minimum cost (min-cost) flow network model was designed that integrated text line extraction and word bounding box generation into a single process, so as to eliminate error accumulation. Sliding window based methods are simple and easy to implement. Nevertheless, they are computationally expensive as a large number of windows need to be classified. CC based methods first segment pixels with consistent region properties (i.e., color, stroke width, density, *etc.*) into candidate components, and then filter out non-text components using heuristically designed rules or well trained classifiers. For example, Huang et al. [2013] proposed a low-

level filter called the Stroke Feature Transform (SFT). It extended the widely-used Stroke Width Transform (SWT) by incorporating color cues of text pixels, leading to enhanced performance on inter-component separation and intra-component connection. A component-level and a text-line-level Text Covariance Descriptors (TCD) were proposed and used to build classifiers, instead of the commonly-used heuristic filtering methods for robust component and text-line classification. Busta et al. [2015] proposed an easy-to-implement stroke detector based on an efficient pixel intensity comparison to surrounding pixels. Stroke-specific keypoints were efficiently detected and text fragments were subsequently extracted by local thresholding guided by keypoint properties. The stroke-specific keypoints can detect 25% more characters than the commonly used MSER. The method is scale- and rotation- invariant and supports a wide variety of scripts (Latin, Hebrew, Chinese, *etc.*) and fonts. He et al. [2016c] developed a Contrast Enhancement Maximally Stable Extremal Regions (CE-MSERs) to generate text components in the input image. A novel Text-Attentional Convolutional Neural Network (Text-CNN) was proposed to filter out the non-text components. Text-CNN particularly focused on extracting text-related regions and features from the image components, which led to a strong capability for discriminating ambiguous texts and robustness against complicated background components. CC based methods are more efficient, but can be affected easily by noise in the image, such as blur, low resolution and non-uniform illumination. Some researchers combine both two methods in a pipeline to enhance the detection performance [Huang et al., 2014; Zhu and Zanibbi, 2016].

Text-line based methods detect text lines firstly and then separate each line into multiple words. The motivation is that people usually distinguish text regions intuitively even if characters are not recognized. Based on the observation that a text region usually exhibits high self-similarity to itself and strong contrast to its local background, Zhang et al. [2015] proposed to extract text lines by exploiting the symmetry property. Symmetry feature and appearance feature were designed to capture

the intrinsic properties of text. Zhang et al. [2016] trained a Fully Convolutional Network (FCN) model to predict the salient map of text regions in an image. The salient map provided a powerful guidance for estimating orientations and generating candidate bounding boxes of text lines. The text line candidates were then classified by two criteria based on character centroids of the text line calculated by another FCN model. Tian et al. [2016] proposed a Connectionist Text Proposal Network (CTPN) that detected text lines in a sequence of fine-scale text proposals directly in convolutional feature maps. A vertical anchor mechanism was developed that jointly predicted location and text/non-text score of each fixed-width proposal. The sequential proposals were finally connected as text lines by RNNs. The whole model can be trained end-to-end and achieved a high detection performance.

More recently, a number of approaches are proposed to detect **words** directly in the images, inspired by recent development on object detection such as Faster R-CNN [Ren et al., 2015], YOLO [Redmon et al., 2016] and SSD [Liu et al., 2016]. By extending Faster R-CNN, Zhong et al. [2016] designed an inception region proposal network (Inception-RPN) which applied multi-scale sliding windows over convolutional feature maps to retain local information as well as contextual information, and achieved a high recall. A powerful text detection network that embedded Ambiguous Text Category (ATC) information and Multi-Level Region-of-interest Pooling (MLRP) was designed for text and non-text classification and accurate localization. Ma et al. [2017] presented Rotation Region Proposal Networks (RRPN), which can generate inclined proposals with text orientation angle information. Rotation Region-of-Interest (RRoI) pooling layer was proposed to project arbitrary-oriented proposals to the feature map for text region classification. The network can detect arbitrary-oriented text with both high effectiveness and efficiency. Lyu et al. [2018] proposed to detect scene text by localizing corner points of word bounding boxes and segmenting text regions in relative positions. The method can handle long oriented text and does not need complex post processing. In addition, Yuliang et al. [2017] proposed a polygon based

curve text detector which can directly detect curve text without empirical combination. An ingenious method, namely transverse and longitudinal offset connection, was presented which uses RNN to learn the inherent connection between locating points, and results in more accurate and smooth detection. Motivated by YOLO, Gupta et al. [2016] adopted a Fully-Convolutional Regression Network (FCRN) for efficient text detection and bounding box regression at all locations and multiple scales in an image. Besides, a fast and scalable engine was designed to generate synthetic images with text in clutter. It can alleviate the requirement of large annotated dataset, and make the design and training of different deep neural networks possible. Similar to SSD, Liao et al. [2017] proposed "TextBoxes" by combining predictions from multiple feature maps with different resolutions. It can detect scene text with both high accuracy and efficiency in a single network forward pass, involving no post-process except for a standard NMS. This method can only detect horizontal texts. He et al. [2017b] also presented a single-shot text detector based on SSD framework. A text attention module was introduced which was built upon the aggregated inception convolutional features, and can significantly suppress background interference. The orientation angle can also be regressed via this framework.

Our first method on text detection is a character based method using a sliding window manner. The detected characters are then grouped together into text lines for further separation and recognition. Our second method is inspired by Faster R-CNN, which is a word based approach. In fact, it integrates both text detection and recognition into one network, so that words are localized and recognized all at once, without word separation or image cropping.

2.3.2 Related Work on Text Recognition

Text recognition aims to recognize the character sequence from the cropped word image. Existing methods on text recognition can be roughly divided into segmentation-needed and segmentation-free ones.

Traditional approaches on text recognition usually need to separate a word into characters firstly, recognize individual characters, and then integrate them together into word by means of beam search [Bissacco et al., 2013], dynamic programming [Jaderberg et al., 2014c], *etc.*. For instance, Mishra et al. [2012] proposed a framework that exploited both bottom-up and top-down cues for word recognition. A sliding window based classifier was adopted to get the local maximum character detections. CRF model was built on these detections to jointly model the strength of the detections and the interactions between them. Top-down language model cues were imposed during the optimization. Novikova et al. [2012] adopted MSER as the primitives in the probabilistic model. Word recognition was performed by estimating the Maximum A Posteriori (MAP) solution under the joint posterior distribution of character appearance and language model, where MAP inference was performed by Weighted Finite-State Transducers (WFSTs).

However, character segmentation by itself is a really challenging task, which is easy to be influenced by the uneven illumination, blur, *etc.* in the image. A bad segmentation or separation result will directly affect the following recognition, even if the imposed top-level language model may make some correction. With the development of CNNs and RNNs, recent methods on word recognition are mostly segmentation free. For example, Jaderberg et al. [2014a] considered word recognition as a multi-class classification problem, and categorized each word over a large dictionary (about 90K words, i.e., class labels) using a deep CNN. With the success of RNNs on handwriting recognition [Graves et al., 2006], He et al. [2016b] and Shi et al. [2015] regarded word recognition as a sequence labeling problem. RNNs were employed to generate sequential probabilities of arbitrary length without character segmentation, and CTC was adopted to decode the sequence. The difference is that the model in [Shi et al., 2015] (called "CRNN") can be trained end-to-end for both CNNs and RNNs, while He et al. [2016b] extracted convolutional features using a pre-trained CNN model in a sliding window manner. Inspired by the sequence to

sequence learning framework on machine translation [Sutskever et al., 2014; Kim et al., 2016], Lee and Osindero [2016] and Shi et al. [2016] proposed to recognize text using an attention-based sequence-to-sequence learning structure. In this manner, RNNs can automatically learn the character-level language model hidden in the word strings from the training data. The soft-attention mechanism allows the model to selectively exploit local image features during recognition. These networks can be trained end-to-end with cropped word image patches as input. Moreover, Shi et al. [2016] introduced a Spatial Transformer Network (STN) to handle words with irregular shapes.

Our first method on text recognition adopts the sequence labeling framework, where CTC is extended for both text line separation and word recognition. In order to exploit the top-level language model clue, our second method integrates the attention-based sequence-to-sequence learning structure, which, to some extend, is expected to recognize words even with inaccurate bounding boxes.

Chapter 3

A Stepwise Method for Text Detection and Recognition

3.1 Introduction

Text in natural scene images contains rich semantic information and is of great value for image understanding. As an important task in image analysis, scene text spotting, including both text detection and word recognition, attracts much attention in computer vision field. Car license plates can be regarded as a special kind of scene text, as they are both composed of characters and appear in natural scenes. Automatic car license plate detection and recognition is an important part of the intelligent transportation system and can be applied widely in traffic management.

Due to the large variability of text patterns and the highly complicated background, text spotting in natural scene images is much more challenging than from scanned documents. In this chapter we present a stepwise method for text detection and recognition in natural scene images, leveraging the high capability of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). CNNs consist of multiple layers of neurons with some essential merits as we described in Chapter 2, which can learn discriminate features efficiently from a large amount of labeled training data, while RNNs have a powerful mechanism to exploit past contextual information, which would benefit sequence recognition.

We start from car license plate detection and recognition, then we extend the

method to general scene text spotting. For both tasks, we treat characters as the atomic building block. A sliding window based character detection is performed firstly, based on a strong CNN classifier. The detected characters are then grouped into license plates or text lines for further recognition. Some specific ideas are designed for each task, considering their own characteristics. Experiments are conducted to verify the effectiveness of the proposed method.

3.2 A Stepwise Method for License Plate Detection and Recognition

A number of work has been done on the topic of car License Plate Detection and Recognition (LPDR) over the past two decades, and some of them have demonstrated success in certain specific tasks. However, most of the existing algorithms work well either under controlled conditions or with sophisticated image capture systems. It is still a challenging task to read license plates accurately in an open environment. The difficulty lies in the extreme diversity of character patterns, such as different fonts, distortion, occlusion or blurring, and the highly complicated backgrounds, like the general text in shop boards, windows, guardrails or bricks.

Previous work on LPDR usually relies on some handcrafted image features that capture certain morphological, color or textural attributes of the license plate, as we introduced before in Chapter 2.2. These features can be sensitive to image noises, and result in many false alarms under complex backgrounds. In this section, we tackle LPDR based on the powerful deep neural networks via a stepwise methodology. We regard a license plate as a string of characters. A character CNN is applied firstly to examine the presence of characters, and a plate/non-plate CNN is followed then to eliminate false alarms. This cascade framework shows high discriminative ability and strong robustness against the complicated background. As to license plate recognition, we formulate it as a sequence labeling problem. The license plate

image is viewed as an unsegmented sequence. CNNs are used to extract image features. RNNs with connectionist temporal classification (CTC) output layer [Graves et al., 2009a] are adopted to label the sequential data. With this method, we do not need to deal with the challenging character segmentation task. The recurrent property of RNNs also helps to exploit the contextual information and improve the recognition performance. The overall framework of the stepwise LPDR system is shown in Figure 3.1.

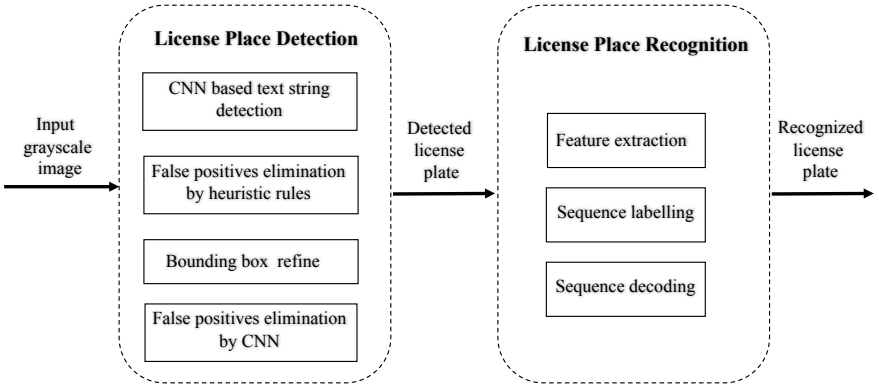


Figure 3.1: Our stepwise framework for car license plate detection and recognition.

The main contributions of this work are as follows.

- We propose a cascade framework that uses different CNN classifiers for different purposes. To begin with, a 4-layer 37-class (10 digits, 26 uppercase letters plus a negative non-character category) CNN classifier (we denote it as CNN-I) is employed in a sliding-window fashion across the entire image to detect the presence of character and generate a text saliency map. Text-like regions are extracted based on the clustering nature of characters. Then another plate/non-plate CNN classifier (denoted as CNN-II) is adopted to reject false positives and distinguish license plates from general text. With this framework, our system can detect license plates in complicated backgrounds with both high recall and precision. Moreover, it can be used to detect license plates of various styles (eg from different countries), regardless of diverse plate colors, fonts or sizes.

- we develop a deep recurrent model which can read all characters in the license plate one-off. To the best of our knowledge, this is the first work that recognizes license plates without character segmentation. We extract features from the whole license plate without pre-segmentation by CNN. Several layers of CNN features are concatenated together, which combine both local and global information. Bidirectional Long Short-Term Memories (BLSTMs) are employed to recognize the feature sequence. CTC is applied to the output of BLSTMs to decode the character string in the plate. This approach takes advantage of both deep CNNs for feature learning and BLSTMs for contextual information exploiting, and results in appealing performance.

In the following we will describe the details on license plate detection and recognition separately. Experimental verifications are followed to demonstrate the effectiveness of our method.

3.2.1 Car License Plate Detection

License plate detection is the first stage of LPDR pipeline. Ideally it is required to generate plate bounding boxes with both high recall and high precision. In this work, we take advantage of the highly discriminative ability of CNNs and perform a character-based plate detection using a multiple scales sliding window manner. Firstly, we train a character CNN classifier based on samples from general text. One benefit that comes along with this is that the trained classifier can be used to detect plates from various countries, as long as the plates are composed of digitals and upper-case letters. In order to distinguish license plate from other text or text-like outliers appeared in the image, another plate/non-plate CNN classifier is employ to remove false positives. We also refine the bounding boxes based on projection based method to improve the overlap ratio. The overall detection process is illustrated in Figure 3.2.

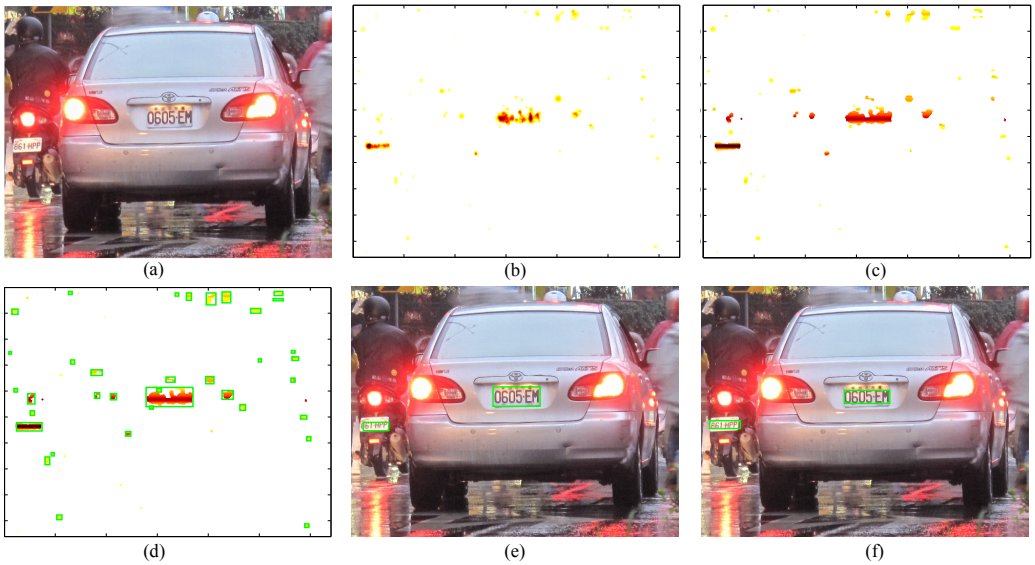


Figure 3.2: License plate detection procedure in a single scale. (a) Input image. (b) Text salience map generated after sliding window based detection. (c) Text salience map after NMS and RLSA. (d) Candidate bounding boxes (green rectangles) generated by CCA. (e) Candidate bounding boxes (green rectangles) after false positive elimination. (f) Final bounding boxes (green rectangles) after box refining and plate/non-plate classification.

3.2.1.1 Candidate Detection Generation

In order to accelerate the detection process, we train a 4-layer CNN to classify text from background. Due to the convolutional structure of CNN, we compute the character saliency map by running CNN classifier across the entire image in one go, instead of calculating on each cropped sliding window, which saves processing time largely.

The configuration of 4-layer character CNN model (CNN-I) is shown in Table 3.1. ReLUs are followed after each convolutional layer. Here we train a 37-class CNN classifier for 26 upper-case letters, 10 digitals and a non-character class, instead of a binary text/non-text classifier. Patches classified as either letters or digitals are all regarded as characters later. In this way the features learned for each class are more specific and discriminative, which will lead to a better detection result with a higher recall.

Table 3.1: Configuration of the 4-layer Character CNN model (CNN-I). "k", "s" and "p" represent kernel size, stride and padding size respectively.

Layer Type	**Parameters**
Soft-max	37 classes
Fully connected	#neurons: 37
Dropout	Prop: 0.5
Fully connected	#neurons: 512
Maxpooling	k:2 $\times$ 2, s:2
Convolution	#filters:384, k:2 $\times$ 2, s:1, p:0
Maxpooling	k:4 $\times$ 4, s:4
Convolution	#filters:120, k:5 $\times$ 5, s:1, p:0
Input	24 $\times$ 24 pixels gray-scale image

For license plate detection, the first phase is to generate candidate license plate bounding boxes with a high recall. Given an input image, we resize it into 12 different scales, and calculate the character saliency map at each scale by evaluating the well-trained CNN classifier in a sliding window fashion across the image. After getting these saliency maps, the candidate bounding boxes are generated independently at each scale by using the Run Length Smoothing Algorithm (RLSA) [Jaderberg et al., 2014c] and Connected Component Analysis (CCA). In detail, for each row in the saliency map, we do Non-Maximal Suppression (NMS) at first to remove detection noise. NMS response for the pixel located at row r column c with classification probability $P(c,r)$ is defined as follows:

$$\hat{P}(c,r) = \begin{cases} P(c,r) & if\ P(c,r) \geq P(c',r), \forall c', \|c'-c\| < \delta \\ 0 & Otherwise \end{cases} \tag{3.1}$$

where δ defines a width threshold. Then we calculate the mean and standard deviation of the spacings between probability peaks. Neighboring pixels are connected together if the spacing between them is less than a threshold. CCA is applied subsequently to produce the initial candidate boxes. The process is shown in Figure 3.2 (a)-(d).

(a) (b)

Figure 3.3: Generated bounding boxes before and after bounding box refining. The top line shows the initial bounding boxes, which include extra background or incomplete letter, while the bottom line shows the results after refining.

3.2.1.2 False Positives Elimination and Bounding Box Refining

The produced bounding boxes are firstly filtered based on some geometric constraints (boxes length, height, aspect ratio, *etc.*). Then we score each box by averaging the character confidence scores within it. Boxes whose scores are less than the average value are eliminated. NMS is employed again on the bounding box level to remove the ones that overlapped by more than 50% with another candidate box of a higher score.

We find that some bounding boxes are too big or too small, as shown in the first line of Figure 3.3, which will affect the following plate verification and recognition. For example, the textual-like background contained in the bounding box in Figure 3.3 (a) will impact the following plate verification. The bounding box in Figure 3.3 (b) that does not contain the whole license plate will definitely lead to an incorrect recognition result. Therefore, a process for refining bounding boxes is performed according to the edge feature of license plate [Zheng et al., 2013]. In detail, for each detected bounding box, we enlarge the box with 15% on each side. Considering the strong connectivity of characters in vertical direction than in horizontal direction, we perform vertical edge detection on the cropped license plate images using Sobel operator. When we get the vertical edge map, a horizontal projection is performed to find the top and bottom boundaries of the license plate. Then a vertical projection is carried out to get the left and right bounds of the license plate. The process is

presented in Figure 3.4.

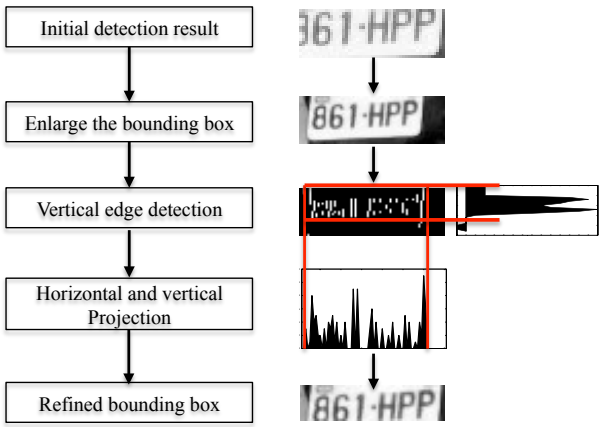


Figure 3.4: The process of bounding box refining.

Finally we use another plate/non-plate CNN classifier (CNN-II) to verify the remaining bounding boxes. The structure of CNN-II is presented in Table 3.2. It is trained with positive samples of gray-scale license plates from different countries, either cropped from real images or synthesized by ourselves, and negative samples constituted by non-text image patches as well as some general text strings. The size of the input image is 100×30 pixels. Data augmentation and bootstrapping are applied here to improve the classification performance. For each candidate license plate, we evaluate it by averaging the probabilities of five predictions over random image translation, so as to remove noises. The ones that are classified as license plates are fed to the next step.

3.2.2 Car License Plate Recognition

The second stage of LPDR system is to recognize the characters in the license plate. In the traditional framework of LPDR, character segmentation has a great influence on the success of plate recognition. The license plate will be recognized incorrectly if the segmentation is improper, even if we have a strong recognizer that can deal with characters of various sizes, fonts and rotations. However, the character segmentation process by itself is a really challenging task that is prone to be influenced by uneven

Table 3.2: Configuration of the 4-layer plate/non-plate CNN model (CNN-II). "k", "s" and "p" represent kernel size, stride and padding size respectively.

Layer Type	**Parameters**
Soft-max	2 classes
Fully connected	#neurons: 2
Dropout	Prop: 0.5
Fully connected	#neurons: 500
Maxpooling	k:3 $\times$ 3, s:3
Convolution	#filters:256, k:5 $\times$ 5, s:1, p:0
Maxpooling	k:2 $\times$ 2, s:2
Convolution	#filters:96, k:5 $\times$ 5, s:1, p:0
Input	30 $\times$ 100 pixels gray-scale image

lighting, shadow and noise in the image. Different rules are employed in previous work [Yoon et al., 2011; Zheng et al., 2013; Hsu et al., 2013] to modify the improperly segmented blocks, which are still not robust. In this subsection, we use a novel recognition technique that treats the characters in license plate as an unsegmented sequence, and solves the problem from the viewpoint of sequence labeling.

The overall procedure of our sequence labeling based plate recognition method is presented in Figure 3.5. It mainly consists of three parts.

3.2.2.1 Sequence Feature Generation

In order to improve the recognition performance, we train another deeper 36-class CNN classifier (CNN-III, as presented in Table 3.3) for sequential feature extraction from the cropped license plate image. Inspired by the work of [Su and Lu, 2014; He et al., 2016b], the features are extracted in a sliding window manner across the image.

