Detection and Recognition of Traffic Scene Objects with Deep Learning

Rongqiang Qian

School of Electrical Engineering, Electronics & Computer Science
University of Liverpool

This dissertation is submitted for the degree of
Doctor of Philosophy

February 2018

I would like to dedicate this thesis to my loving parents and wife ...

Declaration

I hereby confirm that except where specific reference is made to the work of others, the contents presented in this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Rongqiang Qian
February 2018

Acknowledgements

I would like to express my deepest appreciation and gratitude for the help and support from the following persons.

Prof. Yong Yue, Dr. Bailing Zhang and Prof. Frans Coenen, my supervisors, for their industrious guidance throughout my PhD study. Their patience and encouragement are invaluable to both my research and life, for which I consider myself very fortunate.

Dr. Wenjin Lv and Dr. Keith Dures, my advisors, for their valuable critique and advice.

Dr. Chao Yan, Yizhang Xia, Shiyang Yan and Zhao Wang, my colleagues, for their useful discussion, suggestion and help to my research.

I am particularly grateful for the PhD scholarship funding from Xi'an Jiaotong Liverpool University.

Finally, I would like to offer most grateful thanks to my family for all the support, love and belief over all these years.

Abstract

Mobility is an element that is highly related to the development of society and the quality of individual life. Through mass of automobile production and traffic infrastructure construction, advanced countries have reached a high degree of individual mobility. In order to increase the efficiency, convenience and safety of mobility, advanced traffic infrastructure construction, transportation systems and automobiles should be developed.

Among all the systems for modern automobiles, cameras based assistance systems are one of the most important components. Recently, with the development of driver assistance systems and autonomous cars, detection and recognition of traffic scene objects based on computer vision become more and more indispensable. On the other hand, the deep learning methods, in particular convolutional neural networks have achieved excellent performance in a variety of computer vision tasks. This thesis mainly presents the contributions to the computer vision and deep learning methods for traffic scene objects detection and recognition.

The first approach develops numbers of methods for traffic sign detection and recognition. For traffic sign detection, template matching is applied with new features extended from chain code. Moreover, the region based convolutional neural networks are applied for detecting traffic signs painted on road surface. For traffic sign recognition, convolutional neural networks with a variety of architectures are trained with different training algorithms.

The second approach focuses on the detection related to traffic text. A novel license plate detection framework is developed that is able to improve detection performance by simultaneously completing detection and segmentation. Due to the larger number and complex layout of Chinese characters, Chinese traffic text detection faces more challenges than English text detection. Therefore, Chinese traffic texts are detected by applying convolutional neural networks and directed acyclic graph.

The final approach develops a method for pedestrian attribute classification. Generally, there are irrelevant elements included in features of convolutional neural networks. In order to improve classification performance, a novel feature selection algorithm is developed to refine features of convolutional neural networks.

Table of contents

List of figures

List of tables

List of abbreviations

ADAS	Advanced Driver Assistance Systems
AP	Average Precision
BN	Batch Normalization
CNN	Convolutional Neural Network
DAG	Directed Acyclic Graph
GANs	Generative Adversarial Networks
GTSRB	German Traffic Sign Recognition Benchmark
HOG	Histogram of Oriented Gradient
IoU	Intersection over Union
ITS	Intelligent Transportation System
LPD	License Plate Detection
LPR	License Plate Recognition
MAP	Mean Average Precision
MCCH	Multi-level Chain Code Histogram
MLP	Multi-Layer Perception
MPPs	Max Pooling Positions
MSERs	Maximally Stable Extremal Regions
MTL	Multi-Task Learning
ReLU	Rectified Linear Unit

RF	Random Forest
SGD	Stochastic Gradient Descent
STD	Scene Text Detection
SVM	Support Vector Machine
TPD	Traffic Panel Detection
TSD	Traffic Sign Detection
TSR	Traffic Sign Recognition
TTD	Traffic Text Detection

List of symbols

Δ	Difference operator
$*$	Convolution operator
ε	Constant for ensuring numerical stability
η	Learning rate
log	Logarithms
B	Mini-batch
∇	Gradient
$\odot$	Element-wise multiplication
$\sum$	Summation operator
e	Exponential function

Chapter 1

Introduction

1.1 Motivation

Mobility is an element that is highly related to the development of society and the quality of individual life. It is the backbone of commercial trading and services, and therefore, the basis for developing economy. Through mass of automobile production and traffic infrastructure construction, advanced countries have reached a high degree of individual mobility. [2] However, the development of mobility also faces adverse effects. The mass mobilisation costs a great quantity of resource on the production, provision and maintenance of automobiles. The noise and exhaust pollution caused by automobiles also become more and more serious. Moreover, frequent traffic congestion reduces traffic efficiency, and traffic incidents cause personal safety risk.

In order to increase the efficiency, convenience and safety of mobility: for environment, better traffic infrastructure construction and Intelligent Transportation System (ITS) should be provided [3]; for automobile, more reliable automobile design and more powerful Advanced Driver Assistance Systems (ADAS) should be developed [2–6]; for driver, better human-automobile systems should be focused, and autonomous cars in the future [7].

Among all the systems for modern automobiles, systems based on cameras and computer vision are one of the most important components. The applications of computer vision can be traced to very early in 1990s, the researchers tried to develop camera-based systems for autonomous mobile robots and autonomous driving. [8–10] Several comprehensive surveys of the applications of computer vision in intelligent vehicles are presented in [11] and [12], the applications such as lane, pedestrian, and obstacle detection are described and analysed. At the same time, computer vision for enhancing the safety of automobiles are also discussed in [13, 14].

Recently, with the development of ADAS and autonomous cars, detection and recognition of traffic scene objects from images captured by vehicle mounted cameras based on computer vision become more and more indispensable. [13, 4, 3, 5, 15, 7, 6, 16, 17] The existing works have focused on traffic sign recognition [18–23], vehicle detection [6], pedestrian detection [5, 15], road layout detection [16] and road marking detection [4]. On the other hand, the deep learning [24–28] has attracted much attention in computer vision and machine learning area. In particular, Convolutional Neural Network (CNN)s have achieved outstanding performance due to the generalisation ability and learning capacity, therefore, CNNs have been widely used in various computer vision tasks, such as image recognition [29–32] and object detection [33–36].

This research has been conducted with the following objectives.

(i) To evaluate and apply the existing computer vision methods including image processing, feature extraction and machine learning for traffic scene objects detection and recognition.

(ii) To evaluate and apply the existing deep learning methods, in particular CNNs for traffic scene objects detection and recognition.

(iii) To develop new algorithms for training of CNNs. The research should be focused on new network layers and architectures for improving performance such as training speed, generalisation ability and computational cost.

(iv) To apply developed algorithms for traffic scene objects detection and recognition. Traffic scene objects like text and pedestrian have complex appearance, in order to detect and recognise them with high accuracy, CNNs should be applied with different training strategies.

1.2 Thesis Organisation

In order to present a legible introduction of this research, the relationships of the chapters are summarised in Fig. 1.1. The thesis has been divided into ten chapters, which are listed as follows:

Chapter 1 describes the motivation, organisation, contributions and publications of the thesis.

Chapter 2 presents an overview of the research domain. The basic computer vision methods are firstly presented. Then, the particular topics that have been focused in this thesis

are reviewed, including Traffic Sign Recognition (TSR), License Plate Recognition (LPR), Traffic Text Detection (TTD) and CNNs.

Chapter 3 proposes a real-time system for Traffic Sign Detection (TSD), which performs detection by template matching with features based on shapes, namely Multi-level Chain Code Histogram (MCCH)). For the three shapes, circle, triangle and inverted triangle that are associated with Chinese traffic signs, MCCH is an excellent feature with good representation ability and low computational cost.

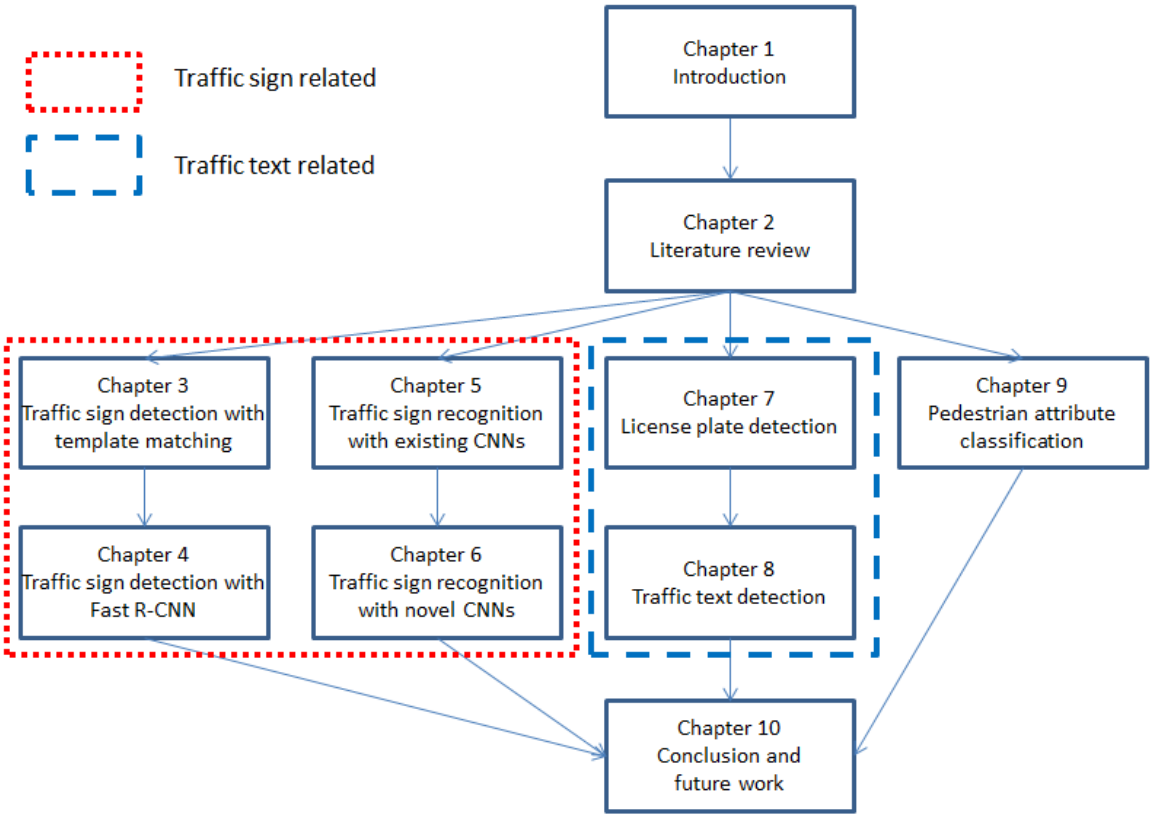


Fig. 1.1 Thesis Organisation

Chapter 4 proposes a road surface TSD system by applying a hybrid region proposal method and a CNN. The proposed system consists of two main stages: (i) a hybrid region proposal method to hypothesise the traffic sign locations by taking into account complementary information of colour and edge; (ii) a multi-task network to simultaneously implement category prediction and bounding box regression based on the Fast R-CNN framework.

Chapter 5 presents a set of detailed experiments about TSR with CNNs. Traffic signs are recognised with the most influencing CNN architectures, including the very deep networks, residual networks and dense networks. Extensive experiments are presented and discussed

based on different training algorithms, such as weight initialisation, regularisation and optimisation.

Chapter 6 proposes four TSR systems which are trained by new strategies or constructed by new network architectures. First, different from the most existing approaches that apply single level classifiers for TSR, a hierarchical classification system is applied for improving the performance of TSR. Second, unsupervised feature learning is a challenge in computer vision. By introducing Generative Adversarial Networks (GANs), a TSR system is proposed based on features that are learnt without labels. Third, in comparison with previous CNNs that only apply max pooling for reducing feature dimension, a novel layer is proposed, which applies Max Pooling Positions (MPPs) of max pooling as information for sparse feature learning. Finally, in order to overcome gradient vanishing and degradation problems, a novel CNN architecture design that learns with multiple loss functions is developed. The performance of the proposed methods are verified on the German Traffic Sign Recognition Benchmark (GTSRB), and excellent results have been achieved.

Chapter 7 proposes a License Plate Detection (LPD) system for detecting Chinese and English license plates with large skew angles. The proposed system consists of three main stages: (i) a character proposal stage to find candidate characters based on Extremal Regions (ERs); (ii) feature extraction and classification relies on CNN; (iii) LPD by linking individual characters based on a new region linking method. By leaving out character segmentation, the proposed method has better robustness compared with existing methods.

Chapter 8 proposes a context-aware Chinese TTD system by applying CNNs and Directed Acyclic Graph (DAG). First, the proposed system jointly applies Maximally Stable Extremal Regions (MSERs) and a regressor as region generator to ensure high recall. Second, all of the generated regions are not only assigned with confidence scores for character/non-character classification, but also inter-similarities for context-aware detection. Finally, characters are refined and linked into texts by applying an unified character linking method, which is based on DAG. With the help of these improvements, the proposed system can handle non-horizontal text lines and crossed text lines, while some existing text detection systems only are able to detect horizontal or near horizontal text lines. The error accumulation problem that usually occurs in the systems that use individual and sequential stages is also avoided.

Chapter 9 proposes a pedestrian attribute classification system based on CNNs and a novel feature selection algorithm. CNNs have demonstrated promising performance in image recognition and object detection. Such networks are able to automatically learn a hierarchy of discriminate features that richly describe image content. However, dimensions of features of CNNs are usually very large, and there are irrelevant elements included in features of

CNNs. Therefore, feature selection could be exploited to remove the irrelevant or redundant features of CNNs.

Chapter 10 states the conclusions of this thesis, followed by possible future research directions.

1.3 Contributions

The major contributions have been made in this research are listed as follows.

(i) Introduction of a new feature MCCH that has good representation ability and low computational cost. (**Chapter 3**)

(ii) Creation of a new road surface traffic sign dataset, which is able to facilitate the research and development of road surface TSD. (**Chapter 4**)

(iii) Introduction of a road surface TSD system based on a hybrid region proposal method and Fast R-CNN. (**Chapter 4**)

(iv) Introduction of a TSR system based on hierarchical classification, which improves performance by recognising different subsets of traffic signs at different levels. (**Chapter 6**)

(v) Introduction of a TSR system base on classifiers trained with unsupervised features that are learnt by GANs. (**Chapter 6**)

(vi) Introduction of a novel sparse layer based on MPPs, which is able to learn sparse feature and improve classification performance effectively. (**Chapter 6**)

(vii) Introduction of a novel CNN architecture design that is able to ease the training problems of deep CNNs, including gradient vanishing and degradation. (**Chapter 6**)

(viii) Introduction of a framework for LPD which simultaneously completes detection and segmentation. The method also able to detect both Chinese and English license plate with extremely skewed angles. (**Chapter 7**)

(ix) Creation of a new Chinese traffic text dataset, which is able to fill the gap of Chinese traffic text detection and recognition. (**Chapter 8**)

(x) Introduction of a Chinese TTD system that has the following advantages: (a) a region proposal method that jointly uses of MSERs and a CNN based regressor to hypothesise

the character locations; (b) a classification method that uses both information from a local model and a contextual model; (c) an unified character linking method based on DAG, which can handle non-horizontal and crossed text lines. (**Chapter 8**)

(xi) Introduction of a novel feature selection algorithm to remove irrelevant or redundant features of CNNs. (**Chapter 9**)

1.4 Publications

1.4.1 Journal Papers

(i) B. Zhang, **R. Qian** and X. Yang, "Reliable License Plate Recognition by Hierarchical Support Vector Machines," IET Intelligent Transport Systems (Accepted)

(ii) **R. Qian**, Y. Yue, F. Coenen and B. Zhang, "Road Surface Traffic Sign Detection with Hybrid Region Proposal and Fast R-CNN," Journal of Computer and System Sciences (Under review)

(iii) **R. Qian**, Y. Xia, S. Yan, B. Zhang, F. Coenen and Y. Yue, "Context-Aware Chinese Traffic Text Detection with Convolutional Neural Network and Directed Acyclic Graph" IEEE Transactions on Intelligent Transportation Systems (Under review)

In the first paper, a License Plate Recognition (LPR) system is presented by applying MCCH for plate detection and Hierarchical SVMs for plate recognition. In the second paper, a road surface TSD system is presented, the details will be presented in **Chapter 4**. In the third paper, a context-aware Chinese TTD system is presented by applying CNNs and DAG, the details will be presented in **Chapter 8**.

1.4.2 Conference Papers

(i) **R. Qian**, B. Zhang, Y. Yue, Z. Wang and F. Coenen, "Robust Chinese Traffic Sign Detection and Recognition with Deep Convolutional Neural Network," 2015 11th International Conference on Natural Computation (ICNC), Zhangjiajie, 2015, pp. 791-796.

(ii) **R. Qian**, B. Zhang, Y. Yue and F. Coenen, "Traffic Sign Detection by Template Matching Based on Multi-level Chain Code Histogram," 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), Zhangjiajie, 2015, pp. 2400-2404.

(iii) **R. Qian**, Y. Yue, F. Coenen and B. Zhang, "Traffic Sign Recognition with Convolutional Neural Network Based on Max Pooling Positions," 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Changsha, 2016, pp. 578-582.

(iv) **R. Qian**, Q. Liu, Y. Yue, F. Coenen and B. Zhang, "Road Surface Traffic Sign Detection with Hybrid Region Proposal and Fast R-CNN," 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Changsha, 2016, pp. 555-559.

(v) **R. Qian**, Y. Yue, F. Coenen and B. Zhang, "Visual Attribute Classification Using Feature Selection and Convolutional Neural Network," 2016 IEEE 13th International Conference on Signal Processing (ICSP), Chengdu, 2016, pp. 649-653.

(vi) Z. Wang, S. Liu, **R. Qian**, T. Jiang, X. Yang and J. J. Zhang, "Human Motion Data Refinement Unitizing Structural Sparsity and Spatial-temporal Information," 2016 IEEE 13th International Conference on Signal Processing (ICSP), Chengdu, 2016, pp. 975-982.

(vii) **R. Qian**, B. Zhang, F. Coenen and Y. Yue, "Multilingual and Skew License Plate Detection Based on Extremal Regions," 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Guilin, 2017.

In the first paper, Chinese traffic sign are detected and recognised with multi-task CNNs. In the second paper, a real-time system for Traffic Sign Detection (TSD) is proposed by template matching based on shapes Multi-level Chain Code Histogram (MCCH)), the details will be presented in **Chapter 3**. In the third paper, traffic signs are recognised based on Max Pooling Positions (MPPs), the details will be presented in **Chapter 6**. In the fourth paper, a road surface TSD system is presented, the details will be presented in **Chapter 4**. In the fifth paper, a pedestrian attribute classification system is proposed based on CNNs and a novel feature selection algorithm, the details will be presented in **Chapter 9**. In the final paper, a License Plate Detection (LPD) system for detecting Chinese and English license plates with large skew angles is proposed, the details will be presented in **Chapter 7**.

Chapter 2

Literature Review

2.1 Computer Vision

2.1.1 Background

Computer vision is an interdisciplinary field that involves how computers get high-level understanding of digital images or videos. From the perspective of scientific discipline, it is concerned with the theory related to artificial intelligence. From the perspective of technological discipline, it seeks to apply its theories and models for the construction of computer vision systems. From the perspective of engineering discipline, it aims to automate tasks that the human visual system can do. [1, 37]

Computer vision tasks involves a series of methods, such as acquiring, processing, analyzing and understanding of digital images or videos. [38] With the help of computer vision methods, high-dimensional real world data can be transferred into numerical or symbolic information, which can be further interfaced with other processes and elicited appropriate actions. This image understanding can be seen as the disentangling of symbolic information from image data using models constructed with the aid of geometry, physics, statistics, and learning theory. [39]

2.1.2 Related Fields

There are a wide range of fields closely related to computer vision, such as machine vision, image processing, machine learning and deep learning.

Machine vision is an engineering discipline, which can be considered distinct from computer vision. A large number of methods are applied in machine vision, including software and hardware products, integrated systems, actions, methods and expertise. It aims

to combine existing methods in new ways for solving real world problems, in particular industry. The detailed applications include automatic inspection, process control, and robot guidance. [40]

Image processing aims to process images with mathematical operations by applying any form of signal processing methods. Its input can be an image, a series of images or a video, and its output may be image or a set of parameters related to the raw image. Generally, image processing methods involve isolating the individual colour planes of an image and treating them as two-dimensional signal and applying standard signal-processing techniques to them. [41]

Machine learning is a field of computer science, which provides computers the ability to learn and act without being explicitly programmed. Based on the study of pattern recognition and computational learning theory in artificial intelligence, machine learning focuses on the methods that can learn from and predict based on data. Therefore, the methods can avoid strictly static program instructions by data driven prediction or decision making. Machine learning is applied in a wide range of computing tasks where designing and programming explicit methods with good performance is difficult or infeasible. [42]

Deep learning is a part of a broader family of machine learning methods. In contrary to task specific methods, deep learning focuses on data representations with supervised, weakly supervised or unsupervised learning. Deep learning methods use a cascade of many layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input. Higher level features are derived from lower level features to form a hierarchical representation. Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks have been widely applied in the fields such as computer vision, speech recognition and natural language processing. [43, 44]

2.1.3 Applications

Computer vision has been applied in a variety of real world applications [1] that are listed as follows.

(i) Optical character recognition (OCR): reading handwritten postal codes on letters (Fig. 2.1a) and automatic License Plate Recognition (LPR).

(ii) Machine inspection: rapid parts inspection for quality assurance using stereo vision with specialised illumination (Fig. 2.1b).

(iii) Retail: object recognition for automated checkout lanes (Fig. 2.1c).

(iv) Medical imaging: registering pre-operative and intra-operative imagery (Fig. 2.1d).

(v) Motion capture: using retro-reflective markers viewed from multiple cameras or other vision-based techniques to capture actors for computer animation.

(vi) Automotive safety: detecting unexpected obstacles such as pedestrians on the street, under conditions where active vision techniques such as radar or lidar do not work well (Fig. 2.1e).

(vii) Surveillance: monitoring for intruders, analyzing highway traffic 2.1f), and monitoring pools for drowning victims.

(viii) Fingerprint recognition and biometrics: for automatic access authentication and forensic applications.

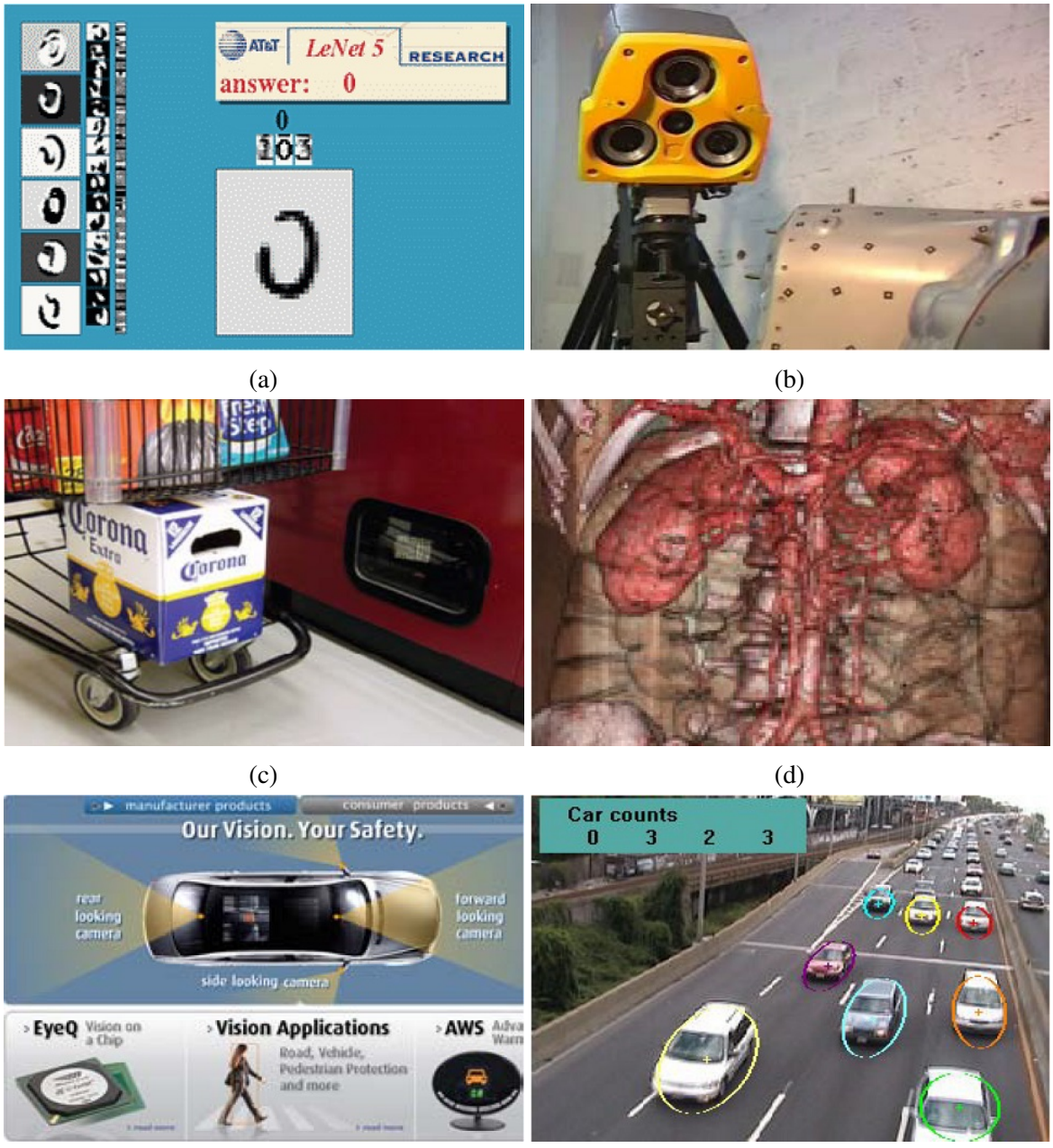
2.1.4 Methods

The organisation of a computer vision system is highly dependent on specific applications. Some simple and stand-alone applications may only solve a specific detection or recognition problem, while some applications are very complex and consist of a series of subsystems, such as data acquisition, control of mechanical components, planning and decision. The specific organisation of a computer vision system also depends on whether its functionality is fixed or flexible during operation. There are a lot of methods have been applied in computer vision. Generally, most methods are typical and can be found in many computer vision systems, while some methods are unique to specific application. [1]

(i) Image acquisition: a digital image is captured by image sensors, such as a variety of light-sensitive cameras, range sensors, tomography devices, radar and ultra-sonic cameras. Depending on the type of sensor, the captured image data is an 2D image, a 3D volume or an image sequence. Typically, the pixel values correspond to light intensity in one or several spectral bands (gray images or RGB images).

(ii) Data preprocessing: raw image data is usually preprocessed before computer vision methods can be applied for extracting specific piece of information. Generally, data preprocessing includes techniques such as normalisation, re-sampling, contrast enhancement and scale space.

(iii) Feature extraction: features at various levels of complexity are extracted from image data. Typically, low level features are colours, lines, edges, corners and ridges; middle



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

Fig. 2.1 Illustration of some industrial applications of computer vision: (a) optical character recognition; (b) mechanical inspection; (c) retail; (d) medical imaging; (e) automotive safety; (f) surveillance and traffic monitoring. [1]

level features include texture and shape; high level features include semantic information. Some famous and powerful hand-crafted features have been widely applied, such as Histogram of Oriented Gradient (HOG) [45], Scale-Invariant Feature Transform (SIFT) [46], Speeded-Up Robust Features (SURF) [47], Aggregate Channel Features (ACF) [48] and Integral Channel Features (ICF)..

(iv) Detection/segmentation: some points or regions of image are selected for further processing. Decisions are made with extracted features based on classification methods, such as simple matching, or machine learning methods like Support Vector Machine (SVM) [49] and AdaBoosting [50],

(v) Recognition: generally, categories of regions are classified with machine learning methods, such as SVM, Random Forest (RF) [51], Multi-Layer Perception (MLP) and Convolutional Neural Network (CNN).

(vi) Decision making: making final decisions required from applications. For example, pass or fail in automatic inspection applications, match or no-match in recognition applications and flag for further human review in more complex applications.

2.2 Traffic Sign Recognition

Acquisition of information from various traffic signs is crucial in many applications, such as Advanced Driver Assistance Systems (ADAS) and autonomous cars. Generally, a traffic sign recognition system involves two related issues: Traffic Sign Detection (TSD) and Traffic Sign Recognition (TSR). The former aims to accurately detect traffic signs in images, while the later intends to identify the labels of detected traffic signs into specific categories/subcategories. Though the topic has attracted research interests in computer vision community for more than two decades [52, 53], there are still challenges due to various complexities, such as diversified illumination conditions, capturing angles and open set environment.

Traffic Sign Detection Methods: TSD is similar to other object detection tasks in computer vision. More specifically, identifying the image regions with bounding boxes that tightly contain traffic signs. Comparing with common objects, traffic signs are usually designed to have rigid and simple shapes, uniform and attractive colours. The spatial relationship with other visual objects along the road is also an important cue that could be exploited in the development of TSD. Therefore, depending on how traffic signs are segmented, approaches can be divided into colour-based methods, geometry-based methods and machine learning-based methods.

Colour-based methods implement object detection based on colour information, which usually have low computational costs and strong robustness to projective distortion. A road sign detection system based on HSI colour space segmentation and SVM [49] classification is proposed in [54]. Colour feature clustering is adopted to perform image segmentation in [55]. Maximally Stable Extremal Regions (MSERs) is introduced in [56]for traffic sign candidates proposal, followed by SVM classifiers trained based on HOG [45] features. A method that combines the information of the colour, saliency, spatial and contextual relationship is proposed in [57]. In [58], colour probability model and MSERs are jointly used to generate traffic sign candidates. A hybrid region proposal method which consists of MSERs and Wave-based Detector (WaDe) is reported in [59]. On the other hand, as geometric shape is another important cue for detection of traffic signs, works have been proposed along this line, including Radial Symmetry Detector [60] and Triangular Detector [61]. TSD methods via both colour segmentation and shape matching are employed in [62, 63].

TSD has also become inseparable with machine learning. For example, AdaBoosting [50] is employed for TSD in [64]. An enhanced Common-Finder AdaBoost (CF.AdaBoost) algorithm is proposed in [65]. ACF [48] and ICF detectors are reported in [66]. In [67], an accurate and efficient traffic sign detection approach is proposed by exploring both AdaBoost and support vector regression for discriminative detector learning. Due to the power of representational learning from raw data, deep learning has acquired general interests in recent years. Therefore, CNNs have also been applied in TSD systems. A real-time TSD system that based on CNN features with sliding window scheme is reported in [63]. In [68], a TSD system is proposed with two deep learning methods, including Fully Convolutional Network (FCN) for traffic sign proposal and deep CNN for object classification.