For each cropped license plate image, we convert it to gray-scale, and pad with 12 pixels on both left and right sides. Then we resize the plate to 24 $\times$ 94 pixels, with the height the same as the input height for CNN-III.

After that we use a sub-window of size 24 $\times$ 24 pixels to partition the padded image convolutionally, with a step size of 1. For each partitioned image patch, we

Table 3.3: Configuration of the 9-layer CNN model (CNN-III). "k", "s" and "p" represent kernel size, stride and padding size respectively.

Layer Type	**Parameters**
Softmax	36 classes
Fully connected	#neurons: 36
Dropout	Prop: 0.5
Fully connected	#neurons: 1000
Dropout	Prop: 0.5
Fully connected	#neurons: 1000
Maxpooling	k:3 $\times$ 3, s:1
Convolution	#filters:512, k:3 $\times$ 3, s:1, p:1
Convolution	#filters:512, k:3 $\times$ 3, s:1, p:1
Maxpooling	k:3 $\times$ 3, s:2
Convolution	#filters:256, k:3 $\times$ 3, s:1, p:1
Convolution	#filters:256, k:3 $\times$ 3, s:1, p:1
Maxpooling	k:3 $\times$ 3, s:2
Convolution	#filters:128, k:3 $\times$ 3, s:1, p:1
Maxpooling	k:3 $\times$ 3, s:1
Convolution	#filters:64, k:3 $\times$ 3, s:1, p:1
Input	24 $\times$ 24 pixels gray-scale image

feed it into the well-trained CNN-III, and extract the $4 \times 4 \times 256$ features from the output of the 4th convolutional layer, as well as the 1000 features from the output of the first fully connected layer. The two feature vectors are then concatenated together into one feature vector with length 5096. These features contain both local and global information of the image patch, which will bring better recognition performance compared with the ones extracted only from the fully connected layer. Principle Component Analysis (PCA) is applied to reduce the feature dimension to 256-d, followed by a feature normalization.

With this operation, features are extracted from left to right on each sub-window in the candidate license plate image, and form a feature sequence array $\mathbf{x} = \{x_1, x_2, \ldots, x_L\}$, where $x_t \in \mathcal{R}^{256}$, L is the number of sub-windows. It not only keeps the order information, but also captures sufficient contextual information for RNNs to exploit. The inter-relation between feature vectors contributes a lot to character recognition.

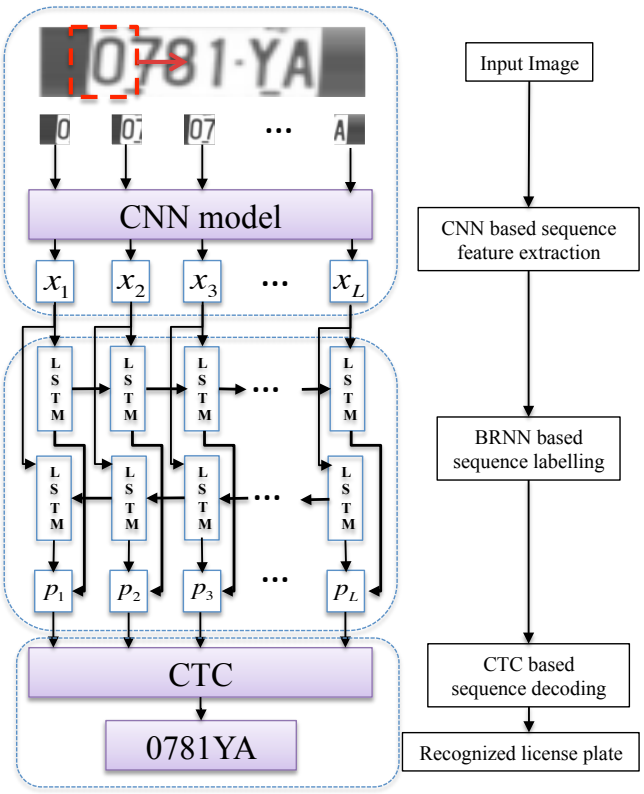


Figure 3.5: Overall structure of the sequence labeling based plate recognition. Sequential features are extracted in a sliding window manner by CNNs, which are then fed to BLSTMs for sequence labeling, without character separation. CTC is followed for sequence decoding.

3.2.2.2 Sequence Labeling

RNNs are special neural networks which provide a powerful mechanism to exploit past contextual information. These contextual cues will make the sequence recognition more stable than treating each feature independently. To overcome the shortcoming of gradient vanishing or exploding during RNN training, LSTMs are employed. They contain memory blocks which can store the contexts for a long period of time.

For our task of character string recognition, it would be helpful to have access to the contexts both in the past and in the future. Hence BLSTMs [Graves et al., 2009a] are applied here. As illustrated in Figure 3.5, there are two separated hidden layers in BLSTMs, one of which processes the feature sequence forward, while the other one processes it backward. For each hidden layer, all LSTMs share the same

parameters. Both hidden layers are connected to the same output layer, providing it with information in both directions along the input sequence.

Sequence labeling is processed by recurrently implementing BLSTMs for each feature in the feature sequence. Each time the state h_t is updated according to computation (3.2) which takes both current feature x_t and neighboring state h_{t-1} or h_{t+1} as inputs, i.e.,

$$\begin{cases} h_t^{(f)} = \text{LSTM}_1(x_t, h_{t-1}^{(f)}), \\ h_t^{(b)} = \text{LSTM}_2(x_t, h_{t+1}^{(b)}). \end{cases} \tag{3.2}$$

where (f) means recurrent forward and (b) means backward. A Softmax layer follows which transforms the BLSTMs' states into a probability distribution over 37 classes, i.e.,

$$p_t(c = c_k|x_t) = \text{Softmax}([h_t^{(f)}, h_t^{(b)}]),\ k = 1, \ldots, 37 \tag{3.3}$$

where the additional 37th class is used to describe the space between characters. The whole feature sequence is finally transformed into a sequence of probability estimation $\mathbf{p} = \{p_1, p_2, \ldots, p_L\}$ with the same length as the input sequence.

3.2.2.3 Sequence Decoding

Lastly, we need to transform the sequence of probability estimation $\mathbf{p}$ into a character string. We follow the handwriting recognition system [Graves et al., 2009a] by applying CTC on top of the output layer of BLSTMs. CTC is specifically designed for sequence classification without data pre-segmentation. It decodes the predicted probability sequence into output labels directly.

The objective function for CTC is defined as the negative log probability of the network correctly labeling the entire training set, i.e.,

$$\mathcal{O} = -\sum_{(\mathbf{c},\mathbf{z})\in\mathcal{S}} \ln P(\mathbf{z}|\mathbf{c}), \tag{3.4}$$

where $\mathcal{S}$ is the training dataset, which consists of pairs of input and target sequences

$(\mathbf{c}, \mathbf{z})$. $P(\mathbf{z}|\mathbf{c})$ denotes the conditional probability of obtaining target sequence $\mathbf{z}$ through the input $\mathbf{c}$. The target is to minimize $\mathcal{O}$, which is equivalent to maximize $P(\mathbf{z}|\mathbf{c})$.

This object function is differentiable to input. We connect it directly to the outputs of BLSTMs, i.e., the input of CTC $\mathbf{c}$ is exactly the output activation of BLSTMs $\mathbf{p}$, and

$$P(\mathbf{z}|\mathbf{c}) = \sum_{\pi:\mathcal{B}(\pi)=\mathbf{z}} P(\pi|\mathbf{p}). \tag{3.5}$$

This part of network can then be trained with gradient descent and back propagation. The operator $\mathcal{B}$ is defined as to remove the repeated labels and the space labels from the path. For example, $\mathcal{B}(a-a-b-) = \mathcal{B}(-aa--ab-b) = (aab)$.

Once the network is well trained, the aim of sequence decoding is to find an approximately optimal path π with maximum probability through the BLSTMs output sequence, i.e.,

$$\mathbf{l}^* \approx \mathcal{B}(\arg\max_{\pi} P(\pi|\mathbf{p})). \tag{3.6}$$

Details of CTC can refer to [Graves et al., 2006, 2009a].

3.2.3 Experiments

In this section, experiments are performed to verify the effectiveness of the proposed methods. Our experiments are implemented on NVIDIA Tesla K40c GPU with 6GB memory. The CNN models are trained using MatConvNet [Vedaldi and Lenc, 2015].

3.2.3.1 Datasets

A sufficiently large training dataset is essential for the success of a CNN model. CNN-I is trained on roughly 1.38×10^5 character images and 9×10^5 non-character images. CNN-III is trained only with the character images. The character images comprise 26 upper-case letters and 10 digitals sampled from the datasets created by Jaderberg et al. [2014c] and Wang et al. [2011]. The non-character image patches

are cropped by ourselves from the ICDAR datasets [Lucas et al., 2003; Lucas, 2005; Shahab et al., 2011] and Microsoft Research Cambridge Object Recognition Image database [Criminisi, 2004]. All images are gray-scale and resized to 24×24 pixels for training. Data augmentation is carried out by image translations and rotations to reduce overfitting. Bootstrapping, which collects hard negative examples and re-trains the classifier, is also used to improve the classification accuracy.

As to CNN-II, we crop around 3000 license plate images from public available datasets [Zhou et al., 2012; Anagnostopoulos et al., 2008]. We also synthesize nearly 5×10^4 license plates using ImageMagic. Around 4×10^5 background images are used here including patches without any characters and patches with some general text. All the images are gray-scale and resized to 30×100 pixels for training.

We test the effectiveness of the proposed detection and recognition algorithms on two datasets. The first one is the Caltech Cars (Rear) 1999 dataset [car, 2003] which consists of 126 images with resolution of 896×592 pixels. The second dataset is the Application-Oriented License Plate (AOLP) benchmark [Hsu et al., 2013], which has 2049 images of Taiwan license plates. This database is categorized into three subsets with different level of difficulty for detection and recognition: Access Control (AC), Traffic Law enforcement (LE), and Road Patrol (RP). AC refers to the case that a vehicle passes a fixed passage with a lower speed or full stop. This is the easiest situation. The images are captured under various illuminations and weather conditions. LE refers to the case that a vehicle violates traffic laws and is captured by roadside camera. The backgrounds are really cluttered, with road signs and multiple plates in one image. RP refers to the case that the camera is held on a patrolling vehicle, and the images are taken with arbitrary viewpoints and distances. The detailed introduction of this AOLP dataset can be found in [Hsu et al., 2013].

3.2.3.2 Evaluation Criteria

For license plate detection, we follow the evaluation criterion in [Zhou et al., 2012] for fair comparison. The detection results are quantified by precision and recall rates. Precision is defined as the number of correctly detected license plates divided by the total number of detected regions. It gives us information on the amount of false alarms. Systems that over-estimate the number of bounding boxes are punished with a low precision score. Recall is defined as the number of correctly detected license plates divided by the total number of ground-truths. It measures how many ground-truth objects have been detected. Systems that under-estimate the number of ground-truths are punished with a low recall score. In addition, we also presented a F-measure is a single measure of the test performance for reference, which is calculated as

$$\text{F-measure} = \frac{2 \times (\text{precision} \times \text{recall})}{(\text{precision} + \text{recall})} \tag{3.7}$$

A detection is considered to be correct if the license plate is totally encompassed by the bounding box, and the overlap between the detection and ground-truth bounding box is greater than 0.5. The overlap means the area of intersection divided by the area of union containing both rectangles (IoU), i.e.,

$$\text{IoU} = \frac{\text{area}(R_{\text{det}} \cap R_{\text{gt}})}{\text{area}(R_{\text{det}} \cup R_{\text{gt}})} \tag{3.8}$$

where R_{det} and R_{gt} are regions of the detected bounding box and ground-truth respectively.

As to license plate recognition, we evaluate it by recognition accuracy, which is defined as the number of correctly recognized license plates divided by the total number of ground-truths. A correctly recognized license plate means all the characters on the plate are recognized properly. The license plates for recognition are from the detection result, rather than cropped directly from the ground-truths. Therefore,

the detection performance affects the final recognition result greatly, not only on quantity, but also on quality. In order to compare with previous work, we also give out the character recognition accuracy as that defined in Hsu et al. [2013], i.e., the number of correctly recognized characters divided by the total number of characters from the ground-truths.

3.2.3.3 Experimental Results on License Plate Detection

The detection performance of our cascade CNN based method is shown in Table 3.4 for Caltech cars dataset, and in Table 3.5 for AOLP dataset. Results with previous approaches are also provided for comparison. The works of Le and Li [2006], Bai and Liu [2004] and Hsu et al. [2013] are edge-based methods, where color information is integrated in Le and Li [2006]'s to remove false positives. Lim and Tay [2010] and Zhou et al. [2012]'s works are character-based methods, where MSER and principal visual word are employed respectively.

Table 3.4: Comparison of plate detection results by different methods on Caltech cars dataset. Our cascade CNN based method produced the best detection result, with both the highest precision and recall. ('P' for Precision, 'R' for Recall, and 'F' for F-measure)

Method	P (%)	R (%)	F (%)
Le and Li [2006]	71.40	61.60	66.14
Bai and Liu [2004]	74.10	68.70	71.30
Lim and Tay [2010]	83.73	90.47	86.97
Zhou et al. [2012]	95.50	84.80	89.83
Ours (37-way outputs)	**97.56**	95.24	**96.38**
Ours (2-way outputs)	97.39	89.89	93.49
Ours (37-way outputs without plate/non-plate CNN used)	92.96	**96.58**	94.73

Based on the evaluation criterion described above, our approach outperforms all the five methods in both precision and recall on both datasets. To be specific, on Caltech cars dataset, our method achieves a recall of 95.24%, which is 4.77% higher than the second best one achieved by the method in [Lim and Tay, 2010]. The precision of our approach is 97.56%, which is also the best, with 2.06% higher than

Table 3.5: Comparison of plate detection results by different methods on the AOLP dataset. Our method produces better results than the previous one on all three subsets. ('P' for Precision, 'R' for Recall, and 'F' for F-measure)

Method \ Subset	AC			LE			RP		
	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)
Hsu et al. [2013]	91	96	93.43	91	95	92.96	91	94	92.48
Ours	98.53	98.38	98.45	97.75	97.62	97.69	95.28	95.58	95.43

the second. On AOLP dataset, our method gives the highest precisions and recalls on all three sub-datasets, with an even obvious superiority on precision. With GPU, it needs about 5 seconds to process an image from Caltech cars dataset, and 2-3 seconds for images from AOLP dataset.

The second last row in Table 3.4 shows a detection result using our framework but with 2-way classification outputs (text/non-text). As we described before, in our detection phase, we use a 37-way CNN classifier instead of a binary text/non-text classifier. The 37-way CNN classifier can learn more discriminative and specific features of each character. By contrast, the 2-way CNN classifier that sorts all characters in one class may omit features specific for certain characters, which would be inaccurate and may miss some characters during detection. The detection result on Caltech cars dataset proves this point, where the 2-way classifier produces a lower recall compared to the 37-way classifier.

The last row in Table 3.4 shows a detection result without the plate/non-plate CNN classifier used. The precision decreases about 5%, which demonstrates the effectiveness of that plate/non-plate CNN classifier in eliminating false positives.

Some plate detection results on Caltech cars dataset are shown in Figure 3.6.

3.2.3.4 Experimental Results on License Plate Recognition

In this part, we only test the recognition performance on AOLP dataset as we do not have training data with similar pattern and distribution as Caltech cars license plates. For AOLP dataset, the experiments are carried out by using license plates

Figure 3.6: Examples of license plate detection on Caltech cars dataset. The green rectangles indicate our detection results. The results show that our method is able to detect license plates in images under various capture conditions, such as strong lighting, bluring, *etc.*. The last image shows a failure case where the last letter is not contained in the bounding box although after bounding box refining.

Table 3.6: Comparison of plate recognition results by different methods on AOLP dataset. The recognition accuracy is presented in percentage.

Method \ Subset	AC		LE		RP	
	Plate (%)	Character (%)	Plate (%)	Character (%)	Plate (%)	Character (%)
Hsu et al. [2013]	88.5	96	86.6	94	85.7	95
baseline approach	93.53	97.84	89.83	97.27	86.58	95.57
Our approach	94.85	–	94.19	–	88.38	–

from different sub-datasets for training and test separately. For example, to test on AC sub-dataset, we use the license plates from LE and RP sub-datasets to train the model. Data augmentation is implemented via image translation and affine transformation to reduce over-fitting. Since the license plates in RP have a large degree of rotation and projective orientation, features extracted horizontally through sliding window are inaccurate for each character. Hence Hough transform is employed here to correct rotations [Rasheed et al., 2012]. Experimental results are presented in

Table 3.6.

In [Hsu et al., 2013], MSER is applied firstly for character segmentation. LBP features are extracted from each character and classified using linear discriminant analysis. We also present a recognition result with a baseline method, which carries out character segmentation by a CC based method and then recognizes each character using CNN-III. Experimental results in Table 3.6 show that our CNN model gives better character recognition performance than the model used in [Hsu et al., 2013]. In addition, our sequence labeling based method gives even higher accuracy for the whole plate recognition. It not only skips the challenging task of character separation, but also takes advantage of the abundant contextual information via BLSTMs which helps to enhance the recognition accuracy furthermore.

To show the advantage of BLSTMs, we visualize the recognition results from the Softmax layer of CNNs and BLSTMs respectively. CNN-III is retrained by adding background images and using bootstrapping so that it can distinguish characters from various kinds of background. The recognition probability distributions from the

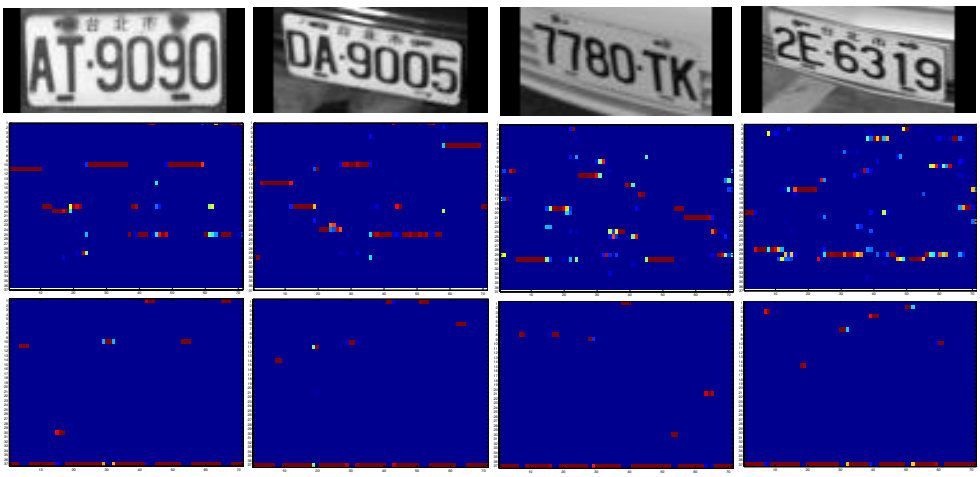


Figure 3.7: License plate recognition confidence maps. The first row is the detected license plates. The second row is the recognition probabilities from the Softmax layer after CNNs. The third row is the recognition probabilities from the Softmax layer after BLSTMs. For each confidence map, the recognition probabilities of current sub-window on 37 classes are shown vertically (with classes order from top to bottom: non-character, 0-9, A-Z). BLSTMs produce much better recognition results. Characters on each license plate can be read straightforward from the outputs of BLSTMs.

Softmax layer of CNNs and BLSTMs are compared in Figure 3.7. It can be observed that the character recognition probabilities are more clear and correct on the output maps of BLSTMs. Characters can then be separated naturally, and the final license plate reading is straightforward by applying CTC on these maps.

Some experimental results on car license plate detection and recognition are shown in Figure 3.8 for AOLP dataset. Some failure cases are also presented which shows that the car logos, front cross and wheels are easy to cause false positives.

Figure 3.8: Examples of license plate detection and recognition on AOLP dataset. The green rectangles indicate our detection results, with the yellow tags above showing the recognition results. Our method can detect and recognize license plates under various illuminations and certain orientations. It should be note that there are still some oversize bounding boxes, e.g., the third image in the first row. Bounding box refining does not work well if there is noise in the license plate, such as extra letters, strong lighting, *etc.*. In addition, car logos, front cross, *etc.*can easily cause false positives, and sometimes are difficult to get rid of.

3.3 A Stepwise Method for General Scene Text Detection and Recognition

In this section, we extend the stepwise method to tackle general text spotting in natural scene images. Considering that there is no definitive borderlines for general text in natural scene images, like the border of license plates, some new ideas are proposed so as to let the framework more suitable for general scene text. We begin by detecting characters from the input image via a sliding window manner based on the well-trained CNN classifier. The detected characters are then grouped into text lines instead of words as it is not easy to group characters into word directly from heaps of detections. In contrast to the separately trained CNNs and RNNs in car license plate recognition, we integrate them together as an end-to-end trainable deep neural network. The network is expected to complete three tasks simultaneously: 1) separating the text lines into words; 2) recognizing each word; 3) getting rid of false positives. The pipeline of the proposed method is shown in Figure 3.9.

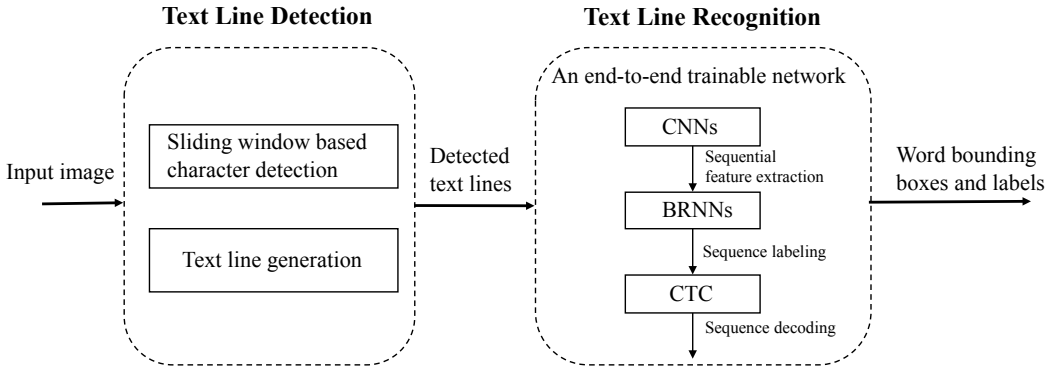


Figure 3.9: The stepwise framework for text spotting.

The main difference between this work and the method used in the previous section is the end-to-end trainable deep neural network that is specifically designed for recognizing sequence-like objects in images. Compared with existing systems for text spotting and our stepwise method for car license plate detection and recognition, the proposed architecture has four distinctive properties:

- Not only character separation free, in this method, the challenging word separation task is also handed over to the deep neural network. The separation task is accomplished via supervised learning instead of employing heuristic rules, which would be more robust and accurate.

- In contrast to most of the existing approaches where the components are separately trained or tuned, e.g., several CNNs used in the previous section, the character CNNs used in [Jaderberg et al., 2014c; He et al., 2016b], the second stage in our method adopts an end-to-end trainable network for sequence recognition, including CNNs for feature learning, RNNs and CTC for sequence labeling. Instead of training a separate character CNN classifier for features extraction, the CNN parameters can be tuned together with RNNs according to the final word recognition loss. The learning process is simplified and the learned features would be more specific and discriminative.