Traffic Sign Recognition Methods: TSR can be generally treated as a pattern recognition problem, with many mature off-the-shelf techniques from machine learning. Among the plenteous models, SVM demonstrates its good performance, which has been applied in [54, 69, 56]. SIFT [46] and SURF [47] descriptors trained with MLP achieved high recognition rates in [70]. RF [51] and k-d tree with HOG features are reported in [18]. Radial Basis Function (RBF) neural network and K-D tree are used in [71] to identify the content of traffic signs. A sign similarity measurement with SimBoost and fuzzy regression tree method is proposed in [72]. An ensemble of classifiers based on the Error-Correcting Output Code (ECOC) method is introduced in [64], where the ECOC is designed through a forest of optimal tree structures that are embedded in the ECOC matrix. Sparse Representation Classification (SRC) [73] that proposed for face recognition is used for TSR in [74], sparse representation based graph embedding method achieves high performance by combining feature selection and subspace learning. In [75], multi-modal tree-structure embedded Multi-

Task Learning (MTL) is proposed to solve TSR problem, which selects features for better recognition and shares correlated features for similar tasks.

As for any recognition problem, feature representation is the critical factor for system performance. How to design discriminative and representative features has been in the central stage of computer vision research. Benefiting from powerful task-specific features, CNNs based methods achieved state-of-the-art performance in the competition of German Traffic Sign Recognition Benchmark (GTSRB) [22] which is held by International Joint Conference on Neural Networks 2011 (IJCNN 2011). A multi-scale CNN is introduced in [19], different from traditional CNN that the output of the last stage is fed to a classifier, in multi-scale CNN, the outputs of all the stages are fed to a classifier, yielding excellent performance. A committee CNN and MLP is proposed in [20] which is able to automatically learn task-specific invariant features in a hierarchical manner. Both these two methods outperform the human performance, however, the computational cost is relatively high. Hinge-loss CNN [23] further improved the recognition performance by introducing a cost function that widely used in SVM.

2.3 License Plate Recognition

License Plate Recognition (LPR) methods enable automated detection and recognition of the registration numbers of vehicles via digital imaging. It is a key component for many security and Intelligent Transportation System (ITS). Examples include car park access control, electronic highway toll payment systems, suspect vehicle analysis and tracking, and automatic identification of expired registrations. LPR has been studied for many years [76–78], with a variety of methods proposed for different kinds of license plates. Many countries have deployed the technology as a main constituent of urban transportation infrastructures.

It is hard to make a comprehensive survey of the published works on LPR. Fortunately, several review papers provide an important source of information [79–82]. There is a great deal of variability between license plates between different countries with regards to the design, colours, characters used, and layout. Previously published LPR methods has been mainly centered upon a small number of the countries, including: mainland China [83–85], Taiwan [86–89], American [90], Korea [91], and certain European countries [79, 76, 77].

Most of the existing LPR systems are based on a common structure, the consecutive composition of: (i) LPD; (ii) character segmentation; (iii) character recognition. The plate detection is considered as the bottleneck problem in a number of recent publications [85]. In order to design an effective LPR system, license plates not only need to be detected correctly, but also fast enough to meet the requirements of real-time applications. The various image

conditions also should be taken into account. Many methods have been proposed in recent years, including different approaches based on edge statistical analysis [92], morphological filtering [93], Bag Of Visual Words [85], and AdaBoost [94]. It should be noticed that the published methods are often proposed with respect to certain controlled environments.

LPD is usually followed by character segmentation, which breaks the cropped license plate region into single characters. Effective segmentation is another critical step for LPR as the last step character classification depends largely on the quality of the segmentation output. Segmentation is usually difficult in practice, especially when dealing with images with inherently noisy, low resolution and poor weather conditions. The most common practice of character segmentation is based on histogram analysis and thresholding [79–82]. A priori knowledge about the layout of the characters on the plate is often exploited, for example, the spacing between two characters. Despite the efforts made, there is no generally accepted method that can well solve the problems such as skewed or illuminated plates detection. Moreover, separately handling LPD and character segmentation has the disadvantage that incorrect detection can directly lead to wrong character segmentation. Therefore, simultaneously plate detection and character segmentation are essential for a reliable LPR system.

2.4 Traffic Text Detection

Generally, Traffic Text Detection (TTD) is closely related to Scene Text Detection (STD). Therefore, the literatures involved with these two topics are reviewed.

2.4.1 Traffic Text Detection Methods

Despite there are numerous methods that pay particular attention to the detection of symbol-based traffic signs [60, 56, 66], the method specifically focuses on the detection of the text-based traffic signs has attracted much less attention. During the past decades, only a few methods [95–103] have been proposed.

Most existing methods [95, 96, 99, 101–103] for TTD follow the pipeline: (i) Traffic Panel Detection (TPD); (ii) TTD on panels. In [95], traffic panels are detected by using of Shi and Tomasi features [104], Gaussian mixture models and K-means algorithm. Traffic panel candidates are segmented based on white and blue colours, followed with Fast Fourier Transform (FFT) for shape classification in [96]. A TPD method is reported in [101], which detected panels based on colour segmentation and Bag of visual words (BOVW) [105]. MSERs are applied in [102] and [103] for TPD. However, TPD still remains challenges

due to the huge variability of traffic panels. In real world, every panel is specially designed for displaying different information, leading to unpredictable variations in size, colour, and shape. Therefore, there maybe detection failure of TTD that caused by detection failure of TPD. The direct detection of traffic texts should be a better choice.

In the methods [101–103], traffic texts are detected by applying MSERs. Considering the following two aspects: (i) MSERs selects regions that are extremely stable and uniform in their colour; (ii) in order to be easily noticed by drivers and pedestrians, traffic texts are usually designed to have uniform and attractive colours. It is reasonable and effective to apply MSERs as region generator. Inspired by these methods [101–103], MSERs is also employed in the proposed system. However, the methods [101–103] are designed for detecting English and Spain words, where the words are commonly composed of several consistent letters. Different from English and Spain words, Chinese characters are usually constructed of several inconsistent strokes, the performance of system will be influenced if only MSERs is applied as region generator.

2.4.2 Scene Text Detection Methods

Since both TTD and STD aim to detection text in the wild, TTD can be significantly inspired by STD. Over the past decade, as the increasing of powerful computer vision methods, STD has been rapidly developed [106–108]. Based on how candidate regions are proposed, approaches to this problem can be generally divided into two groups: connected component based methods [109–136]. and sliding window based methods. [137–149]

Connected component based methods aim to segment individual text components based on information such as colour, edge, gradient and stroke width, and then group the text components with similar properties together to construct texts. The representative methods in this group are Stroke Width Transform (SWT) [119], MSERs [150], and Extremal Regions (ERs) [151]. SWT is a local image operator which computes width of the strokes associated to each pixel, and it has been applied in the methods [115, 116, 130]. In contrast to SWT, MSERs selects regions that are extremely stable in their colour, and MSERs based methods [109, 110, 117, 118, 120, 122, 125–127, 131, 132, 135] have achieved excellent performance in STD. Rather than regions with extremely stable colour, ERs are connected components of an image binarised at different thresholds without stability requirements, excellent results also have been achieved in ERs based methods [111, 113, 133, 114].

Sliding window based methods implement text detection as a common binary classification problem. A variety of machine learning methods have been applied in these methods. For example, AdaBoosting [50] is applied in [143, 144, 148, 149] for STD, and WaldBoost [152] is used with HOG [45] features in [142]. Rather than using boosted cascade classifier,

SVMs [49] trained on HOG features in a sliding window scenario are reported in [147]. Instead of using HOG features, CNN features are applied in [146, 137], and the results indicate that CNN features can work effectively in sliding window scheme. Later on, a region generator based on ACF [48] detector and Edge Boxes [153] is proposed in [138]. Recently, FCN [36] are introduced for STD in [141, 139], resulting in excellent performance.

Candidate region proposal is the first stage of a text detection system, its performance is of crucial importance to success of entire system. The connected component based methods have exhibited the great advantages in computational speed and robustness to scale, aspect and rotation variations. However, it is difficult to detect text components, which are constructed of isolated parts, blurred or partially covered. In order to address these problems, Stroke Feature Transform (SFT) is proposed in [130] by incorporating colour cues of text pixels. Region topology is applied in [110] as a selector, resulting in an extended version of MSERs, which is called MSERs++. In [135], Contrast-Enhanced MSERs (CE-MSERs) is developed based on MSERs, which allows to detect highly challenging text patterns. A Colour-Enhanced Contrasting Extremal Regions (CERs) is reported in [133]. On the other hand, sliding window based methods have strengths of robustness to noise and blur, since strong features and classifiers are introduced. Text components with isolated parts can also be proposed as a whole. The major limitation is the high computational cost when texts with different scales, aspects and rotations have to be detected.

Most of the existing methods have focused on detecting horizontal or near-horizontal text lines, there are only a few methods [115, 116, 141, 117, 118, 121, 131, 135] can handle non-horizontal text lines. Multi-oriented text detection systems [115, 116] are developed by pairing candidates and aggregating pairs with similar orientations. Later on, this method is also applied by In [135]. By constructing a graph of MSERs for each input image, text line detection problem is transferred into a graph partitioning problem in [131]. Multi-orientation STD systems based on text candidates clustering are reported in [117, 118]. Generally, texts on traffic panels are skewed due to capturing angles. Furthermore, the complex layout of texts on traffic panels result in not only multi-oriented text lines, but also crossed text lines. Therefore, in order to solve the problems of multi-oriented and crossed text line detection, improved method should be developed.

As the natural proprieties of English, there is plenty of intra-word and inter-word contextual information within texts. It is extremely rare that English text only contains one letter. Therefore, many existing methods [119, 115, 126, 116, 135] assumed: texts should have at least 2 components, otherwise they are directly regarded as noises. In contrast to English text detection, Chinese text detection is more complex and difficult due to the lack of intra-character contextual information. In China, most of traffic texts contain only two or

three characters, even many stand-alone characters. Therefore, powerful classifiers should be applied in Chines TTD systems, so that individual characters can be accurately detected.

2.5 Convolutional Neural Networks

While the deep CNNs [29] achieve a series of breakthroughs in various computer vision tasks, there are a lot of researches that focused on improving the network architecture of the original AlexNets. In [154], multi-scale and sliding window scheme are implemented within CNN for simultaneously classify, locate and detect objects in images. A novel visualisation technique is proposed in [155] for visualising and understanding the function of intermediate feature layers and the operation of entire network. Based on the observation, two modifications are made: (i) the kernel size of the first layer is adjusted from 11×11 to 7×7; (ii) a stride value 2 is adopted in the first layer rather than 4, resulting in the ZFNets that achieved the best performance in ILSVRC 2013. Furthermore, the CNNs [30] proposed by Visual Geometry Group (VGG) indicate another important direction of network architecture design, which is network depth. By introducing small kernel size (3×3) in all layers, the computational cost is significantly reduced and it becomes feasible to add more convolutional layers. As a result, the famous VGGNets with network depth from 11 to 19 are created and have been widely used in lots of subsequent works. Later, the 22-layer GoogLeNets [31] are reported, and the network architectures are finely optimised based on Hebbian principle and multi-scale processing. The new deeper and wider architectures are code-named Inception, because of the Inception modules in networks. Due to the carefully crafted architectures, GoogLeNets win the champion in ILSVRC 2014.

Deep CNNs naturally extract low-level features to high-level features in an end-to-end multi-layer scheme [155], and the levels of feature can be enriched by increasing depth of networks. The importance of network depth has been proven by both VGGNets [30] and GoogLeNets [31], which employ more convolutional layers and achieve better performance than the AlexNets [29]. However, with the increasing of network depth, the training of network becomes more and more difficult. The most notorious problems are gradient vanishing and exploding [156]. Fortunately, with the aid of methods such as normalised initialisation [156, 157] and normalisation layer [158], the gradients problems have been largely addressed, and networks with very deep depth start to converge. Subsequently, another critical problem is exposed in [159, 160]: as the network depth increasing, both training and testing accuracy get saturated firstly, and then degraded. This problem is known as degradation [32], which is not caused by overfitting, but the depth of network. In order to solve the degradation problem, networks based on deep residual learning [32] are proposed.

By introducing identity shortcut connections, each few stacked layers are able to learn residual mapping rather than direct mapping. Formally, denoting a desired direct mapping $H(x)$ and a residual mapping $F(x) = H(x) - x$, then the original desired mapping can be represented as $H(x) = F(x) + x$. The results of experiments prove that the optimisation of residual mapping is much easier than original direct mapping. As a result, the depth of networks and accuracy of CNNs are dramatically increased, and ResNets win the champion in ILSVRC 2015.

Recently, in order to overcome gradient vanishing and degradation problems, short path from early layers to later layers have been widely applied in a number of CNN architectures. For instance, Highway Networks [159] and ResNets [32, 161] introduce identity shortcut connections for training deep CNNs. Stochastic Depth Networks [162] utilise a novel training strategy which randomly drop a subset of layers of ResNets and bypass them with identity connection. FractalNets [163] develop an alternative strategy called drop path for training ultra deep networks. Later, the connection strategy is further distilled in [164]. In order to ensure the maximum information flow inside networks, each layer receives inputs from all preceding layers and passes to all subsequent layers. Due to the densely connected structure, the network architectures are named as DenseNets. Different from ResNets, features from preceding layers are combined by concatenation instead of summation. Comparing to ResNets, DenseNets have the following advantages: (i) higher parameter efficiency; (ii) easier training; (ii) stronger regularising effect for reducing overfitting.

2.6 Summary

In this chapter, the literatures related to research are reviewed. In section 2.1, an overview of computer vision is presented, and the related methods of computer vision are introduced, including image acquisition, data preprocessing, feature extraction, detection/segmentation, recognition and decision making. In section 2.2, TSR methods are reviewed. Depending on how traffic signs are segmented, TSD approaches can be divided into colour-based methods, geometry-based methods and machine learning-based methods. For TSR, machine learning methods such as SVM, RF and CNN have been widely applied, and excellent results have been achieved. In section 2.3, LPR methods are reviewed. Most of the existing LPR systems are designed for certain controlled environments, otherwise, the performance maybe largely reduced. In order to design a robust and effective LPR system, it should be able to applied in a wide range of environment, and detect license plates with high accuracy. In section 2.4, TTD and STD methods have been reviewed in detail. Over the past decade, as the increasing of powerful computer vision methods, text detection has been rapidly developed.

However, there are still problems, such as Chinese text detection, multi-oriented and crossed text lines detection. In section 2.5, the development of CNNs have been reviewed. As the increasing of depth of networks, CNNs face training problems such as gradient vanishing, gradient exploding and degradation. With the help of normalisation and learning methods, the problems have been well eased. However, they are not totally solved, new method are expected to tackle these problems.

Chapter 3

Traffic Sign Detection with Multi-level Chain Code Histogram

This chapter proposes a real-time system for Traffic Sign Detection (TSD), which performs detection by template matching with features based on shapes, namely Multi-level Chain Code Histogram (MCCH)). For the three shapes, circle, triangle and inverted triangle that are associated with Chinese traffic signs, MCCH is an excellent feature with good representation ability and low computational cost. Therefore, MCCH is very suitable for detecting traffic signs in real-time. The proposed system consists of three stages: (i) segmentation based on colour; (ii) feature extraction with MCCH; (iii) template matching based on histogram intersection. Experiments are conducted using collected datasets, demonstrating good performance with regard to high processing speed and robustness to rotation, scale, and illumination.

3.1 Introduction

Traffic signs are designed to streamline traffic flow and provide information to road users for improving the road efficiency and safety. Drivers or pedestrians should make appropriate response to different traffic signs such as various warning and speed limits to secure safety. The detection and recognition of traffic sign have been studied for more than two decades [52, 53], mainly motivated by the applications such as Advanced Driver Assistance Systems (ADAS) and autonomous cars.

TSD methods are expected to capture information from wild environment, three main stages are usually involved: (i) candidate segmentation; (ii) region detection; (iii) category recognition. Generally, traffic signs are designed with regular shapes and attractive colour to

be easily noticed. Therefore, the information of shape and colour are widely used in TSD. Based on however prior knowledge is used, TSD could be roughly categorised into colour-based methods and shape-based methods. Colour-based methods rely on colour to segment the regions that belong to traffic signs, with main advantages such as low computational cost and robust to projective deformation. However, colour-based methods are sensitive to illumination changes. On the other hand, shape-based methods are robust to illumination, the disadvantage is the high computational cost.

In order to address the mentioned problems, a system for real-time TSD is proposed, which generates candidates by colour segmentation, extracts shape information based on MCCH and detects traffic signs with template matching. The rest of this chapter is organised as follows. Section 3.2 presents a brief introduction of the entire TSD system. Section 3.3 explains the colour space thresholding algorithm for segmentation. Section 3.4 introduces proposed feature extraction method MCCH. Section 3.5 presents the procedures of template matching. Section 3.6 introduces the experimental results, followed by summary in Section 3.7.

3.2 System Overview

The proposed TSD system consists of three main stages, as illustrated in Fig. 3.1. In the first stage, the input image is segmented by using Ohta Space Thresholding. In the second stage, features are extracted with MCCH. In the third stage, the traffic signs are detected based on template are matching.

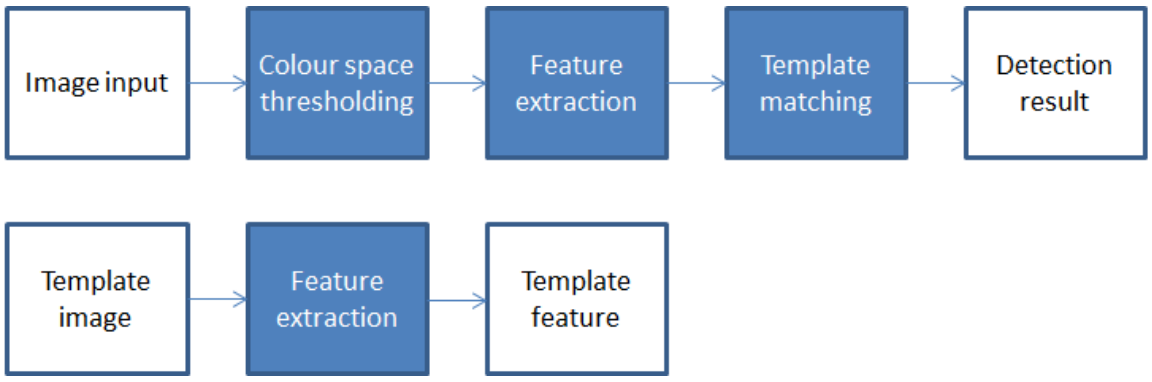


Fig. 3.1 System overview. The up steps are applied in testing stage. The down step is applied for extracting features of templates.

The TSD system can be described in the following steps in detail.

(i) Colour space thresholding. The original input RGB image is converted into binary image by applying Ohta Space Thresholding [53, 165], followed by connected component analysis and edge detection to produce corresponding contour images. In order to decrease computational cost and false positive rate, regions with small areas (≤ 20 pixels) are directly removed since traffic signs in image are impossible to have such small areas.

(ii) Feature extraction. For each contours produced in the Step 1, they features are extracted by using proposed MCCH.

(iii) Template matching. By comparing the MCCHs of the three shapes in Fig. 3.2 with all of the MCCHs of the contours produced from last step, Histogram Intersection [166] distances are calculated which estimate the matching degree between templates and contours. The Histogram Intersection distances are regarded as confidential scores of contours. Finally, threshold value is used to select true traffic signs and reject poor matches.

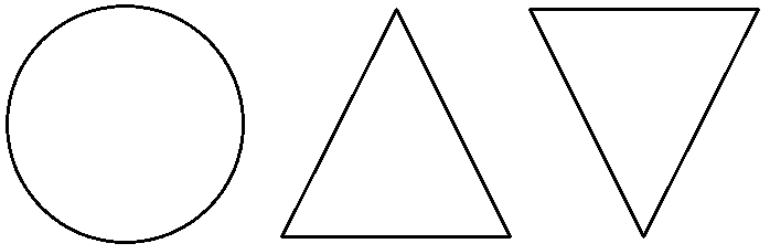


Fig. 3.2 Chinese traffic signs have three kinds of shapes, which are circle, triangle and inverted triangle.

3.3 Colour Space Thresholding

Ohta space thresholding: a colour space is a specific organisation of colours. The research for an effective colour space in colour-based segmentation has resulted in variety of colour models, such as CIEXYZ [167], CIELUV[167], CIELAB[167], and YIQ[167]. Among the variety of spaces, Ohta space [53] shows some significant characteristics. First, the implementation of Ohta space is quite simple and it has a very low computational cost. Second, the aim of Ohta space is focus on finding the best uncorrelated components, thus, all the components are independent to each other. After extensive experiments, the authors of

[165] invented a set of features for the three colours that are derived from RGB. The set of colours can be directly applied for image segmentation:

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 1 & 0 & -1 \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.1}$$

where the component I_1 is mapped to illumination and the components I_2 and I_3 are related to colour categorisation. Therefore, these two components can be applied in segmentation directly. Moreover, in order to increase the robustness to illumination changes, normalisation is also performed. The specific designed normalised components P_1 and P_2 are formulated as:

$$P_1 = \frac{1}{\sqrt{2}} \frac{R-B}{R+G+B} = \frac{1}{3\sqrt{2}} \frac{I_2}{I_1} \tag{3.2}$$

$$P_2 = \frac{1}{\sqrt{6}} \frac{2G-R-B}{R+G+B} = \frac{2}{3\sqrt{6}} \frac{I_3}{I_1} \tag{3.3}$$

Finally, by introducing the normalised components, the colours can be acquired according to the following formulations:

$$Red(i,j) = \begin{cases} True,\ if\ P_1(i,j)) \geq ThR_1\ and\ P_2(i,j)) \leq\ ThR_2 \\ False,\ Otherwise \end{cases} \tag{3.4}$$

$$Blue(i,j) = \begin{cases} True,\ if\ P_1(i,j)) \geq ThB_1\ and\ |P_2(i,j))| \leq\ ThB_2 \\ False,\ Otherwise \end{cases} \tag{3.5}$$

$$Yellow(i,j) = \begin{cases} True,\ if\ P_1(i,j)) \geq ThY_1\ and\ |P_2(i,j))| \leq\ ThY_2 \\ False,\ Otherwise \end{cases} \tag{3.6}$$

where i and j are coordinates in image. All the threshold values are given in Table.

Based on this method, the original input RGB image is converted into binary image, followed by feature extraction that will be detailed in the next section.

3.4 Feature Extraction

A chain code is a traditional method used to describe an object boundary with an ordered sequence of n straight line segments $c_i, i = 1,2,...,n$, where c_i is a vector of connecting

Threshold name	Threshold value
ThR_1	0.024
ThR_2	-0.027
ThB_1	-0.04
ThB_2	0.082
ThY_1	0.071
ThY_2	0.027

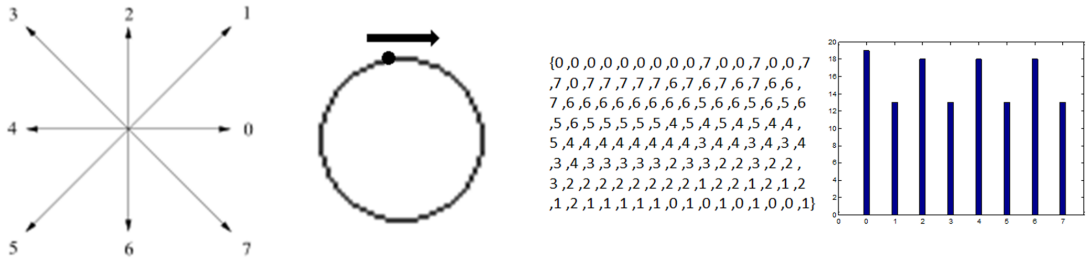
Table 3.1 Parameters of Ohta space thresholding.

neighboring contour pixels. The directions of c_i are coded with integer values $k = 0, 1, ..., K-1$ in a counter clockwise fashion starting from the direction of the positive x-axis, where K is the number of directions defined by an integer value 4 or 8 [168].

To better describe a contour, the histogram of the chain code can be calculated and applied. The Chain Code Histogram (CCH) based on a chain code can be formulated by the following discrete function:

$$CCH(i) = \frac{n_k}{k},\ k = 0, 1, 2..., K-1 \tag{3.7}$$

where n_k is the accumulated number of each direction k in a chain code, and n is the total length of a chain code.



(a) (b) (c) (d)

Fig. 3.3 Illustration of CCH for describing a contour. (a) Definition of an eight directions chain code. (b) A sample circle and its contour. (c) Chain code of the circle. (d) CCH of the circle.

The details of CCH have been illustrated in Fig. 3.3. Fig. 3.3a shows an eight directions chain code, and Fig. 3.3b gives a sample circle and its contour. The starting point for the chain code is marked with a black circle, and clockwise is adopted as the moving direction of the chain code. The chain code and the corresponding CCH of the circle are illustrated in Fig. 3.3c and 3.3d.

A CCH is a global feature for describing the shape of a contour. In order to achieve higher CCH distance between different contours, a cascaded CCH can be created by breaking the original contour into small contour fragments and concatenating their CCHs. To be more specific, a cascaded CCH can be achieved by the following stages. First, the chain code of a contour is divided into a number of small fragments, the fragment size is chosen empirically. And then the CCH of each fragment is individually calculated. Finally, all the achieved CCHs are sequentially concatenated. Supposing the original chain code is equally divided into m parts, all the CCHs at different parts are concatenated to form a cascaded CHH. Thus, the cascaded CCH descriptor is a vector with dimension of $8m$. The improvements can be illustrated by Fig. 3.4, where the chain code of the contour in Fig. 3.3 is divided into 4 fragments and the corresponding CCHs are concatenated together.

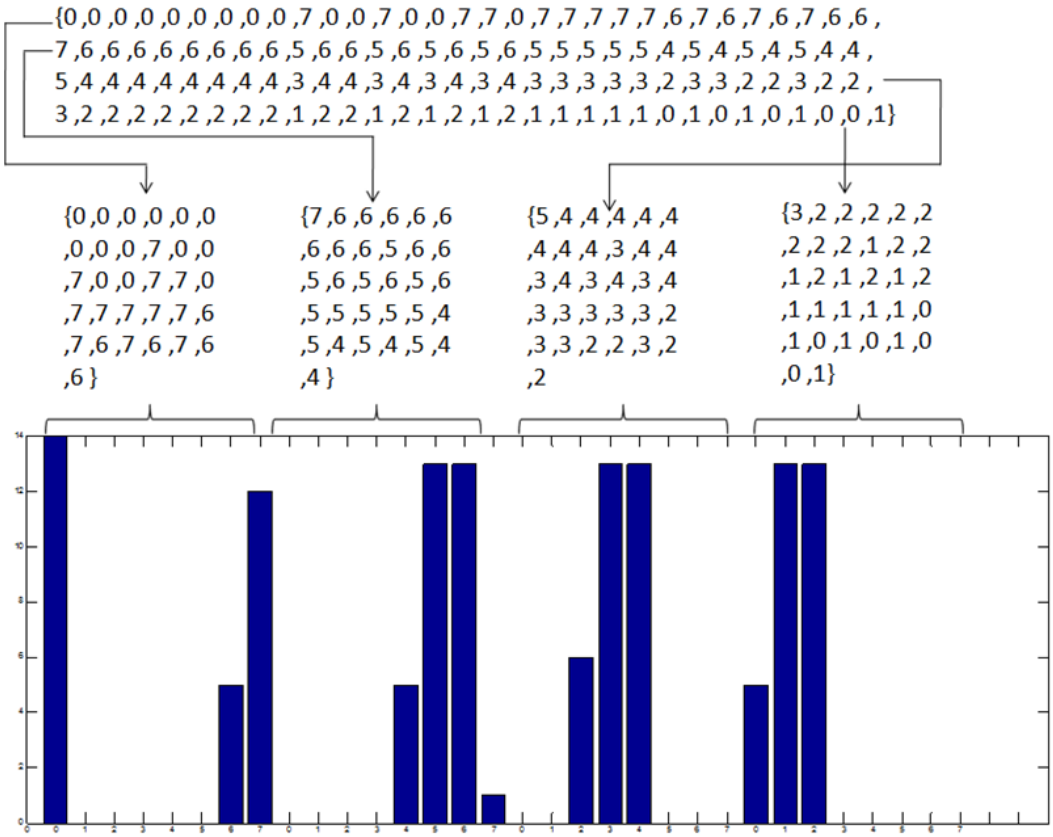


Fig. 3.4 Illustration of a cascaded CCH.

Comparing with the original CCH, a cascaded CCH is more powerful and representative. However, in real world TSD, skew and distortion are very common. In order to achieve better robustness and rotation invariance, cascaded CCH can be further extended to MCCH.

More specifically, a MCCH can be defined as follows. The first level CCH is the original CCH. From the rest levels, chain code of the contour is divided into different number of small fragments. And then the CCH of each fragment is calculated and concatenated. As Fig. 3.5 illustrates, a *2-level* MCCH can be simply achieved by cascading the original CCH and a *4-parts* cascaded CCH. Since the length of each small CCH is 8, the length of *2-level* MCCH is 40, which is small compared with common features, such as Histogram of Oriented Gradient (HOG) [45].

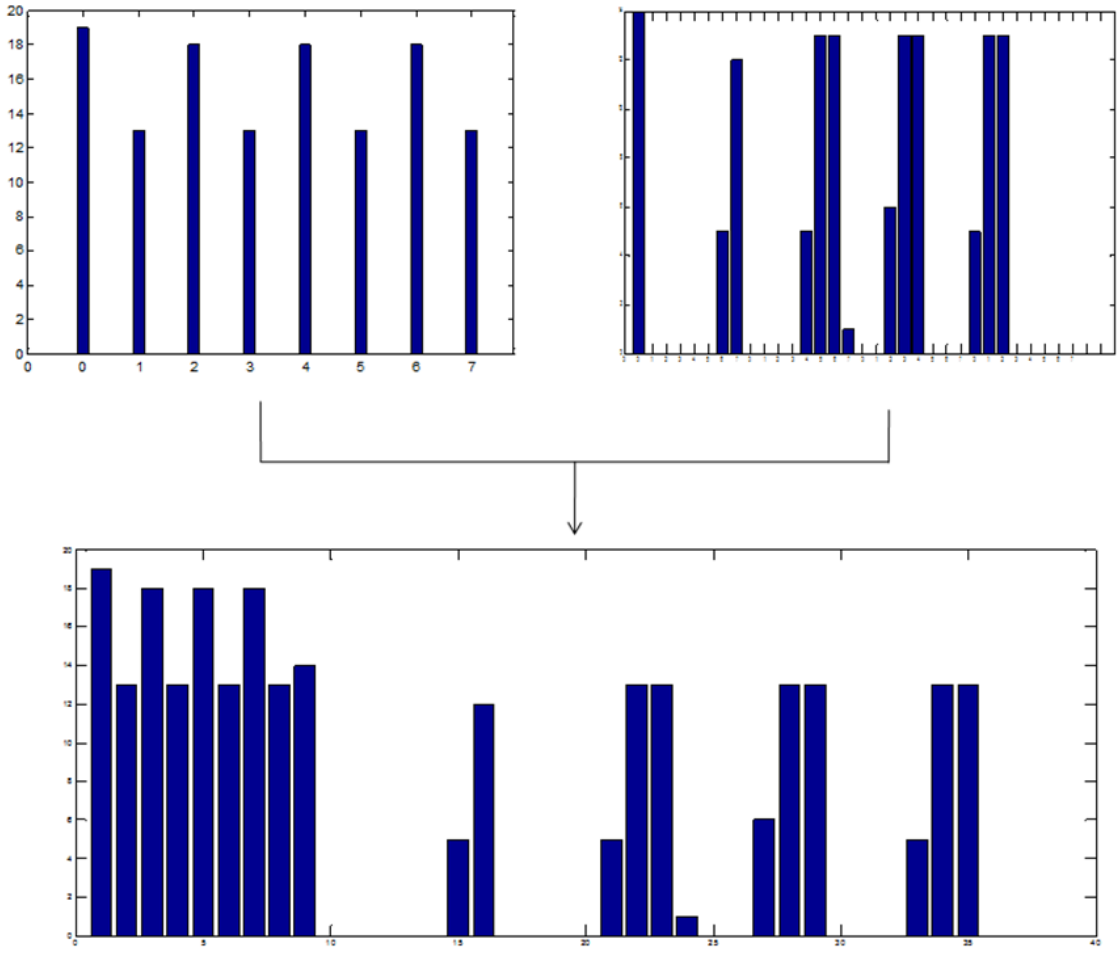


Fig. 3.5 Illustration of a *2-level* MCCH.

3.5 Template Matching

Template matching is a technique for detecting a part of an image that matches a template. The main advantages include the efficiency of its implementation and the effectiveness with respect to the detection of low-textured objects that mainly characterised by shape.

In this stage, four algorithms are applied to calculate distances between MCCHs of templates and candidates, including Histogram Intersection [166], Chi-square, match distance and Euclidean distance. In order to test the performance of different algorithms, a dataset with 100 positive images and 50 negative images is built. The four methods are tested based on this dataset. The results are compared in the Table 3.2.

Method	True positive	False positive
Histogram Intersection	95	3
Chi-square	94	4
match distance	94	3
Euclidean distance	95	5

Table 3.2 Performance comparison of different distance measuring algorithms.

Based on the results, Histogram Intersection is the best algorithm that gives the most true-positive number with the least false-positive number. Therefore, Histogram Intersection is used in proposed system.

3.6 Experiments

In this section, the proposed system is verified on a self collected traffic sign dataset. The details of the experiments will be presented in the following subsections.

3.6.1 Data Collection

The dataset is recorded by a camera set up in vehicle. It includes three categories of traffic signs that based on the shapes and colours:

Prohibitory: circle, red rim, white or red inner.

Mandatory: circle, blue rim, blue inner.

Danger: triangular, black rim, yellow inner.

Images in dataset are captured by a camera with 640×480 resolution. The total number of the images is 300. To be more specific, 150 images contain prohibitory signs, 80 images contain mandatory signs and 70 images contain danger signs. The sizes of traffic sign in the images vary from 25×25 to 80×80. Some examples are illustrated in Fig. 3.6.

Fig. 3.6 Illustration of example images from the collected data, traffic signs are highlighted with green rectangles. Top: prohibitory signs, middle: mandatory signs, bottom: danger signs.

3.6.2 Performance Evaluation

In order to observe the performance of the proposed TSD system, different decision thresholds of template matching are used. The details are shown in the following precision-recall curves 3.7.

Fig. 3.7a illustrates the detection results of prohibitory signs. With the increasing of threshold in template matching, the recall rate rises with the decreasing of the precision rate. According to the curve, the best trade-off is achieved at the position that recall value equals 0.95 and precision value equal to 0.9. The detection results of mandatory signs have been provided in Fig. 3.7b. The performance is slightly lower, because of low resolution and poor quality that caused during data collection. The detection results of danger signs have been shown in Fig. 3.7c. Different from the previous two categories of traffic signs, the shape of danger signs is triangular, which means the detection of danger signs is easily affected by skewed angle or deformation. Therefore, the detection of danger signs faces more challenges than the other two categories of the traffic signs.

3.6.3 Computational Time Evaluation

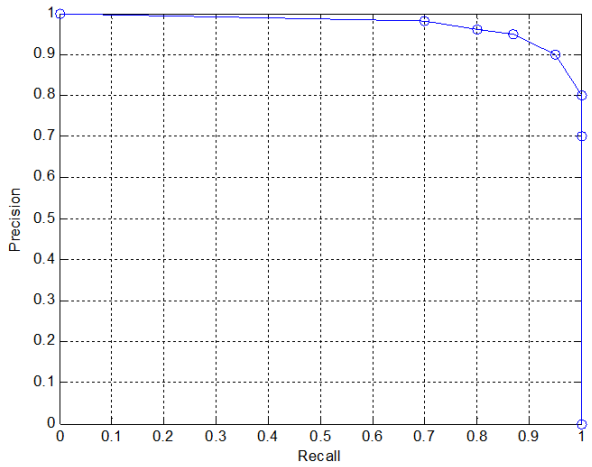
To implement the proposed system, a workstation with an Intel Xeon 3.4GHz CPU and 16GB is employed. The programs are compiled with OpenCV and ran on a 64-bit Windows 7 operating system. The computational time for each image is about 50ms, which means this system can work in real-time application.

3.6.4 Error Evaluation

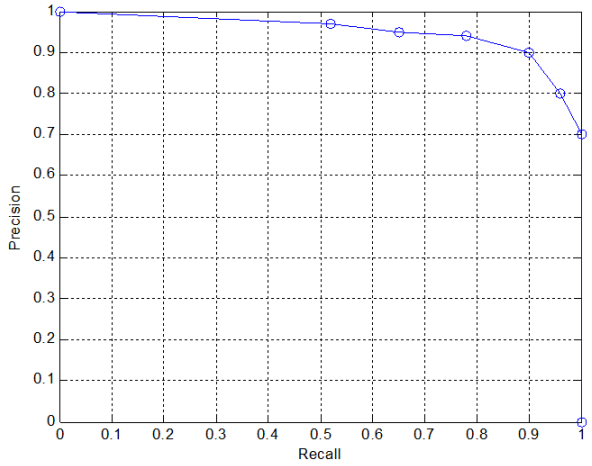
The proposed TSD system failed to detect traffic signs under some extreme conditions, such as poor illumination, occlusion and low resolution. More specifically, poor illumination may make traffic signs have unstable colours, leading to failures in colour segmentation; occlusion and low resolution make traffic signs have unstable shapes, resulting in failures in template matching. Some of the failed example are illustrated in Fig. 3.8.

3.7 Summary

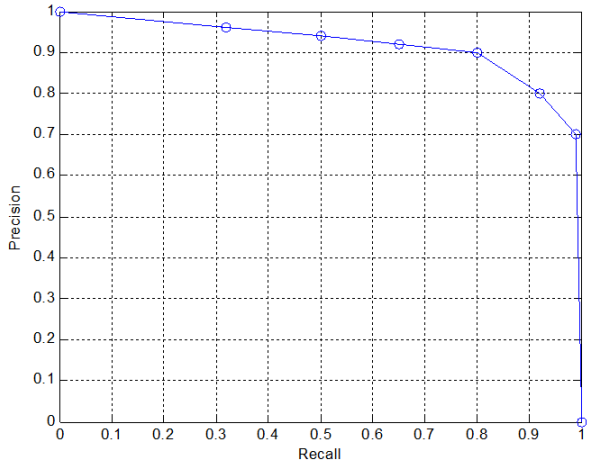
In this chapter, a TSD system is proposed, which mainly consists of three stages. In the first stage, the input image is segmented using Ohta space thresholding. In the second stage, connected component analysis and edge detection are performed, followed by proposed feature extraction algorithm that is named MCCH. In the final stage, the confidence of each



(a)



(b)



(c)

Fig. 3.7 Precision-recall curves of different traffic signs. (a) Prohibitory sign. (b) Mandatory sign. (c) Danger sign.

Fig. 3.8 Illustration of example images that failed in detection, traffic signs are highlighted with red rectangles.

traffic sign is estimated by applying Histogram Intersection. Experiments verified that the feature MCCH has good representation ability and low computational cost. And the proposed system achieves good performance for detecting prohibitory, mandatory and danger traffic signs with skew and deformation in real-time environment.

Chapter 4

Road Surface Traffic Sign Detection with Fast R-CNN

Previous research in Traffic Sign Detection (TSD) generally focused on traffic signs over roads, traffic signs painted on road surface have not been well discussed. In this chapter, a road surface TSD system is proposed by applying a hybrid region proposal method and a Convolutional Neural Network (CNN). The proposed system consists of two main stages: (i) a hybrid region proposal method to hypothesise the traffic sign locations by taking into account complementary information of colour and edge; (ii) a multi-task network to simultaneously implement category prediction and bounding box regression based on the Fast R-CNN framework. Experiments have been conducted with a self collected road surface traffic sign dataset, demonstrating excellent performance with regard to the high recall and precision rate, the overall Average Precision (AP) is about 88.7%.

4.1 Introduction

The extraction of information from various symbols-based and text-based traffic signs is a key component in many applications, such as Advanced Driver Assistance Systems (ADAS), autonomous cars, traffic surveillance and many other tasks in Intelligent Transportation System (ITS). Though the endeavour of the automatic detection of traffic signs for the past two decades, it is still a challenging problem due to the variations of scale, viewpoint and illumination condition.

Generally, traffic signs are erected at the side of roads, above roads or painted on the surface of roads. While there have a lot of works focused on the automatic detection of symbol-based traffic signs [54–62, 64–67, 63, 68] and text-based traffic signs [95–97, 99, 101–

103] at the side of or above roads, traffic signs painted on road surface have not been well discussed. Different from traffic signs at the side of or above roads, the detection of traffic signs on road surface faces more challenges, for example, larger scale and viewpoint variation, colours fading, reflection and occlusion.

Recently, the development of deep learning has attracted lots of attention in computer vision and pattern recognition research as more and more promising results are published on a range of different tasks. Benefit from the huge success achieved by R-CNN [33], object detection based on CNN has been significantly progressed. Furthermore, by introducing RoI pooling and multi-task CNN, Fast R-CNN [34] achieves better performance and faster speed. More specifically, Selective Search [169] is firstly applied for generated candidate bounding boxes, which are subsequently labeled and regressed with a multi-task CNN. The simultaneous classification and regression design results in improved performance. The training and testing speeds are boosted up to 10 times faster.

Due to the mentioned advantages, Fast R-CNN is employed in the proposed road surface TSD system. In order to decrease computational cost, the original region proposal method Selective Search [169] is not adopted. Considering traffic signs are usually designed to have rigid and simple shapes, uniform and attractive colours, MSERs [150] and Edge Boxes [153] are applied together for taking into account the complementary information from colour and edge. The main contributions of this research are presented as follows.

(i) Creation of a new road surface traffic sign dataset. This dataset contains 2223 images and 4204 road surface traffic signs, which have been published to facilitate the research and development of road surface TSD.

(ii) Introduction of a hybrid region proposal method, which takes into account the complementary information from both colour and edge by applying MSERs and Edge Boxes. The integration of these two region proposal methods improves the detection performance.

(iii) A multi-task network based on Fast R-CNN to perform classification and bounding box regression simultaneously. The advantages of Multi-Task Learning (MTL) include fast detection speed and high detection performance.

The rest of this chapter is organised as follows: Section 4.2 presents the proposed road surface traffic sign dataset; Section 4.3 gives a detailed introduction of the proposed TSD system; experimental results are provided in Section 4.4, followed by summary in Section 4.5.

4.2 Road Surface Traffic Sign Dataset

4.2.1 Details of Dataset

The proposed dataset is built as the following stages.

(i) A dashboard camera is set up in a car with a fixed shooting angle for capturing videos with a resolution of 1280×720 pixels.

(ii) All the videos are decompressed and the frames that contain road surface traffic signs are manually collected.

(iii) In order to protect privacy, all the regions with privacy information are blurred, such as faces and vehicle license plates.

(iv) The ground-truth of all the road surface traffic sign in the images are annotated .

(v) All the annotated images are further divided into two parts, namely training and testing sets.

Consequently, a dataset with 2223 images and 4204 road surface traffic signs are built. The details are shown in Table 4.1. Some of the examples with different capturing conditions are illustrated in Fig. 4.1 .

	Training	Testing	Total
Images	1606	617	2223
Traffic signs	2996	1208	4204

Table 4.1 Details of the proposed road surface traffic sign dataset.

The traffic signs in proposed dataset have large variations in resolutions, with distribution of resolutions provided in Table 4.2.

4.2.2 Evaluation Protocols

In order to evaluate detection performance on proposed dataset, the evaluation protocols that adopted in PASCAL VOC object detection competition [170] are employed. Four evaluation metrics are included, which are *Intersection over Union (IoU)*, *Precision*, *Recall* and *AP* respectively.

IoU is introduced to evaluate the predicted bounding boxes. All predicted bounding boxes are assigned to ground-truth and discriminated to be true/false positives based on

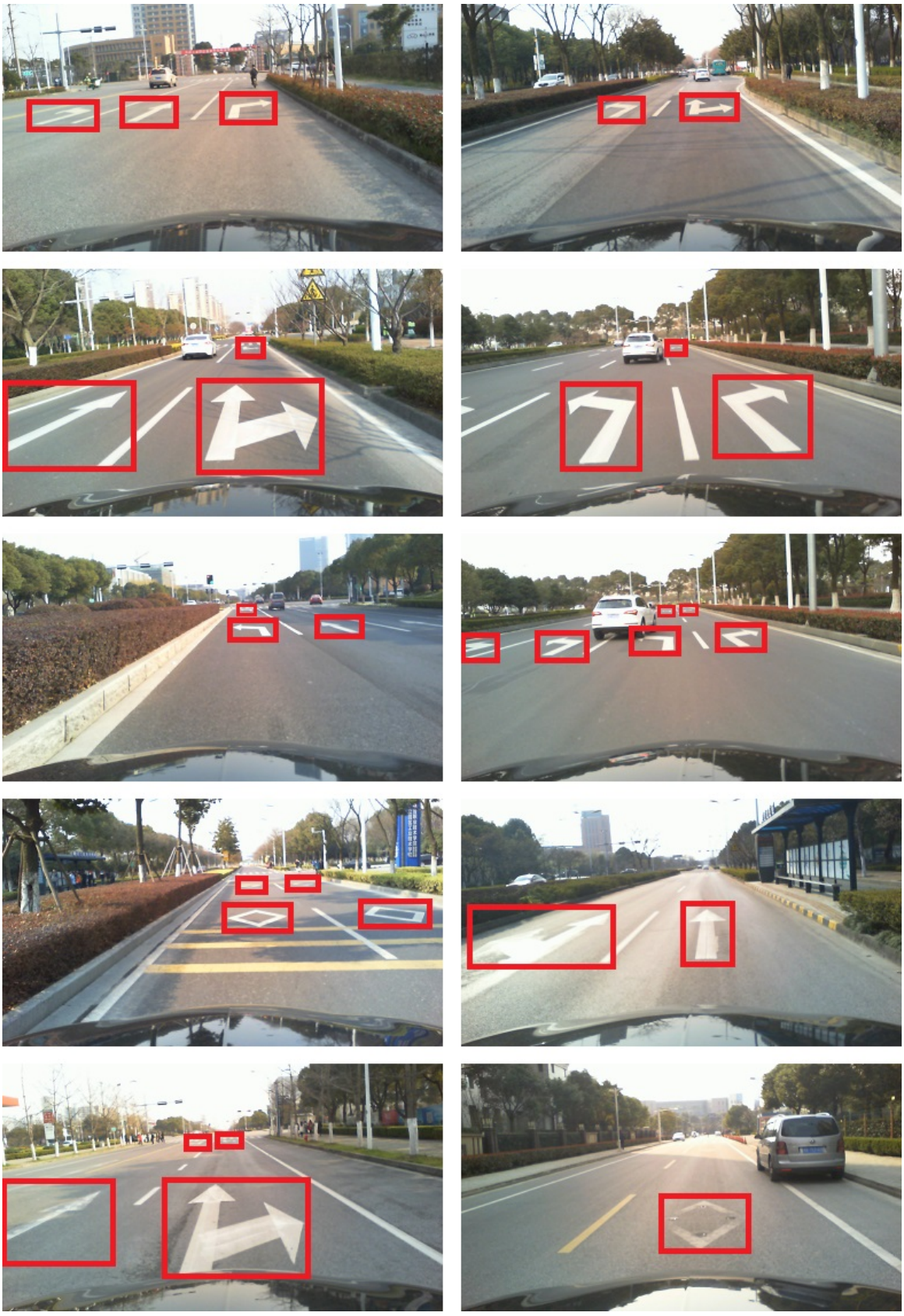


Fig. 4.1 Examples of the collected dataset. The annotated road surface traffic signs are marked with red rectangles. The top row: traffic signs in easy case. The second row: traffic signs with large resolution. The Third row: traffic signs with small resolution. The fourth row: traffic signs with strong illumination. The bottom row: traffic signs with colour fading.

Resolution of Traffic Signs	Training	Testing	Total
$\leq 60 \times 60$	55	28	83
$60 \times 60 \sim 120 \times 120$	437	203	640
$120 \times 120 \sim 180 \times 180$	775	305	1080
$180 \times 180 \sim 240 \times 240$	718	266	984
$240 \times 240 \sim 300 \times 300$	451	174	625
$300 \times 300 \sim 360 \times 360$	242	112	354
$360 \times 360 \sim 420 \times 420$	165	50	215
$420 \times 420 \sim 480 \times 480$	122	51	173
$\geq 480 \times 480$	31	19	50
Total	2996	1208	4204

Table 4.2 Details of traffic signs from the proposed road surface traffic sign dataset.

measuring bounding box overlap. For a correct detection, the overlap ratio (IoU) between the predicted bounding box B_p and ground-truth bounding box B_{gt} should be at least 0.5. The definition of IoU is formulated as:

$$IoU = \frac{Area(B_p \bigcap B_{gt})}{Area(B_p \bigcup B_{gt})} \tag{4.1}$$

where $B_p \bigcap B_{gt}$ and $B_p \bigcup B_{gt}$ denote the intersection and union of the predicted and ground-truth bounding boxes respectively.

Precision is defined as the proportion between all true positive samples and all predicted samples, and *Recall* is defined as the proportion between all true positive samples and all ground-truth samples. *Precision* and *Recall* are formulated as:

$$Precision = \frac{|TP|}{|D|} \tag{4.2}$$

$$Recall = \frac{|TP|}{|G|} \tag{4.3}$$

where TP, D and G are the sets of true positive samples, all detected samples and all ground-truth samples respectively.

The AP collects the mean precision under different recall values, which are eleven equally spaced recall levels $[0, 0.1, \ldots, 1]$:

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} P_{interp}(r) \tag{4.4}$$

For every recall level r, the maximum precision is selected from all the precisions that have corresponding recalls larger than r:

$$P_{interp}(r) = \max_{\tilde{r}:\tilde{r}\geq r}(P(\tilde{r})) \tag{4.5}$$

where $P(\tilde{r})$ are all the precisions that have corresponding recalls larger than r.

4.3 Approach

The overall detection system is demonstrated in Fig. 4.2 which mainly consists of two stages. In the first stage, candidate regions that may contain the desired traffic signs are generated by region generators. In the second stage, all the generated regions will be passed to Fast R-CNN [34] with following processing steps: (i) feature extraction with CNN; (ii) classification and bounding box regression by multi-task CNN; (iii) Non-Maximal Suppression (NMS) for the final outputs.

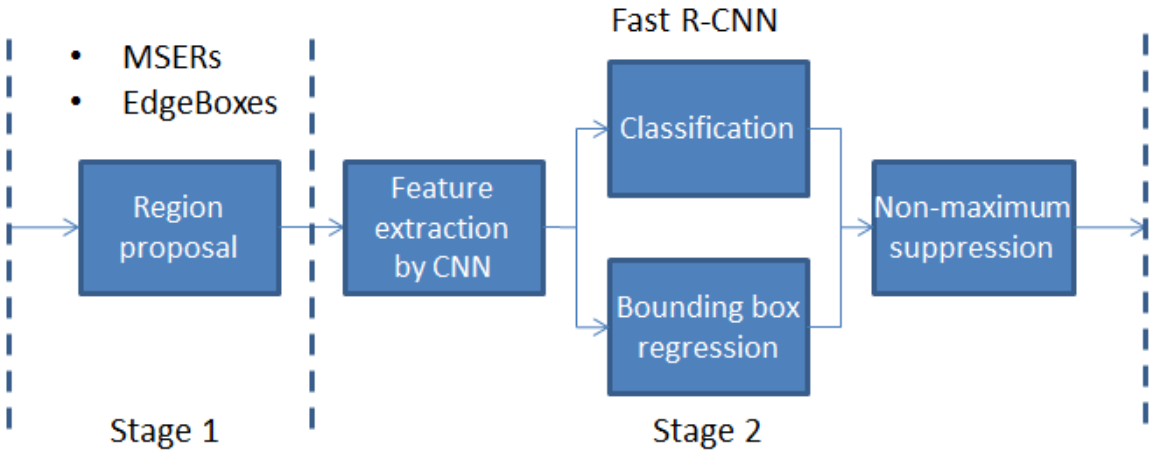


Fig. 4.2 System overview of the proposed TSD system.

4.3.1 Region Proposal

The first stage of the end-to-end TSD pipeline relies on the generation of candidate traffic signs, or object proposals. Generally, a precision-recall tradeoff is required to reduce the computational complexity. And in this system, high recall is pursued in this region proposal stage while high precision will be achieved by rejecting false positive candidates in later stages.

The detection targets include a number of different traffic signs, as illustrated in Fig. 4.3. Due to the difficulties such as large scale variation, viewpoint change, colours fading,

reflection and occlusion, targets may have difference in shape, scale and colour. Therefore, bounding boxes are combined together from two proposal methods, namely, MSERs and Edge Boxes.

Fig. 4.3 Illustration of road surface traffic signs.

Maximally Stable Extremal Regions

MSERs [150] denote a set of outstanding regions that are proposed in an input image. The extremal property of these regions are defined by its intensity function. More specifically, an input image is firstly binarised using different threshold levels, and then connected components analysis is used to find all the connected components at each threshold level. Finally, the regions that maintain their shape and area at several thresholds are selected as MSERs. MSERs are scale-invariant and affine-invariant. Due to these advantages, MSERs have been broadly applied in various detection tasks, which usually have targets can be generated by binarisation and connected components analysis.

Edge Boxes

Recently, Edge Boxes [153] has shown excellent performance comparing with widely used method Selective Search [169]. The key intuition of EdgeBoxes is that: the number of contours wholly covered by a bounding box is a cue of the possibility of containing object. To be more specific, objects usually have obvious and enclosed edges. Following this intuition, a bounding box that contains enclosed edges tends to have object inside. As road

surface traffic signs have distinguished and sharp boundaries, it seems to be particularly true that objects are composed of plentiful boundaries. The pipeline in [153] has the following stages: (i) structured edge map detectors are exploited to compute dense edge response maps; (ii) NMS is applied orthogonal to the edge response in order to find edge peaks, resulting in sparse edge maps; (iii) for each candidate bounding box B, a score S_B is assigned based on the number of edges wholly contained within the bounding box B.

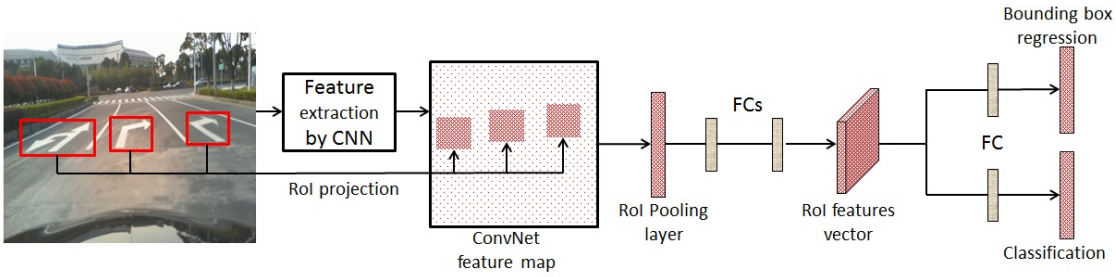


Fig. 4.4 System overview of Fast R-CNN.

4.3.2 Classification and Localisation by Fast R-CNN

R-CNN [33] has demonstrated state-of-the-art performance in PASCAL VOC object detection competition [170]. However, R-CNN has its bottlenecks in three aspects: (i) high computational cost in training and testing stages; (ii) huge storage capacity request in training stage; (iii) separated training for classification and bounding box regression. In order to solve the three problems, an improved scheme has been designed and proposed, which is called Fast R-CNN [34]. With the help of MTL and Regions of Interest (RoI) pooling, the training and testing speed is significantly improved, detection accuracy is also benefited from the new design.

The system overview of Fast R-CNN is illustrated in Fig. 4.4. Different from R-CNN, Fast R-CNN takes a whole image as input. In the first step, a feature map of entire input image is extracted at the final convolution layer, with the addition of position of each region proposal, the features of each regions are obtained. Therefore, it is not necessary to repetitively extract features of same areas that contained by different regions, which largely decreases computational cost. In the second step, all the feature maps are further processed by a RoI pooling layer, which is a special case of the Spatial Pyramid Pooling [171] layer that samples arbitrary size of region proposal to fixed size. In the third step, Fast R-CNN performs classification and bounding box regression by a multi-task network. Therefore, different from the separate training in R-CNN, Fast R-CNN simultaneously trains the two tasks based on this network design. Since the entire system can be trained with standard

back propagation, it is not necessary to store features for training classifiers like in R-CNN, storage capacity requirement is largely reduced.

4.4 Experiments

In this section, first, implementation details will be introduced, including system setup and network configuration. And then, the experimental results will be presented and discussed. Finally, computational time evaluation and error evaluation will be provided.

4.4.1 Implementation Details

System Setup

To implement the road surface TSD system, a workstation with an Intel Xeon 3.3GHz CPU, 48GB memory and a NVIDIA GTX Titan GPU is employed. The programs run on a 64-bit Windows 7 operating system with CUDA 7.5, CUDNNv5, Matlab 2015b and MatConvNet 1.0-beta23 [172] deep learning toolbox.

Network Configurations

The architecture of CNN used in the system is based on the 16-layer network (VGG16) [30], which have demonstrated excellent performance in image classification and object detection. The detailed network configurations are illustrated in Table 4.3. The input are RGB images have been preprocessed to 1066×600. All the convolutional layers are activated by Rectified Linear Unit (ReLU) [173], and zero padded for preserving dimensionality. All max pooling layers have kernel size 2×2 and stride 2. The RoI pooling layer samples all the input feature maps to 7×7. The first and second fully connected layers are also activated by ReLU layers, followed by dropout [174] layers with drop rate 0.5 for preventing over-fitting.

The training of the network consists of two stages, namely, pre-training and fine-tuning. In pre-training stage, a pre-trained VGG16 network initialised on ImageNet is loaded. The last max pooling layer is replaced by a RoI pooling layer. And then the last fully connected layer is replaced with a new fully-connected layer that initialised with *Improved Xavier* [157], followed by softmax loss and smooth L_1 loss. Furthermore, the network inputs are modified to use information from both images and region bounding boxes in the images. In fine-tuning stage, Stochastic Gradient Descent (SGD) is adopted to update network parameters. Each mini-batch is constructed from 2 images, which are selected randomly and uniformly. For each image, 16 regions have $IoU \geq 0.5$ with a ground-truth bounding box are selected as

positive samples. 48 regions have maximum $IoU < 0.5$ with all ground-truth bounding boxes are selected as negative samples. Consequently, a mini-batch B with 128 samples is constructed. Data augmentation is applied by rotating and zooming samples during training. The learning rate η is set to be 0.01, momentum 0.9, weight decay 0.0005 and maximum training epoch 200 as applied in [34].

Weighted layers	Classification	Bounding box regression
	Input	
1	Convolution $3 \times 3, 64$	
2	Convolution $3 \times 3, 64$	
	Max pooling	
3	Convolution $3 \times 3, 128$	
4	Convolution $3 \times 3, 128$	
	Max pooling	
5	Convolution $3 \times 3, 256$	
6	Convolution $3 \times 3, 256$	
7	Convolution $3 \times 3, 256$	
	Max pooling	
8	Convolution $3 \times 3, 512$	
9	Convolution $3 \times 3, 512$	
10	Convolution $3 \times 3, 512$	
	Max pooling	
11	Convolution $3 \times 3, 512$	
12	Convolution $3 \times 3, 512$	
13	Convolution $3 \times 3, 512$	
	RoI pooling	
14	FC 4096	
15	FC 4096	
16	FC 2	FC 4
	Softmax loss	Smooth L_1 loss

Table 4.3 Network configurations of CNN used in the system.

4.4.2 Experimental Results

Performance Evaluation of Region Proposal

The recall rates for region proposal stage are shown in Fig. 4.5. At IoU threshold of 0.5, the recall rate for Edge Boxes ($\alpha = 0.75$, $\beta = 0.55$, *minimum score* $= 0.01$ and *maximum boxes* $= 1000$), MSERs and the combination of the two methods are 84.6%,

87.6% and 92.2% respectively. Therefore, the combination of the two proposal methods shows boosted performance. The missed proposal regions are mainly caused by scale variations, changes of illumination conditions and colours fading.