- Apart from word recognition, the network designed in the second stage is also trained to eliminate false positives, which helps improve detection precision.

- The end-to-end network for sequence recognition is unconstrained to the lengths of sequence-like objects, requiring only height normalization in both training and test phases, which avoids image deformation that may happen with the method in the previous section.

Because this method is an extension of the stepwise method for car license plate detection and recognition in the previous section, in the following we will explain each part in brief, with more description on the difference. Experiments on general text spotting will be implemented to demonstrate the effectiveness of this method.

3.3.1 Text Line Detection

Sliding window based character detection method is adopted here to get the text saliency map of the input image, as we used in Chapter 3.2.1. To be specific, the

well-trained 37-class CNN classifier is applied in a sliding window fashion across the input image. Positions with characters will results in a high probability while the background positions will have a low probability close to zero, as demonstrated in Figure 3.10. The input image is resized into 16 scales so that characters of different size can be covered. At each scale, the probability map is thresholded and non-maximal suppressed to eliminate noise. The high probability pixels are then connected as text lines with 4-connected neighborhood. Here, we remove the ones that are nearly flat, i.e., regions with height less than 10 pixels. The rest text lines are kept for further classification and recognition.

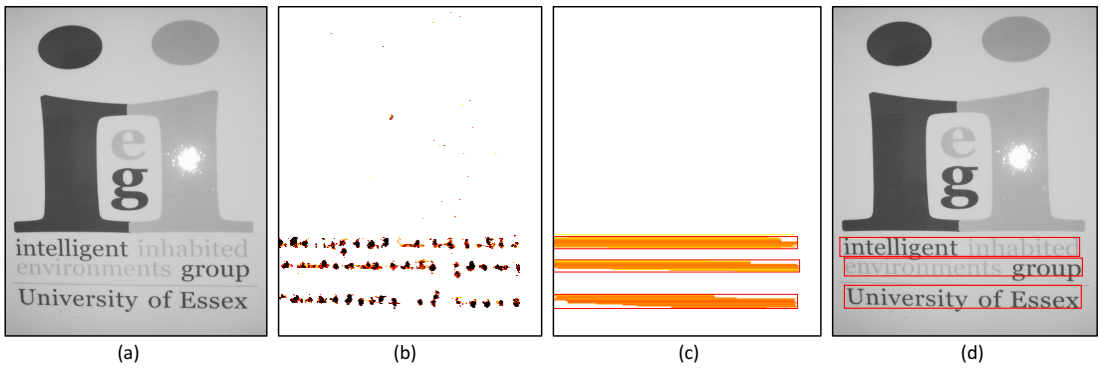


Figure 3.10: Text line detection procedure in a single scale. (a) is the input image, (b) is the text heat map by sliding window based detection, (c) shows the candidate bounding boxes after CCA, (d) shows the detected text lines annotated by red bounding boxes.

3.3.2 Text Line Separation and Word Recognition

In this part, an end-to-end trainable deep neural network is proposed which can recognize the text line and separate it into words simultaneously. Moreover, it has the ability to remove false positives as well. The architecture of the network refers to "CRNN" [Shi et al., 2015], as shown in Table 3.7. The beginning of the network is several convolutional layers to get a feature sequence from each text line image. Then bi-direction LSTMs are built on top of the convolutional layers directly to predict each frame of the feature sequence. CTC is adopted lastly as the transcription layer

to translate the frame prediction into a label sequence. In contrast to the model we used in Chapter 3.2.2 for license plate recognition, both CNNs and RNNs are trained jointly in a unified network with one loss function. Compared with "CRNN" [Shi et al., 2015], the end-to-end trainable network for word recognition, our model involves another background class, besides the 36 character classes (26 case-insensitive alphabets and 10 digits). We denoted it as "*". It is used to represent the background as well as the spaces between words. Hence, the network is trained to distinguish background and identify word intervals in text lines.

Table 3.7: Configuration of the end-to-end network for text line recognition (from up to bottom, referring to Figure 3.9). "k", "s" and "p" represent kernel size, stride and padding size respectively.

Layer Type	**Parameters**
Input	W $\times$ 32 pixels gray-scale image
Convolution	#filters:64, k:3 $\times$ 3, s:1, p:1
Maxpooling	k:2 $\times$ 2, s:2, p:0
Convolution	#filters:128, k:3 $\times$ 3, s:1, p:1
Maxpooling	k:2 $\times$ 2, s:2, p:0
Convolution	#filters:256, k:3 $\times$ 3, s:1, p:1
Batch Normalization	
Convolution	#filters:256, k:3 $\times$ 3, s:1, p:1
Maxpooling	p:2 $\times$ 2, s:1 $\times$ 2 p:1 $\times$ 0
Convolution	#filters:512, k:3 $\times$ 3, s:1, p:1
Batch Normalization	
Convolution	#filters:512, k:3 $\times$ 3, s:1, p:1
Batch Normalization	
Maxpooling	k:2 $\times$ 2, s:1 $\times$ 2, p:0
Convolution	#filters:512, k:2 $\times$ 2, s:1, p:0
Batch Normalization	
Transform to Sequence	
BLSTMs	#hidden units: 256
BLSTMs	#hidden units: 256
Softmax	
CTC for Transcription	

Sequential Feature Extraction The architecture of convolutional layers is similar to VGG-16, with small convolutional kernels (3 $\times$ 3) and deep layers (15 layers), as

presented in Table 3.7. We keep the aspect ratio of the obtained text line images, but scale them into the same height (32 pixels), so that after convolutions and poolings, a sequence of feature vector can be generated. Each frame in the sequence represents the feature of corresponding rectangular region of the input image (i.e., the receptive field), as illustrated in Figure 3.11. The features are arranged from left to right with the same order as the characters appeared in the input image. They not only include the information for each character, but also contain sufficient contextual information. We denote the feature sequence as $\mathbf{x} = \{x_1, x_2, \ldots, x_L\}$, where $x_t \in \mathcal{R}^{512}$, L is the length of feature sequence.

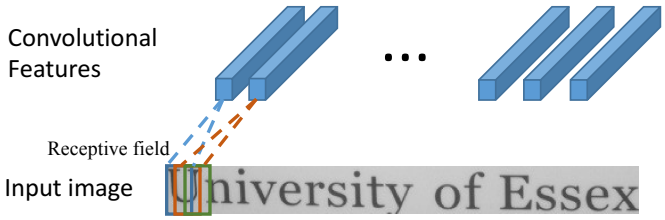


Figure 3.11: The illustration of the receptive field. Each frame of the convolutional feature corresponds to a rectangular region of the input image. The features are complementary with each other.

Sequence Labeling Two layers of bi-directional LSTMs are adopted after the convolutional layers, as presented in Table 3.7. As we described and verified in car license plate recognition, RNNs have a strong capability of exploiting contextual information, which would benefit a lot the sequence recognition. For example, several successive frames of features will be complementary with each other in recognizing a wide character, like "W". Besides, it would be easier to recognize "il" by contrasting the character heights if the contextual information is available than by recognizing each of them separately. LSTM can also back-propagate errors to its input, which allows the joint training of both CNNs and RNNs in a unified network. A Softmax layer is followed which transforms the BLSTMs' states into a probability distribution

over 38 classes, i.e.,

$$p_t(c = c_k|x_t) = \text{Softmax}([h_t^{(f)}, h_t^{(b)}]),\ k = 1, \ldots, 38, t = 1, \ldots, L \tag{3.9}$$

where $h_t^{(f)}$ and $h_t^{(b)}$ are the hidden states of the forward LSTMs and backward LSTMs at the top layer respectively. The forward and backward hidden states are concatenated together and transformed into a probability by a Softmax layer. The 38 classes includes 36 characters, 1 background denoted as "*", and 1 "blank" between characters in one word denoted as "-".

Sequence Decoding The method for sequence decoding is the same as what we used in the previous section, with CTC employed to transform the probabilities into a character string. For instance, the image in Figure 3.11 will finally be recognized as "University*of*Essex", and then be separated into words intuitively as "University", "of", "Essex". Note that if the transformed string is only composed of "*", it means the corresponding image is background and should be abandoned.

3.3.3 Experiments

Experiments are carried out in this section to demonstrate the effectiveness of this method. The character CNN is implemented by MatConvNet [Vedaldi and Lenc, 2015], while the end-to-end recognition network is implemented by Torch 7. The experiments are conducted on NVIDIA Tesla K40c GPU with 6GB memory.

3.3.3.1 Datasets and Model Training

The 37-class character CNN well trained in Chapter 3.2.3 is adopted again for character detection in the first step. The end-to-end recognition model is fine-tuned from the well-trained "CRNN" model [Shi et al., 2015] with additional text line images and background images. We crop text lines from the commonly used scene text training datasets including ICDAR2003 [Lucas et al., 2003], ICDAR2011 [Shahab et al., 2011], ICDAR2015 [Karatzas et al., 2015] and Street View Text (SVT) [Wang et al., 2011]. We

also photograph some natural scene images with text inside, and annotate the text lines by ourself. Totally we harvest 5485 text line images. 7000 background images are cropped from the above mentioned scene text training datasets, with height set to 32 pixels and width randomly sampled from 50 to 800 pixels. There are ultimately around 12000 training images in total, which are used to fine-tune the end-to-end recognition network. We rescale the height of all the images to 32 pixels, while the width is calculated according to the image aspect ratio but no more than 800 pixels. The network is trained with ADADELTA [Zeiler, 2012], with the decay rate ρ set to 0.9. Data augmentation is added with certain degrees of rotation (no more than 15 degree).

We evaluate the proposed approach on the test dataset of ICDAR2015 [Karatzas et al., 2015] "Focused Scene Text", which composed of 233 images. The dataset also provides three types of lexicons for reference during test phase, i.e., "Strongly Contextualized", "Weakly Contextualized" and "Generic". "Strongly Contextualized" lexicon means 100 words are provided for each image including all words that appear in the image as well as a number of distractors. "Weakly Contextualized" lexicon provides all proper words that appear in the entire test set for reference, with totally around 600 words. "Generic" is a general vocabulary with 90k words.

3.3.3.2 Evaluation Criteria

We follow the evaluation criterion used in ICDAR2015 Robust Reading Competition [Karatzas et al., 2015]: a bounding box is considered as correct if its IoU ratio with any ground-truth is greater than 0.5 and the recognized word also matches, ignoring the case. The words that contain alphanumeric characters and no longer than three characters are ignored.

There are two evaluation protocols used in the task of scene text spotting: "End-to-End" and "Word Spotting". "End-to-End" protocol requires that all words in the image are to be recognized, with independence of whether the string exists or not

Table 3.8: Text spotting results on ICDAR2015 "Focused Scene Text" dataset with two evaluation protocols. F-measures are presented in the table in percentage. Our method achieves the best performance under all three lexicons with both evaluation criteria.

Method	ICDAR 2015 Word-Spotting			ICDAR 2015 End-to-End		
	Strong	Weak	Generic	Strong	Weak	Generic
TextSpotter [Neumann and Matas, 2016]	85	66	57	77	63	54
Deep2Text II+ [Yin et al., 2014]	84.84	83.43	78.90	81.81	79.47	76.99
Jaderberg *et al.* [Jaderberg et al., 2016]	90.49	–	76	86.35	–	–
Ours	**91.39**	**90.16**	**82.91**	**87.19**	**86.39**	**80.12**

in the provided contextualized lexicon, while "Word Spotting" protocol, on the other hand, only looks at the words that actually exist in the lexicon provided, ignoring all the rest that do not appear in the lexicon. It should be note that the dictionaries provided for each image contain a standardized form of words (e.g., if a word finishes in 's or an exclamation mark, this is not included in the dictionary). Strings like telephone numbers, prices, *etc*.are not included either. The dictionaries are only supposed to provide a context for the "End-to-End" protocol, while if we evaluate with "Word Spotting" protocol, we only focus on the subset of words that actually exist in the dictionary.

As to each protocol, both precision and recall are evaluated, and F-measure is calculated according to equation 3.10, which provides a comprehensive evaluation of the result.

$$\text{F-measure} = \frac{2 \times (\text{precision} \times \text{recall})}{(\text{precision} + \text{recall})} \tag{3.10}$$

3.3.3.3 Experimental Results

As shown in Table 3.8, our system achieves the best performance on the "Focused Scene Text" Task in ICDAR2015 Competition at that time, under both "End-to-End"

Figure 3.12: Examples of Text spotting results on ICDAR2015 benchmark. The detection and recognition results are presented in red bounding boxes with red labels above. Our proposed method can detect and recognize text in scene images even if it is a long word with a big aspect ratio, or it consists of digits. However, because of the horizontal text line generation method, it is hard to detect text with some orientations. In addition, small text and text with unfamiliar fonts are difficult to be detected or recognized. Some failure cases are presented here in the fourth line, where the green bounding boxes and green labels show the ground-truth.

and "Word Spotting" protocols. Especially when we test with the "Generic" lexicon, our method produces the F-measure of 82.91% under "Word Spotting", which is about 4% higher than the second best result. Some examples of the text spotting results are presented in Figure 3.12, which show the effectiveness of our method.

Some failure cases are also provided, which shows that our method can not deal with text with some orientations or very small text.

3.4 Conclusion

In this chapter, we proposed a stepwise method to address the problems of car license plate detection and recognition, and general scene text spotting respectively, using the promising CNN techniques. To be specific, for car license plate detection and recognition, a cascade detection framework was proposed with a sliding window based approach for character level detection, heuristic rules for character grouping and bounding box refining, and a plate/non-plate classifier for false positives eliminating. It is robust under various conditions. The plate recognition was formulated as a sequence labeling problem. BRNNs and CTC were employed for sequence recognition, which skipped the challenging task of character segmentation. The recurrent property of RNNs enables contextual information exploration, which contributes a lot to the final recognition result. As to text spotting in natural scene images, a similar sliding window based method was employed at the beginning for character detection. The detected characters were grouped into text lines instead of generally used words, which skipped the challenging word separation tasks. An end-to-end trainable deep neural network was trained to recognize the text lines, split words and remove false positives. Experimental results on both car license plates and general scene text demonstrated the validity of these ideas.

However, this framework has a few drawbacks. An obvious one is that the sliding winow based detection method is too slow to real-time application. Therefore, methods will be explored to improve the processing speed. In addition, the detection method works in an unsupervised manner, where the information of ground-truth bounding box position has not been used in the whole process. Although some post-processing like bounding boxes refining can improve the overlap ratio, it is affected easily by the unwanted noises in the image, as shown in the last image of Figure 3.6

and the 3rd and 5th images of Figure 3.8. In the future, we will exploit methods that can take full advantage of the provided ground-truth information to improve the detection accuracy.

Chapter 4

Towards End-to-End Car License Plate Detection and Recognition

4.1 Introduction

In the previous chapter, we provide a stepwise method for car license plate detection and recognition, where two separate modules are used respectively for plate detection and recognition. Although it shows good performance on several datasets, this framework has a number of weaknesses: 1) There are several CNNs trained separately for different subtasks, which seems to be cumbersome. 2) The sliding window based detection method is too slow to real-time application. 3) The sequential features for recognition are extracted by a separate CNN, whose parameters cannot be tuned anymore according to the final recognition loss. 4) The character grouping and bounding box refining work in a heuristic manner, which are not robust.

With these shortcomings in mind, in this chapter, an integrated deep neural network is proposed for simultaneous car license plate detection and recognition. In fact, the tasks of plate detection and recognition are highly correlated as we can see from the previous chapter. Accurate bounding boxes obtained via detection methods can improve the recognition accuracy, while the recognition result can be used to eliminate false positives vice versa. Thus in this chapter, we propose a unified framework to jointly tackle these two tasks at the same level. A deep neural network is designed, which takes an image as input and outputs the locations of the license

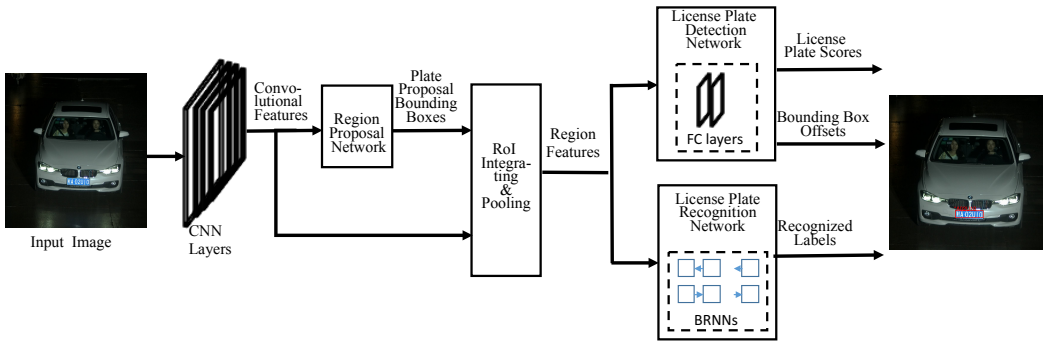


Figure 4.1: The overall structure of our model. It consists of several convolutional layers, a region proposal network for license plate proposals generation, proposal integrating and pooling layer, multi-layer perceptrons for plate detection and bounding box regression, and RNNs for plate recognition. Given an input RGB image, with a single forward evaluation, the network outputs scores of predicted bounding boxes being license plates, bounding box offsets with a scale-invariant translation and log-space height/width shift relative to a proposal, as well as the recognized license plate labels at the same time. The extracted region features are used by both detection and recognition, which not only shares computation, but also reduces model size.

plates as well as the plate labels simultaneously in a single forward pass, with both high efficiency and accuracy. We prove that the low level convolutional features can be used for both detection and recognition.

With CNNs, RNNs and CTC layer integrated in one network, and with a multi-task loss jointly optimized, the whole network is trained end-to-end, without using any heuristic rules. Both training and test processes are simplified largely compared to the ones in the previous chapter. The bounding box localization works in a supervised manner, where the information of the ground-truth bounding box position provides guidance for accurate bounding box regression. The well-trained model achieves higher accuracies on both license plate detection and recognition, using a shorter processing time. An overview of the network architecture is shown in Figure 4.1.

It should be note that although a number of integrated methods have been proposed for end-to-end text detection and recognition [Wang et al., 2011; Neumann and Matas, 2013a], as we described in Chapter 1.1.3, the most remarkable difference

between our method and the existing ones is that our network can be trained end-to-end for both detection and recognition, while the other methods need to combine results from separately trained models to obtain the final detection and recognition results. With this innovation, some pre-processings, like character detection or character grouping, can be eliminated, which not only simplifies the whole process, but also avoids the intermediate error accumulation. To our best knowledge, this is the first work at that time that integrates both license plate detection and recognition into a single network with an end-to-end optimization. The main contributions of this work are as follows:

- A single unified deep neural network is proposed, which can detect license plates from an image and recognize the labels all at once. The whole framework involves no heuristic processes, such as the use of plate colors or character space, and avoids intermediate procedures like character grouping or separation. It can be trained end-to-end, with only the image, plate positions and labels needed for training. The resulting system achieves high accuracy on both plate detection and letter recognition.

- Secondly, the convolutional features are shared by both detection and recognition, which leads to fewer parameters compared to using separated models. Moreover, with the joint optimization of both detection and recognition losses, the extracted features would have richer information. Experiments show that both detection and recognition performance can be boosted via using the jointly trained model.

- By integrating plate recognition directly into the detection pipeline, the resulting system is more efficient. With our framework, we do not need to crop the detected license plates from the input image and then recognize them by a separate network. The whole framework takes about 0.31 second for an input image of 600×600 pixels on a Titan X GPU, which is much more faster than the stepwise method in the previous chapter.

In the following sections, we will introduce the integrated model as well as each component in detail. Experiments are also performed to verify the effectiveness and efficiency of this method.

4.2 Model

Our approach addresses both license plate detection and recognition by a single deep network. As illustrated in Figure 4.1, our model consists of a number of convolutional layers to extract discriminative features for license plates, a region proposal network tailored specifically for car license plates, a Region of Interest (RoI) pooling layer, multi-layer perceptrons for plate detection and bounding box regression, and RNNs with CTC for plate recognition. With this architecture, the plate detection and recognition can be achieved simultaneously, with one network and a single forward evaluation of the input image. Moreover, the whole network is trained end-to-end, with both localization loss and recognition loss being jointly optimized, and shows improved performance.

4.2.1 Low-level Feature Extraction

The VGG-16 network Simonyan and Zisserman [2015] which is pre-trained on ImageNet Russakovsky et al. [2015] is adopted here to extract low level CNN features. VGG-16 consists of 13 layers of 3×3 convolutions followed by Rectified Linear Unit (ReLU) non-linearity, 5 layers of 2×2 max-pooling, and fully connected layers. Here we keep all the convolutional layers and abandon the fully connected layers as we require local features at each position for plate detection. Given that the license plates are small compared with the whole image size, we use 2 pooling layers instead of 5, in case the feature information of license plates is vanished after pooling. So the resulting feature maps are one fourth size of the original input image. The higher-resolution feature maps will benefit the detection of small objects Redmon and Farhadi [2017]. They are used as a base for both detection and recognition.

4.2.2 Plate Proposal Generation

Ren et al. [2015] designed a Region Proposal Network (RPN) for object detection, which can generate candidate objects in images. RPN is a fully convolutional network that takes the low-level convolutional features as input, and outputs a set of potential bounding boxes. It is trained end-to-end so that high quality proposals can be generated. In this work, we modify RPN slightly to make it suitable for car license plate proposal.

Car license plates generally have a larger width than height, but the aspect ratios are nearly fixed for license plates in one country. According to the scales and aspect ratios of license plates in our datasets, we designed 6 scales anchors (the heights are respectively 5, 8, 11, 14, 17, 20) with an aspect ratio (width/height $= 5$), which results in $k = 6$ anchors at each position of the input feature maps. In addition, inspired by inception-RPN [Zhong et al., 2016], we use two 256-d rectangle convolutional filters ($W_1 = 5, H_1 = 3$ and $W_2 = 3, H_2 = 1$) instead of the regularly used one filter size 3×3, as shown in Figure 4.2. The two convolutional filters are applied simultaneously across each sliding position. The extracted local features are concatenated along the channel axis and formed a 512-d feature vector, which is then fed into two separate fully convolutional layers for plate/non-plate classification and box regression. On one hand, these rectangle filters are more suitable for objects with larger aspect ratios (i.e., license plates). On the other hand, the concatenated features keep both local and contextual information, which will benefit the plate classification later.

For k anchors at each sliding position on the feature map, the plate classification layer outputs $2k$ scores which indicate the probabilities of the anchors as license plates or not. The bounding box regression layer outputs $4k$ values which are the offsets of anchor boxes to a nearby ground-truth. Given an anchor with the center at (x_a, y_a), width w_a and height h_a, the regression layer outputs 4 scalars (t_x, t_y, t_w, t_h) which are the scale-invariant translations and log-space height/width shifts. The

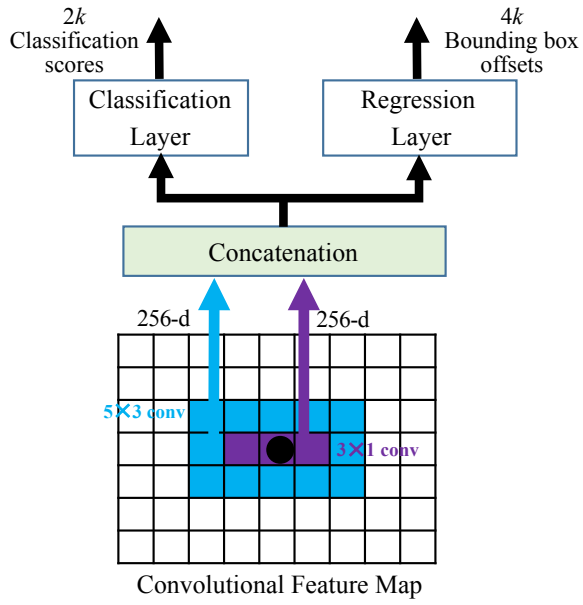


Figure 4.2: Plate Proposal Generation. Two rectangular convolutional filters are applied in each sliding window, which include rich contextual information. The features are concatenated and then fed into the classification layer for plate/non-plate classification and the regression layer to calculate coordinate offsets, with respect to k anchors at each position.

bounding box after regression is given by

$$\begin{aligned} x &= x_a + t_x w_a, \qquad y = y_a + t_y h_a, \\ w &= w_a \exp(t_w), \qquad h = h_a \exp(t_h), \end{aligned} \tag{4.1}$$

where x, y are the center coordinates of the bounding box after regression, and w, h are its width and height.

For a convolutional feature map with size $M \times N$, there will be $M \times N \times k$ anchors in total. Those anchors are redundant and highly overlapped with each other. Moreover, there are much more negative anchors than positive ones, which will lead to bias during training if we use all those anchors. We randomly sample 256 anchors from one image as a mini-batch, where the ratio between positive and negative anchors is up to $1:1$. The anchors that have Intersection over Union (IoU) scores larger than 0.7 with any ground-truth bounding boxes are selected as positives, while anchors with IoU lower than 0.3 as negatives. The anchors with the highest IoU scores with any ground-truth are also regarded as positives, so as to make sure that every ground-truth box has at least one positive anchor. If there are not enough positive anchors, we pad with negative ones.

The binary logistic loss is used here for box classification, and smooth L_1 loss [Ren et al., 2015] is employed for box regression. The multi-task loss function used for training RPN is

$$L_{RPN} = \frac{1}{N_{cls}} \sum_{i=1}^{N_{cls}} L_{cls}(p_i, p_i^\star) + \frac{1}{N_{reg}} \sum_{i=1}^{N_{reg}} L_{reg}(\mathbf{t}_i, \mathbf{t}_i^\star), \tag{4.2}$$

where N_{cls} is the size of a mini-batch and N_{reg} is the number of positive anchors in this batch. Bounding box regression is only for positive anchors, as there is no ground-truth bounding box matched with negative ones. p_i is the predicted probability of anchor i being a license plate and $p_i^\star$ is the corresponding ground-truth label (1 for positive anchor, 0 for negative anchor). $\mathbf{t}_i$ is the predicted coordinate offsets $(\mathbf{t}_{i,x}, \mathbf{t}_{i,y}, \mathbf{t}_{i,w}, \mathbf{t}_{i,h})$ for anchor i, and $\mathbf{t}_i^\star$ is the associated offsets for anchor i relative to the ground-truth. RPN is trained end-to-end with back-propagation and Stochastic Gradient Descent (SGD).

At test time, the forward evaluation of RPN will result in $M \times N \times k$ anchors with objectiveness scores as well as bounding box offsets. We employ Non-Maximum Suppression (NMS) to select 100 proposals with higher confidences based on the predicted scores. To be specific, we sort the proposed bounding boxes according to their classification scores firstly. The one who has an overlap ratio with another bounding box larger than 0.7, but has a smaller classification score as plate, will be suppressed. Then we choose the top 100 proposals with higher scores for the following processing.

4.2.3 Proposal Processing and Pooling

As we state before, 256 anchors are sampled from the $M \times N \times k$ anchors to train RPN. After bounding box regression, the 256 samples will later be used for plate detection and recognition.

We denote the bounding box samples as $p = (x^{(1)}, y^{(1)}, x^{(2)}, y^{(2)})$, where $(x^{(1)}, y^{(1)})$ is the top-left coordinate of the bounding box, and $(x^{(2)}, y^{(2)})$ is the bottom-right coor-

dinate of the bounding box. For all the positive proposals $p_{i,j} = (x_{i,j}^{(1)}, y_{i,j}^{(1)}, x_{i,j}^{(2)}, y_{i,j}^{(2)})$, $i = 1, \ldots, n$ that are associated with the same ground-truth plate g_j, a bigger bounding box $b_j = (x_j^{(1)}, y_j^{(1)}, x_j^{(2)}, y_j^{(2)})$ is constructed that encompasses all proposals $p_{i,j}$, i.e.,

$$\begin{cases} x_j^{(1)} = \min_{i=1,\ldots,n}(x_{i,j}^{(1)}), \\ y_j^{(1)} = \min_{i=1,\ldots,n}(y_{i,j}^{(1)}), \\ x_j^{(2)} = \max_{i=1,\ldots,n}(x_{i,j}^{(2)}), \\ y_j^{(2)} = \max_{i=1,\ldots,n}(y_{i,j}^{(2)}). \end{cases} \tag{4.3}$$

This process is also illustrated in Figure 4.3. The constructed bounding boxes b_j, $j = 1, \ldots, m$ will then be used as positive samples for later plate localization and recognition. To avoid the bias caused by the unbalanced distribution between positive and negative samples, we randomly choose $3m$ negative ones from the 256 samples and form a mini-batch with $4m$ samples.

Considering that the sizes of the samples are different from each other, in order to interface with the following plate detection network as well as the recognition network, RoI pooling [Girshick, 2015] is adopted here to extract a fixed-size feature representation from each sample. Each RoI is projected into the image convolutional feature maps, and results in feature maps of size $\mathrm{H}' \times \mathrm{W}'$. The varying size feature maps $\mathrm{H}' \times \mathrm{W}'$ are then divided into $\mathrm{X} \times \mathrm{Y}$ grids, where boundary pixels are aligned by rounding. Features are max-pooled within each grid. Here we choose $\mathrm{X} = 4$ and $\mathrm{Y} = 28$ instead of 7×7 that is used in [Girshick, 2015], because of the subsequent plate recognition task. To be specific, since we need to recognize each character in the license plate, it would be better if we keep more feature horizontally. However, the model size p from this layer to the next fully connected layer is closely related to X and Y, i.e., $p \propto XY$. A large feature map size will result in more parameters and increase the computation burden. After experimental analysis, we adopt a longer width $Y = 28$ and a shorter height $X = 4$.

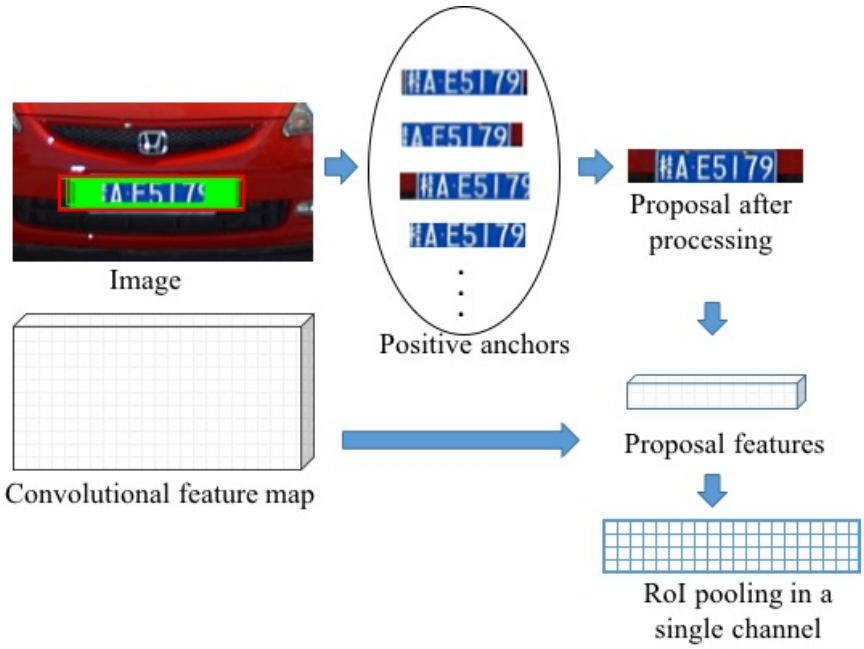


Figure 4.3: Proposal processing and pooling. Since some positive anchors (the green ones in the image) cannot cover all letters in the license plate, the corresponding features may not be included after RoI pooling, which is not suitable for recognition. Hence we construct a new proposal (the red one in the image) that encircles all positive anchors and big enough to contain sufficient features for recognition. RoI pooling is then followed to get fixed size feature maps. It should be note that this process is only implemented in training stage, while in test stage, the select 100 proposals with higher plate classification scores are processed directly for detection and recognition.

4.2.4 Plate Detection Network

Plate detection network aims to judge whether the resulted RoIs are car license plates or not, and refine the coordinates of the plate bounding boxes.

Two fully connected layers with 2048 neurons and a dropout rate of 0.5 are employed here to extract discriminative features for license plate detection. The features from each RoI are flattened into a vector and passed through the two fully connected layers. The encoded features are then fed concurrently into two separate linear transformation layers respectively for plate classification and bounding box regression. The plate classification layer has 2 outputs, which indicate the softmax probability of each RoI as plate/non-plate. The plate regression layer produces the bounding box

coordinate offsets for each proposal, as used in the region proposal network.

4.2.5 Plate Recognition Network

Plate recognition network aims to recognize each character in RoIs based on the extracted region features. Similarly, to avoid the challenging task of character segmentation, we regard the plate recognition as a sequence labeling problem as what we did in the previous chapter. Bidirectional RNNs (BRNNs) with CTC loss [Graves et al., 2009a] are employed to label the sequential features, which is illustrated in Figure 4.4.

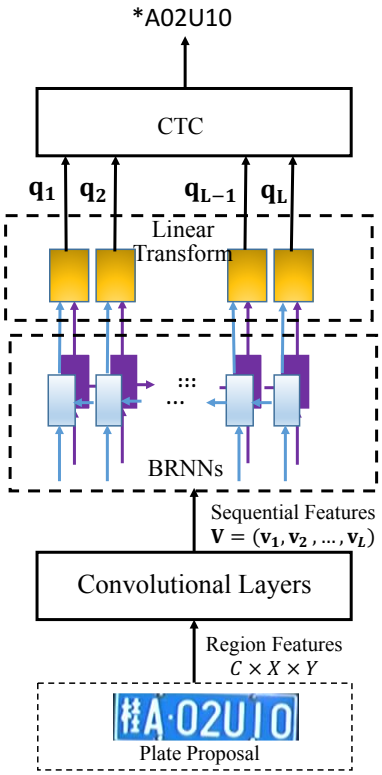


Figure 4.4: Plate Recognition Network. The pooled region features are regarded as a feature sequence, and encoded by BRNNs, which capture the contextual information in both sides. CTC is used for plate decoding without character separation.

The region features after RoI pooling are denoted as $\mathbf{Q} \in \mathbb{R}^{C \times X \times Y}$, where C is the channel size. First of all, we add two additional convolutional layers with ReLUs. Both of them have 512 filters. The kernel sizes are 3 and 2 respectively, with a padding of 1 used in the first convolutional layer. A rectangular pooling window with $k_W = 1$ and $k_H = 2$ is applied between them, which would be beneficial for

recognizing characters with narrow shapes, such as "1" and "I", referring to [Shi et al., 2015]. These operations will transform the region features $\mathbf{Q}$ into a sequence of the size $D \times L$, where $D = 512$ and $L = 19$. We denote the resulting features as $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_L)$, where $\mathbf{v}_i \in \mathbb{R}^D$.

Then BRNNs are applied on top of the sequential features. As presented in Figure 4.4, Two separated RNN layers with 512 units are used. One processes the feature sequence forward, with the hidden states updated via $\mathbf{h}_t^{(f)} = \mathrm{g}(\mathbf{v}_t, \mathbf{h}_{t-1}^{(f)})$. The other one processes it backward with the hidden states updated via $\mathbf{h}_t^{(b)} = \mathrm{g}(\mathbf{v}_t, \mathbf{h}_{t+1}^{(b)})$. The two hidden states are concatenated together and fed to a linear transformation with 37 outputs. A Softmax layer is followed to transform the 37 outputs into probabilities, which correspond to the distributions over 26 capital letters, 10 digits, and the space between characters. We record the probabilities at each time step. Hence, after BRNNs encoding, the feature sequence $\mathbf{V}$ is transformed into a sequence of probability estimation $\mathbf{q} = (\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_L)$ with the same length as $\mathbf{V}$. BRNNs capture abundant contextual information from both directions, which will make the character recognition more accurate. Similarly, LSTMs [Hochreiter and Schmidhuber, 1997] are adopted here, to overcome the shortcoming of gradient vanishing or exploding during traditional RNN training.

Then CTC layer [Graves et al., 2009a] is adopted for sequence decoding as we used in the previous chapter, which is to find an approximately optimal path $\mathbf{l}^*$ with the maximum probability through the Softmax output $\mathbf{q}$, i.e., ,

$$\mathbf{l}^* \approx \mathcal{B}\left(\arg\max_{\pi} P(\pi|\mathbf{q})\right). \tag{4.4}$$

Here a path π is a label sequence based on the output of Softmax layer, and $P(\pi|\mathbf{q}) = \prod_{t=1}^{L} P(\pi_t|\mathbf{q})$. The operator $\mathcal{B}$ is the operation of removing the repeated labels and the character spaces from the path, as we used in the previous chapter. The optimal label sequence $\mathbf{l}^*$ is exactly the recognized plate label.

4.2.6 Loss Functions and Training

As we demonstrate previously, the whole network takes as inputs an image, the plate bounding boxes and the associated labels during training time. After obtaining the samples as well as the region features, we combine the loss terms for plate detection and recognition, and jointly train the detection and recognition networks. Hence, the multi-task loss function is defined as

$$L_{DRN} = \frac{1}{N}\sum_{i=1}^{N} L_{cls}(p_i, p_i^\star) + \frac{1}{N_+}\sum_{i=1}^{N_+} L_{reg}(\mathbf{t}_i, \mathbf{t}_i^\star) + \frac{1}{N_+}\sum_{i=1}^{N_+} L_{rec}(\mathbf{q}^{(i)}, \mathbf{s}^{(i)}) \tag{4.5}$$

where N is the size of a mini-batch used in the detection network and N_+ is the number of positive samples in this batch. The definitions of L_{cls} and L_{reg} are the same as that used in RPN. $p_i, p_i^\star, \mathbf{t}_i, \mathbf{t}_i^\star$ also use the same definition as that used in RPN. $\mathbf{s}^{(i)}$ is the ground-truth plate label for sample i and $\mathbf{q}^{(i)}$ is the corresponding output sequence after BRNNs and Softmax.

It is observed that the length of BRNNs' outputs $\mathbf{q}^{(i)}$ is not consistent with the length of target label $\mathbf{s}^{(i)}$. Following CTC loss in [Graves et al., 2009a], the objective function for plate recognition is defined as the negative log probability of the network outputting correct label, i.e.,

$$L_{rec}(\mathbf{q}^{(i)}, \mathbf{s}^{(i)}) = -\log P(\mathbf{s}^{(i)}|\mathbf{q}^{(i)}) \tag{4.6}$$

where

$$P(\mathbf{s}^{(i)}|\mathbf{q}^{(i)}) = \sum_{\pi:\mathcal{B}(\pi)=\mathbf{s}^{(i)}} P(\pi|\mathbf{q}^{(i)}) \tag{4.7}$$

which is the sum of probabilities of all π that can be mapped to $\mathbf{s}^{(i)}$ by $\mathcal{B}$.

We use the approximate joint training process [Ren et al., 2015] to train the whole network, ignoring the derivatives with respect to the proposed boxes' coordinates. Fortunately, this does not have a great influence on the performance, as demonstrated in [Ren et al., 2015]. We train the whole network using SGD. CNNs for extracting

low-level features are initialized from the pre-trained VGG-16 model. We do not fine-tune the first four convolutional layers for efficiency. The rest of CNN layers are fine-tuned only in the first $50K$ iterations. The other weights are initialized according to Gaussian distribution. For optimization, we use ADAM [Kingma and Ba, 2014], with an initial learning rate of 10^{-5} for parameters in the pre-trained VGG-16 model, and 10^{-4} for other parameters. The latter learning rate is halved every $10K$ iterations until 10^{-5}. The network is trained for $200K$ iterations. Each iteration uses a single image sampled randomly from the training dataset. For each training image, we resize it to the shorter side of 700 pixels, while the longer side no more than 1500 pixels.

4.3 Experiments

In this section, we conduct experiments to verify the effectiveness of the proposed method. The network is implemented using Torch 7 and the experiments are performed on NVIDIA Titan X GPU with 12GB memory.

4.3.1 Datasets

In this phase, we obtained some more datasets that can be used to train the model and evaluate the effectiveness of our proposed method.

The first dataset is collected from China, denoted as "CarFlag-Large". We collected 460K images in total. The images are captured from frontal viewpoint by fixed surveillance cameras under different weather and illumination conditions, e.g., in sunny days, in rainy days, or at night, with a resolution of 1600×2048. The plates are nearly horizontal. Only the nearest license plate in the image is labeled in the ground-truth file. We use 322K images for training, and 138K images for test.

The second one is the AOLP database [Hsu et al., 2013], as we introduced before in the previous chapter. Considering the small number of training images, data augmentation is implemented by rotation and affine transformation.

The third dataset is Caltech-cars (Real) 1999 dataset car [2003], which is also used in the previous chapter. To test the effectiveness of our framework on this dataset, we collect extra 1626 images with America license plates from Dlagnekov [2015]; ope [2013] to train the end-to-end model. Data augmentation is implemented by translation and rescaling.

The fourth dataset is issued by Yuan *et al.* Yuan et al. [2017], and denoted as "PKUData". It has 3977 images with Chinese license plates captured from various scenes. This dataset is also used only for test, by applying the well-trained model from CarFlag-Large. It is categorized into 5 groups (i.e., , G1-G5) corresponding to different configurations. As there are only the plate bounding boxes given in the ground-truth file, we merely evaluate the detection performance on this dataset.

We summarize related information of all these datasets in Table 4.1 for reference.

Table 4.1: Car plate datasets summary

Dataset		Image quantity	Image resolution (height × width)	Number of training/test Images	Number of plate	Plate size (height × width)
CarFlag-Large		460000	1600 × 2048	322000/138000	138000	$(20 \sim 56) \times (85 \sim 265)$
AOLP	AC	2049	240 × 352	1368/681	681	$(25 \sim 32) \times (70 \sim 87)$
	LE		480 × 640	1292/757	757	$(28 \sim 48) \times (80 \sim 133)$
	RP		240 × 320	1438/611	611	$(30 \sim 58) \times (70 \sim 120)$
Caltech-cars		126	592 × 896	1626/126	126	$(23 \sim 59) \times (70 \sim 87)$
PKUData	G1	810	728 × 1082	322000/810	810	$(35 \sim 57) \times (145 \sim 184)$
	G2	700	728 × 1082	322000/700	700	$(30 \sim 62) \times (160 \sim 184)$
	G3	743	728 × 1082	322000/743	743	$(29 \sim 53) \times (145 \sim 184)$
	G4	572	1236 × 1600	322000/572	572	$(30 \sim 58) \times (158 \sim 170)$
	G5	1152	1200 × 1600	322000/1152	1438	$(20 \sim 60) \times (136 \sim 168)$

4.3.2 Evaluation Criteria

To evaluate the “End-to-end” performance with both detection and recognition results considered, we refer to the "End-to-end" evaluation protocol for general text spotting in natural scenes [Karatzas et al., 2015] as they have similar application scenario. The bounding box is considered to be correct if its IoU with the ground-truth bounding box is more than 50% (IoU $>$ 0.5), and the plate label matches, i.e., all characters in the plate are correctly recognized. F-measures are calculated and compared in our experiments which synthesize both precision and recall according to the

following equation 4.8:

$$\text{F-measure} = \frac{2 \times (\text{precision} \times \text{recall})}{(\text{precision} + \text{recall})} \tag{4.8}$$

It should be note that we denote all Chinese characters in license plates as "*", since the training images in CarFlag-Large are all from one province and use the same Chinese character. The trained network cannot be used to distinguish other Chinese characters.

The detection-only evaluation criterion is the same as what we used in the previous chapter for fair comparison, i.e., a detection is considered to be correct if the license plate is totally encompassed by the detected bounding box, and IoU > 0.5, refering to [Zhou et al., 2012; Yuan et al., 2017]

4.3.3 Performance Evaluation on CarFlag-Large

In this section, we run a number of ablations to analyze the network structure. A comparison experiment between our unified framework and a commonly used two-stage approach is also carried out to demonstrate the superiority of our end-to-end jointly trained network.

4.3.3.1 Network Structure Analysis

Feature size after RoI pooling: In order to determine the feature size X and Y after RoI pooling, a set of experiments are performed with different feature width Y. Experimental results in Table 4.2 show that a longer Y will result in a better performance, but accompany with more parameters. By considering both the performance and the model size, we choose $X = 4$ and $Y = 28$ in the following experiments.

Feature extraction method in plate recognition network: As we illustrated in section 4.2.5, two additional convolutional layers with ReLUs and rectangular poolings are added at the beginning of the plate recognition network, which convert the region feature map $\mathbf{Q}$ into a feature vector $\mathbf{V}$ for BRNNs processing. In Table 4.3, we

compare this to using solely average pooling or max pooling. 2-D average or max pooling with the kernel size as 4×4 and the step size as 1×1 is adopted before BRNNs respectively. Experimental results in Table 4.3 suggest that the additional CNN layers can learn more discriminative features and lead to a better end-to-end performance.

Table 4.2: Affect of feature map size after RoI pooling on performance. The result shows that the performance can be improved by a denser pooling. However, the model size increases accordingly ("M" means million). Balancing the model size and performance, we choose 4×28 in the following experiments.

Size	End-to-end Performance (%)	Model size in detection net
4×16	95.45	$73M$
4×20	96.13	$90M$
4×24	96.76	$107M$
4×28	**97.13**	$124M$
4×32	97.15	$141M$

Table 4.3: Influence of feature extraction method in plate recognition network on performance. The additional 2 convolutional layers plus rectangular max pooling lead to a better performance.