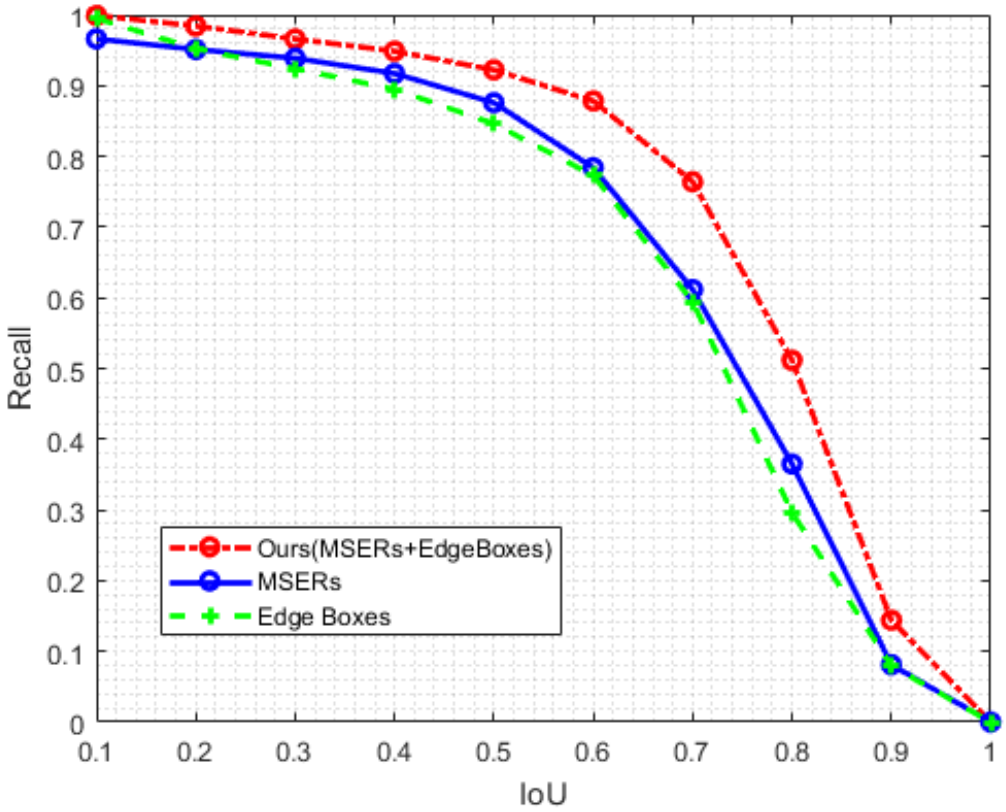


Fig. 4.5 Illustration of the recall rates with the changes of IoU between the range $(0.1, 1)$ for proposed method.

The comparison of the performance with the five region proposal methods that have been widely used is illustrated in Fig. 4.6, including MSERs applied in [56, 58, 59, 102, 103], Selective Search applied in [33], achromatic RGB applied in [54], achromatic HSI applied in [175] and achromatic Ohta applied in [165]. At IoU threshold of 0.5, the recall rates are 92.2%, 87.6%, 88.7%, 68.0%, 69.7% and 69.8% respectively. The average numbers of proposals per image are 1294, 299, 2065, 40, 39 and 39 respectively, the details are presented in Table 4.4. Therefore, the proposed method has the highest recall rate, and the number of region is reasonable.

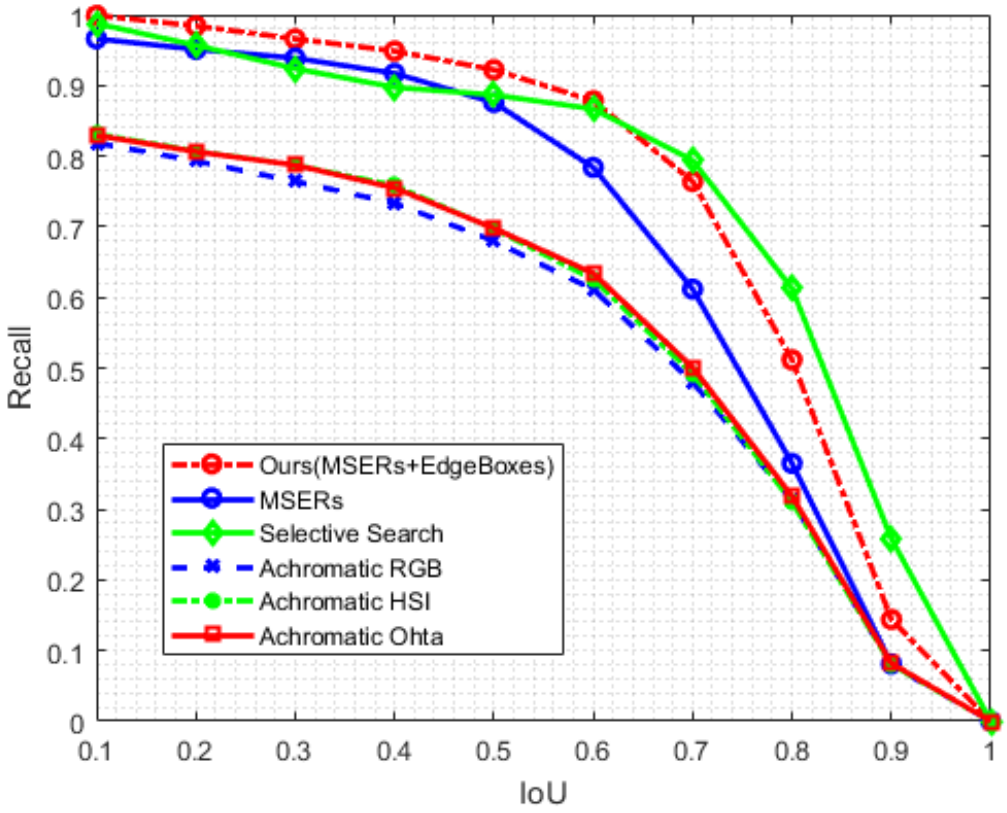


Fig. 4.6 Illustration of the recall rates with the changes of IoU between the range $(0.1, 1)$ for the six methods.

Type	Proposals/image
Ours	1294
MSERs	299
Selective Search	2065
Achromatic RGB	40
Achromatic HSI	39
Achromatic Ohta	39

Table 4.4 Average numbers of proposals per image.

Performance Evaluation of Detection System

The Precision Recall (PR) curves for the overall system are demonstrated in Fig. 4.7. Two experiments are set up for evaluating the performance of bounding box regression and MTL, namely, Fast R-CNN with or without bounding box regression. As the Fig. 4.7 illustrates, the highest recall rates achieved by these two experiments are 93.3% and 92.2% respectively. Comparing with the recall rate 92.2% obtained in region proposal stage, Fast R-CNN with bounding box regression attains 1.1% improvement, which indicates that bounding box regression indeed improves performance, regions with low IoU value: (< 0.5) can be regressed to have higher IoU value: (> 0.5). On the other hand, by introducing MTL for bounding box regression, the precision of multi-task CNN is also higher than single-task CNN, which proofs that MTL is able to improve classification accuracy. The APs of the two experiments are 88.7% and 87.2% respectively.

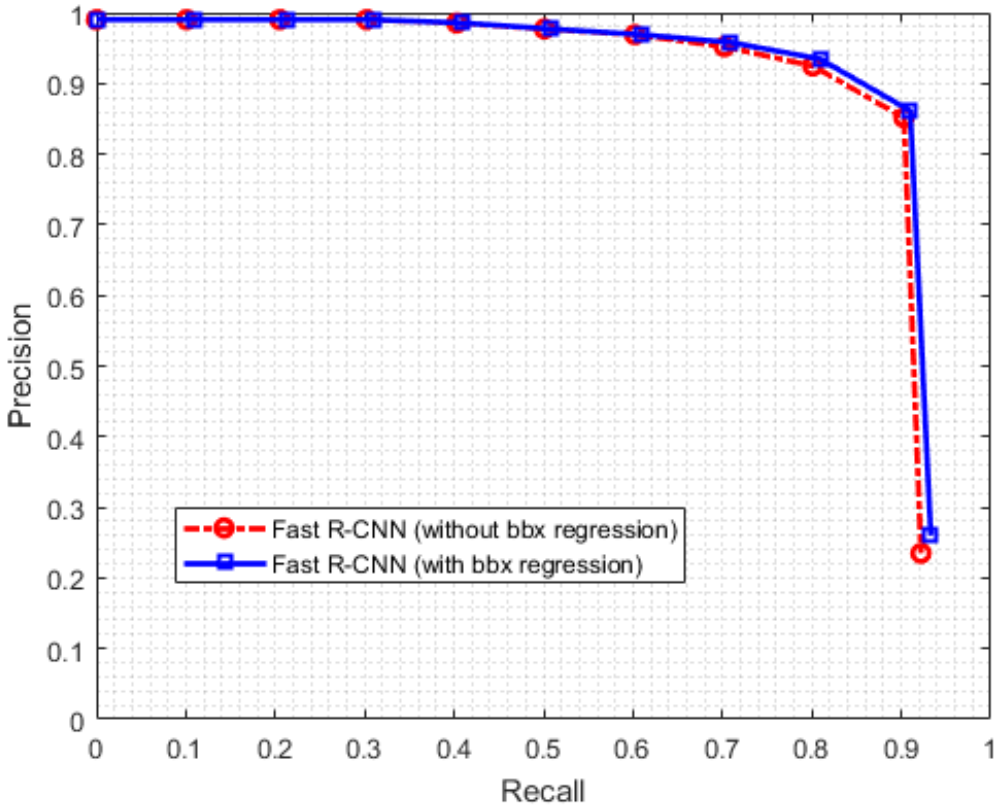


Fig. 4.7 Illustration of PR curves for the proposed system.

The comparison of the performance with the five widely used detection systems have been illustrated in Table 4.5, including R-CNN [33], HOG + SVM [56], Haar + AdaBoosting [50] and ACF + AdaBoosting [66]. As the Table 4.5 shows, R-CNN achieves similar performance

to the system (Fast R-CNN), however, the training and testing speed of R-CNN is much slower. On the other hand, the detection systems HOG + SVM [56], Haar + AdaBoosting [50] and ACF + AdaBoosting [66] are faster, however, due to the weak representative power of the features, the performance of these systems are much lower.

Method	AP
Ours (Fast R-CNN)	88.7%
R-CNN (CNN + MLP) [33]	87.5%
R-CNN (CNN + SVM) [33]	78.3%
HOG + SVM [56]	45.6%
Haar + AdaBoosting [50]	42.5%
ACF + AdaBoosting [66]	51.3%

Table 4.5 Performance comparison of the six detection systems.

4.4.3 Computational Time Evaluation

The detailed computational time are presented in Table 4.6. For each input image, the average computational times (ms) are 2124, 2483, 2539, 653, 243 and 352 for the proposed system, R-CNN, HOG + SVM [56], Haar + AdaBoosting [50] and ACF + AdaBoosting [66] respectively.

4.4.4 Error Evaluation

Although good performance has been achieved on the road surface traffic sign dataset, there still exists missing detections, which are mainly caused by scale variations, changes of illumination and colours fading. Some of the examples are shown in Fig. 4.8. Since the recall rate of regions proposal is only about 92.2%, region proposal is the bottleneck of the proposed system.

4.5 Summary

In this chapter, a road surface traffic sign detection system is proposed, with main contributions including: (i) a hybrid region proposal method which takes into account the complementary information of colour and edge; (ii) fast R-CNN to perform classification and bounding box regression simultaneously. By combining two popular region proposal methods, namely MSERs and Edge Boxes, high recall rate is ensured in region proposal stage. As the features extracted from CNN are discriminative, high recall and precision rates

Method	Stage	Time (ms)
Ours (Fast R-CNN)	Region proposal	898
	Classification	1226
	Total	2124
R-CNN (CNN + MLP) [33]	Region proposal	898
	Preprocessing	1035
	Classification	550
	Total	2483
R-CNN (CNN + SVM) [33]	Region proposal	898
	Preprocessing	1035
	Classification	606
	Total	2539
HOG + SVM [56]	Region proposal	165
	Preprocessing	475
	Classification	13
	Total	653
Haar + AdaBoosting [50]	Detection	243
ACF + AdaBoosting [66]	Detection	352

Table 4.6 Computational time analysis of widely used detection systems.

Fig. 4.8 Examples of failed detections.

of the entire system are achieved on collected dataset. Comparing with the five widely used methods, the system achieves highest AP, and comparable computational time.

Chapter 5

Traffic Sign Recognition with Convolutional Neural Networks

Recognition of traffic signs have been studied for more than two decades, and numbers of machine learning methods have been adopted, such as Random Forest (RF), Support Vector Machine (SVM), Multi-Layer Perception (MLP) and Convolutional Neural Network (CNN). Among all of the methods, CNNs demonstrate dominant performance due to the generalisation ability and learning capacity. In this chapter, traffic signs are recognised with the most influencing CNN architectures, including the very deep networks, residual networks and dense networks. Extensive experiments are presented and discussed based on different training algorithms, such as weight initialisation, regularisation and optimisation.

5.1 Introduction

Traffic Sign Recognition (TSR) with computer vision and machine learning methods is a key component with respect to many applications. Recognition of traffic signs allows a driver to be warned of inappropriate actions and potential dangers. In 2011, with the advent of public benchmark dataset GTSRB [22] that provides a number of difficult challenges such as viewpoint variation, poor illumination condition, motion-blur, occlusion, colours fading, and low resolution, the recognition of traffic signs has been significantly developed. A number of excellent approaches have been reported in the literatures [18–23], and CNNs have achieved the best performance due to the discriminative features. Later, in 2012, the 8-layer AlexNets proposed in [29] attracted extraordinary interests in computer vision and machine learning field by displaying significantly improved performance on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [176, 177]. Following the huge success achieved in

ILSVRC 2012, CNNs have been extended to various tasks, such as object detection [33–35] and image segmentation [178], and more and more promising results have been conducted based on a range of different benchmarks.

In this chapter, traffic signs will be recognised with CNN architectures that have been widely used, including CNNs [19], VGGNets [30], ResNets [32] and DenseNets [164]. Extensive experiments will be presented and discussed based on different weight initialisation algorithms, normalisation layers, regularisation strategies and optimisation algorithms. The rest of this chapter is organised as follows: Section 5.2 introduces the basic blocks of CNNs; Section 5.3 gives brief introduction of the most influencing CNN architectures; Section 5.4 summarises regularisation strategies for improving model generalisation ability; Section 5.5 illustrates optimisation algorithms of gradient descent; experiments will be presented and discussed in Section 5.6, followed by summary in Section 5.7.

5.2 Network Blocks

As the name indicate, convolutional networks [179] or CNNs, are a special kind of neural networks that process data by applying matrix convolution operation instead of matrix multiplication. Simply, CNNs are a sequence of layers that perform one operation to another through different functions. There are five kinds of layers that have been widely applied, including weight, pooling, activation, normalisation and loss layers. The individual network blocks in CNNs are detailed in Appendix A.1

Weight layers contain parameters that can be used to apply mathematical operations such as matrix convolution and multiplication. Depending on the intersections between inputs and outputs, there are three kinds of widely used weight layers, namely, convolution layer, convolution transpose layer and fully-connected layer.

The weight layers performs a set of linear activation. In order to construct nonlinear models, nonlinear activation layers should be introduced. Originally, sigmoid and hyperbolic tangent functions are often applied in neural networks. However, these activation functions face serious gradient vanishing problem as depth of neural network becomes deeper and deeper. Currently, Rectified Linear Unit (ReLU) [173] is normally adopted as activation function in deep models, because of the advantages such as sparse activation, scale invariant, efficient gradient propagation and efficient computation.

Typically, a convolutional block consists of three layers, namely, convolution, nonlinear activation and pooling. Pooling layer aims to further refine the input features at each certain location by summarising a statistic around it. Due to the working principle of pooling, the output features may invariant to small translation of the input features. This is a very useful

property if feature content is more important than feature location. Since pooling summarises the features over a nearby extent, the dimension of features is also reduced, this improves the computational efficiency of the next layers. Furthermore, in some models, the input data has varying size and pooling is essential in these networks.

Normalisation for input data, or known as preprocessing is an essential procedure in many computer vision applications. For example, images are usually normalised into reasonable ranges such as $[-1, 1]$ or $[0, 1]$. Moveover, normalisation layers inside neural networks are also very important for improving generalisation and convergence. In CNNs, there are two main normalisation layers, namely Local Response Normalisation (LRN) [29] and Batch Normalisation (BN) [158].

The function that minimised or maximised by model is called the objective function. If the model is aim to minimise the objective function, it is also called loss function or cost function. The choice of a loss function in the design of a CNN is very important. Fortunately, the design of loss function of a CNN can be largely inspired by lots of parametric models.

5.3 Network Architectures

Four most famous CNN architectures are applied in this chapter, namely CNNs [19], VGGNets [30], ResNets [32] and DenseNets [164]. Inspired by the famous LeNet-5 [180] that developed for handwritten zip code digits recognition, CNNs [19, 20] are also applied for TSR and achieved the best performance on GTSRB in 2011. The VGGNets [30] with network depth from 11 to 19 are developed by VGG from Oxford. These networks have prove the importance of network depth of CNNs. The ResNets [32, 161] are developed for solving the degradation problem [159, 160]: with the increasing of network depth, accuracy becomes statured and then degraded. Recently, short path from early layers to later layers have been widely adopted in a number of CNN architectures, such as Highway Networks [159], ResNets [32], Stochastic Depth Networks [162], FractalNets [163] and DenseNets [164]. Among all these networks, DenseNets achieve the maximum information flow inside the networks, each layer receives inputs from all preceding layers and passes to all subsequent layers. The details of the architectures are presented in Appendix A.2

5.4 Network Regularisation

How to achieve good performance on testing data, not just on training data is a key problem in machine learning field. The strategies that explicitly designed to increase testing performance are collectively known as regularisation. The widely used regularisation strategies in deep

learning are presented in Appendix A.3, including data argumentation, dropout, parameter norm penalty and early stopping.

5.5 Network Optimisation

Generally, most of deep learning algorithms are involved with optimisation that aims to minimise or maximise given objective functions $f(x)$. However, minimisation algorithms are normally focused, because maximisation can be accomplished by minimising $-f(x)$. For example, in the training of CNNs, gradient based optimisations are usually applied, and gradient descent is used to minimise the loss functions. More specifically, giving an input x, a CNN predicts an output $\hat{y}$ through forward propagation. With a target value y, the loss value $J(\theta)$ can be calculated. And then, the gradients from output layer to input layer are computed by back propagation [181]. Finally, gradient descent is adopted to update parameters of each layer of the CNN. The most used gradient descent algorithms are presented in Appendix A.4.

5.6 Experiments

In this section, first, the system environment for implementing all the TSR systems will be introduced. And then, the dataset employed in the experiments will be presented. Finally, the results of all the implemented TSR systems will be provided and discussed.

5.6.1 System Environment

In order to implement all the TSR systems, a desktop with an Intel 3.5GHz CPU, 32GB memory and a NVIDIA GTX Titan GPU is employed. All the programs run on a 64-bit Windows 10 operating system with CUDA 8.0, CUDNNv5.1, Matlab 2017a and MatConvNet 1.0-beta24 [172] deep learning toolbox.

5.6.2 Dataset Details

German Traffic Sign Recognition Benchmark

The GTSRB [22] is the most famous traffic sign dataset that has been published at IJCNN 2011. It contains 51839 traffic sign images that belong to 43 classes. Some examples of the traffic signs are illustrated in Fig. 5.1. The class distribution of the GTSRB is given in Fig. 5.2.

Fig. 5.1 Examples of the 43 traffic sign classes of the GTSRB.

The 51839 traffic sign images are collected from more than 1,700 traffic sign instances. The traffic signs are not necessarily squared, and their sizes vary from 15×15 to 222×193. The resolution distribution of the GTSRB is given in Table 5.1. Each image has a border of 10% (at least 5 pixels) around the traffic sign to allow for edge-based approaches. Finally, all the traffic signs are randomly split into two subsets, which are training set and testing set that contain 39209 and 12630 traffic signs respectively.

Resolution of traffic signs	Training	Testing	Total
≤ 25	117	40	157
≤ 35	11793	3970	15763
≤ 45	10604	3333	13937
≤ 55	6416	2063	8479
≤ 65	3791	1168	4959
≤ 75	2198	669	2867
≤ 85	1313	435	1748
≤ 95	907	294	1201
> 95	2070	658	2728
Total	39209	12630	51839

Table 5.1 Resolution distribution of the GTSRB (in pixel).

Data Preprocessing

The data preprocessing is applied based on three steps. First, the borders of all the traffic signs are removed according to the annotations. And then, based on the resolution distribution of the GTSRB, all the cropped traffic signs are resized to $48 \times 48 \times 3$ pixels. In other wards, the traffic signs are processed to have square bounding boxes. Finally, the colour normalisation of the traffic signs is utilised by linearly scaling all the pixels to the range $[-1, 1]$.

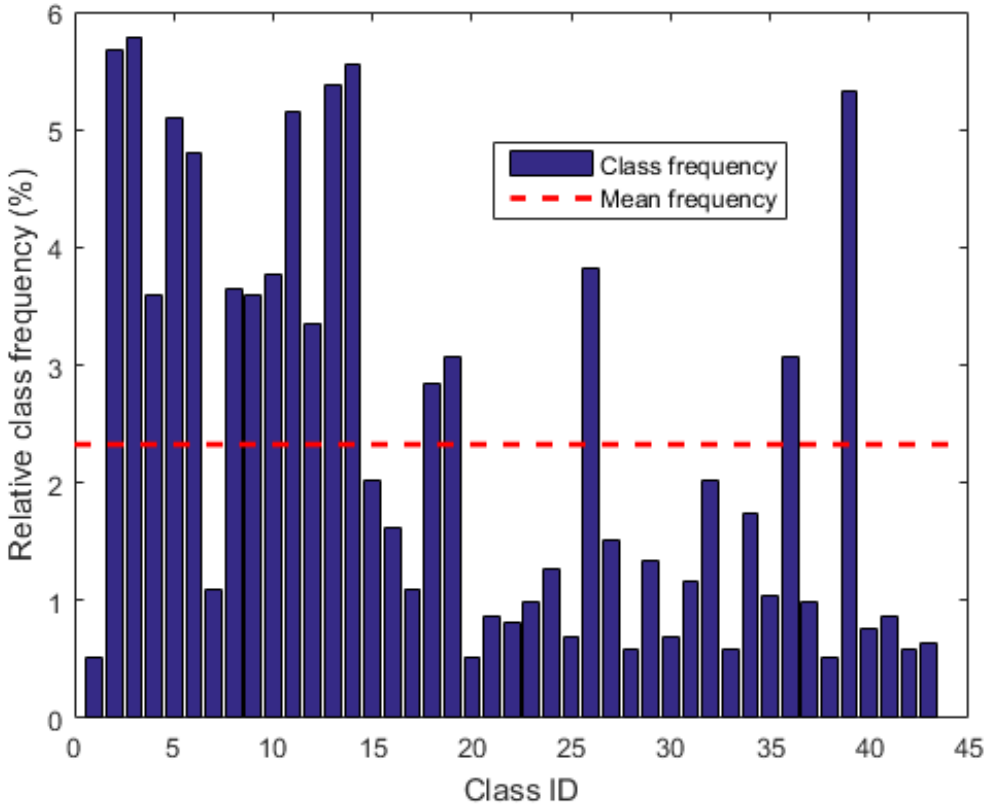


Fig. 5.2 Relative class frequencies of the GTSRB. The class ID is enumerated the classes in Fig. 5.1 from top-left to bottom-right.

5.6.3 Performance Evaluation

The results of the experiments will be presented in detail in this section. The comparative experiments are implemented based on different network architectures, regularisation strategies and optimisation algorithms. For all the experiments, data argumentation is not applied, and the generalisation ability is imposed via network architectures and some other regularisation strategies, including parameter norm penalty and dropout. In order to achieve more accurate

results, all the experiments with same setups are utilised for 10 times, and the experiments with the best performance will be presented. Except as otherwise indicated, all the training configurations are the same as Table 5.2.

Batch size	128
Epoch	10
Learning rate	10 logarithmically equally spaced points between the values: 10^{-2} and 10^{-3}.
Parameter initialisation	*Improved Xavier* [157]
Regularisation	L^2 parameter regularisation with value: 0.0005
	Dropout: 0.5
Optimisation	Momentum optimisation algorithm with momentum factor: 0.9

Table 5.2 The training configurations of all the experiments.

Network Architectures

First, the comparative experiments are presented and discussed based on the four famous network architectures listed as follows.

(i) PlainNets-5 from [20], as shown in Table 5.3;

(ii) PlainNets-7 similar to VGGNets [30], as shown in Table 5.4;

(iii) PlainNets-10/18/34 and ResNets-10/18/34 from [32], the details have been displayed in Tabel 5.5;

(iv) DenseNets-16/27/38 illustrated in Table 5.6, which are based on the networks in [164].

For all the experiments, the size of mini-batch B is set to be 64, learning rate η is sampled from 10 logarithmically equally spaced points between 10^{-1} to $10^{-1.5}$, L^2 regularisation value is chosen to 0.0001, and dropout is not applied so that the regularisation mainly depends on the design of network architectures.

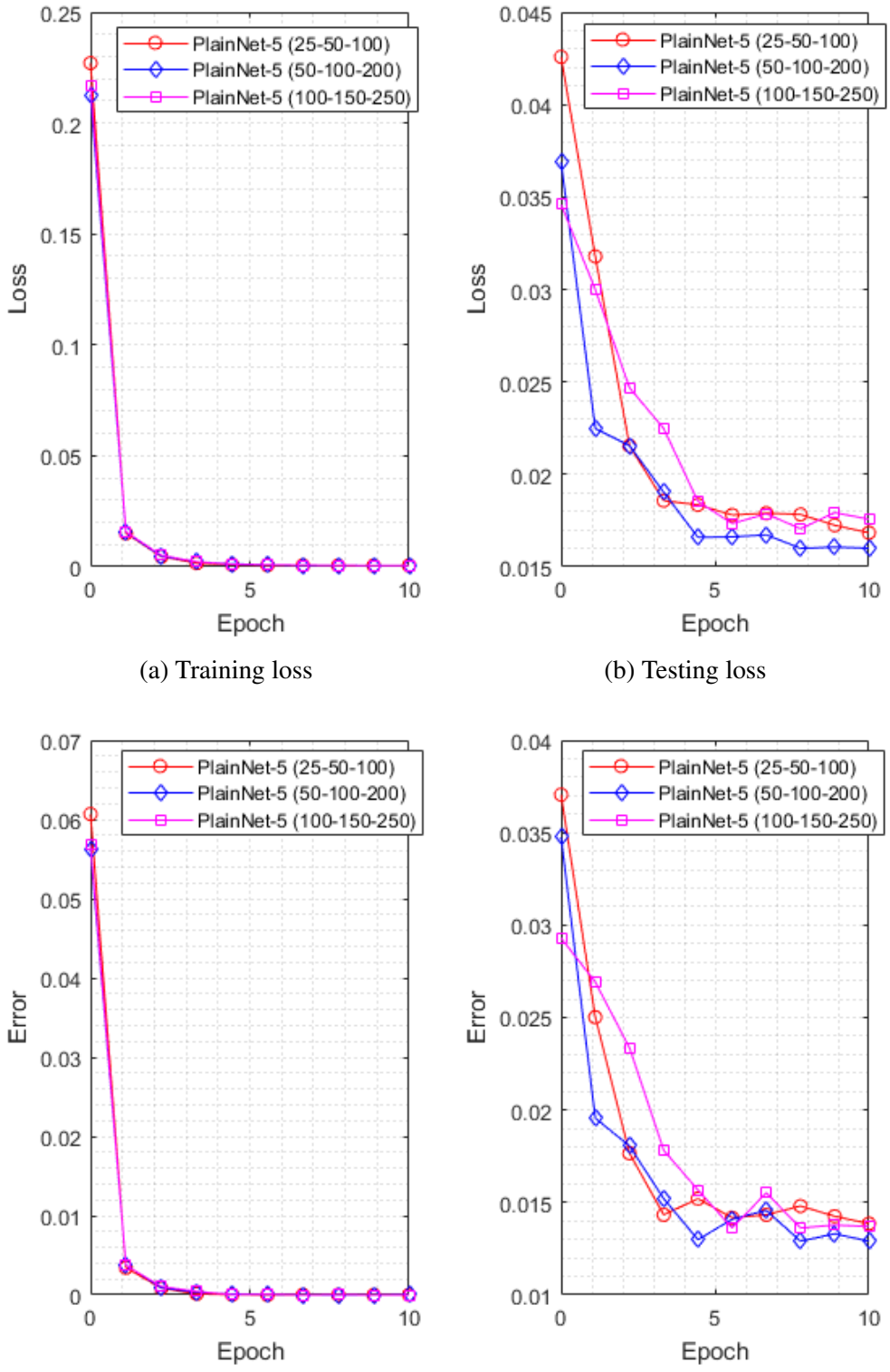
The first networks are PlainNets-5 that are similar to the famous LeNet-5 [180]. Three PlainNets-5 with different width are employed, as shown in Table 5.3. According to the training loss and error in Fig. 5.3a and 5.3c, all of the three networks have similar convergence rates in the training stage, the optimiser is able to find a good solution within 4 epoches every time. The final testing error for the PlainNets-5 (25-50-100, 50-100-200, 100-150-250) are also similar, which means that the networks have similar generalisation abilities. Considering

Layer name	Output size	PlainNet-5		
Conv_1	42×42	$7 \times 7, 25$	$7 \times 7, 50$	$7 \times 7, 100$
Pool_1	21×21	2×2 max poling with stride 2		
Conv_2	18×18	$4 \times 4, 50$	$4 \times 4, 100$	$4 \times 4, 150$
Pool_2	9×9	2×2 max poling with stride 2		
Conv_3	6×6	$4 \times 4, 100$	$4 \times 4, 200$	$4 \times 4, 250$
Pool_3	3×3	2×2 max poling with stride 2		
Fully-connected_1	1×1	$3 \times 3, 256$		
Fully-connected_2	1×1	$1 \times 1, 43$		
Softmax	1×1			

Table 5.3 The architectures of PlainNets-5 for experiments. Each *Conv_x* corresponds to the stacked layers (*Conv-BN-ReLU*). The *Fully-connected_1* corresponds to the stacked layers *FC-BN-ReLU*.

Layer name	Output size	PlainNet-7			
Conv_1	48×48	$3 \times 3, 8$	$3 \times 3, 16$	$3 \times 3, 32$	$3 \times 3, 64$
Pool_1	24×24	2×2 max poling with stride 2			
Conv_2	24×24	$3 \times 3, 16$	$3 \times 3, 32$	$3 \times 3, 64$	$3 \times 3, 128$
Pool_2	12×12	2×2 max poling with stride 2			
Conv_3	12×12	$3 \times 3, 32$	$3 \times 3, 64$	$3 \times 3, 128$	$3 \times 3, 256$
Pool_3	6×6	2×2 max poling with stride 2			
Conv_4	6×6	$3 \times 3, 64$	$3 \times 3, 128$	$3 \times 3, 256$	$3 \times 3, 512$
Pool_4	3×3	2×2 max poling with stride 2			
Fully-connected_1	1×1	$3 \times 3, 1024$			
Fully-connected_2	1×1	$1 \times 1, 256$			
Fully-connected_3	1×1	$1 \times 1, 43$			
Softmax	1×1				

Table 5.4 The architectures of PlainNets-7 for experiments. Each *Conv_x* corresponds to the stacked layers (*Conv-BN-ReLU*). The *Fully-connected_1 and Fully-connected_2* correspond to the stacked layers (*FC-BN-ReLU*).