Method	End-to-end Performance (%)
Convs.+pooling	**97.13**
Average pooling	91.83
Max pooling	95.61

4.3.3.2 Unified *vs.* Stepwise Framework

As illustrated in Figure 4.5, a commonly used two-stage approach implements plate detection and recognition by two separated models. Plate detection is carried out firstly. The detected objects are cropped out and then recognized by another different model. In contrast, our proposed network outputs both detection and recognition results at the same time, with a single forward pass and requiring no image cropping. The convolutional features are shared by both detection and recognition net-

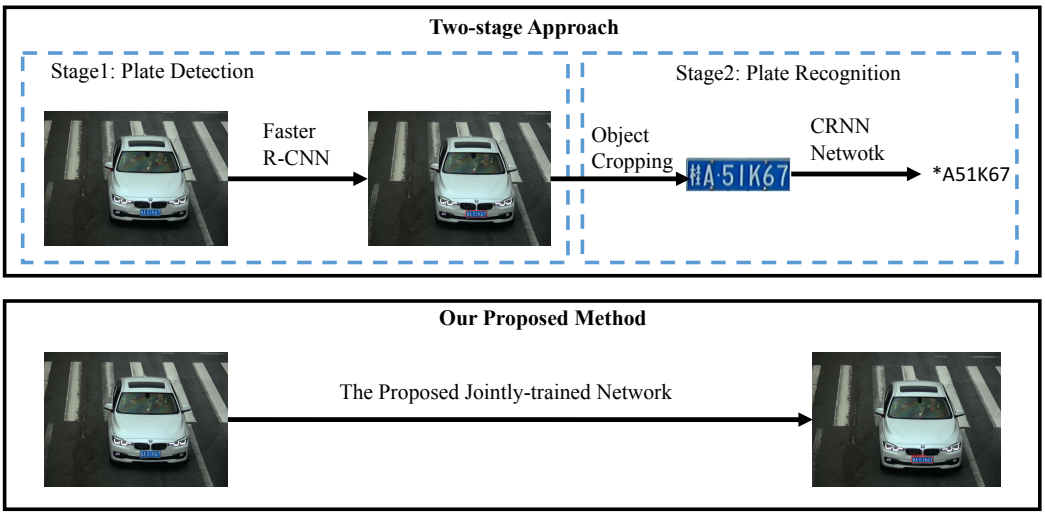


Figure 4.5: A two-stage approach VS. our proposed method. In the two-stage approach, after license plate detection by Faster R-CNN, we crop the detected license plates from the image, and then recognize them by another separate model (fine-tuned "CRNN" [Shi et al., 2015] in this paper). The features need to be re-computed during recognition phase. In contrast, our proposed unified network takes an image as input, and produces license plate bounding boxes and plate labels in one-shot. It avoids some intermediate processes like image cropping, and share computation for convolutional feature extraction.

works, which omits feature re-computation. For fair comparison with the proposal based detection framework, instead of adopting the sliding window based detection method that we designed in the previous chapter, we train another independent Faster R-CNN model [Ren et al., 2015] solely for car license plate detection, without recognition. We denote it as "Ours (Detection-only)". The two-stage approach we used in this chapter is denoted as "Ours (New-Two-stage)" and the jointly trained network is "Ours (Jointly-trained)".

In "Ours (New-Two-stage)", the detection only Faster R-CNN model [Ren et al., 2015] ("Ours (Detection-only)") is trained by the 322K training images too. We modify the scales and shapes of anchors as the ones we used in section 4.2.2 so as to fit the license plates. The network is also trained with $200K$ iterations, using the same initial parameters and learning rate. As to the plate recognition, we employ "CRNN" framework [Shi et al., 2015], which produces the state-of-the-art performance on gen-

eral text recognition. It is an end-to-end framework for cropped word recognition, including CNN layers, RNN layers and CTC for transcription, from bottom to top. We crop the groud-truth license plates from the 322K training images, and resize them to 160×32 pixels. Then we fine-tune the "CRNN" model with these training data.

In order to boost the performance, we rescale the input image into multiple sizes in test phase for both our proposed network "Ours (Jointly-trained)" and the detection-only Faster R-CNN network "Ours (Detection-only)". The input images are resized to the shorter side of 600, 1200 pixels respectively, with the longer side less than 1500 pixels. With "Ours (Jointly-trained)", both detection and recognition results come out together, while with "Ours (New-Two-stage)", we have to crop the detected bounding boxes from the input images, resize them to 160×32 pixels, and then feed into the well-trained "CRNN" model for recognition. Only bounding boxes with classification scores larger than 0.95 are kept and merged via NMS, with the overlap ratio threshold setting to 0.7. Considering that there is only one plate labeled as ground-truth per image, we finally choose the one that has 7 characters recognized and/or with the highest detection score for evaluation. The evaluation results are presented in Table 4.4. "Ours (Jointly-trained)" gives the "End-to-end" performance with F-measure of 97.13% on 138K test images. It is around 3% higher than the result produced by "Ours (New-Two-stage)", which demonstrates the advantage of end-to-end training for both detection and recognition in a unified network. The learned features are more informative, and the two subtasks can promote each other.

In terms of the computational speed, the unified framework takes about 310*ms* per image for a forward evaluation on the single small input scale, while the two-stage approach needs around 450*ms* to get both detection and recognition results, as it needs to implement image cropping and CNN feature re-calculation.

We also compare the detection-only performance. "Ours (Jointly-trained)" produces a detection accuracy of 98.33%, which is 1% higher than the result given by

Table 4.4: Experimental results on CarFlag-Large dataset. We compare both performance and running speed of our jointly trained network with a two-stage baseline method based on "Faster R-CNN + CRNN" for fair comparison. F-measures are presented here in percentage. The jointly trained network not only achieves higher accuracies on both detection and "End-to-end" performance, but also in a shorter time.

Method	End-to-end Performance (%)	Detection-only Performance (%)	End-to-end Speed (per image single scale) (ms)
Ours (Jointly-trained)	**97.13**	**98.33**	**310**
Ours (Two-stage)	94.09	97.05	450

"Ours (Detection-only)". This result illustrates that car license plate detection can be improved with the multi-task loss used during training phase.

Some experimental results using "Ours (Jointly-trained)" are presented in Figure 4.6, which show that the jointly trained model can deal with images under different illumination conditions.

4.3.4 Performance Evaluation on AOLP

In this subsection, we compare the "End-to-end" performance of our method with other state-of-the-art methods on AOLP dataset. Note that the network is only trained with 15K iterations because of the small number of training images in this dataset. Moreover, since the sizes of license plates in AOLP are almost the same, and the ratios between license plates and images sizes are also similar, for this dataset, we only use a single image scale with the shorter side as 700 pixels in test phase.

The detection and recognition results are presented in Figure 4.7. Comparison results with other methods in Table 4.5 show that "Ours (Jointly-trained)" performs better on AC and LE subsets under "End-to-end" evaluation. It also produces the best performance for plate detection on all three subsets, with averagely 2% higher than the results in the previous chapter, and 4% higher than the results by the edge

Figure 4.6: Experimental results on CarFlag-Large dataset by our jointly trained model. The detection and recognition results are presented in red bounding boxes, with red labels above. The third line demonstrates some failure cases, which shows that the car license plate cannot be recognized correctly if it is too dark or partially occluded.

Table 4.5: Experimental results on AOLP dataset. We compare our proposed method with other state-of-the-art methods on both performance and running speed. The performance is presented here in F-measures. Our jointly-trained network shows improved performance for images with license plates in nearly horizontal position.

Method	End-to-end Performance (%)			Detection-only Performance (%)			End-to-end Speed (per image single scale) (ms)
	AC	LE	RP	AC	LE	RP	
Hsu et al. [2013]	–	–	–	93.43	92.96	92.48	**260**
The stepwise method in the previous chapter	94.85	94.19	**88.38**	98.45	97.69	95.43	1000-2000
Ours (Jointly-trained)	**95.29**	**96.57**	83.63	**99.56**	**99.34**	**98.85**	400

based method used in [Hsu et al., 2013]. As to the computational speed, out network takes about 400*ms* to get both detection and recognition results, while the stepwise

Figure 4.7: Experimental results on AOLP dataset by our jointly trained model. The red bounding box and red label in each image present the detection and recognition results respectively. The third line shows some failure cases.

method described in previous chapter costs 2-3s. Hsu et al. [2013]'s method needs averagely 260ms since it adopts a shallow network.

It should be noted that in Table 4.5, the "End-to-end" performance on RP subset is worse than that in the previous chapter. This is because the license plates in RP have a large degree of rotation and projective orientation. In the previous chapter, the detected license plates are cropped out and Hough transform is adopted to correct the orientation. Nevertheless, the integrated method in this chapter does not explicitly handle the rotated plates. Integrating the spatial transform network into our end-to-end framework may be a solution, referring to [Shi et al., 2016], which is a future work.

4.3.5 Performance Evaluation on Caltech-cars

To further evaluate the effectiveness of the proposed framework, in this section, we test the "End-to-end" performance of our method on the Caltech-cars dataset car [2003], which has license plates from America. We fine-tune the model trained by CarFlag-Large dataset with about 1626 training images and test the model on Caltech-cars. Experimental results in Table 4.6 show that our framework is also feasible for detecting and recognizing license plates from America. It achieves the best detection performance compared to the previous methods. It should be note that this dataset does not include any training data. Although we collect 1626 images with America car license plates, they are not very consistent with each other. We believe the end-to-end performance can be further improved if we have enough consistent training data. Experimental results on Caltech-cars are presented in Figure 4.8. Compared to the results in the previous chapter, the generated bounding boxes tend to regard the license plate as a whole object, and lead to much better result. Failure cases in the third line of Figure 4.8 show that the model fails when the image is too dark or overexposure. The license plate with unusual characters cannot be correctly recognized.

Table 4.6: Experimental results on Caltech-cars dataset. Our method achieves the best detection performance compared to the previous methods.

Method	Detection-only Performance (%)	End-to-end Performance (%)
Zhou *et al.* Zhou et al. [2012]	89.83	-
Tian *et al.* Tian et al. [2017]	90.70	-
Kim *et al.* Kim et al. [2017]	97.60	-
The stepwise method in the previous chapter	96.38	-
Ours(Jointly-trained)	**98.04**	**94.12**

Figure 4.8: Experimental results on Caltech-cars dataset by our jointly trained model. The red bounding box and red label in each image present the detection and recognition results respectively. The third line shows some failure cases.

Table 4.7: Experimental results on PKUData. G1 - G5 correspond to different image capturing conditions. Our jointly trained network achieves a average detection ratio of 99.73%, which is 2% higher than the previous best performance. In addition, the jointly trained network performs better than that trained only with the detection information.

Method	Detection Performance (%)						Detection Speed (per image single scale)	End-to-end Speed (per image single scale)
	G1	G2	G3	G4	G5	Average		
Zhou et al. [2012]	95.43	97.85	94.21	81.23	82.37	90.22	475 ms	-
Li et al. [2013]	98.89	98.42	95.83	81.17	83.31	91.52	672 ms	-
Yuan et al. [2017]	98.76	98.42	97.72	96.23	97.32	97.69	**42** ms	-
Ours(Detection-only)	**99.88**	99.71	99.46	99.83	98.68	99.51	283 ms	-
Ours(Jointly-trained)	**99.88**	**99.86**	**99.60**	**100**	**99.31**	**99.73**	279 ms	310 ms

4.3.6 Performance Evaluation on PKUData

Because the ground-truth file in PKUData only provides the plate bounding boxes, we simply evaluate the detection performance on this dataset. Both the detection accuracy and computing speed are compared with other methods [Li et al., 2013;

Zhou et al., 2012; Yuan et al., 2017]. We use the same model trained by the CarFlag-Large dataset, as they are both datasets with Chinese license plates.

Images in Figure 4.9 show examples with both detection and recognition results. The detection-only results by our approach and other three methods are presented in Table 4.7. Our jointly trained model demonstrates absolute advantage on all 5 subsets, especially on G4, where we achieve 100% detection rate. This result proves the robustness of our approach in face of various scenes and diverse conditions. Qualitatively, our jointly trained network achieves a average detection ratio of 99.73%, which is 2% higher than the previous best performance.

Figure 4.9: Experimental results on PKUData by our jointly trained model. The red bounding box and red label in each image present the detection and recognition results respectively. The third line shows some failure cases, where plates are not completely detected.

In addition, the detection performance by our jointly trained network is slightly better than that by the detection-only network as seen from Table 4.7. This is consis-

tent with the outcome on CarFlag-Large dataset, and proves again that the detection performance can be boosted when training with the label information.

In terms of the computational speed, Yuan *et al.*'s method [Yuan et al., 2017] is relatively faster than ours', because they use simple linear SVMs, while we adopt deep CNNs and RNNs.

4.4 Conclusion

In this chapter we have presented a jointly trained network which integrates CNNs and RNNs for simultaneous car license plate detection and recognition. With this network, car license plates can be detected and recognized all at once in a single forward pass, with both high accuracy and efficiency. By sharing convolutional features with both detection and recognition networks, the model size decreases largely. The whole network can be trained approximately end-to-end, without intermediate processing like image cropping or character separation. The whole computational time decreases a lot compared to the stepwise method used in the previous chapter. Comprehensive evaluation and comparison on three datasets with different approaches validate the advantage of our method.

In the future, we will extend our network to process multi-oriented car license plates by integrating the spatial transform network [Jaderberg et al., 2015]. On the other hand, although the processing speed accelerates a lot, it remains to be improved. After time analysis, it is found that NMS takes about half of the whole processing time. Hence, we will optimize the code on NMS to speed up the whole process. In addition, existing researches show that the direct regression based methods, such as YOLO or SSD, are relatively faster than the proposal based methods. So we will also try to exploit the direct regression based architectures to accelerate the detection and recognition speed furthermore.

Chapter 5

Towards End-to-end Text Spotting with Convolutional Recurrent Neural Networks

5.1 Introduction

As we described in Chapter 1.1.3, previous methods on text spotting can be categorized into stepwise or integrated manners. Stepwise methods conduct text detection firstly with a high recall to get candidate text regions. Word recognition is performed on the cropped text bounding boxes by a totally different model, in accompany with text line separation or character grouping sometimes. Integrated methods, on the other hand, tackle both word detection and recognition jointly. However, a separate character detection step is usually needed to be carried out in the existing integrated methods [Wang et al., 2011; Neumann and Matas, 2013a]. The detected characters are then grouped into words and recognized simultaneously. In this chapter, we leave out the character detection step furthermore, and propose a unified framework to detect and recognize word level bounding boxes directly and concurrently. As a matter of fact, the tasks of word detection and recognition are highly correlated. Firstly, the feature information can be shared between them. Furthermore, these two tasks can complement each other: better detection improves recognition accuracy, and the recognition information can refine detection results vice versa.

In the previous chapter, we have proposed a unified deep neural network for simultaneous car license plate detection and recognition. However, general scene text is more flexible and varied than car license plates, and with some special characteristics. For example, in contrast to the nearly fixed aspect ratio of license plates in one country, the length of words may vary drastically, like the words of "I" and "congratulations". In addition, there is a kind of a potential language model embedded in words, while the characters appeared in car license plates are totally random.

With these particularities into mind, we propose an end-to-end trainable text spotter in this chapter, which jointly detects and recognizes words in an image. An overview of the network architecture is presented in Figure 5.1. It consists of a number of convolutional layers, a region proposal network tailored specifically for text (refer to as Text Proposal Network, TPN), an Recurrent Neural Network (RNN) encoder for embedding proposals of varying sizes to fixed-length vectors, multi-layer perceptrons for detection and bounding box regression, and an attention-based RNN decoder for word recognition. Via this framework, both text bounding boxes and word labels are generated with a single forward evaluation of the network. We do not need to process the intermediate issues such as character grouping [Zhu and Zanibbi, 2016; Tian et al., 2015] or text line separation [Zhang et al., 2015], and thus avoid the error accumulation. The whole network takes an image as input and outputs the coordinates of word bounding boxes as well as text labels directly and simultaneously, with both high efficiency and accuracy. The main contributions are thus three-fold.

(1) An end-to-end trainable DNN is designed to optimize the overall accuracy and share computations. The network integrates both text detection and word recognition. With the end-to-end optimization of multiple tasks, the learned features are more informative, which can promote the detection results as well as the overall performance. The convolutional features are shared by both detection and recognition, which saves processing time. To our best knowledge, this is the first successful

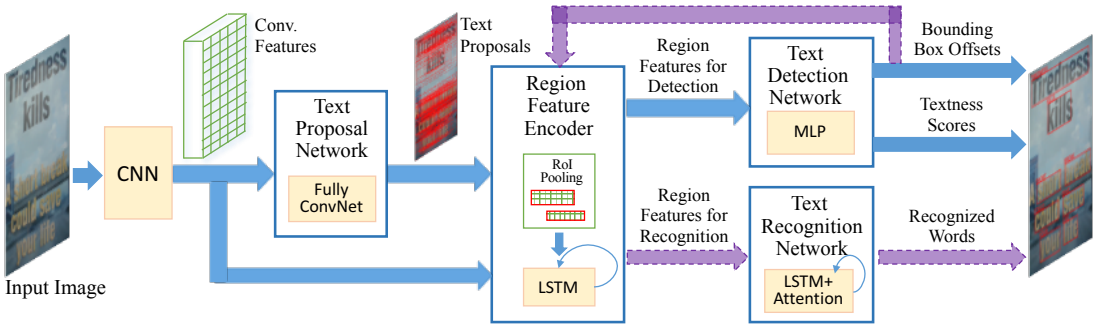


Figure 5.1: Model overview. The network takes an image as input, and outputs both text bounding boxes and text labels in one forward pass. The whole network is trained end-to-end.

attempt to integrate word level detection and recognition into a single end-to-end trainable network. It should be note that as we mentioned in Chapter 1.1.3, Busta et al. [2017] also proposed a unified text localization and recognition framework. However, in their architecture, the convolutional features are not shared. It is only the multi-task loss that be jointly optimized. The detected bounding boxes need to be cropped out from the image as usual, and features are extracted again by another recognition network.

(2) We propose a new method for region feature extraction. In previous work [Girshick, 2015; Ren et al., 2015] as well as our integrated network for car license plate detection and recognition in Chapter 4, Region-of-Interest (RoI) pooling layer converts regions of different sizes and aspect ratios into feature maps with a fixed size. Considering the significant diversity of aspect ratios in text bounding boxes, it is sub-optimal to fix the size after pooling. To accommodate the original aspect ratios and avoid distortion, RoI pooling is tailored to generate feature maps with varying lengths. An RNN encoder is then employed to encode feature maps of different lengths into the same size.

(3) A curriculum learning strategy is designed to train the system with gradually more complex training data. Starting from synthetic images with simple appearance and a large word lexicon, the system learns a character-level language model and

finds a good initialization of appearance model. By employing real-world images with a small lexicon later, the system gradually learns how to handle complex appearance patterns. We conduct a set of experiments to explore the capabilities of different model structures. The best model outperforms state-of-the-art results on several standard benchmarks, including ICDAR2011, 2015.

Notation For legibility, the following notations are defined. All bold capital letters represent matrices and all bold lower-case letters denote column vectors. $[\mathbf{a}; \mathbf{b}]$ concatenates the vectors $\mathbf{a}$ and $\mathbf{b}$ vertically, while $[\mathbf{a}, \mathbf{b}]$ stacks $\mathbf{a}$ and $\mathbf{b}$ horizontally (column wise). In the following sections, the bias terms in neural networks are omitted.

5.2 Model

Our goal is to design an end-to-end trainable network, which simultaneously detects and recognizes all words in images. Our model is motivated by recent progresses in DNN models such as Faster R-CNN [Ren et al., 2015] and sequence-to-sequence learning [Shi et al., 2016; Lee and Osindero, 2016], but we take the special characteristics of text into consideration.

The whole system architecture is illustrated in Figure 5.1. Firstly, the input image is fed into a convolutional neural network that is modified from VGG-16 net [Simonyan and Zisserman, 2015]. VGG-16 consists of 13 layers of 3×3 convolutions followed by Rectified Linear Units (ReLUs), 5 layers of 2×2 max-pooling, and Fully-Connected (FC) layers. Here we remove FC layers. As long as text in images can be relatively small, we only keep the 1st, 2nd and 4th layers of max-pooling, so that the down-sampling ratio is increased from $1/32$ to $1/8$.

Given the computed convolutional features, TPN provides a list of text region proposals (bounding boxes). Then, Region Feature Encoder (RFE) converts the convolutional features of proposals into fixed-length representations. These representations are further fed into Text Detection Network (TDN) to calculate their textness

scores and bounding box offsets. Next, RFE is applied again to compute fixed-length representations of text bounding boxes provided by TDN (see purple paths in Figure 5.1). Finally, Text Recognition Network (TRN) recognizes words in the detected bounding boxes based on their representations. In the following subsections, we will present a detailed description about each part.

5.2.1 Text Proposal Network

Text Proposal Network (TPN) is inspired from RPN [Ren et al., 2015; Zhong et al., 2016], which can be regarded as a fully convolutional network. As presented in Figures 5.2, it takes convolutional features as input, and outputs a set of bounding boxes accompanied with textness scores and coordinate offsets which indicate scale-invariant translations and log-space height/width shifts relative to pre-defined anchors, as in [Ren et al., 2015].

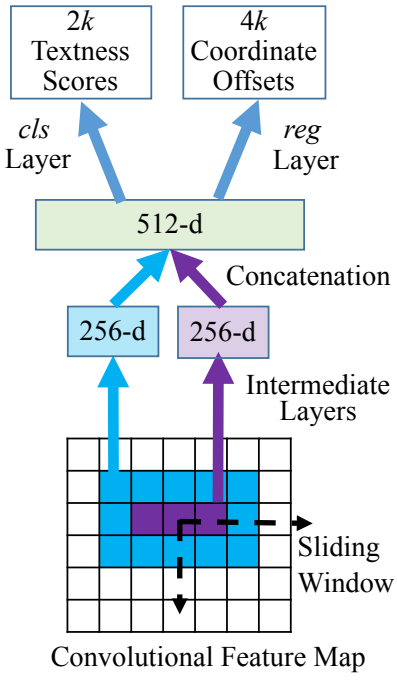


Figure 5.2: Text Proposal Network (TPN). We apply multiple scale sliding windows over the convolutional feature maps. Both local and contextual information are retained which helps to propose high quality text bounding boxes. The concatenated local and contextual features are further fed into the *cls* layer for computing textness scores and the *reg* layer to calculate coordinate offsets, with respect to k anchors at each position.

Considering that word bounding boxes usually have larger aspect ratios (W/H) and varying scales, we designed $k = 24$ anchors with 4 scales (with box areas of 16^2, 32^2, 64^2, 80^2) and 6 aspect ratios ($1:1$, $2:1$, $3:1$, $5:1$, $7:1$, $10:1$).

Inspired by the work in [Zhong et al., 2016], we apply two 256-d rectangle convolutional filters of different sizes ($W = 5, H = 3$ and $W = 3, H = 1$) on the feature maps to extract both local and contextual information. The rectangle filters lead to wider receptive fields, which is more suitable for word bounding boxes with large aspect ratios. The resulting features are further concatenated to 512-d vectors and fed into two sibling layers for text/non-text classification and bounding box regression.

5.2.2 Region Feature Encoder

To process RoIs of different scales and aspect ratios in a unified way, most existing work re-samples regions into *fixed-size* feature maps via pooling [Ren et al., 2015]. However, for text, this approach may lead to significant distortion due to the large variation of word lengths. For example, it may be unreasonable to encode short words like "Dr" and long words like "congratulations" into feature maps of the same size. In this work, we propose to re-sample regions according to their respective aspect ratios, and then use RNNs to encode the resulting feature maps of different lengths into fixed length vectors. The whole region feature encoding process is illustrated in Figure 5.3.

For an RoI of size $h \times w$, we perform spatial max-pooling with a resulting size of

$$H \times \min(W_{max}, 2Hw/h), \tag{5.1}$$

where the expected height H is fixed and the width is adjusted to keep the aspect ratio as $2w/h$ (twice the original aspect ratio) unless it exceeds the maximum length W_{max}. Note that here we employ a pooling window with an aspect ratio of $1 : 2$, which benefits the recognition of narrow shaped characters, like "i", "l", etc., as stated in [Shi et al., 2015].