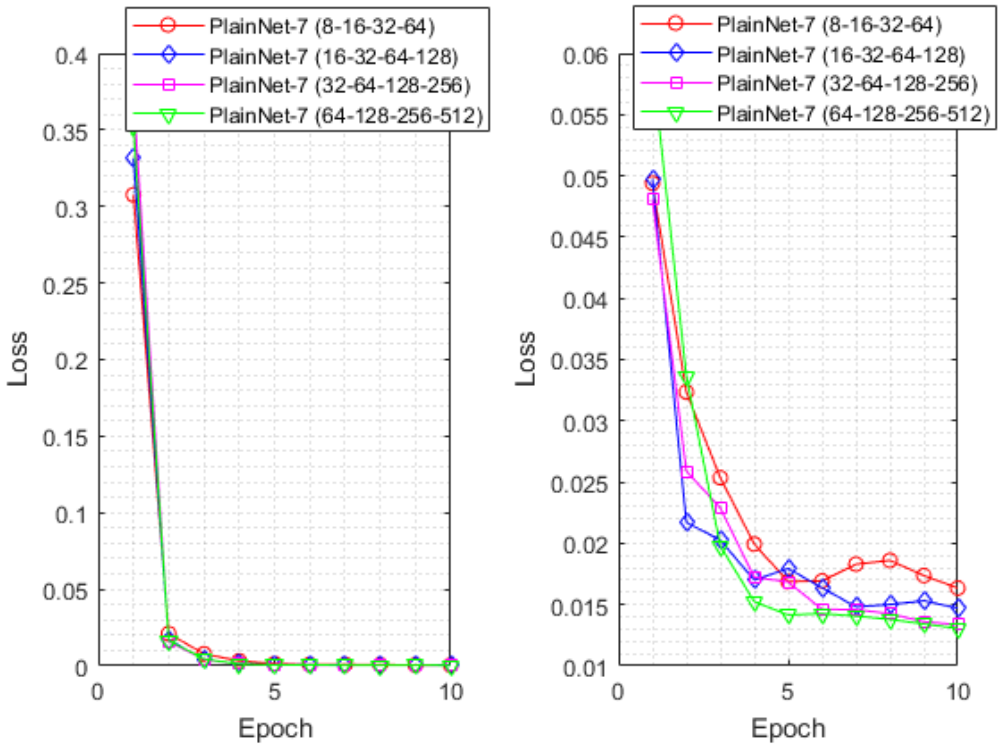
(a) Training loss

(b) Testing loss

(c) Training error

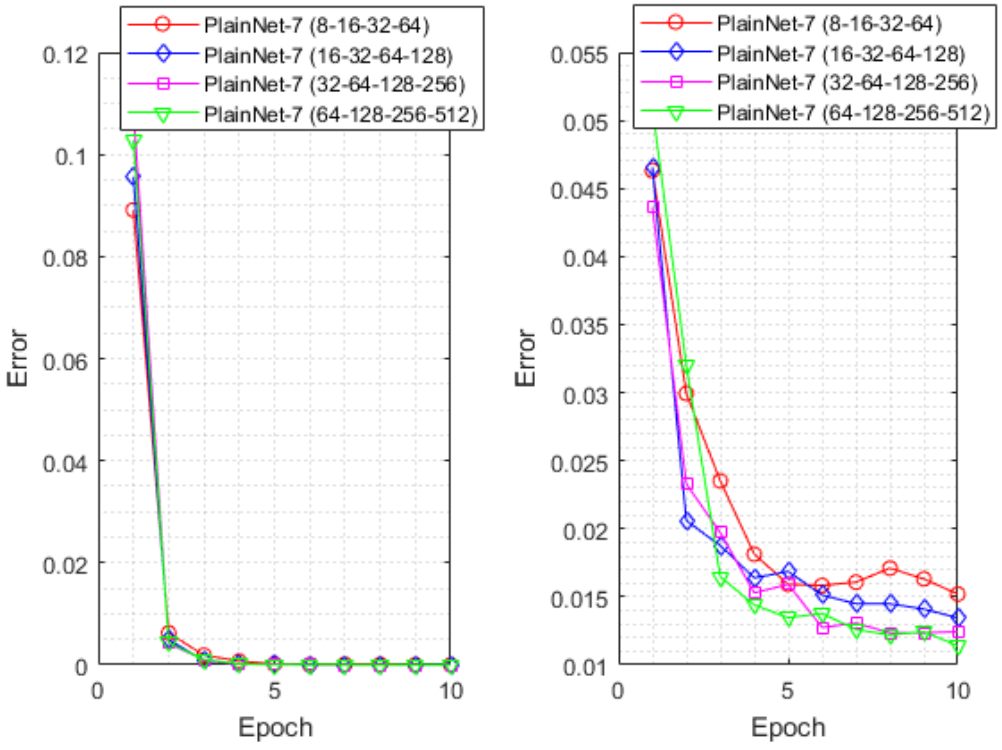
(d) Testing error

Fig. 5.3 Training details of PlainNets-5 with different width.



(a) Training loss

(b) Testing loss



(c) Training error

(d) Testing error

Fig. 5.4 Training details of PlainNets-7 with different width.

both computational cost and generalisation ability, the PlainNet-5 (50-100-200) is slightly better than the other two networks.

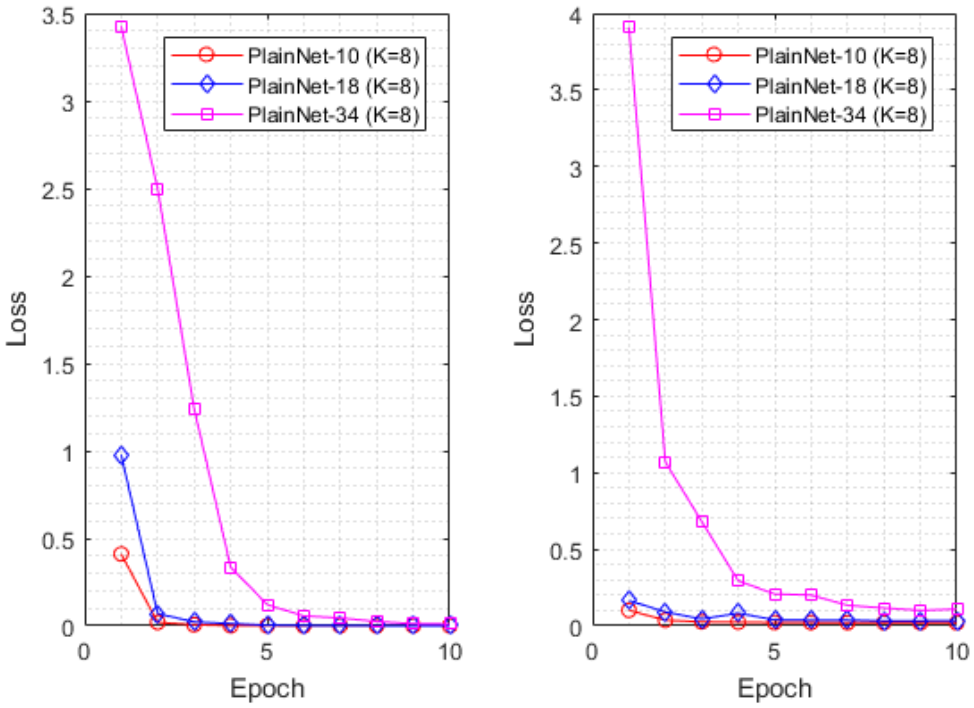
The second networks are PlainNets-7 that are similar to the famous VGGNets [30]. Four PlainNets-7 with different width are employed, as shown in Table 5.4. The training loss and error are illustrated in Fig. 5.4a and 5.4c, and testing loss and error are illustrated in Fig. 5.4b and 5.4d. As these figures indicate, all of the four networks have similar training speed and have finished the training within 5 epoches. Moreover, the testing error of the PlainNets-7 (8-16-32-64, 16-32-64-128, 32-64-128-256, 64-128-256-512) decreases with the increasing of the width of networks. Therefore, PlainNet-7 (64-128-256-512) achieves the highest accuracy in the four networks. In fact, it also demonstrates the best performance among all the network architectures in this section.

Layer name	Output size	10-layer	18-layer	34-layer
Conv_1	48×48	$3 \times 3, 8K$		
Conv_2_x	24×24	$\begin{bmatrix} 3 \times 3, 8K \\ 3 \times 3, 8K \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3, 8K \\ 3 \times 3, 8K \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 8K \\ 3 \times 3, 8K \end{bmatrix} \times 3$
Conv_3_x	12×12	$\begin{bmatrix} 3 \times 3, 16K \\ 3 \times 3, 16K \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3, 16K \\ 3 \times 3, 16K \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 16K \\ 3 \times 3, 16K \end{bmatrix} \times 4$
Conv_4_x	6×6	$\begin{bmatrix} 3 \times 3, 32K \\ 3 \times 3, 32K \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3, 32K \\ 3 \times 3, 32K \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 32K \\ 3 \times 3, 32K \end{bmatrix} \times 6$
Conv_5_x	3×3	$\begin{bmatrix} 3 \times 3, 64K \\ 3 \times 3, 64K \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3, 64K \\ 3 \times 3, 64K \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64K \\ 3 \times 3, 64K \end{bmatrix} \times 3$
Classification	1×1	BN, ReLU		
		3×3 global average pooling with stride 3		
		43D fully-connected, softmax		

Table 5.5 The architectures of PlainNets-10/18/34 and ResNets-10/18/34 for experiments. The stacked layers (*BN-ReLU-Conv-BN-ReLU-Conv*) are shown in brackets, followed by the stacked numbers. The convolutional strides of Conv_2_1, Conv_3_1, Conv_4_1 and Conv_5_1 are 2 for down sampling. The width of the networks is controlled by the factor K with possible values 1, 2, 4 and 8.

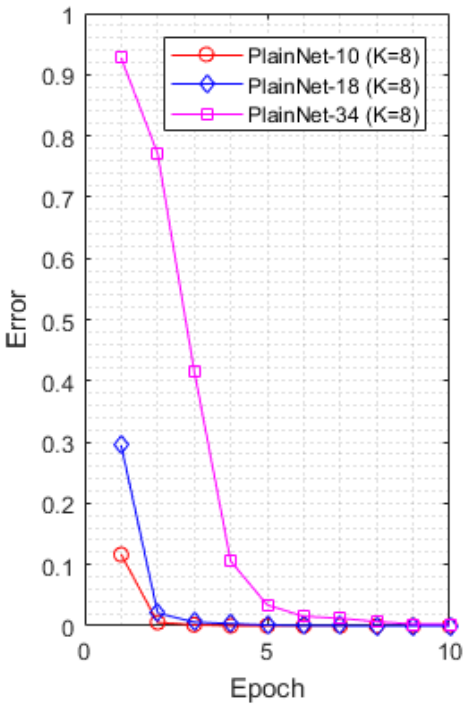
The third networks are composed of two kinds of network architectures, including common PlainNets without shortcut connections, and PlainNets with shortcut connections that also known as ResNets [32]. Three different network depth are employed, which are 10, 18 and 34 respectively. In order to figure out the detailed relationship between network depth, width and performance, a factor K is introduced for controlling the width of networks. The networks with different depth and width are shown in Table 5.5.

The training and testing details of PlainNets-10/18/34 with $K = 8$ have been illustrated in Fig. 5.5. The curves in Fig. 5.5d indicate that deeper plain networks have higher testing

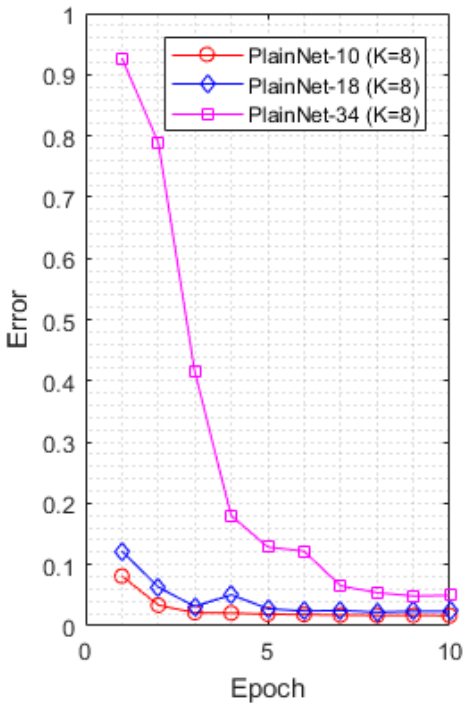


(a) Training loss

(b) Testing loss

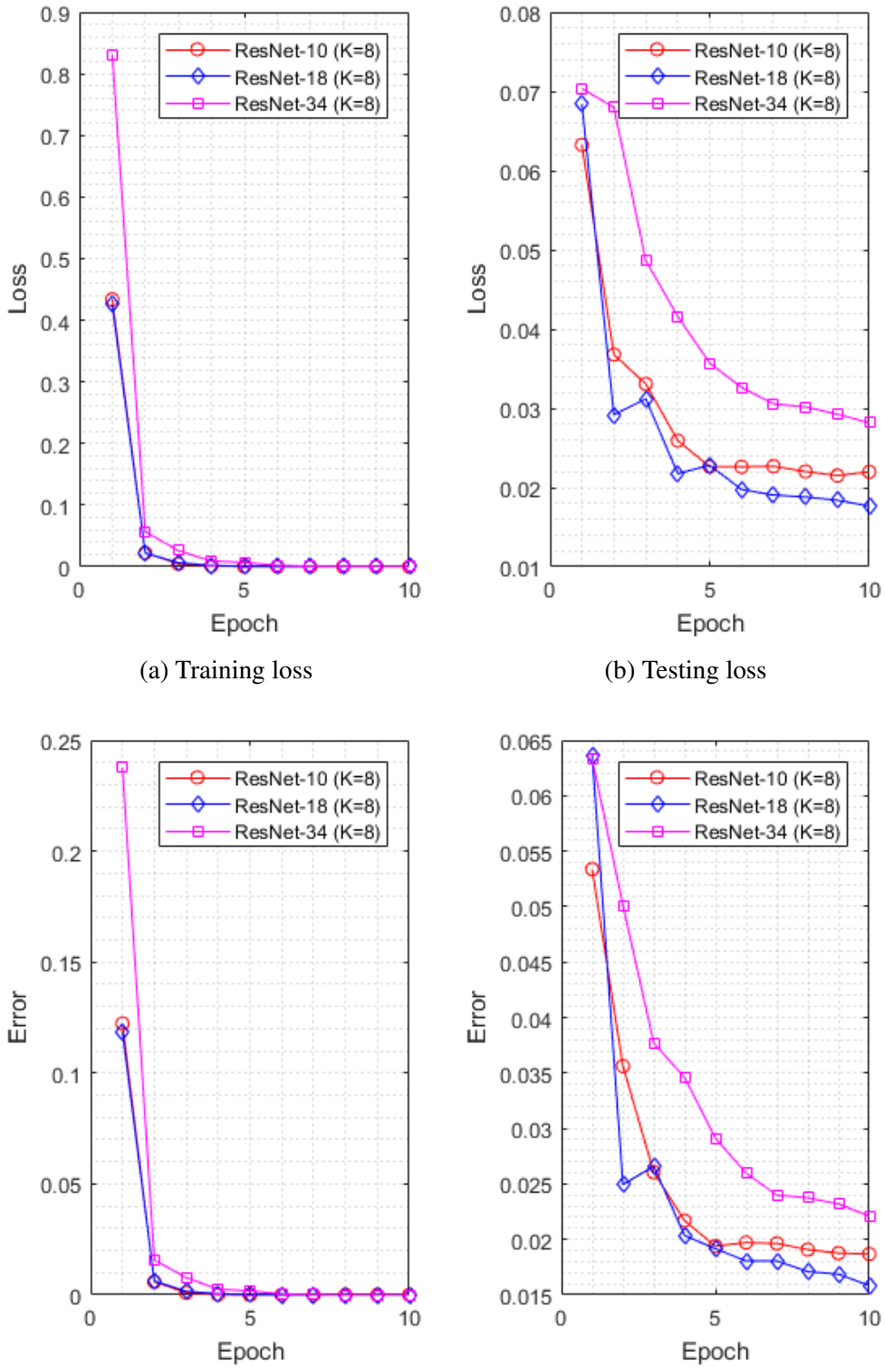


(c) Training error



(d) Testing error

Fig. 5.5 Training details of PlainNets-10/18/34.



(a) Training loss

(b) Testing loss

(c) Training error

(d) Testing error

Fig. 5.6 Training details of ResNets-10/18/34.

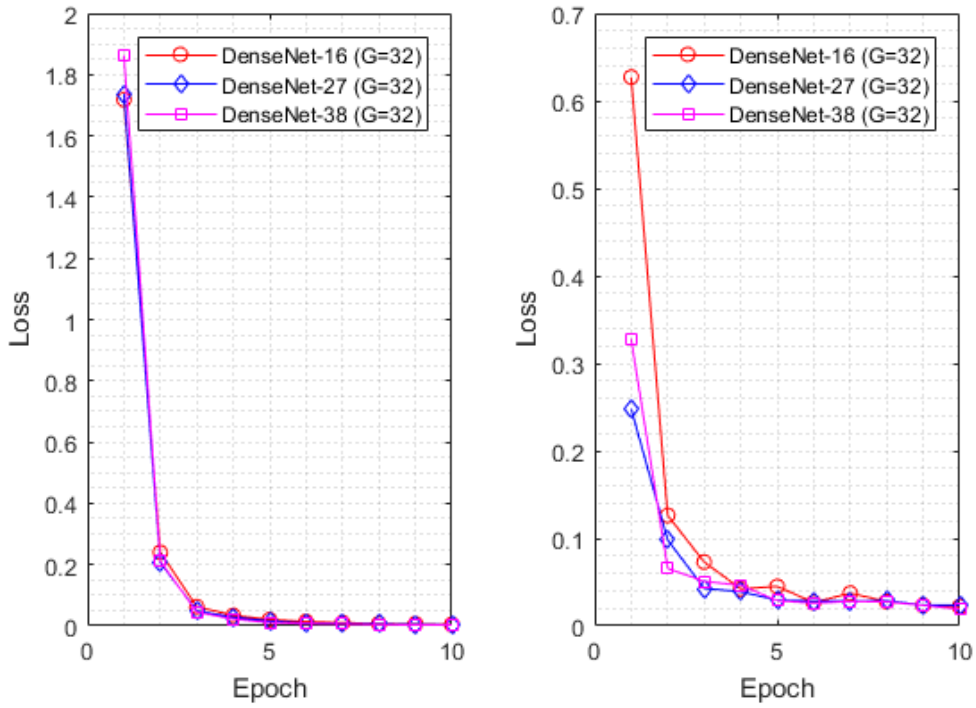
error than shallower plain networks, especially the PlainNet-34 has much higher testing error than PlainNet-10 and PlainNet-18. To reveal the reasons, the training error in Fig. 5.5c can be compared, and it is obvious that PlainNet-34 also has higher training error than PlainNet-10 and PlainNet-18. Therefore, the degradation problem already appears. Because all the networks are trained with BN layers, which ensure forward propagated features and backward propagated gradients have healthy values. It can be argued that this degraded performance is nearly impossible to be caused by gradient vanishing. In fact, at the end of the training, PlainNet-34 is still able to converge and achieve not much degraded accuracy, which means that the optimiser still works but the convergence rate is largely decreased by the deep and plain network architecture.

The training and testing details of ResNets-10/18/34 with $K = 8$ have been illustrated in Fig. 5.6. By introducing residual learning, the training speed of ResNet-34 is significantly increased, the ResNets-10/18/34 become have similar training speed and error. Moreover, although residual learning does not bring deeper networks better testing error, all the networks achieve comparable testing error, which means the degradation problem is well addressed.

Layer name	Output size	16-layer	27-layer	38-layer
Convolution	24×24	$3 \times 3, 2G$ with stride 2		
Dense_Block_1	24×24	$[3 \times 3\ conv] \times 2$	$[3 \times 3\ conv] \times 4$	$[3 \times 3\ conv] \times 6$
Transition_Block_1	24×24	$1 \times 1\ conv$		
	12×12	2×2 average poling with stride 2		
Dense_Block_2	12×12	$[3 \times 3\ conv] \times 2$	$[3 \times 3\ conv] \times 4$	$[3 \times 3\ conv] \times 6$
Transition_Block_2	12×12	$1 \times 1\ conv$		
	6×6	2×2 average poling with stride 2		
Dense_Block_3	6×6	$[3 \times 3\ conv] \times 4$	$[3 \times 3\ conv] \times 8$	$[3 \times 3\ conv] \times 12$
Transition_Block_3	6×6	$1 \times 1\ conv$		
	3×3	2×2 average poling with stride 2		
Dense_Block_4	3×3	$[3 \times 3\ conv] \times 3$	$[3 \times 3\ conv] \times 6$	$[3 \times 3\ conv] \times 6$
Classification	1×1	BN, ReLU		
		3×3 global average pooling with stride 3		
		43D fully-connected, softmax		

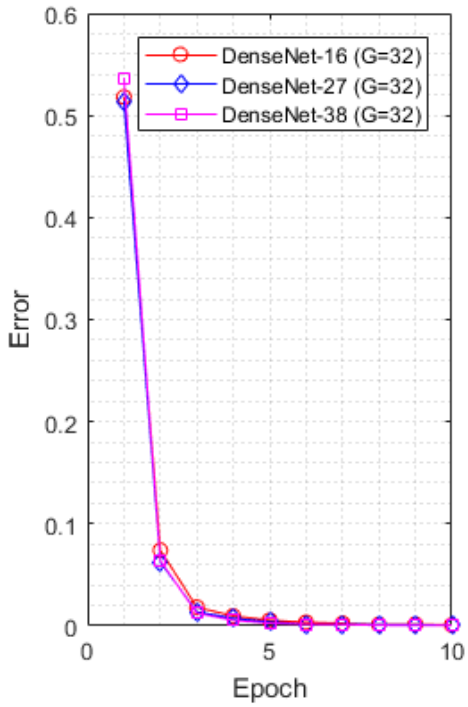
Table 5.6 The architectures of DenseNets-16/27/38 for experiments. The *conv* stands for composite layers (*BN-ReLU-Conv*), and the numbers followed by the brackets are the stacked numbers. The width of the networks is controlled by the stacked numbers of dense blocks and the growth rate G with possible values 8, 16 and 32.

The final networks are DenseNets-16/27/38 that built based on the original DenseNets [164]. The networks with different depth and width are shown in Table 5.6. G is the growth rate that is able to control the width of networks. The training details of DenseNets-16/27/38

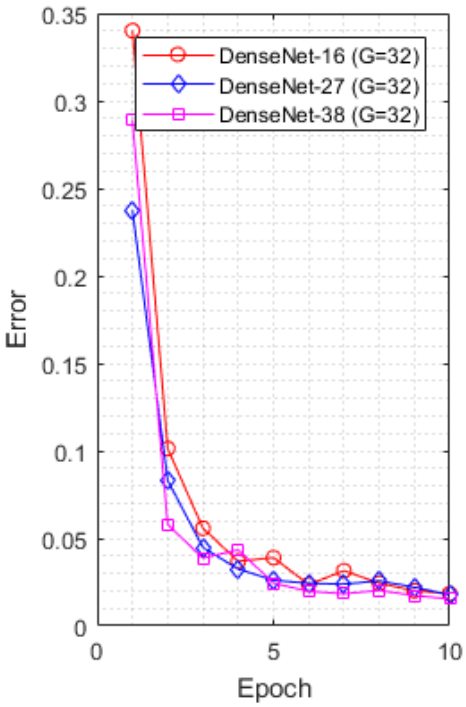


(a) Training loss

(b) Testing loss



(c) Training error



(d) Testing error

Fig. 5.7 Training details of DenseNets-16/27/38 with different width.

Network name	Average top-1 testing error	Lowest top-1 testing error
PlainNet-5 (25-50-100)	1.54%	1.39%
PlainNet-5 (50-100-200)	**1.51%**	1.29%
PlainNet-5 (100-150-250)	1.53%	1.36%
PlainNet-7 (8-16-32-64)	2.38%	1.52%
PlainNet-7 (16-32-64-128)	1.63%	1.35%
PlainNet-7 (32-64-128-256)	1.60%	1.24%
PlainNet-7 (64-128-256-512)	**1.51%**	**1.14%**
PlainNet-10 ($K = 1$)	3.01%	2.32%
PlainNet-10 ($K = 2$)	2.66%	2.26%
PlainNet-10 ($K = 4$)	2.10%	1.82%
PlainNet-10 ($K = 8$)	1.98%	1.68%
PlainNet-18 ($K = 1$)	3.69%	3.06%
PlainNet-18 ($K = 2$)	2.86%	2.44%
PlainNet-18 ($K = 4$)	2.66%	2.26%
PlainNet-18 ($K = 8$)	2.78%	2.23%
PlainNet-34 ($K = 1$)	7.24%	5.08%
PlainNet-34 ($K = 2$)	6.32%	5.00%
PlainNet-34 ($K = 4$)	6.07%	4.77%
PlainNet-34 ($K = 8$)	8.00%	4.88%
ResNet-10 ($K = 1$)	2.95%	2.53%
ResNet-10 ($K = 2$)	2.52%	2.27%
ResNet-10 ($K = 4$)	2.06%	1.79%
ResNet-10 ($K = 8$)	2.00%	1.87%
ResNet-18 ($K = 1$)	2.75%	2.29%
ResNet-18 ($K = 2$)	2.33%	1.92%
ResNet-18 ($K = 4$)	1.92%	1.43%
ResNet-18 ($K = 8$)	1.94%	1.58%
ResNet-34 ($K = 1$)	2.65%	2.21%
ResNet-34 ($K = 2$)	2.36%	1.83%
ResNet-34 ($K = 4$)	2.18%	1.78%
ResNet-34 ($K = 8$)	2.19%	2.10%
DenseNet-16 ($G = 8$)	3.14%	2.24%
DenseNet-16 ($G = 16$)	2.01%	1.54%
DenseNet-16 ($G = 32$)	2.15%	1.84%
DenseNet-27 ($G = 8$)	2.20%	1.84%
DenseNet-27 ($G = 16$)	1.92%	1.52%
DenseNet-27 ($G = 32$)	2.10%	1.77%
DenseNet-38 ($G = 8$)	2.39%	1.71%
DenseNet-38 ($G = 16$)	2.05%	1.56%
DenseNet-38 ($G = 32$)	2.37%	1.54%

Table 5.7 The performance of all the network architectures. The networks achieve the best performance are highlighted with bold font.

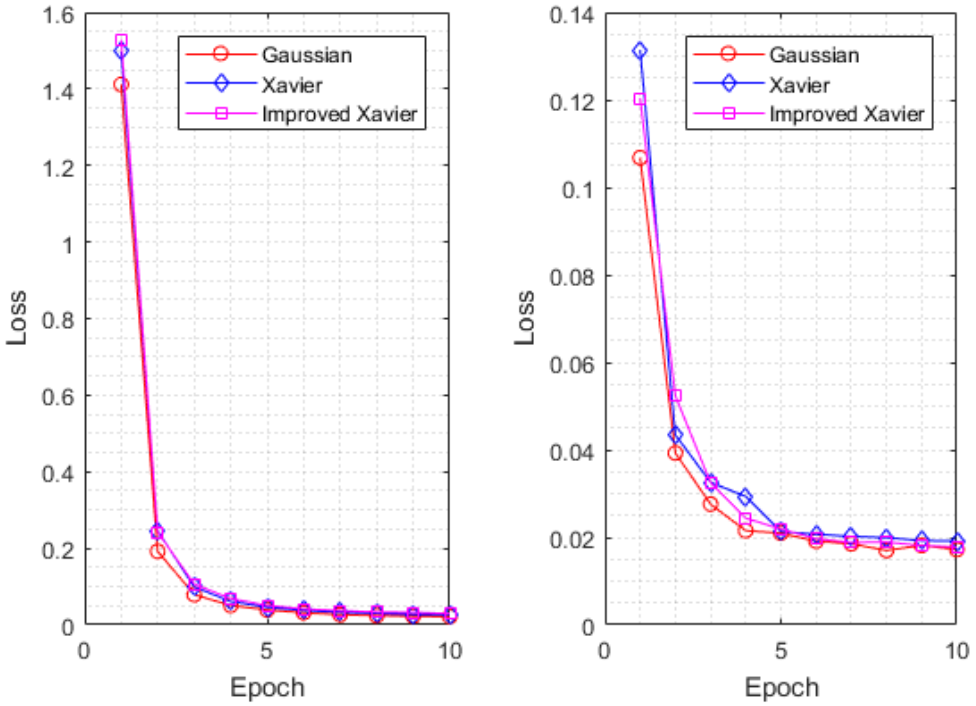
with $G = 32$ have been illustrated in Fig. 5.7. Based on the observation of the training and testing error, it can be concluded that degradation problem has not been found in DenseNets. The convergence rates of DenseNets are also not visibly affected by the depth of networks. These are benefited from the special design of DenseNets, which allows each layer receives inputs from all preceding layers and passes to all subsequent layers. Therefore, feature reuse exists everywhere inside DenseNets. The reuse of feature makes DenseNets very compact in network parameters. Furthermore, the compact network architecture significantly improves parameter efficiency. High parameter efficiency ensures less parameter number, resulting in less overfitting and better generalisation ability.

Finally, the performance of all the network architectures are summarised and displayed in Table 5.7. Three major observation can be found from Table 5.7, first, the testing error of PlainNets-34 is much higher than other networks. It is not caused by overfitting, because the network parameters of PlainNets-34 are less than ResNet-34. Therefore, the PlainNets-34 faces serious degradation problem. Second, with the help of shortcut connections, both ResNets and DenseNets ease the degradation problem well. These networks can achieve similar performance no matter how deep the networks are. Finally, the PlainNets-5 and PlainNets-7 achieve excellent performance even though the depth of these networks are relatively shallow. In particular, PlainNet-7 (64-128-256-512) achieves the lowest testing error which is 1.14%. This is because that the GTSRB is not such hard, so that simple networks can achieve good performance. On the other hand, the degradation problem is not totally solved. Further improvement should be made to enhance network architecture.

Parameter Initialisation Methods

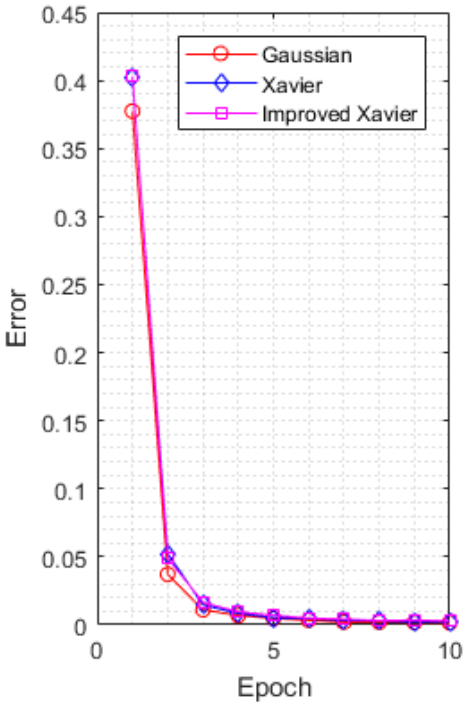
Three parameter initialisation methods are employed, namely *Gaussian distribution*, *Xavier*[156] and *Improved Xavier*[157]. For the gaussian distribution initialisation, the parameters are sampled from $[-0.02, 0.02]$. The employed network is PlainNet-7 (64-128-256-512) that has been shown in Table 5.4.