Next, the resampled feature maps are considered as a sequence and fed into RNNs for encoding. Here we also use LSTMs instead of vanilla RNNs to overcome the shortcoming of gradient vanishing or exploding. The feature maps af-

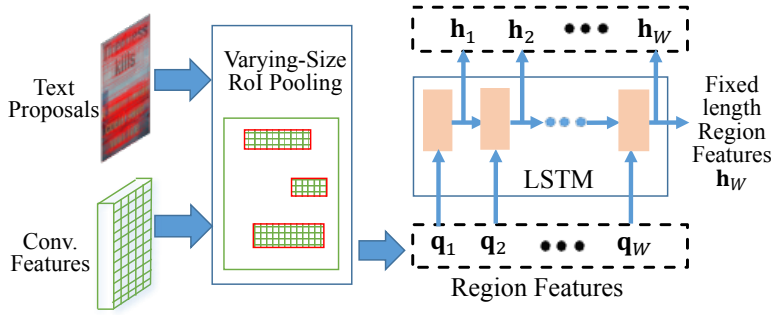


Figure 5.3: Region Features Encoder (RFE). The region features after RoI pooling are not required to be of the same size. In contrast, they are calculated according to aspect ratio of each bounding box, with height normalized. LSTM is then employed to encode different length region features into the same size.

ter the above varying-size RoI pooling are denoted as $\mathbf{Q} \in \mathbb{R}^{C\times H\times W}$, where $W = \min(W_{max}, 2Hw/h)$ is the number of columns and C is the channel size. We flatten the features in each column, and obtain a sequence $\mathbf{q}_1, \ldots, \mathbf{q}_W \in \mathbb{R}^{C\times H}$ which are fed into LSTMs one by one. Each time LSTM units receive one column of feature $\mathbf{q}_t$, and update their hidden state $\mathbf{h}_t$ by a non-linear function: $\mathbf{h}_t = \mathrm{f}(\mathbf{q}_t, \mathbf{h}_{t-1})$. In this recurrent fashion, the final hidden state $\mathbf{h}_W$ (with size $R = 1024$) captures the holistic information of $\mathbf{Q}$ and is used as an RoI representation with fixed dimension.

5.2.3 Text Detection and Recognition

Text Detection Network (TDN) aims to judge whether the proposed RoIs are text or not and refine the coordinates of bounding boxes once again, based on the extracted region features $\mathbf{h}_W$. Two fully-connected layers with 2048 neurons are applied on $\mathbf{h}_W$, followed by two parallel layers for classification and bounding box regression respectively.

The classification and regression layers used in TDN are similar to those used in TPN. Note that the whole system refines the coordinates of text bounding boxes twice: once in TPN and then in TDN. Although RFE is employed twice to calculate features for proposals produced by TPN and later the detected bounding boxes provided by TDN, the convolutional features only need to be computed once.

Text Recognition Network (TRN) aims to predict the text in the detected bounding boxes based on the extracted region features. As shown in Figure 5.4, we adopt LSTMs with attention mechanism [Luong et al., 2015; Shi et al., 2016] to decode the sequential features into words.

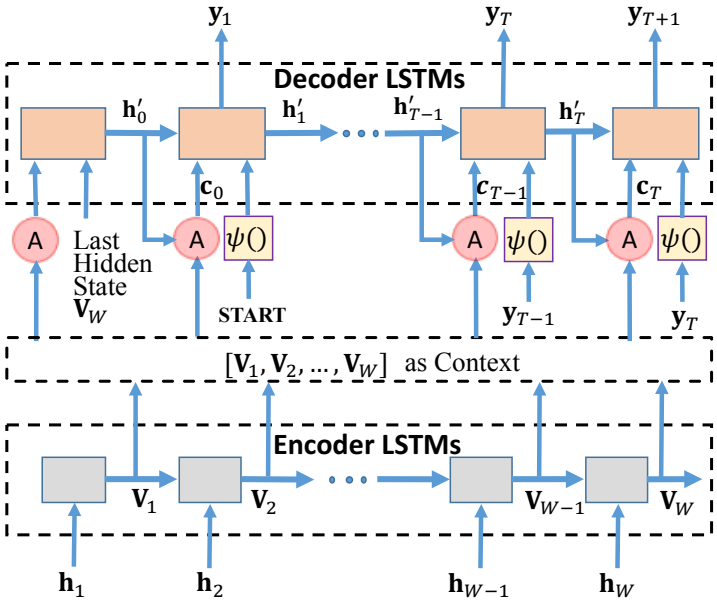


Figure 5.4: Text Recognition Network (TRN). The region features are encoded by one layer of LSTMs, and then decoded in an attention based sequence to sequence manner. Hidden states of encoder at all time steps are reserved and used as context for attention model.

Firstly, hidden states at all steps $\mathbf{h}_1, \ldots, \mathbf{h}_W$ from RFE are fed into an additional layer of LSTMs with 1024 units to encode once more. We record the hidden state at each time step and form a sequence of $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_W] \in \mathbb{R}^{R \times W}$. It includes local information at each time step and works as the context for the attention model.

As for decoder LSTMs, the ground-truth word label is adopted as input during training phase. It can be regarded as a sequence of tokens $\mathbf{s} = \{s_0, s_1, \ldots, s_{T+1}\}$ where s_0 and s_{T+1} represent the special tokens START and END respectively. We feed decoder LSTMs with $T+2$ vectors: $\mathbf{x}_0$, $\mathbf{x}_1$, ..., $\mathbf{x}_{T+1}$, where $\mathbf{x}_0 = [\mathbf{v}_W; \text{Atten}(\mathbf{V}, \mathbf{0})]$ is the concatenation of the encoder's last hidden state $\mathbf{v}_W$ and the attention output with the guidance equals to zero; and $\mathbf{x}_i = [\psi(s_{i-1}); \text{Atten}(\mathbf{V}, \mathbf{h}'_{i-1})]$, for $i = 1, \ldots, T+1$, is

made up of the embedding $\psi()$ of the $(i-1)$-th token s_{i-1} and the attention output guided by the hidden state of decoder LSTMs in the previous time-step $\mathbf{h}'_{i-1}$. The embedding function $\psi()$ is defined as a linear layer followed by a tanh non-linearity.

The attention function $\mathbf{c}_i = \text{Atten}(\mathbf{V}, \mathbf{h}'_i)$ is defined as follows:

$$\begin{cases} \mathbf{g}_j = \tanh(\mathbf{W}_v \mathbf{v}_j + \mathbf{W}_h \mathbf{h}'_i), \ j = 1, \ldots, W, \\ \boldsymbol{\alpha} = \text{Softmax}(\mathbf{w}_g^\top \cdot [\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_W]), \\ \mathbf{c}_i = \sum_{j=1}^{W} \alpha_j \mathbf{v}_j, \end{cases} \tag{5.2}$$

where $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_W]$ is the variable-length sequence of features to be attended, $\mathbf{h}'_i$ is the guidance vector, $\mathbf{W}_v$ and $\mathbf{W}_h$ are linear embedding weights to be learned, $\boldsymbol{\alpha}$ is the attention weights of size W, and $\mathbf{c}_i$ is a weighted sum of input features.

At each time-step $t = 0, 1, \ldots, T+1$, the decoder LSTMs compute their hidden state $\mathbf{h}'_t$ and output vector $\mathbf{y}_t$ as follows:

$$\begin{cases} \mathbf{h}'_t = \text{f}(\mathbf{x}_t, \mathbf{h}'_{t-1}), \\ \mathbf{y}_t = \varphi(\mathbf{h}'_t) = \text{Softmax}(\mathbf{W}_o \mathbf{h}'_t) \end{cases} \tag{5.3}$$

where LSTM is used for the recurrence formula $\text{f}()$, and $\mathbf{W}_o$ linearly transforms hidden states to the output space of size 38, including 26 case-insensitive characters, 10 digits, a token representing all punctuations like "!" and "?", and a special END token.

At test time, the token with the highest probability in previous output $\mathbf{y}_t$ is selected as the input token at step $t+1$, instead of the ground-truth tokens $s_1, \ldots, s_T$. The process is started from the START token, and repeated until we get the special END token.

5.2.4 Loss Functions and Training

Loss Functions As we demonstrate above, our system takes as input an image, word bounding boxes and their labels during training. Both TPN and TDN employ the binary logistic loss L_{cls} for classification, and smooth L_1 loss L_{reg} [Ren et al., 2015] for regression. So the loss for training TPN is

$$L_{TPN} = \frac{1}{N}\sum_{i=1}^{N} L_{cls}(p_i, p_i^\star) + \frac{1}{N_+}\sum_{i=1}^{N_+} L_{reg}(\mathbf{d}_i, \mathbf{d}_i^\star), \tag{5.4}$$

where N is the number of randomly sampled anchors in a mini-batch and N_+ is the number of positive anchors in this batch (the range of positive anchor indices is from 1 to N_+). The mini-batch sampling and training process of TPN are similar to that used in [Ren et al., 2015]. An anchor is considered as positive if its Intersection-over-Union (IoU) ratio with a ground-truth is greater than 0.7 and considered as negative if its IoU with any ground-truth is smaller than 0.3. In this paper, N is set to 256 and N_+ is at most 128. p_i denotes the predicted probability of anchor i being text and $p_i^\star$ is the corresponding ground-truth label (1 for text, 0 for non-text). $\mathbf{d}_i$ is the predicted coordinate offsets $(\mathrm{dx}_i, \mathrm{dy}_i, \mathrm{dw}_i, \mathrm{dh}_i)$ for anchor i, and $\mathbf{d}_i^\star$ is the associated offsets for anchor i relative to the ground-truth. Bounding box regression is only for positive anchors, as there is no ground-truth bounding box matched with negative ones.

For the final outputs of the whole system, we apply a multi-task loss for both detection and recognition:

$$L_{DRN} = \frac{1}{\hat{N}}\sum_{i=1}^{\hat{N}} L_{cls}(\hat{p}_i, \hat{p}_i^\star) + \frac{1}{\hat{N}_+}\sum_{i=1}^{\hat{N}_+} L_{reg}(\hat{\mathbf{d}}_i, \hat{\mathbf{d}}_i^\star) + \frac{1}{\hat{N}_+}\sum_{i=1}^{\hat{N}_+} L_{rec}(\mathbf{Y}^{(i)}, \mathbf{s}^{(i)}) \tag{5.5}$$

where $\hat{N} = 128$ is the number of text proposals sampled from the output of TPN, and $\hat{N}_+ \leq 64$ is the number of positive ones. The thresholds for positive and negative anchors are set to 0.6 and 0.4 respectively, which are less strict than those used for training TPN. In order to mine hard negatives, we first apply TDN on 1000 randomly sampled negatives and select those with higher textness scores. $\hat{p}_i$ and $\hat{\mathbf{d}}_i$ are the outputs

of TDN. $\mathbf{s}^{(i)}$ is the ground-truth tokens for sample i and $\mathbf{Y}^{(i)} = \{\mathbf{y}_0^{(i)}, \mathbf{y}_1^{(i)}, \ldots, \mathbf{y}_{T+1}^{(i)}\}$ is the corresponding output sequence of decoder LSTMs. $L_{rec}(\mathbf{Y}, \mathbf{s}) = -\sum_{t=1}^{T+1} \log \mathbf{y}_t(s_t)$ denotes the cross entropy loss on $\mathbf{y}_1, \ldots, \mathbf{y}_{T+1}$, where $\mathbf{y}_t(s_t)$ represents the predicted probability of the output being s_t at time-step t and the loss on $\mathbf{y}_0$ is ignored.

Following [Ren et al., 2015], we use an approximate joint training process to minimize the above two losses together (ADAM [Kingma and Ba, 2014] is adopted), ignoring the derivatives with respect to the proposed boxes' coordinates.

Data Augmentation We sample one image per iteration in the training phase. Training images are resized to the shorter side of 600 pixels and the longer side of at most 1200 pixels. Data augmentation is also implemented to improve the robustness of our model, which includes:

1) randomly rescaling the width of the image by ratio 1 or 0.8 without changing its height, so that the bounding boxes have more variable aspect ratios;

2) randomly cropping a subimage which includes all text in the original image, padding with 100 pixels on each side, and resizing to 600 pixels on shorter side.

Curriculum Learning In order to improve generalization and accelerate the convergence speed, we design a curriculum learning [Bengio et al., 2009] paradigm to train the model from gradually more complex data.

1) We generate 48k images containing words in the "Generic" lexicon [Jaderberg et al., 2016] of size 90k by using the synthetic engine proposed in [Gupta et al., 2016]. The words are randomly placed on simple *pure colour backgrounds* (10 words per image on average), as shown in Figure 5.5. We lock TRN initially, and train the rest parts of our proposed model on these synthetic images in the first 30k iterations, with convolutional layers initialized from the trained VGG-16 model and other parameters randomly initialized according to Gaussian distribution. For efficiency, the first four convolutional layers are fixed during the entire training process. The learning rate is set to 10^{-5} for parameters in the rest of convolutional layers and 10^{-3} for randomly initialized parameters.

Figure 5.5: Synthetic images with words randomly placed on simple pure colour backgrounds. These images are used to pre-train our model.

2) In the next 30k iterations, TRN is added and trained with a learning rate of 10^{-3}, together with other parts in which the learning rate for randomly initialized parameters is halved to 5×10^{-4}. We still use the 48k synthetic images as they contain a comprehensive 90k word vocabulary. With this synthetic dataset, a character-level language model can be learned by TRN.

3) In the next 50k iterations, the training examples are randomly selected from the "Synth800k" [Gupta et al., 2016] dataset, which consists of 800k images with averagely 10 synthetic words placed on each *real scene background*, as presented in Figure 5.6. The learning rate for convolutional layers remains at 10^{-5}, but that for others is halved to 10^{-4}. These images have more complex background, so that the model will be further fine-tuned to handle complicated appearance patterns.

4) Totally 2044 *real-world* training images from ICDAR2015 [Karatzas et al., 2015], SVT [Wang et al., 2011] and AddF2k [Zhong et al., 2016] datasets are employed for another 20k iterations. In this stage, all the convolutional layers are fixed and the learning rate for others is further halved to 10^{-5}. These real images contain much less words than synthetic ones, but their appearance patterns are much more complex. Some example images are shown in Figure 5.7.

Figure 5.6: Synthetic images from [Gupta et al., 2016], which are created via blending rendered words into real natural scene images. The background is more complex than that of images in Figure 5.5.

5.3 Experiments

In this section, we perform experiments to verify the effectiveness of the proposed method. All experiments are implemented on a NVIDIA Tesla M40 GPU with 24GB memory. We rescale the input images into multiple sizes during test phase in order to cover the large range of bounding box scales, and sample 300 proposals with the highest textness scores produced by TPN. The detected bounding boxes are then merged via NMS according to their textness scores with the overlap threshold of 0.3 and fed into TRN for recognition.

5.3.1 Datasets

As we describe before, in order to train the deep network, three kinds of datasets are adopted step by step in a curriculum learning process, including two synthetic datasets and one real image dataset. To show the superiority of our method, we evaluate the well-trained model on four test datasets, including commonly used scene text benchmarks: "Focused Scene Text" in ICDAR2015 [Karatzas et al., 2015], ICDAR2011 [Shahab et al., 2011], Street View Text (SVT) [Wang et al., 2011] and another electronic document dataset: "Born-Digital Images (Web and Email)" in

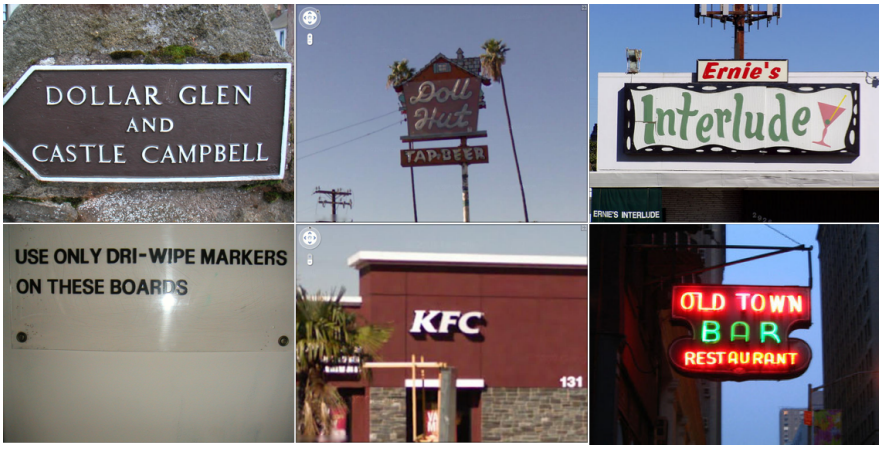


Figure 5.7: The real-world images captured from natural scene, with the 1st, 2nd and 3rd columns corresponding to images from datasets ICDAR2015, SVT and AddF2k respectively.

ICDAR2015 [Karatzas et al., 2015].

The dataset for “Focused Scene Text" in ICDAR2015 Robust Reading Competition consists of 229 images for training and 233 images for test. These images are captured in natural scene, and focused around the text content of interest. In addition, it provides 3 specific lists of words as lexicons for reference in test phase, i.e., “Strong”, “Weak” and “Generic”, as we introduced in Chapter 3.3.

ICDAR2011 dataset is quite similar to ICDAR2015, with 255 images for test. But it does not provide any lexicon. So we only use the 90k “Generic” vocabulary as context.

SVT dataset consists of 100 images for training and 249 images for test. These images are harvested from Google Street View and often have a low resolution which makes this dataset more challenging. It also provides a “Strong” lexicon with 50 words per-image. As there are unlabeled words in SVT, we only evaluate the “Word-Spotting” performance on this dataset.

“Born-Digital Images (Web and Email)” dataset in ICDAR2015 is harvest from electronic documents in web-pages and Emails. Automatically extracting text from born-digital images is useful. It enables a number of applications such as indexing

and retrieval of web content, content filtering, *etc.*. In contrast to natural scene text images which are usually high-resolution camera captured ones, born-digital images are inherently low-resolution (made to be transmitted online and displayed on a screen) and text is digitally created on the image. They may suffer from compression artefacts and severe anti-aliasing. The dataset of "Born-Digital Images (Web and Email)" provides 410 images for training and 141 images for test. It also provides 3 lexicons for reference in test.

5.3.2 Evaluation Criteria

We also follow the evaluation protocols used in ICDAR2015 Robust Reading Competition [Karatzas et al., 2015] for fair comparison: a bounding box is considered as correct if its IoU ratio with any ground-truth is greater than 0.5 and the recognized word also matches, ignoring the case. One- or two- character words as well as words deemed unreadable (annotated as "do no care" in the dataset) are ignored. Similarly, both "End-to-End" and "Word Spotting" protocols (as we introduced in Chapter 3.3) are adopted in evaluating the ICDAR2015 test data. For ICDAR2011 and SVT, we only evaluate the "Word-Spotting" performance. F-measures are reported for comprehensive comparison.

5.3.3 Evaluation under Different Model Settings

In order to show the effectiveness of our proposed varying-size RoI pooling (see Section 5.2.2) and the attention mechanism (see Section 5.2.3), we examine the performance of our model with different settings in this subsection. With the fixed RoI pooling size of 4×20, we denote the models with and without the attention mechanism as "Ours Atten+Fixed" and "Ours NoAtten+Fixed" respectively. The model with both attention and varying-size RoI pooling is denoted as "Ours Atten+Vary", in which the size of feature maps after pooling is calculated according to Equation (5.1) with $H = 4$ and $W_{max} = 35$.

Although the last hidden state of LSTMs encodes the holistic information of RoI image patch, it still lacks details. Particularly for a long word image patch, the initial information may be lost during the recurrent encoding process, even if LSTMs are used instead of vanilla RNNs. Thus, we keep the hidden states of encoder LSTMs at each time step as context. The attention model can choose the corresponding local features for each character during decoding process, as illustrated in Figure 5.8. From Table 5.2, we can see that the model with attention mechanism, namely "Ours Atten+Fixed", achieves higher F-measures on all scene text data than "Ours NoAtten+Fixed" which does not use attention.

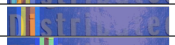
One contribution of this work is a new region feature encoder, which is composed of a varying-size RoI pooling mechanism and an LSTM sequence encoder. To validate its effectiveness, we compare the performance of models "Ours Atten+Vary" and "Ours Atten+Fixed". Experiments shows that varying-size RoI pooling performs significantly better for long words. For example, "Informatikforschung" can be recognized correctly by "Ours Atten+Vary", but not by "Ours Atten+Fixed" (as shown in Figure 5.8 (a)), because a large portion of information for long words is lost by fixed-size RoI pooling. Similarly, as illustrated in Table 5.2, adopting varying-size RoI pooling ("Ours Atten+Vary") instead of fixed-size pooling ("Ours Atten+Fixed") makes F-measures increase around 1% for ICDAR2015, 4% for ICDAR2011 and 3% for SVT with strong lexicon used.

5.3.4 Joint Training vs. Separate Training

Previous works [Jaderberg et al., 2016; Gupta et al., 2016; Liao et al., 2017] on text spotting typically perform in a two-stage manner, where detection and recognition are trained and processed separately. The text bounding boxes detected by a model need to be cropped from the image and then recognized by another model. By contrast, our proposed model is trained jointly for both text detection and recognition. By sharing convolutional features and RoI encoder, the knowledge learned from the

Image				
	"Ours Atten+Vary"		"Ours Atten+Fixed"	
Time Step	Decoder Output	Attention Weights (Length=35)	Decoder Output	Attention Weights (Length=20)
t=1	I		I	
t=2	N		N	
t=3	F		F	
t=4	O		O	
t=5	R		M	
t=6	M		A	
t=7	A		T	
t=8	T		F	
t=9	I		O	
t=10	K		R	
t=11	F		S	
t=12	O		C	
t=13	R		H	
t=14	S		U	
t=15	C		N	
t=16	H		G	
t=17	U			
t=18	N			
t=19	G			
Recognition Result		INFORMATIKFORSCHUNG		INFOMATFORSCHUNG

(a)

Image				
	"Ours Atten+Vary"		"Ours Atten+Fixed"	
Time Step	Decoder Output	Attention Weights (Length=35)	Decoder Output	Attention Weights (Length=20)
t=1	D		D	
t=2	I		I	
t=3	S		S	
t=4	T		T	
t=5	R		R	
t=6	I		I	
t=7	B		N	
t=8	U		G	
t=9	T		E	
t=10	E		D	
t=11	D			
Recognition Result		DISTRIBUTED		DISTRINGED

(b)

Figure 5.8: Attention mechanism based sequence decoding process by "Ours Atten+Vary" and "Ours Atten+Fixed" separately. The heat maps show that at each time step, the position of the character to be decoded has higher attention weights, so that the corresponding local features will be extracted and assist the text recognition. However, if we use the fixed size RoI pooling, information may be lost during pooling, especially for a long word, which leads to an incorrect recognition result. In contrast, "Ours Atten+Vary" gives the correct result, even if some parts of the word image are missed, such as "I", "n" in the first example.

correlated detection and recognition tasks can be transferred between each other and results in better performance for both tasks.

To fairly compare with the model “Ours Atten+Vary" which is a word based jointly trained model, instead of using the sliding window based character detection method as in Chapter 3.3, we build another two-stage system (denoted as “Ours New-Two-Stage” in order to distinguish with our stepwise method described in Chapter 3.3), in which detection and recognition models are trained separately. The detector in “Ours New-Two-Stage” is built by removing the recognition part from model “Ours Atten+Vary” and trained only with the detection objective (denoted as “Ours DetOnly”). As to recognition, we employ “CRNN” [Shi et al., 2015] that produces state-of-the-art performance on text recognition. Model “Ours New-Two-Stage” firstly adopts “Ours DetOnly” to detect text with the same multi-scale inputs. “CRNN” is then followed to recognize the detected bounding boxes. We can see from Table 5.2 that model “Ours New-Two-Stage” performs worse than “Ours Atten+Vary" on all the evaluated scene text datasets.

Furthermore, we also compare the detection-only performance of these two systems. Note that “Ours DetOnly” and the detection part of “Ours Atten+Vary" share the same architecture, but they are trained with different strategies: “Ours DetOnly” is optimized with only the detection loss, while “Ours Atten+Vary" is trained with a multi-task loss for both detection and recognition. In consistent with the “End-to-End” evaluation criterion, a detected bounding box is considered to be correct if its IoU ratio with any ground-truth is greater than 0.5. The detection results are presented in Table 5.1. Without any lexicon used, “Ours Atten+Vary” produces a detection performance with F-measures of 85.6% on ICDAR2015 Scene text data and 85.1% on ICDAR2011, which are averagely 2% higher than those given by “Ours DetOnly”. This result illustrates that detector performance can be improved via joint training.

Table 5.1: Text detection results on scene text datasets. Precision (P) and Recall (R) at maximum F-measure (F) are reported in percentage. The jointly trained model ("Ours Atten+Vary") gives better detection results than the one trained with detection loss only ("Ours DetOnly").

Method	ICDAR2015 "Focused Scene Text"			ICDAR2011		
	R	P	F	R	P	F
Jaderberg et al. [2016]	68.0	86.7	76.2	69.2	87.5	77.2
FCRNall+multi-filt [Gupta et al., 2016]	76.4	**93.8**	84.2	76.9	**94.3**	84.7
Ours DetOnly	78.5	88.9	83.4	80.0	87.5	83.5
Ours Atten+Vary	**80.5**	91.4	**85.6**	**81.7**	89.2	**85.1**

5.3.5 Comparison with Other Methods

In this part, we compare the text spotting results of "Ours Atten+Vary" with other state-of-the-art approaches on both scene text images and born-digital images. As shown in Table 5.2 and Table 5.3, "Ours Atten+Vary" outperforms all compared methods on both datasets. In particular, our method shows a significant superiority when using a generic lexicon. It leads to a 1.5% higher recall on average than the state-of-the-art "TextBoxes" [Liao et al., 2017] in scene text images, using only 3 input scales compared with 5 scales used by "TextBoxes".

Several text spotting examples are presented in Figure 5.9 for scene text images and in Figure 5.10 for born-digital images. The results demonstrate that model "Ours Atten+Vary" is capable of dealing with words of different fonts, aspect ratios and illuminations.

5.3.6 Running Speed

Using an M40 GPU, model "Ours Atten+Vary" takes approximately 0.9s to process an input image of 600×800 pixels. It takes nearly 0.45s to compute the convolutional features, 0.02s for text proposal calculation, 0.25s for RoI encoding, 0.01s for text detection and 0.15s for word recognition. On the other hand, model "Ours New-Two-Stage" spends around 0.45s for word recognition on the same detected bound-

Table 5.2: Text spotting results on scene text benchmarks. We present the F-measure here in percentage. "Ours New-New-Two-Stage" uses separate models for detection and recognition, while other "Ours" models are end-to-end trained. "Ours NoAtten+Fixed" and "Ours Atten+Fixed" denote models without and with attention respectively, using a fixed pooling size. "Ours Atten+Vary" uses both attention and varying-size RoI pooling. "Ours Atten+Vary" achieves the best performance on almost all datasets.

Method	ICDAR2015 "Focused Scene Text" Word-Spotting			ICDAR2015 "Focused Scene Text" End-to-End			ICDAR2011 Word-Spotting	SVT Word-Spotting	
	Strong	Weak	Generic	Strong	Weak	Generic	Generic	Strong	Generic
Deep2Text II+ [Yin et al., 2014]	84.84	83.43	78.90	81.81	79.47	76.99	–	–	–
Jaderberg et al. [2016]	90.49	–	76	86.35	–	–	76	76	53
FCRNall+multi-filt [Gupta et al., 2016]	–	–	84.7	–	–	–	84.3	67.7	55.7
TextBoxes [Liao et al., 2017]	93.90	91.95	85.92	**91.57**	89.65	83.89	87	84	64
YunosRobot1.0	86.78	–	86.78	84.20	–	84.20	–	–	–
Ours Stepwise in Chapter 3.3	91.39	90.16	82.91	87.19	86.39	80.12	–	–	
Ours New-Two-Stage	92.94	90.54	84.24	88.20	86.06	81.97	82.86	82.19	62.35
Ours NoAtten+Fixed	92.70	90.37	83.83	87.73	85.53	79.18	81.70	79.49	58.70
Ours Atten+Fixed	93.33	91.66	87.73	90.72	87.86	83.98	83.81	81.80	64.50
Ours Atten+Vary	**94.16**	**92.42**	**88.20**	91.08	**89.81**	**84.59**	**87.70**	**84.91**	**66.18**

Figure 5.9: Examples of text spotting results by "Ours Atten+Vary". The first two columns are images from ICDAR2015 "Focused Scene Text", and the rest are images from SVT. Red bounding boxes are both detected and recognized correctly. Green bounding boxes are missed words, and yellow dashed bounding boxes are false positives. The results show that our model is able to detect and recognize words of different aspect ratios. Most missed words have small bounding boxes.

Table 5.3: Text spotting results on born-digital image dataset. We present the F-measure here in percentage. "Ours Atten+Vary" achieves the best performance on this dataset under both evaluation protocols.

Method	ICDAR2015 "Born-Digital Images" Word-Spotting			ICDAR2015 "Born-Digital Images" End-to-End		
	Strong	Weak	Generic	Strong	Weak	Generic
Baseline [Gomez and Karatzas, 2014]	45.95	43.09	29.22	41.28	38.72	26.90
Stradvision-2 [Karatzas et al., 2015]	85.59	81.53	62.27	78.10	74.08	57.01
Deep2Text II+ [Yin et al., 2014]	87.64	86.81	83.68	82.08	80.75	78.77
Megvii-Image++ [Karatzas et al., 2015]	91.05	90.25	84.32	85.35	84.40	78.92
Ours Atten+Vary	**93.43**	**92.16**	**86.62**	**86.33**	**85.06**	**80.12**

Figure 5.10: Examples of text spotting results by "Ours Atten+Vary" on ICDAR2015 "born-digital image". Red bounding boxes are both detected and recognized correctly. Green bounding boxes are missed words, and blue dashed bounding boxes are false positives. Our model can also detect and recognize words in low-resolution images. Small words are difficult to cover.

ing boxes, as it needs to crop the word patches, and re-calculate the convolutional features during recognition.

5.4 Conclusion

In this chapter we presented a unified end-to-end trainable DNN for simultaneous text detection and recognition in natural scene images. With this framework, scene text can be detected and recognized in a single forward pass efficiently and accurately, avoiding intermediate processes like image cropping, feature re-calculation, word separation, or character grouping. Compared to previous integrated methods for text spotting, we eliminated a separate character detection step. In contrast to the integrated method we proposed in the previous chapter for car license plate detection and recognition, some innovations were introduced to make the framework

more suitable for general scene text. A novel RoI pooling and encoding method was proposed to tackle the large diversity of aspect ratios of word bounding boxes. An attention based sequence to sequence learning method was adopted for word recognition, which can incorporate a character level language model implicitly. Through end-to-end training, the learned features can be more informative, which improves the overall performance. The convolutional features are calculated only once and shared by both detection and recognition, which saves processing time. Our proposed method has achieved competitive performance on several benchmark datasets.

One limitation of our work is to handle images with multi-oriented text. Although our method can detect and recognize words with a certain degree of orientation, e.g., the first image on the second row of Figure 5.9, the resulted bounding boxes are horizontal, which cannot fit the words very well. It would be much better if the orientation angle can be estimated. 2-D RNN + 2-D attention model may be a solution. In addition, experimental results show that small text is still hard to be detected and recognized well with this framework. We will try to concatenate features from multiple convolutional layers for improvement.

Chapter 6

Conclusion and Future Directions

In this chapter, we summarize the work we have done in this thesis and discuss some potential directions for future research.

6.1 Conclusion

In this thesis, we concentrated on text detection and recognition in natural scene images. Contributions have been made to improve text spotting performance during the research progress, all built upon the advanced deep learning techniques. Generally speaking, we proposed a stepwise and a unified framework for car license plate detection and recognition, and general scene text detection and recognition, respectively. Car license plates can be regarded as a special case of general scene text, as they are both composed of characters and appear in natural scenes. However, they have their respective characteristics. Car license plates have distinct borders compared to general scene text, but the characters are usually randomly selected from an alphabetic set and do not have inter-dependency. Moreover, they are expected to be distinguished from other text in the background. General scene text, by contrast, does not have noticeable borderlines in the image appearance. It usually appears with the clustering of several words, but a character level language model is potentially embedded within the word. With these differences into mind, some innovations are proposed in each framework to tackle the specific task.

In particular, in Chapter 3 we present a stepwise framework for text detection and recognition, in which the detection and recognition are performed step by step by

two separated models. The framework is firstly applied to car license plates and then extended to general scene text. Basically, a well-trained character CNN classifier is applied in a sliding window manner across the input image for fast character detection. The detected characters are then grouped into car license plates or text lines. A sequence labeling based method is proposed for sequence recognition, which leverages the recurrent property of RNNs and eliminates the challenging character separation or word splitting tasks. To discriminate car license plates with other text in the background, a plate/non-plate CNN classifier is trained. This framework produces promising performance on both detection and recognition compared to conventional methods with shallow features used. Moreover, for general scene text, we incorporate the functions of word recognition, word splitting and false positive removal into a single sequence recognition network, which simplifies the whole pipeline greatly. This method achieves No. 1 on the task of "End-to-End Focused Scene Text Recognition" in ICDAR2015 at that time.

The stepwise method is flexible, where additional components can be added conveniently to improve the performance, e.g., the Hough transform employed to deal with rotated license plates. However, it is quite complex, with several CNNs to be trained and used separately. In addition, the sliding window based detection method is too slow to real-time application. On account of the sequence labeling based recognition framework and Faster R-CNN [Ren et al., 2015], in Chapter 4, we integrate CNNs and RNNs in one network for joint car license plates detection and recognition. The convolutional features are shared by both detection and recognition modules, which results in less parameters and faster computation compared to the stepwise method. The whole network can be trained end-to-end. Via a single evaluation of the input image, both plate bounding boxes and the corresponding labels are generated concurrently, without any heuristic rules used. With both detection and recognition losses jointly optimized, the well-trained model produces even better results. Compared to the previous stepwise method, the end-to-end detection

and recognition performance improves by about 2%, and the computational speed is almost 10 times faster.

Considering the particularities of general scene text compared with car license plates, in Chapter 5, we further propose a unified end-to-end trainable deep neural network for simultaneous text detection and recognition in natural scene images. Apart from sharing convolutional features, which leads to less parameters and faster processing speed, a varying-size RoI encoding method is proposed to tackle the large diversity on aspect ratio of general scene text. An attention based sequence to sequence learning method is adopted in the recognition module, instead of the sequence labeling based method for license plate recognition, which enables a character level language model to be learned, and would benefit the word recognition furthermore. Our end-to-end network achieves state-of-the-art results on several benchmark datasets, including "End-to-End Focused Scene Text Recognition" in ICDAR2011, ICDAR2015, SVT, *etc.*. It is the first successful attempt for text spotting by a single end-to-end trainable network, and attracts much attention in the area [He et al., 2018; Liu et al., 2018; Bartz et al., 2018].

6.2 Future Work

Although this work has made considerable progress to the end-to-end text detection and recognition in natural scene images, some issues are still unsolved. In this part, we discuss some potential research directions.

- **Multi-oriented text** Recently, some more datasets with inclined text or curved text appear [Veit et al., 2016; Yuliang et al., 2017; ChâĂŹng and Chan, 2017], which shows an increasing requirement of solving this problem. He et al. [2018] proposed a unified framework for end-to-end text spotting, with a novel text-alignment layer designed to precisely compute features for oriented text. Liu et al. [2018] introduced "RoIRotate" to share convolutional features between detection and recognition, which can extract the oriented text regions from

convolutional feature maps. The idea of extending our framework to handle multi-oriented text is to take advantage of 2-D RNN and spatial 2-D attention model. To be specific, the oriented angle is expected to be regressed via the detection network, apart from the central point, height and width of the detected bounding box. Then 2-D RNN will be adopted to encode image features both horizontally and vertically, while 2-D attention model will be used to acquire local character features for recognition. The whole network can still be trained end-to-end with a multi-task loss, but can localize and recognize inclined text.

- **Multi-lingual scene text detection and recognition** Our methods can process alphabet (mainly for English) and digits. However, in modern cities where multiple cultures live and communicate together, a lot of images emerge with various languages. How to extend current methods to other languages or characters is also a future work. Two new datasets appear recently [Yuan et al., 2018; Nayef, 2017] which will stimulate future development of detection and recognition algorithms of other languages.

- **Image retrieval with text information** With the improvement of text spotting on both performance and speed, it is possible to retrieve images from a huge corpus that contain the given text query. A simple method is to assign each image a score of containing the query word, and then return the top K images with higher scores. Another trend is to integrate text information and visual appearance to improve the performance. A fast processing speed is vital in this application, since millions of images are needed to be computed.

- **Very deep architectures for text spotting** Stimulated by the great improvement of very deep residual network in image recognition [He et al., 2016a], it seems to be useful to explore deeper CNN and RNN architectures for text detection and recognition. However, how to design the CNN and RNN architecture is a critical issue, which is also investigated widely in other areas. A

great deal of experiments are needed to achieve better performance.

- **Transplant to mobile systems** With the wide use of smart mobile phones, it becomes increasing desirable to incorporate text spotting as a function in mobile phones, for example, the text translation system as we described in Chapter 1. With this application, there would be a high demand to the computation efficiency. How to optimize the framework to speed up the computation is another research direction. A possible solution is to integrate MobileNets [Howard et al., 2017]. MobileNets adopt depthwise separable convolutions to build light weight deep neural networks, which demonstrate good trade off between efficiency and accuracy. Moreover, the model should also fit in the limited memory and processing budget of low-power mobile devices. A possible strategy is to split the pipeline into several parts, where some parts are computed locally on the phone and some parts are processed remotely on a server, so as to achieve the fastest overall processing speed.

Bibliography

2003. Caltech plate dataset. http://www.vision.caltech.edu/html-files/archive.html. (cited on pages 56, 87, and 95)

2013. benchmarks. https://github.com/openalpr/benchmarks. (cited on page 87)

Alsharif, O. and Pineau, J., 2014. End-to-end text recognition with hybrid hmm maxout models. In *Proc. Int. Conf. Learn. Repr.* (cited on page 7)

Anagnostopoulos, C.; Anagnostopoulos, I.; Loumos, V.; and Kayafas, E., 2006a. A license plate-recognition algorithm for intelligent transportation system applications. 7, 3 (2006), 377–392. (cited on page 32)

Anagnostopoulos, C.; Anagnostopoulos, I.; Psoroulas, I.; Loumos, V.; and Kayafas, E., 2008. License plate recognition from still images and video sequences: A survey. 9, 3 (2008), 377–391. (cited on page 56)

Anagnostopoulos, C.-N.; Giannoukos, I.; Loumos, V.; and Kayafas, E., 2006b. A license plate recognition algorithm for intelligent transportation system applications. 7, 3 (2006), 377–392. (cited on pages 33 and 35)

Antol, S.; Agrawal, A.; Lu, J.; Mitchell, M.; Batra, D.; Zitnick, C. L.; and Parikh, D., 2015. VQA: visual question answering. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2425–2433. (cited on pages 29 and 30)

Ashtari, A. H.; Nordin, M. J.; and Fathy, M., 2014. An iranian license plate recognition system based on color features. 15 (2014), 1690–1705. (cited on page 33)

Azizpour, H.; Razavian, A. S.; Sullivan, J.; Maki, A.; and Carlsson, S., 2016.

Factors of transferability for a generic convnet representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38, 9 (2016), 1790–1802. (cited on page 20)

Bahdanau, D.; Cho, K.; and Bengio, Y., 2015. Neural machine translation by jointly learning to align and translate. In *Proc. Int. Conf. Learn. Repr.* (cited on page 24)

Bai, H. and Liu, C., 2004. A hybrid license plate extraction method based on edge statistics and morphology. In *Proc. IEEE Int. Conf. Patt. Recogn.*, 831–834. (cited on page 58)

Bartz, C.; Yang, H.; and Meinel, C., 2018. See: Towards semi-supervised end-to-end scene text recognition. (cited on pages 9 and 124)

Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J., 2009. Curriculum learning. In *Proc. Int. Conf. Mach. Learn.* (cited on page 109)

Bissacco, A.; Cummins, M.; Netzer, Y.; and Neven, H., 2013. Photoocr: Reading text in uncontrolled conditions. In *Proc. IEEE Int. Conf. Comp. Vis.* (cited on pages 7 and 41)

Busta, M.; Neumann, L.; and Matas, J., 2015. Fastext: Efficient unconstrained scene text detector. In *Proc. IEEE Int. Conf. Comp. Vis.* (cited on pages 8, 37, and 38)

Busta, M.; Neumann, L.; and Matas, J., 2017. Deep textspotter: An end-to-end trainable scene text localization and recognition framework. In *Proc. IEEE Int. Conf. Comp. Vis.* (cited on pages 9 and 101)

Chen, R. and Luo, Y., 2012. An improved license plate location method based on edge detection. In *Proc. Int. Conf. Appl. Phys. Industr. Engin.*, 1350âĂŞ1356. (cited on page 32)

Chen, X. and Yuille, A., 2004. Detecting and reading text in natural scenes. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 4)

Cho, K.; van Merrienboer, B.; Bahdanau, D.; and Bengio, Y., 2014. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259 (2014). http://arxiv.org/abs/1409.1259. (cited on page 23)

Chorowski, J. K.; Bahdanau, D.; Serdyuk, D.; Cho, K.; and Bengio, Y., 2015. Attention-based models for speech recognition. In *Proc. Adv. Neural Inf. Process. Syst.*, 577–585. (cited on page 24)

Chung, J.; Gülçehre, Ç.; Cho, K.; and Bengio, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555 (2014). (cited on pages ix, 23, and 24)

ChâĂŹng, C. K. and Chan, C. S., 2017. Total-text: A comprehensive dataset for scene text detection and recognition. In *14th IAPR International Conference on Document Analysis and Recognition ICDAR*, 935–942. doi:10.1109/ICDAR.2017.157. (cited on page 124)

Cortes, C. and Vapnik, V., 1995. Support-vector networks. *Machine Learning*, 20, 3 (1995), 273–297. (cited on page 15)

Criminisi, A., 2004. Microsoft research cambridge object recognition image database. http://research.microsoft.com/en-us/downloads/b94de342-60dc-45d0-830b-9f6eff91b301/default.aspx. (cited on page 56)

Dalal, N. and Triggs, B., 2005. Histograms of oriented gradients for human detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 14)

Deb, K. and Jo, K., 2008. Hsi color based vehicle license plate detection. In *Proc. Int. Conf. Cont. Autom. Syst.*, 687–691. (cited on page 33)

Deng, L. and Yu, D., 2014. Deep learning: Methods and applications. Technical report. https://www.microsoft.com/en-us/research/publication/deep-learning-methods-and-applications/. (cited on page 15)

Dlagnekov, L., 2015. *Video-based car surveillance: License plate, make, and model recognition*. Master's thesis, Comput. Sci., Univ. California, San Diego, CA, USA. (cited on page 87)

Donahue, J.; Jia, Y.; Vinyals, O.; Hoffman, J.; Zhang, N.; Tzeng, E.; and Darrell, T., 2014. DeCAF: A deep convolutional activation feature for generic visual recognition. In *Proc. Int. Conf. Mach. Learn.*, 647–655. (cited on page 20)

Dosovitskiy, A. and Brox, T., 2016. Inverting visual representations with convolutional networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 4829–4837. (cited on page 19)

Du, S.; Ibrahim, M.; Shehata, M.; and Badawy, W., 2013. Automatic license plate recognition (alpr): A state-of-the-art review. *IEEE Trans. Circuits Syst. Video Technol.*, 23, 2 (2013), 311–325. (cited on pages 32 and 36)

Everingham, M.; Gool, L. V.; Williams, C. K. I.; Winn, J.; and Zisserman, A., 2010. The pascal visual object classes (voc) challenge. *Int. J. Comp. Vis.*, 88, 2 (2010), 303âĂŞ338. (cited on page 27)

Fei-Fei, L.; Fergus, R.; and Perona, P., 2006. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28, 4 (2006), 594 – 611. (cited on page 27)

Felzenszwalb, P. F.; Girshick, R. B.; McAllester, D. A.; and Ramanan, D., 2010. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32, 9 (2010), 1627–1645. (cited on page 27)

Giannoukos, I.; Anagnostopoulos, C.-N.; Loumos, V.; and Kayafas, E., 2010. Operator context scanning to support high segmentation rates for real time license plate recognition. *Pattern Recogn.*, 43, 11 (2010), 3866âĂŞ3878. (cited on pages 33, 35, and 36)

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J., 2014. Rich feature hierarchies

for accurate object detection and semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 580–587. (cited on pages 2 and 27)

Girshick, R. B., 2015. Fast R-CNN. In *Proc. IEEE Int. Conf. Comp. Vis.*, 1440–1448. (cited on pages 28, 81, and 101)

Gomez, L. and Karatzas, D., 2014. Scene text recognition: No country for old men? In *Proc. Asian Conf. Comp. Vis.*, 157–168. (cited on page 119)

Goodfellow, I. J.; Warde-Farley, D.; Mirza, M.; Courville, A.; and Bengio, Y., 2013. Maxout networks. *arXiv:1302.4389*, (2013). (cited on page 26)

Gou, C.; Wang, K.; Yao, Y.; and Li, Z., 2016. Vehicle license plate recognition based on extremal regions and restricted boltzmann machines. 17 (2016), 1096–1107. (cited on page 35)

Graves, A.; Fernandez, S.; Gomez, F.; and Schmidhuber, J., 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proc. Int. Conf. Mach. Learn.*, 369–376. (cited on pages 31, 41, and 55)

Graves, A.; Liwicki, M.; and Fernandez, S., 2009a. A novel connectionist system for unconstrained handwriting recognition. 31, 5 (2009), 855–868. (cited on pages 31, 45, 53, 54, 55, 83, 84, and 85)

Graves, A.; Liwicki, M.; FernÃąndez, S.; Bertolami, R.; Bunke, H.; and Schmidhuber, J., 2009b. A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31, 5 (2009), 855 – 868. (cited on page 25)

Graves, A.; Mohamed, A.; and Hinton., G., 2013. Speech recognition with deep recurrent neural networks. In *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*. (cited on page 25)