The training details are illustrated in Fig. 5.8, and the testing error is given in Table 5.8. The results indicate that the three parameter initialisation methods have nearly the same performance in training stage. Subsequently, in testing stage, *Improved Xavier*[157] gives the best average testing error 1.58% while *Gaussian distribution* achieves the lowest testing error 1.24%, and *Xavier* shows a little lower performance. Combining both the average and lowest testing error, *Improved Xavier* slightly outperforms the other two methods.

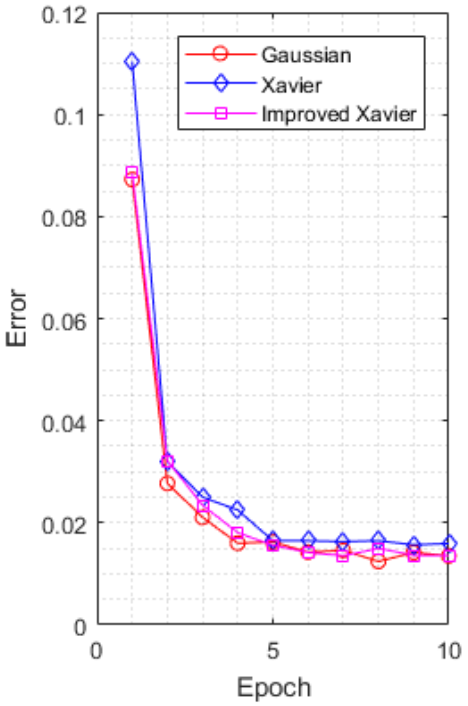


(a) Training loss

(b) Testing loss



(c) Training error



(d) Testing error

Fig. 5.8 Training details of PlainNets-7 (64-128-256-512) with the three parameter initialisation methods.

Method	Average top-1 testing error	Lowest top-1 testing error
Gaussian	1.60%	**1.24%**
Xavier	1.72%	1.57%
Improved Xavier	**1.58%**	1.35%

Table 5.8 The performance of the networks with the three parameter initialisation methods.

Nonlinear Activation Layers

The sigmoid function and ReLU [173] are employed as nonlinear activation functions respectively. The employed network is PlainNet-7 (64-128-256-512) that has been shown in Table 5.4.

The training details are illustrated in Fig. 5.9, and the testing error is given in Table 5.9. As Fig. 5.9a to 5.9d illustrate, the convergence rate of the network with sigmoid function is much slower than the network with ReLU, and the optimiser fails to find a good solution within 10 epoches. Therefore, the sigmoid function causes serious gradient vanishing problem. It is important to notice that BN layers have been employed in all of these networks, so that the gradient vanishing problem has already been partially overcame. Without the help of BN layers, the gradient vanishing problem caused by sigmoid function should be more serious. For this reason, sigmoid function should be carefully applied in deep CNNs.

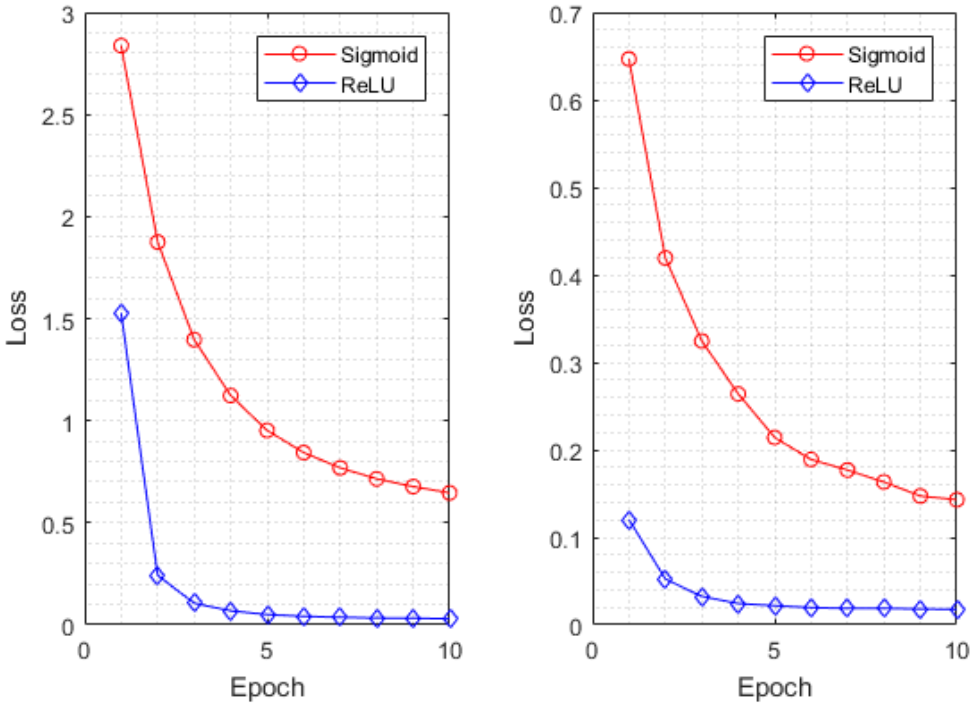
Method	Average top-1 testing error	Lowest top-1 testing error
Sigmoid	9.42%	8.86%
ReLU	**1.58%**	**1.35%**

Table 5.9 The performance of the networks with the two nonlinear activation functions.

Batch Normalisation

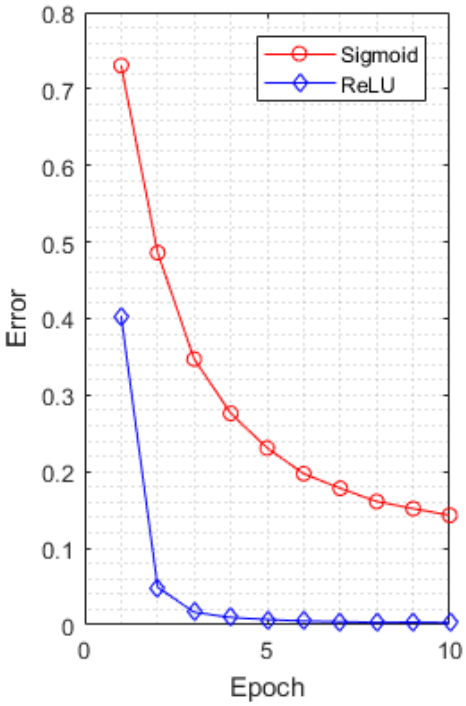
The performance of networks with or without BN [158] is presented here. The applied network is PlainNet-7 (64-128-256-512) that has been shown in Table 5.4.

The training details are illustrated in Fig. 5.10, and the testing error is given in Table 5.10. The results in Fig. 5.10a and 5.10c show that BN indeed boosts training speed. The network with BN is able to finish the training by only 5 epoches, while the network without BN takes 10 epoches to find the solution. Moreover, the results in Fig. 5.10b and 5.10d indicate that BN also provides lower testing error. This is benefited from its strong regularisation ability, because the features of a training example are affected by the randomly created mini-batch.

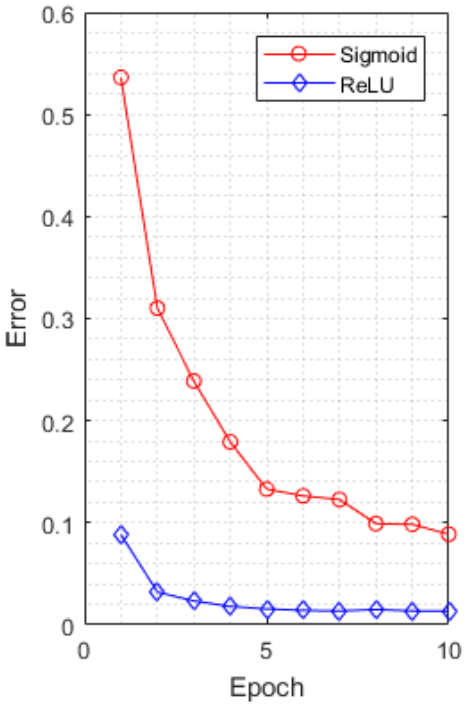


(a) Training loss

(b) Testing loss

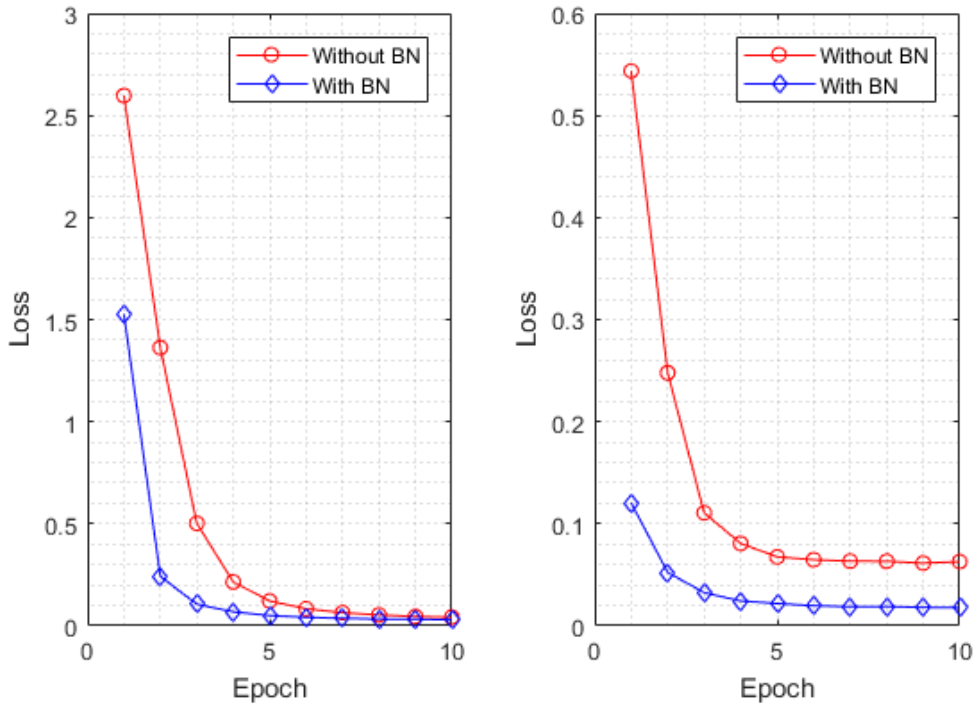


(c) Training error

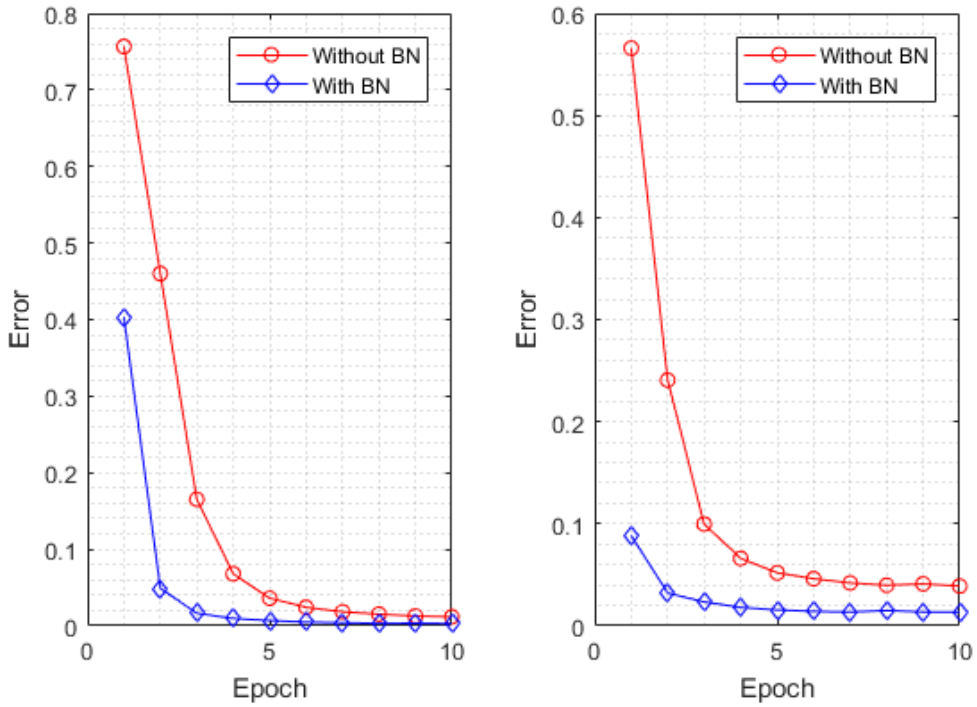


(d) Testing error

Fig. 5.9 Training details of PlainNets-7 (64-128-256-512) with the two nonlinear activation functions.



(a) Training loss (b) Testing loss



(c) Training error (d) Testing error

Fig. 5.10 Training details of PlainNets-7 (64-128-256-512) with or without BN.

Method	Average top-1 testing error	Lowest top-1 testing error
Without BN	4.28%	3.90%
With BN	**1.58%**	**1.35%**

Table 5.10 The performance of the networks with or without BN.

Parameter Norm Penalties

Three parameter norm penalty strategies are applied, including no penalty, L^1 and L^2. The L^1 value is 0.0001 and L^2 value is 0.0005. The employed network is PlainNet-7 (64-128-256-512) that has been shown in Table 5.4.

The training details are illustrated in Fig. 5.11, and the testing error is given in Table 5.11. It is interesting that the networks with L^2 penalty achieve the best average testing error 1.58%, and the network without any penalty provides the lowest testing error 1.31%. The best average testing error of L^2 penalty means it truly provides regularisation, and the lowest testing error achieved by no penalty indicates that the existing regularisation of networks is already quite strong. On the other hand, the testing loss and error of network with L^1 penalty are extremely unstable. Due to the principle of L^1 penalty, it trends to make the network parameters sparse, resulting in sparse features. The sparse features maybe incompatible with the networks.

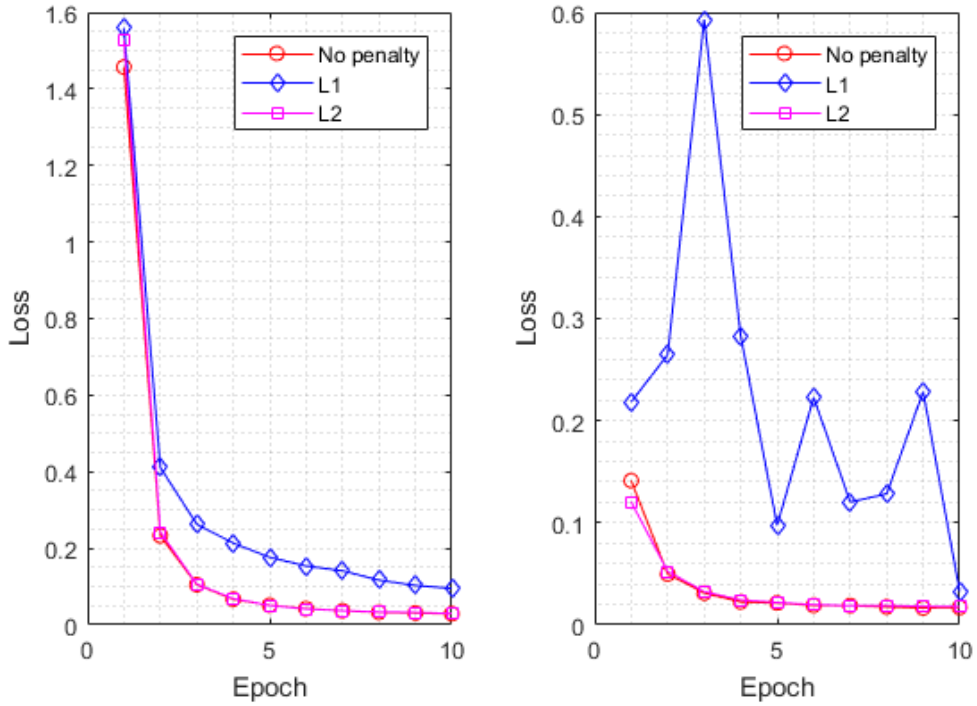
Method	Average top-1 testing error	Lowest top-1 testing error
No penalty	1.64%	**1.31%**
L^1	4.90%	2.61%
L^2	**1.58%**	1.35%

Table 5.11 The performance of the networks trained with the three parameter norm penalties.

Dropout

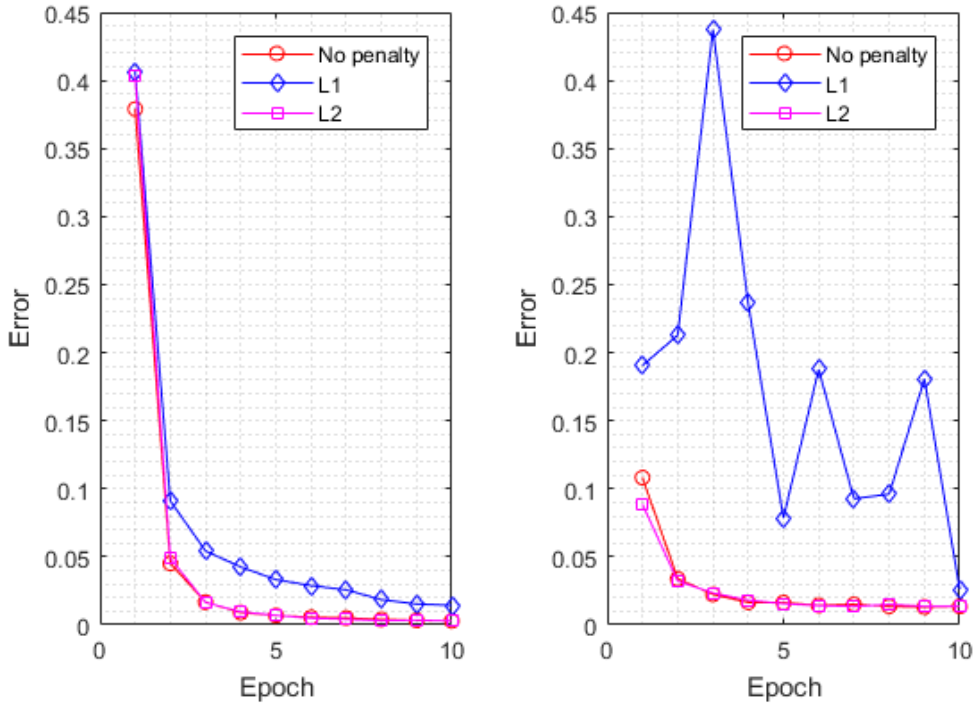
The performance of networks with or without dropout [174] is presented here. The applied network is PlainNet-7 (64-128-256-512) that has been shown in Table 5.4.

The training details are illustrated in Fig. 5.12, and the testing error is given in Table 5.12. Obviously, networks with dropout provide lower testing error due to its strong regularisation. However, the results in Fig. 5.12a and 5.12c show that network without dropout is able to converge within 3 epoches, while network with dropout needs 5 epoches. Therefore, dropout can provide strong regularisation, but it also decreases training speed.



(a) Training loss

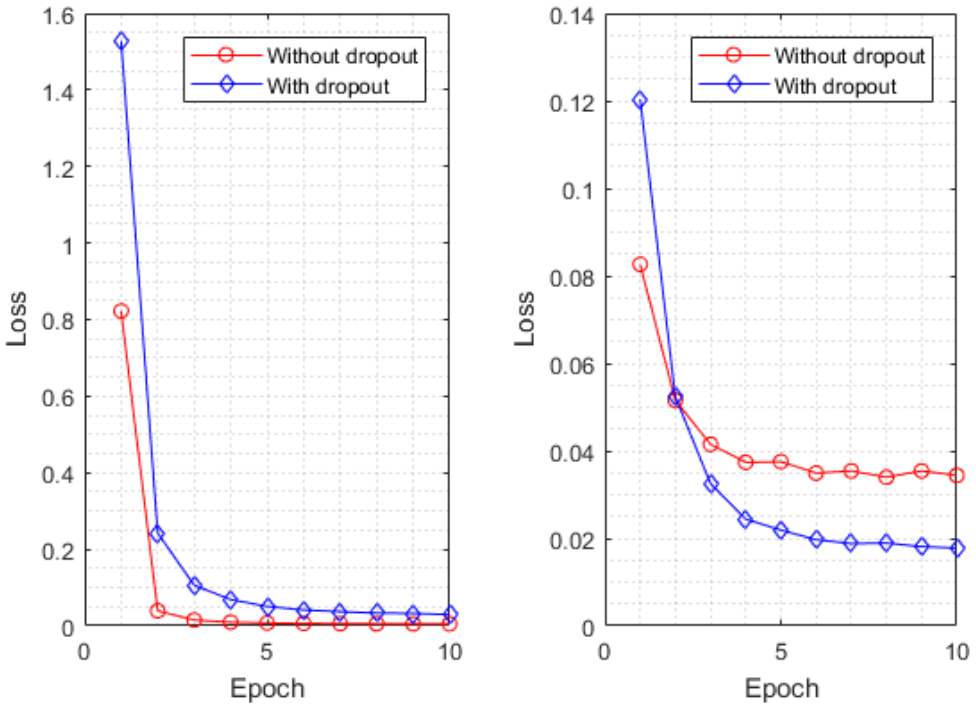
(b) testing loss



(c) Training error

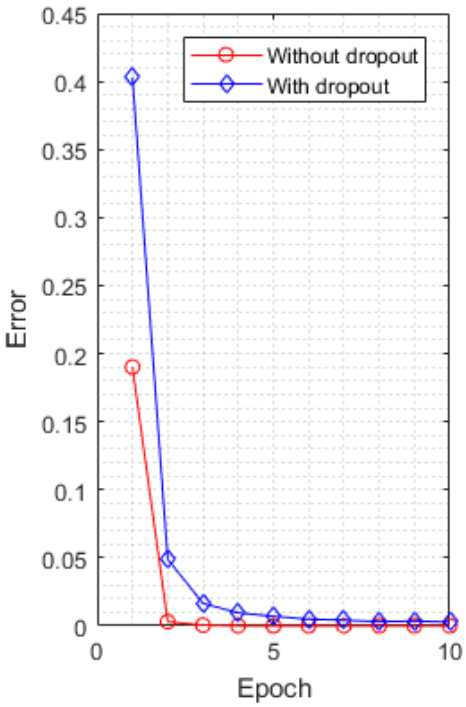
(d) Testing error

Fig. 5.11 Training details of PlainNets-7 (64-128-256-512) with the three parameter norm penalties.

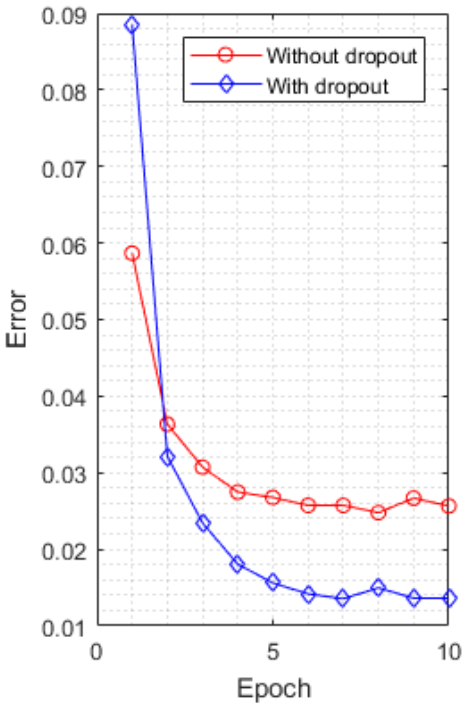


(a) Training loss

(b) Testing loss



(c) Training error



(d) Testing error

Fig. 5.12 Training details of PlainNets-7 (64-128-256-512) with or without dropout.

Method	Average top-1 testing error	Lowest top-1 testing error
Without dropout	3.04%	2.48%
With dropout	**1.58%**	**1.35%**

Table 5.12 The performance of the networks with or without dropout.

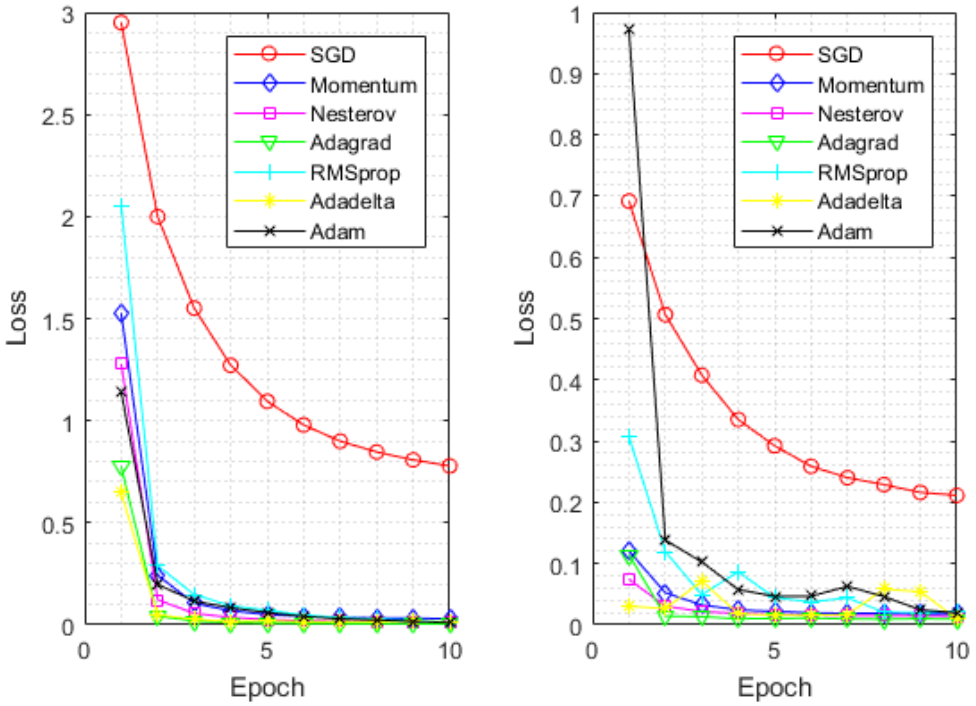
Optimisation Algorithms

In order to demonstrate a detailed comparison, seven optimisation algorithms are employed, namely SGD, Momentum ($\alpha = 0.9$) [182], Nesterov ($\alpha = 0.9$) [183], Adagrad ($\varepsilon = 10^{-10}$) [184], RMSprop ($\alpha = 0.99, \varepsilon = 10^{-8}$) [185], Adadelta ($\alpha = 0.9, \varepsilon = 10^{-6}$) [186] and Adam ($\alpha = 0.9, \beta = 0.999, \varepsilon = 10^{-8}$) [187]. The applied network is PlainNet-7 (64-128-256-512) that has been shown in Table 5.4.

The training details are illustrated in Fig. 5.13, and the testing error is given in Table 5.13. First, it is clear that SGD provides very slow training speed, and it fails to find a good solution within 10 epoches. Second, Momentum significantly increases the training speed of SGD, and then Nesterov further boosts the training speed of Momentum. The testing loss and error of both the two algorithms are very stable during the training. The final testing error is also comparable with other optimisation algorithms. Third, Adagrad demonstrates the fastest training speed and achieves the lowest testing error 0.7%. The training of Adagrad is rapidly stopped after about 2 epoches, which is accords with its characteristic. Fourthly, RMSprop and Adadelta display fast training speed in the early and middle stage, however, the testing error is not stable at the end of the training. Finally, Adam shows improved performance than SGD, however, it does not outperform the other optimisation algorithms in the experiments.

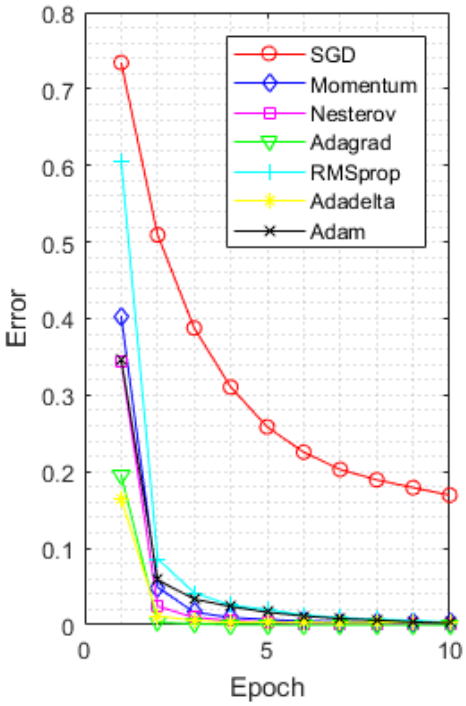
Method	Average top-1 testing error	Lowest top-1 testing error
SGD	16.68%	15.22%
Momentum	1.58%	1.35%
Nesterov	1.32%	1.21%
Adagrad	**0.87%**	**0.70%**
RMSprop	1.96%	1.61%
Adadelta	1.14%	0.78%
Adam	1.89%	1.53%

Table 5.13 The performance of the networks trained with the seven optimisation algorithms.

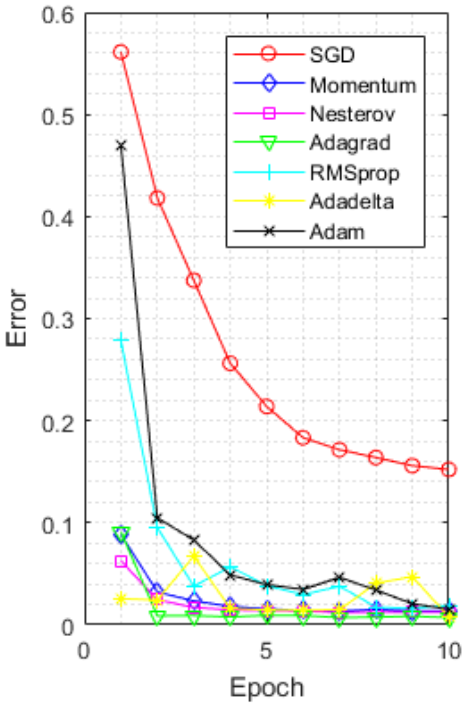


(a) Training loss

(b) Testing loss



(c) Training error



(d) Testing error

Fig. 5.13 Training details of PlainNets-7 (64-128-256-512) with the seven optimisation algorithms.

5.7 Summary

In this chapter, traffic signs are recognised with various CNN architectures that have been widely used, including CNNs [19], VGGNets [30], ResNets [32] and DenseNets [164]. Extensive experiments have been presented and discussed based on different network architectures, parameters initialisation methods, network layers, parameter norm penalty strategies and optimisation algorithms. For network architectures, the very deep plain networks show larger training and testing error than shallow plain networks, which indicates the degradation problem maybe appear when the networks have very deep depths. In order to ease this problem, shortcut connections have been applied in ResNets and DenseNets, resulting in better performance. For parameter initialisation methods, normalised parameters demonstrate good training speed and generalisation ability. For network layers, ReLU, dropout and BN layers show better performance in training speed and generalisation ability. The classical activation function sigmoid should be carefully applied, it shows significantly gradient vanishing problem in very deep networks. For parameter norm penalty strategies, the L^2 penalty shows better regularisation ability. For optimisation algorithms, Adagrad demonstrates the fastest training speed and achieves the lowest testing error in the experiments. On the other hand, SGD provides very slow training speed, and the networks fail to converge in the experiments, therefore, SGD should be carefully applied in the training of deep CNNs.

Chapter 6

Traffic Sign Recognition with Novel Convolutional Neural Networks

Deep Convolutional Neural Network (CNN)s have achieved excellent performance in Traffic Sign Recognition (TSR). However, the construction and training of deep CNNs still remain problems. In this chapter, traffic signs are recognised with new CNNs, which are trained by new strategies or constructed by new network architectures. First, different from the most existing approaches that apply single level classifiers for TSR, a hierarchical classification system is applied for improving the performance of TSR. Second, unsupervised feature learning is a challenge in computer vision. By introducing Generative Adversarial Networks (GANs), a TSR system is proposed based on features that are learnt without labels. Third, in comparison with previous CNNs that only apply max pooling for reducing feature dimension, a novel layer is proposed, which applies Max Pooling Positions (MPPs) of max pooling as information for sparse feature learning. Finally, in order to overcome gradient vanishing and degradation problems, a novel CNN architecture design that learns with multiple loss functions is developed. The performance of the proposed methods are verified on the German Traffic Sign Recognition Benchmark (GTSRB), and excellent results have been achieved.

6.1 Introduction

Traffic Sign Recognition (TSR) is a challenging machine learning and computer vision problem [22]. First, it is a multi-class classification problem that involves with not only imbalanced data, but also very small inter-class difference. Second, in real-world environments, traffic signs have very large variability in their appearance due to illumination changing, shooting angle variations, partial occlusions, weather conditions and colour fading. Finally,

accuracy and real-time capability also should be ensured as TSR are usually applied in real world applications, such as ADAS and autonomous cars.

In order to be easily noticed and understood by pedestrian and driver, traffic signs are designed to have visible appearance that is combined with colour, shape, icon and text. This kind of design principle allows to create a large number of traffic signs, which can be read with a simple glance. Generally, traffic signs can be divided into three big subsets, including prohibitory signs, mandatory signs and danger signs. Signs from different subsets usually have large variations in appearance. However, signs from same subset may share many similarities, which makes they are easily confused with each other during classification. Due to the large intra-class difference and small inter-class similarity, a hierarchical classification TSR system is developed by integrating multiple level CNNs.

Recently, supervised learning based on CNNs has been widely applied in a lot of computer vision applications. However, unsupervised learning with CNNs has attracted much less attention because of its difficult. Moreover, for the most existing TSR methods [18–23], supervised leaning are also dominate methods. Therefore, TSR system are applied with unsupervised features based on the GANs, including Deep Convolutional Generative Adversarial Networks (DCGANs) [188] and Wasserstein Generative Adversarial Networks (WGANs) [189].

Despite CNN based methods have achieved excellent performance, exploring, understanding and interpreting the internal working principles of CNNs are still elusive problems to researchers. Some recent methods apply recognition tasks by the visualisation and activation CNN models [190–192, 155]. Inspired by these approaches, an novel layer is proposed by applying MPPs as information for sparse feature learning. Due to the powerful feature learning ability, CNNs usually generate overcomplete and redundant features, which may cause overfitting. With the help of proposed sparse layer, unnecessary features can be effectively removed, leading to improved performance.

Generally, deeper CNNs generate more discriminative features than shallower CNNs [30]. However, the training of deep CNNs face problems. The first problems are gradient vanishing and exploding [156]. Fortunately, with the help of BN [158], the problems of feature forward propagation and gradient backward propagation have been largely eased. The second critical problem is degradation [32], which have been discussed by *shortcut connections* in many approaches, such as Highway Networks [159], ResNets [32, 161], Stochastic Depth Networks [162], FractalNets [163] and DenseNets [164]. In order to deal with these training problems, an alternative design is proposed, which employs multiple loss functions for training CNNs.

The main contributions in this chapter are listed as follows.

(i) A TSR system based on hierarchical classification, which improves performance by recognising different subsets of traffic signs at different levels.

(ii) A TSR system base on classifiers trained with unsupervised features that are learnt by GANs.

(iii) A novel sparse layer based on MPPs, which is able to learn sparse feature and improve classification performance effectively.

(iv) A novel CNN architecture design that is able to ease the training problems of deep CNNs, including gradient vanishing and degradation.

The rest of this chapter is organised as follows: Section 6.2 presents the proposed methods, including the hierarchical classification system in Subsection 6.2.1, the TSR system based on unsupervised feature learning in Subsection 6.2.2, the novel sparse layer for sparse feature learning in Subsection 6.2.3 and the novel CNN architecture that has multiple loss functions in Subsection 6.2.4. Experiment details are provided and discussed in Section 6.3, followed by the summary in Section 6.4.

6.2 Approachs

6.2.1 Recognition with a Hierarchical Classification System

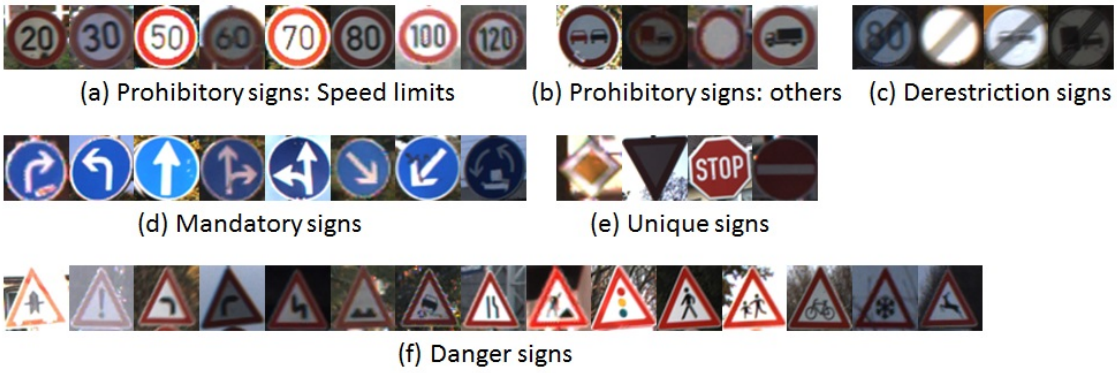


Fig. 6.1 Examples from the six subsets of GTSRB.

Hierarchical Structure of Dataset

Originally, the GTSRB contains 51839 traffic signs that belong to 43 unique classes. According to the appearance and meaning of the traffic signs, the dataset is further split into

six subsets that have been shown in Fig. 6.1. For the subsets from (a) to (f) in Fig. 6.1, the numbers of the signs are 16950, 6030, 1500, 7409, 8190 and 11760 respectively.

Hierarchical Structure of System

The proposed hierarchical classification system includes two levels, which are the rooted level and the branch level, as illustrated in Fig. 6.2 . The rooted level contains 1 classifier that is responsible for classifying the 43 classes of signs into the 6 subclasses. Subsequently, the branch level includes 6 classifiers which focus on discriminating the specific classes. Therefore, the proposed hierarchical classification system is composed of 7 CNNs.

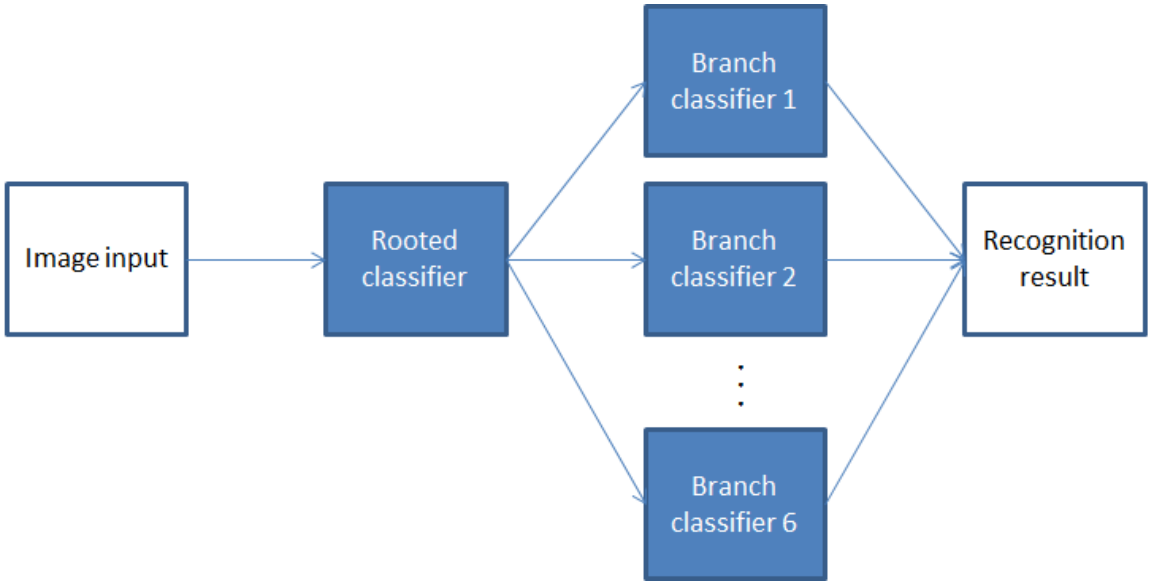


Fig. 6.2 System overview of the proposed hierarchical classification system.

Network Architectures

The CNNs applied in this system are similar to the VGGNets [30], the detailed network architecture is illustrated in Fig. 6.3. Each of the network consists of 4 convolutional layers and 3 fully-connected layers. BN [158] layers are employed for improving training speed and regularisation. Furthermore, dropout [174] layers are introduced to improve performance. Due to the huge difference of the numbers of traffic signs in the subsets, the 7 CNNs applied have different width, which will be detailed in the following section.

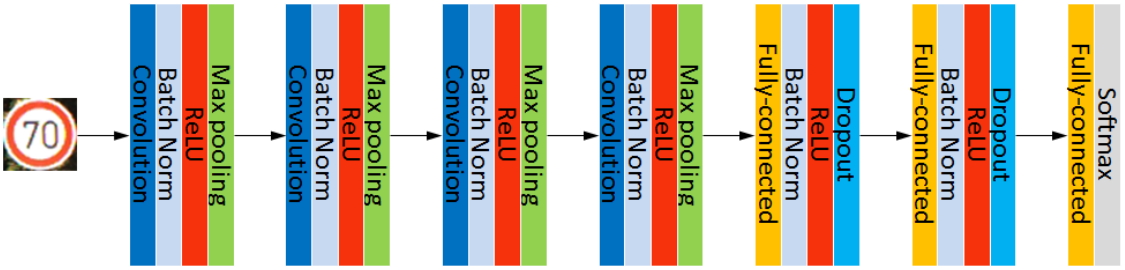


Fig. 6.3 The network architecture of CNNs in the hierarchical classification system.

6.2.2 Unsupervised Feature Learning Based on Generative Adversarial Networks

Generative Adversarial Networks

The adversarial modeling framework [193] consists of two networks: a generative network G that captures the real data distribution, and a discriminative network D that estimates the probabilities of samples from real data and fake data. In the training stage, the aim of G is to generating fake data that makes D predicts it as real with high probability. On the contrary, D tries to discriminate real and fake data as correct as possible. Therefore, the training procedure is a minimax two-player game. The system overview of the original GANs is illustrated in Fig. 6.4.

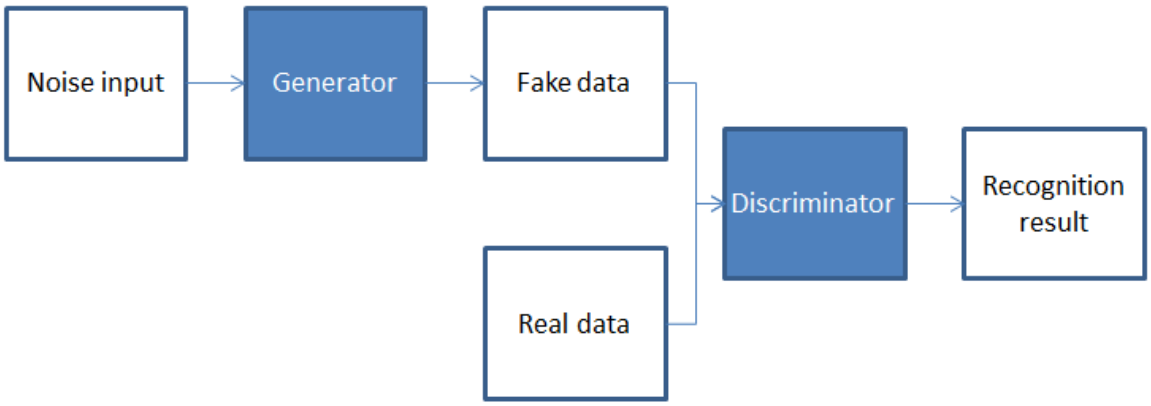


Fig. 6.4 System overview of generative adversarial networks.

In order to learn the distribution of real data $p_{real}(x)$, a set of noise variables $p_z(z)$ are firstly sampled, and then mapped to a new data distribution p_{fake} through the generative network G. Subsequently, the discriminative network D predicts single scalars that represent the probability of real or fake data. Finally, D is trained to maximise the probability of assigning the correct label to both real and fake data, while G is trained to minimise the

probability of assigning the correct label to fake data. Formally, D and G play the following minimax two-player game with value function $V(G,D)$ formulated as:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{real}(x))}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \tag{6.1}$$

where G and D are the generator and discriminator respectively. log is the logarithm function. In practice, at the beginning of training, G is not able to generate high quality data, leading to easily saturated values of $\log(1-D(G(z)))$. Therefore, G can be trained to maximise $\log D(G(z))$. So that the objective is not changed, but gradient is much stronger.

Deep Convolutional Generative Adversarial Networks

Originally, the networks G and D of GANs [193] are MLPs, which are not able to learn complex data distribution and generate high quality data. The training of the GANs faces two problems [188]: (i) delicate and unstable training procedure; (ii) using of CNNs in GANs. In order to solve the problems, the DCGANs [188] are proposed by adopting the following techniques.

(i) Using fractional-strided convolutions (transpose convolution or deconvolution) in generator G, and strided convolutions in discriminator D for performing pooling operations.

(ii) Applying BN in both G and D.

(iii) Removing Fully-connected layers from the deep architectures.

(iv) Using ReLU in G for all the activation layers, expect for the last layer, which is Tanh.

(v) Using Leaky ReLU in D for all the activation layers.

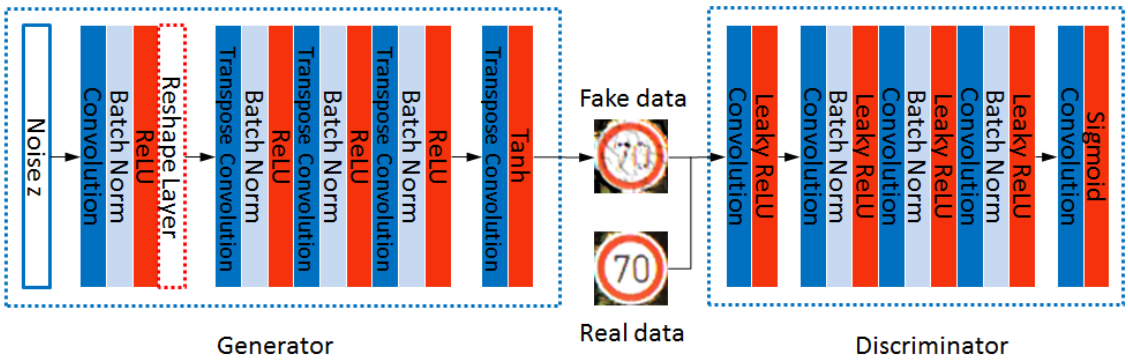


Fig. 6.5 The network architecture of DCGANs.

The network architectures of DCGANs are illustrated in Fig. 6.5. First, in generator G, a uniform distribution noise z is projected to high dimension feature maps by convolution and reshape layers, and then four transpose convolution layers convert the features into fake data. Second, both the fake data and real data are fed into discriminator D. Finally, D estimates the probabilities of data belongs to real or fake.

Wasserstein Generative Adversarial Networks

DCGANs successfully integrated deep CNNs into the architectures of GANs. However, the training of GANs still faces problems. One of the most important problem is collapse modes, which is the situation that generator generate very less kind of fake data, most of the fake data is repeat and lack of variety.

Instead of using popular probability distances and divergences, Earth Mover (EM) distance is applied in WGANs [189], and improved performance is achieved with the following modifications.

(i) Removing the last sigmoid layer in discriminator D.

(ii) Removing log in the loss function.

(iii) Weight clipping after each parameters update.

(iv) Applying optimisation algorithms that do not rely on momentum, such as SGD and RMSProp.

Recognition with Features of Generative Adversarial Networks

With the features learnt from the GANs, traffic signs are recognised with simple MLPs. The implementation details will be presented in the following sections.

6.2.3 Sparse Feature Learning Based on Max Pooling Positions

Network Architecture in Pre-training Stage

The network applied in pre-training stage is composed of 4 convolutional layers and 3 fully-connected layers, the detailed network architecture is illustrated in Fig. 6.6. Subsequently, for all the training samples of GTSRB, their features at the last convolutional layer are extracted. As indicated in Fig. 6.6, the features at the dash line are collected, the size of features of each sample is $3 \times 3 \times 512$.

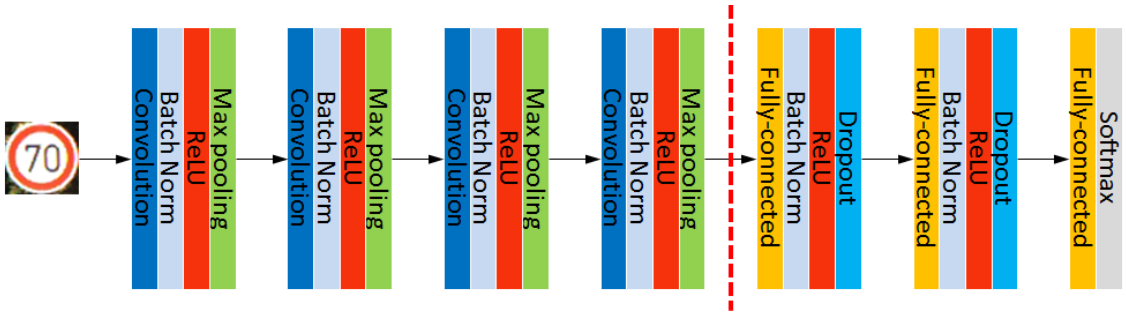


Fig. 6.6 The network architecture in pre-training stage.

Max Pooling Positions

As Fig. 6.7 illustrates, the size of features of each training sample is $3 \times 3 \times 512$. Then, a max pooling operation with kernel size 3×3 and stride 3 is implemented on the features, resulting in 512 max values and the corresponding positions. Finally, the MPPs of each sample are recorded. For example, as displayed in Fig. 6.7, the first feature map has a max values at top-left, so that the feature map is firstly encoded into a 3×3 matrix: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$, and recorded subsequently.

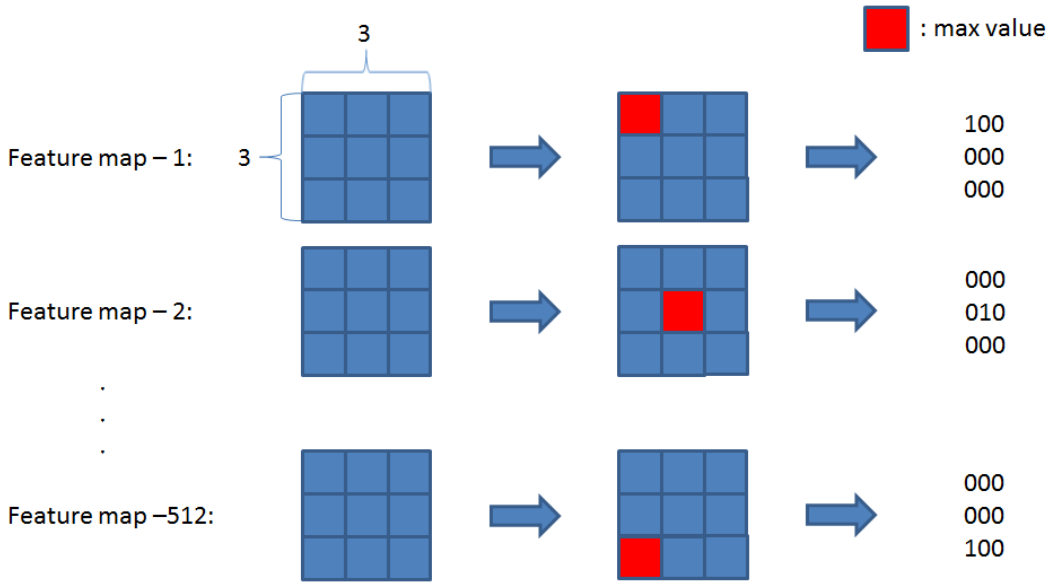


Fig. 6.7 Illustration of the procedures for computing max pooling positions.

One Versus One Classifications

In order to achieve higher performance, the original multi-class classification problem is transferred into a set of one versus one classification problems. Since GTSRB includes 43 unique classes, $\frac{43(43-1)}{2} = 903$ one versus one classifiers should be train for performing the 43-way multi-class problem.

For each classifier, supposing two classes: $i, j \in \{1, 2...43\}$ are related, the training stage can be described in the following steps.

(i) Data collection. All the features and the corresponding MPPs belong to classes i and j are collected.

(ii) Data processing. In order to measure the similarities of MPPs belong to same class, all of the MPPs come from same class are accumulated, and then normalised by dividing the number of them. So that two matrixes are calculated, which can indicate the probability of appearance of the max value positions. The two probability are denoted as p_i and p_j.

(iii) Sparse layer creation. First, a kernel matrix of the sparse layer with size $3 \times 3 \times 512$ is initialised. Second, based on the values of the two probability matrixes p_i and p_j, a difference matrix is computed by $d = \left| p_i - p_j \right|$. And then, the positions of the highest n values of d are selected, where $0 < n \leq 4608$. The hyperparameter n is decided by cross validation. Finally, the kernel matrix of the sparse layer is assigned with 1 at the selected n positions, and 0 otherwise.

(iv) Classifier initialisation. A classifier with with three layers is initialised, as Fig. 6.8 illustrated. The kernel of the sparse layer are obtained from last step.

(v) Classifier training. The classifier is trained by standard back propagation. More specifically, first, the features are passed through the sparse layer, refined by element-wise multiplication with the sparse kernel. Second, the sparse features are fed into the fully-connected layer, and then the softmax layer. Finally, gradients are calculated from the loss function, and back propagated to the fully-connected layer.

Network Architecture in Testing Stage

The network applied in testing stage is illustrated in Fig. 6.8. At the first glance, this network architecture is very similar to a typically multi-task network. However, the feature extraction layers and classification layers are not optimised as a whole, which means MTL is not

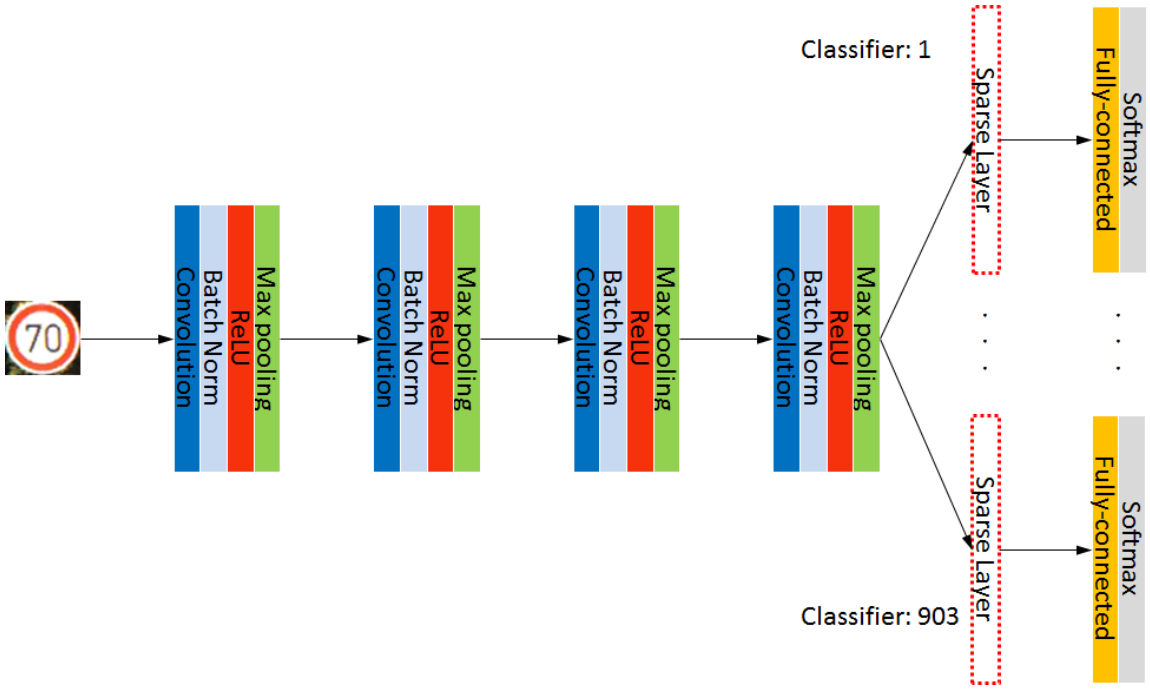


Fig. 6.8 The network architecture in testing stage.

involved. The network includes 903 binary classifiers, which rely on the the same public feature extraction layers and 903 dedicated sparse layers. For each testing sample, 903 binary labels are firstly predicted, and then the final label is voted by the 903 predictions.

6.2.4 Very Deep Network Training with Multi-loss Functions

Learning with Auxiliary Classifiers

Due to the good performance of shallow networks, the features of low-level and middle-level layers are verified to be very discriminative and important. However, in the training of very deep networks, the problem of propagating gradient back through all the layers is a big challenge, low-level layers in particular. In order to improve the ability of gradient back propagation, auxiliary classifiers are employed in the GoogLeNets [31]. These auxiliary classifiers are connected to the intermediate layers of networks, which are able to ease the gradient vanishing problem and provide regularisation.

The network architectures of PlainNets-10/18/34 with auxiliary classifiers are illustrated in Fig. 6.9. For the strategies of GoogLeNets, in the training stage, the loss of auxiliary classifier is added to the main loss with a discount weight (0.3). In testing stage, these auxiliary classifiers are discarded. Therefore, the auxiliary classifiers are only used for generating gradients and regularisation.

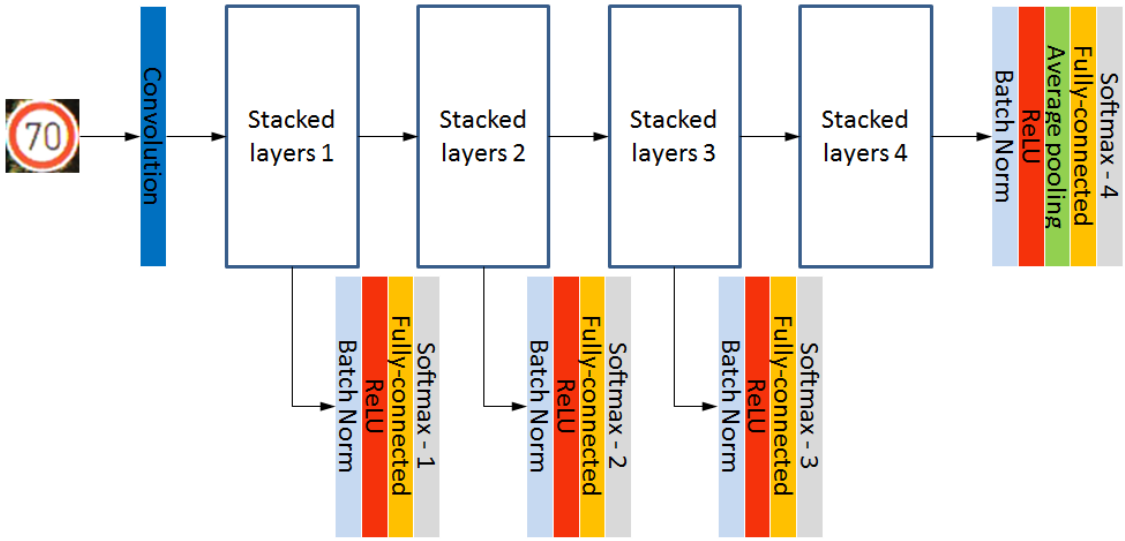


Fig. 6.9 The network architectures of PlainNets-10/18/34 with auxiliary classifiers.

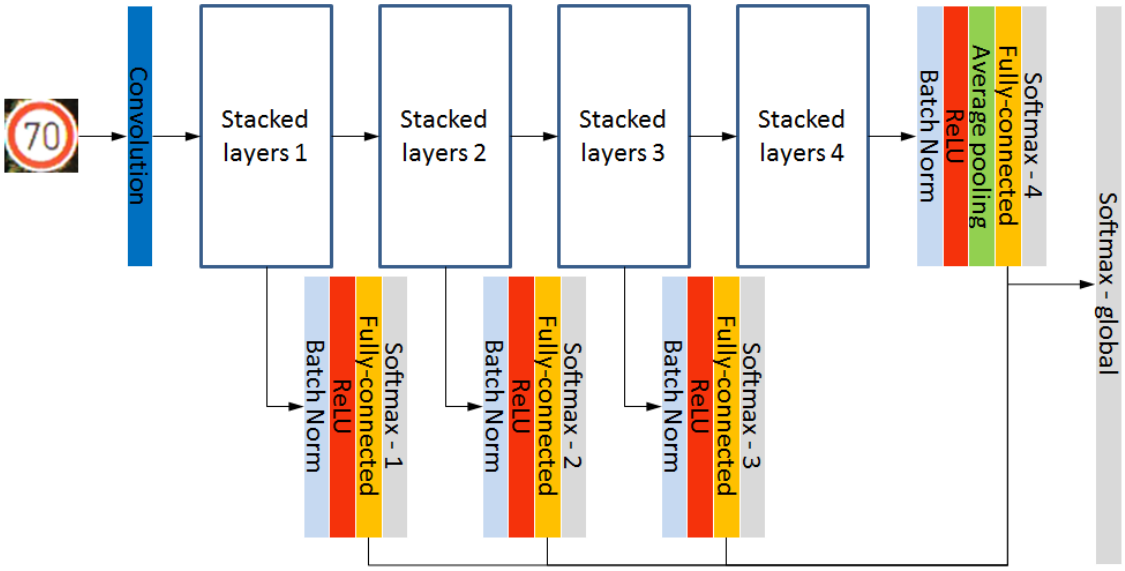


Fig. 6.10 The network architectures of PlainNets-10/18/34 with auxiliary classifiers and a global softmax layer.