Griffin, G.; Holub, A.; and Perona, P., 2006. The caltech 256. Technical report. (cited on page 27)

Guo, J. and Liu, Y., 2008. License plate localization and character segmentation with feedback self-learning and hybrid binarization techniques. 57, 3 (2008), 1417–1424. (cited on page 35)

Gupta, A.; Vedaldi, A.; and Zisserman, A., 2016. Synthetic data for text localisation in natural images. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on pages xiii, 40, 109, 110, 111, 114, 117, and 118)

He, K.; Gkioxari, G.; DollÃąr, P.; and Girshick, R., 2017a. Mask r-cnn. In *Proc. IEEE Int. Conf. Comp. Vis.* (cited on pages 2 and 31)

He, K.; Zhang, X.; Ren, S.; and Sun, J., 2014. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Proc. Eur. Conf. Comp. Vis.*, 346–361. (cited on page 28)

He, K.; Zhang, X.; Ren, S.; and Sun, J., 2016a. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 770–778. (cited on pages 2, 27, and 125)

He, P.; Huang, W.; He, T.; Zhu, Q.; Qiao, Y.; and Li, X., 2017b. Single shot text detector with regional attention. In *Proc. IEEE Int. Conf. Comp. Vis.* (cited on page 40)

He, P.; Huang, W.; Qiao, Y.; Loy, C. C.; and Tang, X., 2016b. Reading scene text in deep convolutional sequences. 3501–3508. (cited on pages 41, 51, and 64)

He, T.; Huang, W.; Qiao, Y.; and Yao, J., 2016c. Text-attentional convolutional neural networks for scene text detection. 25 (2016), 2529–2541. (cited on page 38)

He, T.; Tian, Z.; Huang, W.; Shen, C.; Qiao, Y.; and Sun, C., 2018. An end-to-end textspotter with explicit alignment and attention. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 124)

Hermans, M. and Schrauwen, B., 2013. Training and analyzing deep recurrent neural networks. In *Proc. Adv. Neural Inf. Process. Syst.*, 190–198. (cited on page 24)

Ho, W. T.; Lim, H. W.; and Tay, Y. H., 2009. Two-stage license plate detection using gentle adaboost and SIFT-SVM. In *Proc. Asian Conf. Intel. Infor. Datab. Syst.*, 109–114. (cited on page 34)

Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural Computation*, 9, 8 (1997), 1735–1780. (cited on pages 21 and 84)

Hou, Y.; Qin, X.; Zhou, X.; Zhou, X.; and Zhang, T., 2015. License plate character segmentation based on stroke width transform. In *Proceeding of 8th International Congress on Image and Signal Processing*. (cited on page 35)

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861*, (2017). (cited on page 126)

Hsu, G.; Chen, J.; and Chung, Y., 2013. Application-oriented license plate recognition. 62, 2 (2013), 552–561. (cited on pages 32, 35, 36, 51, 56, 58, 59, 60, 61, 86, 93, and 94)

Huang, G.; Liu, Z.; van der Maaten, L.; and Weinberger, K. Q., 2016. Densely connected convolutional networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 4700–4708. (cited on page 27)

Huang, W.; Lin, Z.; Yang, J.; and Wang, J., 2013. Text localization in natural images using stroke feature transform and text covariance descriptors. In *Proc. IEEE Int. Conf. Comp. Vis.* (cited on page 37)

Huang, W.; Qiao, Y.; and Tang, X., 2014. Robust scene text detection with convolution neural network induced mser tree. In *Proc. Eur. Conf. Comp. Vis.* (cited on page 38)

Ioffe, S. and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. Int. Conf. Mach. Learn.*, 448–456. (cited on page 26)

Jaderberg, M.; Simonyan, K.; Vedaldi, A.; and Zisserman, A., 2014a. Synthetic data and artificial neural networks for natural scene text recognition. In *Proc. Adv. Neural Inf. Process. Syst.* (cited on pages 8 and 41)

Jaderberg, M.; Simonyan, K.; Vedaldi, A.; and Zisserman, A., 2016. Reading text in the wild with convolutional neural networks. *Int. J. Comp. Vis.*, 116, 1 (2016), 1–20. (cited on pages ix, 7, 8, 70, 109, 114, 117, and 118)

Jaderberg, M.; Simonyan, K.; Zisserman, A.; and Kavukcuoglu, K., 2015. Spatial transformer networks. In *Proc. Adv. Neural Inf. Process. Syst.* (cited on page 98)

Jaderberg, M.; Vedaldi, A.; and Zisserman, A., 2014b. Deep features for text spotting. In *Proc. Eur. Conf. Comp. Vis.* (cited on page 1)

Jaderberg, M.; Vedaldi, A.; and Zisserman, A., 2014c. Deep features for text spotting. In *Proc. Eur. Conf. Comp. Vis.*, 512–528. (cited on pages 37, 41, 48, 55, and 64)

Jia, W.; Zhang, H.; and He, X., 2007. Region-based license plate detection. *Jour. Netw. Comp. Appl.*, 30 (2007), 1324–1333. (cited on page 33)

Jiao, J.; Ye, Q.; and Huang, Q., 2009. A configurable method for multi-style license plate recognition. *Pattern Recogn.*, 42 (2009), 358–369. (cited on pages 35 and 36)

Johnson, J.; Karpathy, A.; and Fei-Fei, L., 2016. Densecap: Fully convolutional localization networks for dense captioning. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 30)

Józefowicz, R.; Zaremba, W.; and Sutskever, I., 2015. An empirical exploration of recurrent network architectures. In *Proc. Int. Conf. Mach. Learn.*, 2342–2350. (cited on page 24)

Karatzas, D.; Gomez-Bigorda, L.; Nicolaou, A.; Ghosh, S.; Bagdanov, A.; Iwamura, M.; Matas, J.; Neumann, L.; Chandrasekhar, V. R.; Lu, S.; Shafait, F.; Uchida, S.; and Valveny, E., 2015. ICDAR 2015 robust reading competition. In *Proc. Int. Conf. Doc. Anal. Recog.*, 1156–1160. (cited on pages 68, 69, 87, 110, 111, 112, 113, and 119)

Karpathy, A. Convolutional neural network. https://en.wikipedia.org/wiki/Convolutional_neural_network. Convolutional neural network on Wikipedia. (cited on page 17)

Karpathy, A., 2017. Cs231n convolutional neural networks for visual recognition. http://cs231n.github.io/convolutional-networks/. Stanford CS class. (cited on pages 16 and 17)

Karpathy, A.; Johnson, J.; and Fei-Fei, L., 2016. Visualizing and understanding recurrent networks. In *Proc. Int. Conf. Learn. Repr.* (cited on page 24)

Kim, K.; Jung, K.; and Kim, J., 2003. Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25, 12 (2003), 1631–1639. (cited on page 37)

Kim, S.; Jeon, H.; and Koo, H., 2017. Deep-learning-based license plate detection method using vehicle region extraction. *Electronics Letters*, 53, 15 (2017), 1034 – 1036. (cited on page 95)

Kim, Y.; Jernite, Y.; Sontag, D.; and Rush, A. M., 2016. Character-aware neural language models. In *AAAI*. (cited on page 42)

Kingma, D. and Ba, J., 2014. Adam: A method for stochastic optimization. In *Proc. Int. Conf. Learn. Repr.* (cited on pages 86 and 109)

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E., 2012. Imagenet classification

with deep convolutional neural networks. In *Proc. Adv. Neural Inf. Process. Syst.*, 1106–1114. (cited on pages 2 and 26)

Lafferty, J. D.; McCallum, A.; and Pereira, F. C. N., 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. Int. Conf. Mach. Learn.* (cited on page 15)

Le, W. and Li, S., 2006. A hybrid license plate extraction method for complex scenes. In *Proc. IEEE Int. Conf. Patt. Recogn.*, 324–327. (cited on page 58)

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P., 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86, 11 (1998), 2278–2324. (cited on page 17)

Lee, C.-Y. and Osindero, S., 2016. Recursive recurrent nets with attention modeling for ocr in the wild. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on pages 42 and 102)

Li, B.; Tian, B.; Li, Y.; and Wen, D., 2013. Component-based license plate detection using conditional random field model. 14, 4 (2013), 1690–1699. (cited on pages 34 and 96)

Liao, M.; Shi, B.; Bai, X.; Wang, X.; and Liu, W., 2017. Textboxes: A fast text detector with a single deep neural network. In *AAAI*. (cited on pages 8, 40, 114, 117, and 118)

Lim, H. W. and Tay, Y. H., 2010. Detection of license plate characters in natural scene with MSER and SIFT unigram classifier. In *Proc. IEEE Int. Conf. Sustainable Utilization and Development in Engineering and Technology*, 95–98. (cited on page 58)

Lin, K.; Tang, H.; and Huang, T., 2010. Robust license plate detection using image saliency. In *Proc. IEEE Int. Conf. Patt. Recogn.*, 3945–3948. (cited on page 34)

Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S. E.; Fu, C.; and Berg, A. C., 2016. SSD: single shot multibox detector. In *Proc. Eur. Conf. Comp. Vis.*, 21–37. (cited on pages 29 and 39)

Liu, X.; Liang, D.; Yan, S.; Chen, D.; Qiao, Y.; and Yan, J., 2018. Fots: Fast oriented text spotting with a unified network. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 124)

Llorca, D. F.; Salinas, C.; JimÃl'nez, M.; Parra, I.; Morcillo, A. G.; Izquierdo, R.; Lorenzo, J.; and Sotelo, M. A., 2016. Two-camera based accurate vehicle speed measurement using average speed at a fixed point. In *Proceeding of 19th International Conference on Intelligent Transportation Systems*. (cited on page 34)

Llorens, D.; Marzal, A.; Palazon, V.; and Vilar, J. M., 2005. Car license plates extraction and recognition based on connected components analysis and hmm decoding. *Lecture Notes in Computer Science*, 3522 (2005), 571–578. (cited on page 36)

Lowe, D. G., 2004. Distinctive image features from scale-invariant keypoints. *Int. J. Comp. Vis.*, 60, 2 (2004), 91–110. (cited on page 14)

Lu, J.; Yang, J.; Batra, D.; and Parikh, D., 2016. Hierarchical question-image co-attention for visual question answering. In *Proc. Adv. Neural Inf. Process. Syst.* (cited on pages 29 and 30)

Lucas, S., 2005. ICDAR 2005 text locating competition results. In *Proc. Int. Conf. Doc. Anal. Recog.*, 80–84. (cited on page 56)

Lucas, S. M.; Panaretos, A.; Sosa, L.; Tang, A.; Wong, S.; and Young, R., 2003. ICDAR 2003 robust reading competitions. In *Proc. Int. Conf. Doc. Anal. Recog.*, 682–687. (cited on pages 56 and 68)

Luong, M.-T.; Pham, H.; and Manning, C. D., 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference*

on Empirical Methods in Natural Language Processing. http://aclweb.org/anthology/D15-1166. (cited on page 106)

Lyu, P.; Yao, C.; Wu, W.; Yan, S.; and Bai, X., 2018. Multi-oriented scene text detection via corner localization and region segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 39)

Ma, J.; Shao, W.; Ye, H.; Wang, L.; Wang, H.; Zheng, Y.; and Xue, X., 2017. Arbitrary-oriented scene text detection via rotation proposals. *arXiv:1703.01086*, (2017). (cited on page 39)

Mahendran, A. and Vedaldi, A., 2015. Understanding deep image representations by inverting them. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 5188–5196. (cited on page 19)

Mishra, A.; Alahari, K.; and Jawahar, C. V., 2012. Top-down and bottom-up cues for scene text recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 41)

Nayef, N., 2017. Icdar2017 competition on multi-lingual scene text detection and script identification. http://rrc.cvc.uab.es/?ch=8&com=introduction. (cited on page 125)

Neumann, L. and Matas, J., 2013a. On combining multiple segmentations in scene text recognition. In *Int. Conf. Doc. Anal. Recogn.* (cited on pages 9, 75, and 99)

Neumann, L. and Matas, J., 2013b. Scene text localization and recognition with oriented stroke detection. In *Proc. IEEE Int. Conf. Comp. Vis.* (cited on page 37)

Neumann, L. and Matas, J., 2016. Real-time lexicon-free scene text localization and recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38, 9 (2016), 1872–1885. (cited on page 70)

Nielsen, M. A., 2015. *Neural Networks and Deep Learning*. Determination Press. (cited on page 25)

Novikova, T.; Barinova, O.; Kohli, P.; and Lempitsky, V., 2012. Large-lexicon attribute-consistent text recognition in natural images. In *Proc. Eur. Conf. Comp. Vis.*, 752–765. (cited on page 41)

Ojala, T.; PietikÃĭinen, M.; and Harwood, D., 1996. A comparative study of texture measures with classification based on feature distributions. *Pattern Recogn.*, 29, 1 (1996), 51–59. (cited on page 14)

Pascanu, R.; Mikolov, T.; and Bengio, Y., 2013. On the difficulty of training recurrent neural networks. In *Proc. Int. Conf. Mach. Learn.*, 1310–1318. (cited on page 21)

Permuter, H.; Francos, J.; and Jermyn, I., 2003. Gaussian mixture models of texture and colour for image database retrieval. In *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*. (cited on page 15)

Poirson, P.; Ammirato, P.; Fu, C.-Y.; Liu, W.; Kosecka, J.; and Berg, A. C. Fast single shot detection and pose estimation. In *3*. (cited on page 31)

Qiao, S.; Zhu, Y.; Li, X.; Liu, T.; and Zhang, B., 2010. Research on improving the accuracy of license plate character segmentation. In *Proc. Int. Conf. Front. Comp. Sci. Tech.*, 489–493. (cited on page 35)

Rasheed, S.; Naeem, A.; and Ishaq, O., 2012. Automated number plate recognition using hough lines and template matching. In *Proc. World Cong. Engin. Comp. Sci.*, 199–203. (cited on pages 36 and 60)

Redmon, J.; Divvala, S. K.; Girshick, R. B.; and Farhadi, A., 2016. You only look once: Unified, real-time object detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 779–788. (cited on pages 29 and 39)

Redmon, J. and Farhadi, A., 2017. YOLO9000: Better, faster, stronger. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* https://arxiv.org/abs/1612.08242. (cited on page 77)

Ren, S.; He, K.; Girshick, R. B.; and Sun, J., 2015. Faster R-CNN: towards real-time object detection with region proposal networks. In *Proc. Adv. Neural Inf. Process. Syst.*, 91–99. (cited on pages 2, 28, 39, 78, 80, 85, 90, 101, 102, 103, 104, 108, 109, and 123)

Rosenblatt, F., 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, (1958), 65–386. (cited on page 15)

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M. S.; Berg, A. C.; and Fei-Fei, L., 2015. Imagenet large scale visual recognition challenge. *Int. J. Comp. Vis.*, 115, 3 (2015), 211–252. (cited on page 77)

Schuster, M. and Paliwal, K., 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Proc.*, 45, 11 (1997), 2673–2681. (cited on page 24)

Shahab, A.; Shafait, F.; and Dengel, A., 2011. ICDAR 2011 robust reading competition challenge 2: Reading text in scene images. In *Proc. Int. Conf. Doc. Anal. Recog.*, 1491–1496. (cited on pages 56, 68, and 111)

Sharma, J.; Mishra, A.; Saxena, K.; and Kumar, S., 2014. A hybrid technique for license plate recognition based on feature selection of wavelet transform and artificial neural network. In *Proc. Int. Conf. Optim. Reliab. Infor. Techn.*, 347–352. (cited on page 36)

Shi, B.; Bai, X.; and Yao, C., 2015. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *CoRR*, abs/1507.05717 (2015). (cited on pages xii, 8, 41, 65, 66, 68, 84, 90, 104, and 116)

Shi, B.; Wang, X.; Lv, P.; Yao, C.; and Bai, X., 2016. Robust scene text recognition

with automatic rectification. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on pages 24, 42, 94, 102, and 106)

Shih, K. J.; Singh, S.; and Hoiem, D., 2016. Where to look: Focus regions for visual question answering. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 30)

Simonyan, K.; Vedaldi, A.; and Zisserman, A., 2013. Deep fisher networks for large-scale image classification. In *Proc. Adv. Neural Inf. Process. Syst.*, 163–171. (cited on page 2)

Simonyan, K. and Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. In *Proc. Int. Conf. Learn. Repr.* (cited on pages 27, 77, and 102)

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15, 1 (2014), 1929–1958. (cited on page 25)

Su, B. and Lu, S., 2014. Accurate scene text recognition based on recurrent neural network. In *Proc. Asian Conf. Comp. Vis.*, 35–48. (cited on pages 8 and 51)

Sutskever, I.; Vinyals, O.; and Le, Q. V., 2014. Sequence to sequence learning with neural networks. In *Proc. Adv. Neural Inf. Process. Syst.* (cited on page 42)

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S. E.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A., 2015. Going deeper with convolutions. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 1–9. (cited on page 27)

Tan, J.; Abu-Bakar, S.; and Mokji, M., 2013. License plate localization based on edge-geometrical features using morphological approach. In *Proc. IEEE Int. Conf. Image Process.*, 4549–4553. (cited on page 32)

Tian, J.; Wang, G.; Liu, J.; and Xia, Y., 2017. License plate detection in an open environment by density-based boundary clustering. *Journal of Electronic Imaging*, 26, 3 (2017), 033017(1–11). (cited on page 95)

Tian, S.; Pan, Y.; Huang, C.; Lu, S.; Yu, K.; and Tan, C. L., 2015. Text flow: A unified text detection system in natural scene images. In *Proc. IEEE Int. Conf. Comp. Vis.* (cited on pages 8, 37, and 100)

Tian, Z.; Huang, W.; He, T.; He, P.; and Qiao, Y., 2016. Detecting text in natural image with connectionist text proposal network. In *Proc. Eur. Conf. Comp. Vis.* (cited on page 39)

Vedaldi, A. and Lenc, K., 2015. MatConvNet—convolutional neural networks for MATLAB. In *Proc. ACM Int. Conf. Multimedia*. (cited on pages 55 and 68)

Veit, A.; Matera, T.; Neumann, L.; Matas, J.; and Belongie, S., 2016. Coco-text: Dataset and benchmark for text detection and recognition in natural images. *arXiv:1601.07140*, (2016). (cited on pages 4 and 124)

Vinyals, O.; Toshev, A.; Bengio, S.; and Erhan, D., 2015. Show and tell: A neural image caption generator. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 3156–3164. (cited on page 29)

Wang, J.; Yang, Y.; Mao, J.; Huang, Z.; and Xu, C. H. W., 2016. Cnn–rnn: A unified framework for multi-label image classification. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 31)

Wang, K.; Babenko, B.; and Belongie, S., 2011. End-to-end scene text recognition. In *Proc. IEEE Int. Conf. Comp. Vis.* (cited on pages ix, 8, 9, 37, 55, 68, 75, 99, 110, and 111)

Wang, T.; Wu, D. J.; Coates, A.; and Ng, A. Y., 2012. End-to-end text recognition with convolutional neural networks. In *Proc. IEEE Int. Conf. Image Process.*, 3304–3308. (cited on page 9)

Weinman, J. J.; Learned-Miller, E.; and Hanson, A. R., 2009. Scene text recognition using similarity and a lexicon with sparse belief propagation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31, 10 (2009), 1733–1746. (cited on page 4)

Wen, Y.; Lu, Y.; Yan, J.; Zhou, Z.; von Deneen, K.; and Shi, P., 2011. An algorithm for license plate recognition applied to intelligent transportation system. 12 (2011), 830–845. (cited on page 36)

Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A. C.; Salakhutdinov, R.; Zemel, R. S.; and Bengio, Y., 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proc. Int. Conf. Mach. Learn.*, 2048–2057. (cited on pages 24, 29, and 30)

Ye, Q. and Doermann, D., 2015. Text detection and recognition in imagery: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37, 7 (2015), 1480–1500. (cited on pages 6 and 36)

Yin, X.-C.; Yin, X.; Huang, K.; and Hao, H.-W., 2014. Robust text detection in natural scene images. 36, 5 (2014), 970–983. (cited on pages 70, 118, and 119)

Yoon, Y.; Ban, K.; Yoon, H.; and Kim, J., 2011. Blob extraction based character segmentation method for automatic license plate recognition system. 2192–2196. (cited on page 51)

Yu, S.; Li, B.; Zhang, Q.; Liu, C.; and Meng, M., 2015. A novel license plate location method based on wavelet transform and emd analysis. *Pattern Recogn.*, 48, 1 (2015), 114âĂŞ125. (cited on page 33)

Yuan, T.-L.; Zhu, Z.; Xu, K.; Li, C.-J.; and Hu, S.-M., 2018. Chinese text in the wild. *arXiv:1803.00085*, (2018). (cited on page 125)

Yuan, Y.; Zou, W.; Zhao, Y.; Wang, X.; Hu, X.; and Komodakis, N., 2017. A robust and efficient approach to license plate detection. 26, 3 (2017), 1102–1114. (cited on pages 87, 88, 96, 97, and 98)

Yuliang, L.; Lianwen, J.; Shuaitao, Z.; and Sheng, Z., 2017. Detecting curve text in the wild: New dataset and new solution. *arXiv:1712.02170*, (2017). (cited on pages 39 and 124)

Zeiler, M. D., 2012. Adadelta: An adaptive learning rate method. *arXiv:1212.5701*, (2012). (cited on page 69)

Zeiler, M. D. and Fergus, R., 2014. Visualizing and understanding convolutional networks. In *Proc. Eur. Conf. Comp. Vis.*, 818–833. (cited on pages 18, 19, and 27)

Zhang, H.; Jia, W.; He, X.; and Wu, Q., 2006. Learning-based license plate detection using global and local features. In *Proc. IEEE Int. Conf. Patt. Recogn.*, 1102–1105. (cited on page 33)

Zhang, Z.; Shen, W.; Yao, C.; and Bai, X., 2015. Symmetry-based text line detection in natural scenes. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on pages 38 and 100)

Zhang, Z.; Zhang, C.; Shen, W.; Yao, C.; Liu, W.; and Bai, X., 2016. Multi-oriented text detection with fully convolutional networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on page 39)

Zheng, L.; He, X.; Samali, B.; and Yang, L., 2013. An algorithm for accuracy enhancement of license plate recognition. *J. Comp. & Syst. Sci.*, 79, 2 (2013), 245–255. (cited on pages 35, 49, and 51)

Zhong, Z.; Jin, L.; Zhang, S.; and Feng, Z., 2016. Deeptext: A unified framework for text proposal generation and text detection in natural images. *CoRR*, abs/1605.07314 (2016). (cited on pages 39, 78, 103, 104, and 110)

Zhou, W.; Li, H.; Lu, Y.; and Tian, Q., 2012. Principal visual word discovery for automatic license plate detection. 21, 9 (2012), 4269–4279. (cited on pages 32, 34, 56, 57, 58, 88, 95, 96, and 97)

Zhu, S. and Zanibbi, R., 2016. A text detection system for natural scenes with convolutional feature learning and cascaded classification. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* (cited on pages 8, 37, 38, and 100)

Zhu, Y.; Yao, C.; and Bai, X., 2016. Scene text detection and recognition: recent advances and future trends. *Frontiers of Computer Science*, 10, 1 (2016), 19–36. (cited on page 37)