Network Architectures of Proposed Networks

Inspired by GoogLeNets [31], a novel network architecture is developed, which is able to ease the problems of gradient vanishing and degradation. The network architectures of PlainNets-10/18/34 with proposed design are illustrated in Fig. 6.10. Except the auxiliary classifiers, a *global softmax layer* is added as the last output, its inputs are summarised by the inputs of the four softmax layers (*softmax-1* to *softmax-4*).

Different from the strategies of GoogLeNets, for proposed networks, the loss of auxiliary classifier is added to the main loss with a relative small weight (0.1) in the training stage. The weight is decided by control experiments, which indicate that large weights have negative effects to the convergence of the networks, while too small weights have too less influence to the convergence. Based on this design, the problem of gradient vanishing is eased. In the testing stage, instead of discarding the auxiliary classifiers, the final labels are predicted by summarising the inputs of all the four softmax layers. This ensemble voting scheme can improve performance, even though part of the classifiers give wrong predictions. Therefore, the degradation problem is also well eased.

6.3 Experiments

In this section, first, the system environment for implementing all the TSR systems will be introduced. Second, the dataset employed in the experiments will be presented. And then, the configurations of all the networks applied will be detailed. Finally, the results of the experiments will be provided and discussed.

6.3.1 System Environment

In order to implement all the TSR systems, a desktop with an Intel 3.5GHz CPU, 32GB memory and a NVIDIA GTX Titan GPU is employed. All the programs run on a 64-bit Windows 10 operating system with CUDA 8.0, CUDNNv5.1, Matlab 2017a and MatConvNet 1.0-beta24 [172] deep learning toolbox.

6.3.2 Dataset Details

German Traffic Sign Recognition Benchmark

The GTSRB [22] contains 51839 traffic sign images that belong to 43 classes. The examples and class distribution of the GTSRB have been clearly presented and discussed in Section 5.6.2.

Data Preprocessing

The data preprocessing is applied based on three steps. First, the borders of all the traffic signs are removed according to the annotations. And then, all the cropped traffic signs are resized to $48 \times 48 \times 3$ pixels. Finally, the colour normalisation of the traffic signs are utilised by linearly scaling all the pixels to the range $[-1, 1]$. Data argumentation is not applied in all the experiments.

6.3.3 Network Configurations

Hierarchical Classification Networks

Depending on the complexity of tasks, the CNNs are developed with different width. For the rooted classifier, the kernels of weight layers from input are {3,64,1}, {3,128,1}, {3,256,1}, {3,512,1}, {3,1024,1}, {1,256,1}, {1,6,1} in the form of {*kernel size*, *kernel number*, *stride*} respectively. For the branch classifiers, 6 task are related, as illustrated in Fig. 6.1. The *tasks (a), (d) and (f)* have the following kernels: {3,8,1}, {3,16,1}, {3,32,1}, {3,64,1}, {3,1024,1}, {1,256,1}, {1,*x*,1}, the *x* stands for the number of class of each task. On the other hand, the *tasks (b), (c) and (e)* have the following kernels: {3,8,1}, {3,16,1}, {3,32,1}, {3,64,1}, {3,256,1}, {1,32,1}, {1,*x*,1}. All the max pooling layers have 2×2 kernels with stride 2. The dropout rate is 0.5.

The CNNs are initialised with *Improved Xavier* [157], and trained for 30 epoches by applying Momentum of factor 0.9 and a mini-batch B with size 64. The learning rate η is started from 10^{-1} and uniformly decayed to 10^{-3} at the end of training. Moreover, L^2 parameter regularisation with value 0.0005 is employed for preventing overfitting.

Generative Adversarial Networks

The detailed network configurations of the DCGANs and WGANs have been displayed in Table 6.1. All the applied Leaky ReLU layers have leaky values 0.2. Padding is adopted in convolutions for keeping the size of feature maps, and cropping is applied in transpose convolutions for adjusting the size of feature maps.

For the training of DCGANs, networks are initialised with *Gaussian distribution* in the range $[-0.02, 0.02]$. The size of mini-batch B is set to 128, and learning rate η 0.0002. Adam ($\alpha = 0.5, \beta = 0.999, \varepsilon = 10^{-8}$) [187] optimisation algorithm is employed, and no parameter norm penalty is applied. The discriminator and generator are trained for 25 epoches. For each iteration in the first epoch, discriminator and generator are alternately trained with frequency

5:1. For each iteration in the rest epoches, discriminator and generator are alternately trained with frequency 1:1.

For the training of WGANs, networks are initialised with *Gaussian distribution* in the range $[-0.02, 0.02]$. The size of mini-batch B is set to 64, and learning rate η 0.00005. The weight clipping ensures all the parameters of networks stay in the range $[-0.01, 0.01]$. RMSprop ($\alpha = 0.99, \varepsilon = 10^{-8}$) [185] optimisation algorithm is employed, and no parameter norm penalty is applied. The discriminator and generator are trained for 25 epoches. For each iteration in the first epoch, discriminator and generator are alternately trained with frequency 5:1. For each iteration in the rest epoches, discriminator and generator are alternately trained with frequency 1:1.

Finally, with the features (outputs of Conv_D_4) extracted from the GANs, MLPs are trained for TSR. The MLPs have the following kernels: {3,1024,1}, {1,256,1}, {1,43,1}.

Generator			
Layer name	Output size	DCGANs	WGANs
Input	1×1	Noise with depth=100.	
Conv_G	1×1	$1 \times 1, 4608$, followed by BN and ReLU.	
Reshape	3×3	depth=512	
Convt_G_1	6×6	$5 \times 5, 256$ with stride 2, followed by BN and ReLU.	
Convt_G_2	12×12	$5 \times 5, 128$ with stride 2, followed by BN and ReLU.	
Convt_G_3	24×24	$5 \times 5, 64$ with stride 2, followed by BN and ReLU.	
Convt_G_4	48×48	$5 \times 5, 3$ with stride 2, followed by Tanh.	
Discriminator			
Input	48×48	Fake and Real data with depth=3.	
Conv_D_1	24×24	$3 \times 3, 64$ with stride 2, followed by Leaky ReLU.	
Conv_D_2	12×12	$3 \times 3, 128$ with stride 2, followed by BN and Leaky ReLU.	
Conv_D_3	6×6	$3 \times 3, 256$ with stride 2, followed by BN and Leaky ReLU.	
Conv_D_4	3×3	$3 \times 3, 512$ with stride 2, followed by BN and Leaky ReLU.	
Conv_D	1×1	$3 \times 3, 1$	
Activation	1×1	Sigmoid	-

Table 6.1 The detailed configurations of the generators and discriminators used in DCGANs and WGANs. *Convt_G_x* stands for the transpose convolution in generator, the cropping parameters are [2 1 2 1]. *Conv_D_x* stands for the convolution in discriminator, the padding parameters are [1 1 1 1].

Max Pooling Positions Networks

The network applied in pre-training stage has the following kernels: {3,64,1}, {3,128,1}, {3,256,1}, {3,512,1}, {3,1024,1}, {1,256,1}, {1,43,1}. For each of the 903 binary classifier,

a feature spare layer with kernels $3 \times 3 \times 512$ is included, followed by a fully-connected layer with kernels {3,2,1} and a softmax layer.

The networks are initialised with *Improved Xavier* [157], and trained for 10 epoches by applying Momentum of factor 0.9 and a mini-batch B with size 128. The learning rate η is started from 10^{-1} and uniformly decayed to $10^{-1.5}$ at the end of training. Moreover, L^2 parameter regularisation with value 0.0005 is employed for preventing overfitting.

Multi-loss Networks

The applied baseline networks are PlainNets-10/18/34 and ResNets-10/18/34, the detail network architectures have been shown in Table 5.6. For the proposed networks, auxiliary classifiers are added to *stacked layers 1*, *stacked layers 2* and *stacked layers 3*, as shown in Fig. 6.10. Moreover, global softmax layers are applied to summarise the inputs of the previous softmax layers.

The CNNs are initialised with *Improved Xavier* [157], and trained for 10 epoches by applying Momentum of factor 0.9 and a mini-batch B with size 64. The learning rate η is started from 10^{-1} and uniformly decayed to $10^{-1.5}$ at the end of training. Moreover, L^2 parameter regularisation with value 0.0001 is employed for preventing overfitting.

6.3.4 Presentation of Generated Traffic Signs

The generated traffic signs during the training of the GANs are illustrated in Fig. 6.11. For every two epoches, 6 traffic signs are collected, and each of them belongs to one of the 6 subsets as in Fig. 6.1. According to the generated traffic signs, three main observations are found: (i) both methods can stably produce high quality samples. (ii) WGANs have obviously faster convergence rate than DCGANs. This is mainly caused by the network designs. DCGANs apply sigmoid function in output layer and log function in loss layer, which are easily saturated during training. On the contrary, WGANs discard sigmoid and log functions, resulting in increased training speed; (iii) collapse modes still exist, DCGANs in particular. For example, due to the less number of *prohibitory signs - Derestriction* (the third rows in Fig. 6.11) in the GTSRB, both DCGANs and WGANs trend to generate other kinds of traffic signs.

6.3.5 Performance Evaluation of Hierarchical Classification System

The results in Table 6.2 present the performance of the hierarchical TSR system, together with the results of other TSR systems. The proposed hierarchical system achieved 100.00%

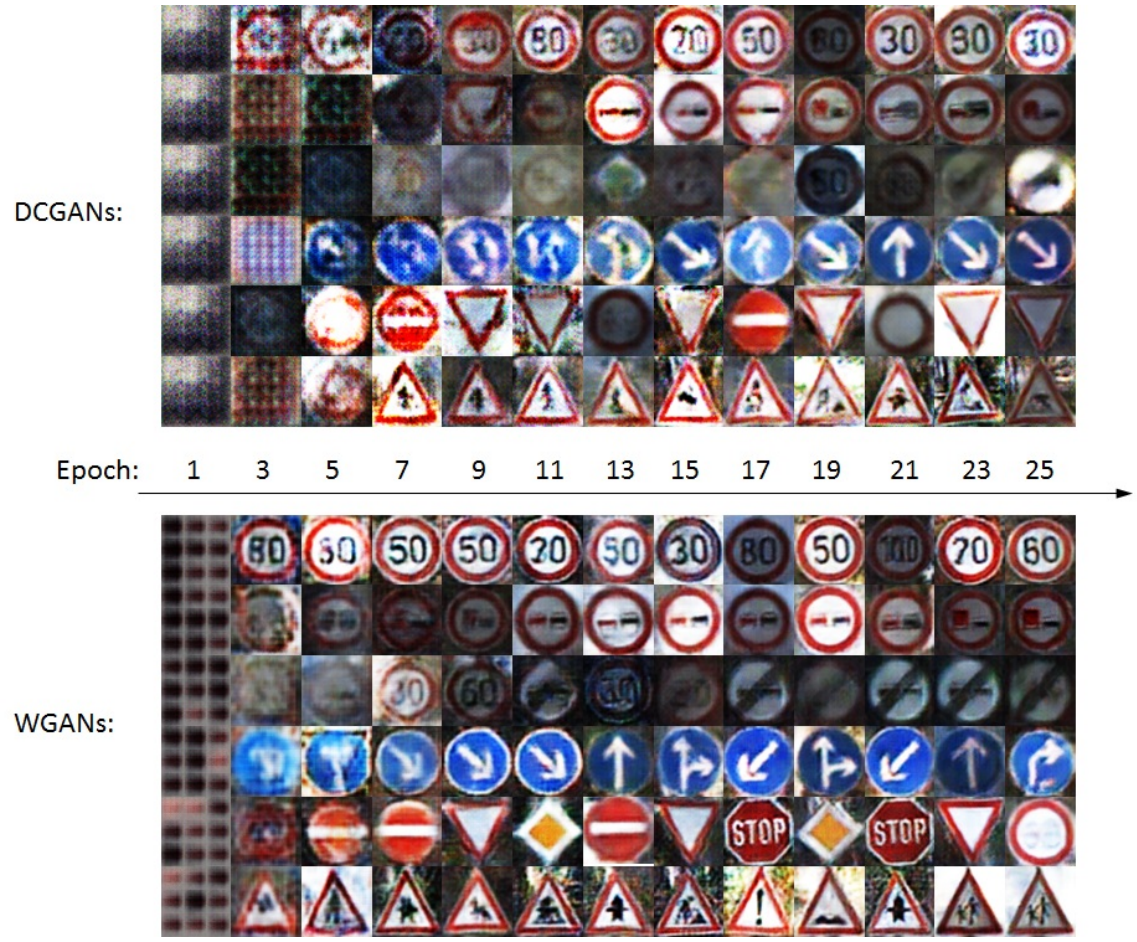
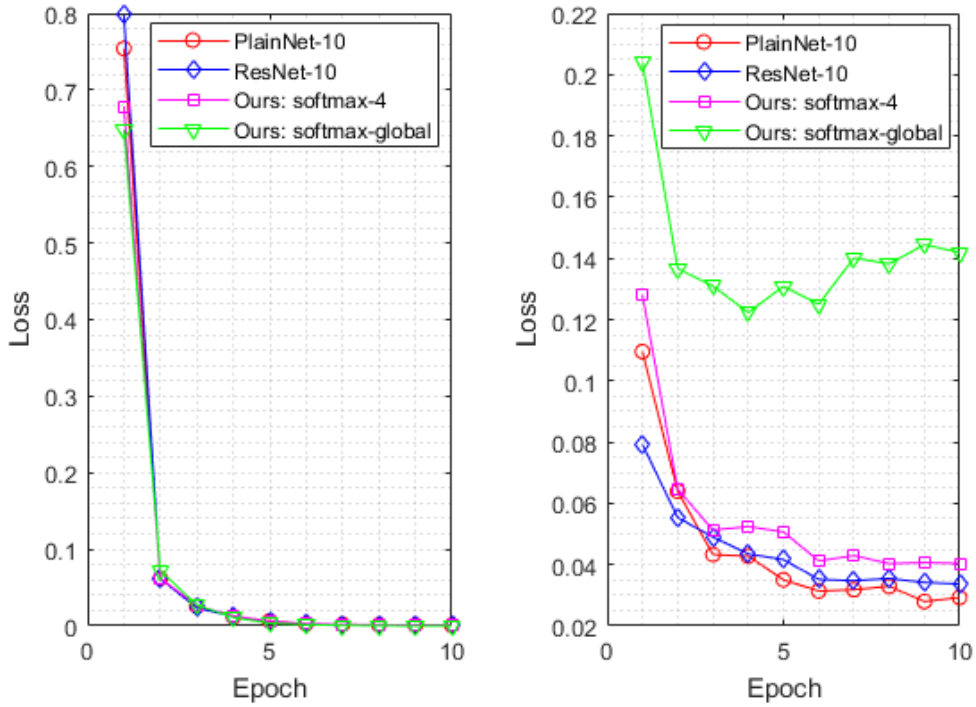


Fig. 6.11 Illustration of generated traffic signs during the training of the GANs. Top: DCGANs, bottom: WGANs. All the generated samples are not cherry-picked.

recognition rates in both *prohibitory signs - Others* and *prohibitory signs - Derestriction*, and the recognition rate of *Unique signs* is the highest except human performance. The performance indeed indicates that hierarchical classification is effective on the tasks with strong hierarchical structure. Comparing with the single CNN system, hierarchical system achieves better performance on most of the subsets, resulting in improved overall performance.

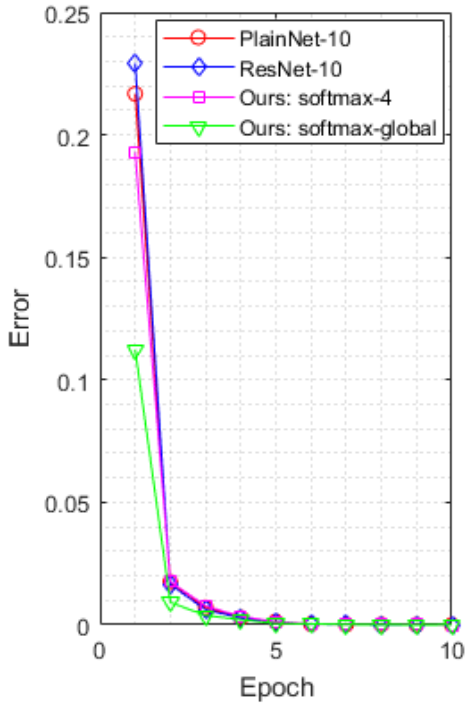
6.3.6 Performance Evaluation of Multi-loss Networks

The training details of PlainNets-10/18/34 ,ResNets-10/18/34 and the proposed multi-loss networks with $K = 1$ are illustrated in Fig. 6.12, Fig. 6.13 and Fig. 6.14. First, the Fig. 6.12a to 6.12d present the training details of the 10-layer networks. The results show that the proposed multi-loss network has the lowest training and testing error, which is benefited from the global summation. The convergence rate of the proposed network is similar to the

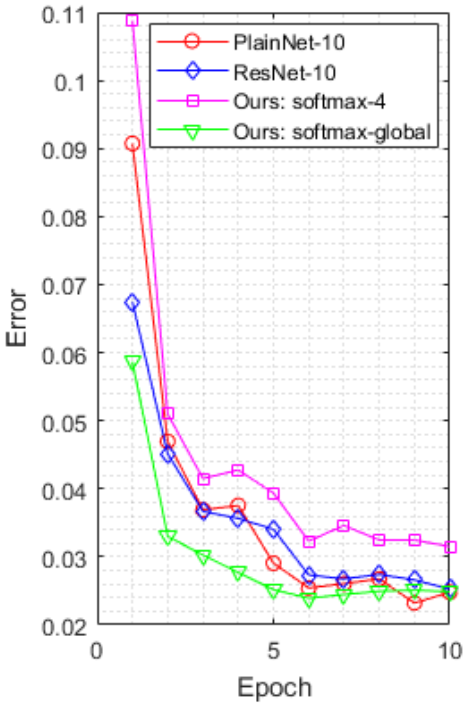


(a) Training loss

(b) Testing loss

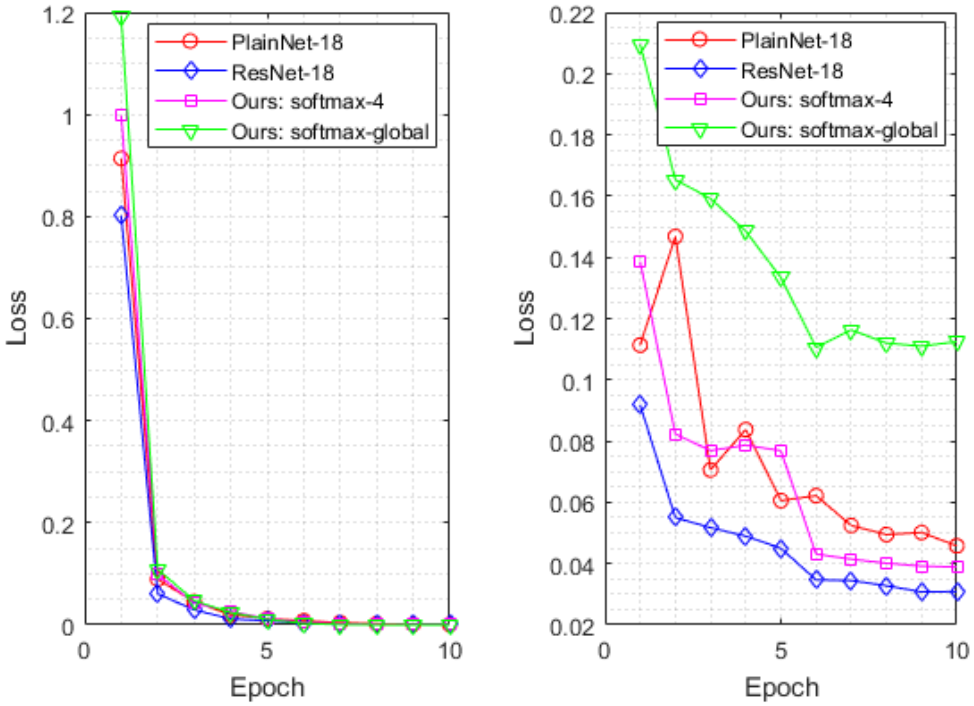


(c) Training error



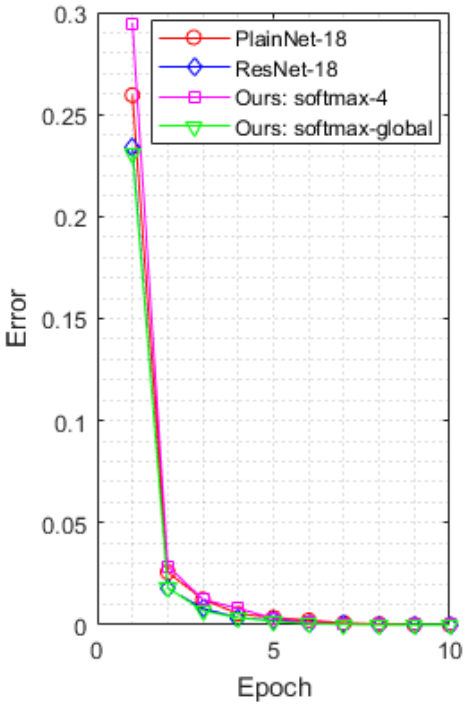
(d) Testing error

Fig. 6.12 Training details of PlainNets-10 ,ResNets-10 and the proposed multi-loss networks. *softmax-4* stands for the original softmax layer, and *softmax-global* stands for the introduced global softmax layer. All the networks are built with width factor $K = 1$.

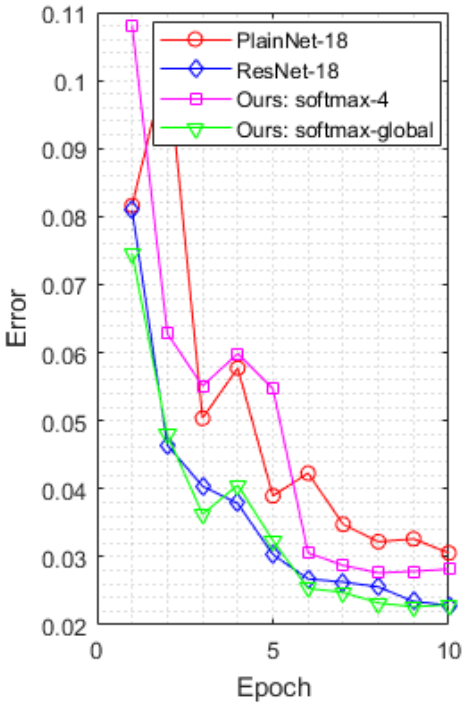


(a) Training loss

(b) Testing loss

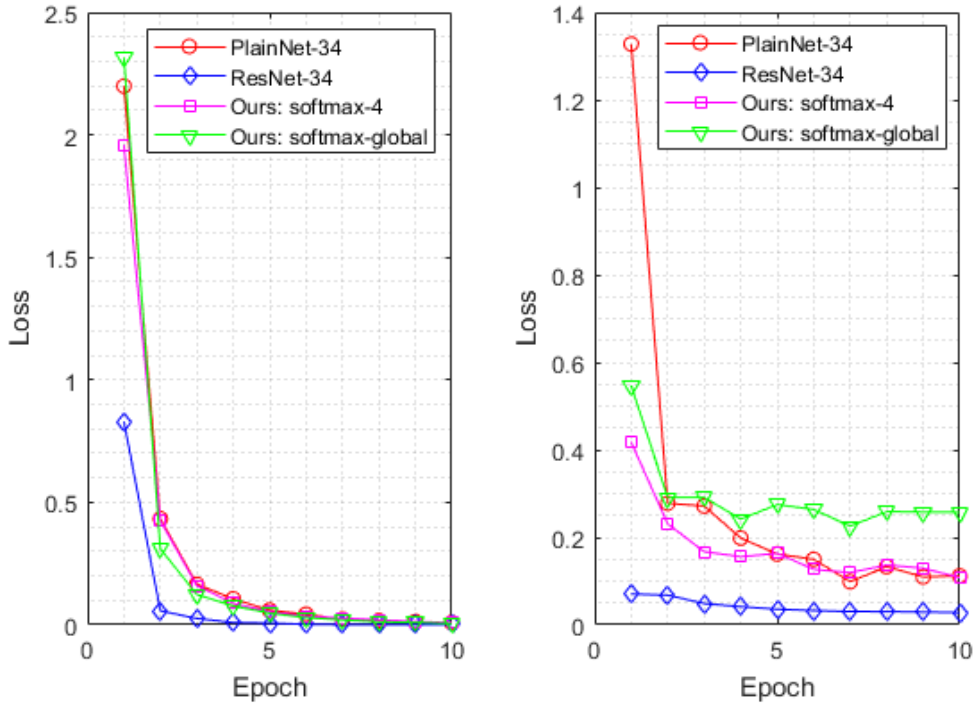


(c) Training error



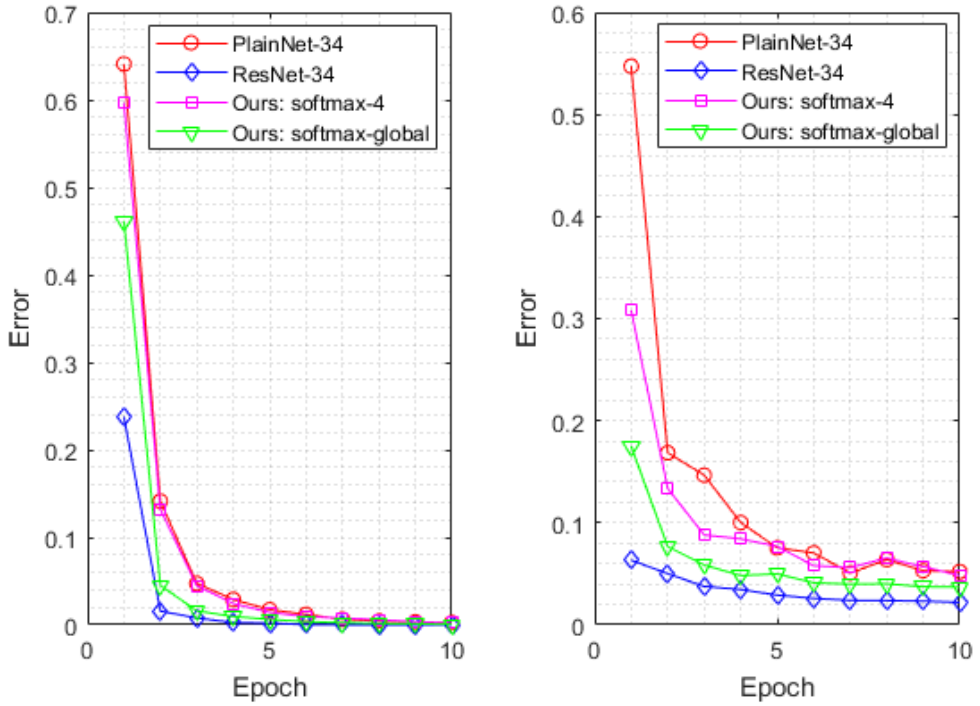
(d) Testing error

Fig. 6.13 Training details of PlainNets-18 ,ResNets-18 and the proposed multi-loss networks. *softmax-4* stands for the original softmax layer, and *softmax-global* stands for the introduced global softmax layer. All the networks are built with width factor $K = 1$.



(a) Training loss

(b) Testing loss



(c) Training error

(d) Testing error

Fig. 6.14 Training details of PlainNets-34 ,ResNets-34 and the proposed multi-loss networks.*softmax-4* stands for the original softmax layer, and *softmax-global* stands for the introduced global softmax layer. All the networks are built with width factor $K = 1$.

System name	Rooted	Speed limits	Others	Derestriction	Mandatory	Danger	Unique
Hierarchical CNNs (Ours)	99.88%	99.09%	**100.00%**	**100.00%**	99.55%	97.17%	99.90%
Single CNN (Ours)	-	99.23%	99.53%	92.78%	99.44%	96.67%	99.75%
Committee of CNNs [20]	-	**99.47%**	99.93%	99.72%	99.89%	99.07%	99.22%
Human (best individual) [22]	-	98.32%	99.87%	98.89%	**100.00%**	**99.21%**	**100.00%**
Human (average) [22]	-	97.63%	99.93%	98.89%	99.72%	98.67%	**100.00%**
Multi-Scale CNN [19]	-	98.61%	99.87%	94.44%	97.18%	98.03%	98.63%
Random Forests (HOG 2) [18]	-	95.95%	99.13%	87.50%	99.27%	92.08%	98.73%
LDA (HOG 2) [22]	-	95.37%	96.80%	85.83%	97.18%	93.73%	98.69%

Table 6.2 Individual performance of the 6 subsets. Bold font denotes the best results.

PlainNet-10 and ResNet-10. In Fig. 6.12b, the loss of *softmax-global* is higher than other softmax layers. The reason is that *softmax-global* summarises the results from 4 different outputs (as illustrated in Fig. 6.10) in forward propagation, but it does not generate gradients in back propagation. Second, the results in Fig. 6.13a to 6.13d also show that the proposed network has the lowest training and testing error, while its convergence rate is comparable to the other two networks. Finally, as Fig. 6.14a indicates, due to the problem of gradient vanishing, the convergence of the PlianNet-34 is significantly affected. By introducing auxiliary classifiers, the proposed network achieves boosted convergence rate. Moreover, the results in Fig. 6.14d show that the proposed network has lower testing error than the PlianNet-34, which means that the degradation problem is partially eased. Besides, based on the shortcut connections, the ResNet-34 achieves the highest convergence rate and the lowest testing error. To sum up, the proposed multi-loss networks have better performance than plain networks under various depth. On the other hand, the proposed multi-loss networks outperform ResNets when depth of networks are not extremely deep. In other words, the proposed design has better performance than plain and residual designs in relative shallow networks. However, in extremely deep networks, it outperforms plain design, but worse than residual design.

6.3.7 Performance Evaluation of Proposed Systems

The performance of the systems have been shown in Table 6.3, followed by the performance of the famous systems. For proposed systems, in order to observe the generalisation ability of networks themselves, data argumentation is not performed. The baseline CNN is the PlainNet-7 (64-128-256-512) that has been shown in Table 5.4. Based on the results, the conclusions are: (i) hierarchical classification can improve performance; (ii) the proposed MPPs based CNN is powerful, and it achieves the highest performance among all the systems, including the famous systems with data argumentation; (iii) the features learnt by unsupervised learning are discriminative, which are comparable with HOG; (iv) the proposed

multi-loss networks achieves increased performance than plain networks, however, gradient vanishing and degradation are not totally addressed.

System name	Accuracy (with data argumentation)	Accuracy (without data argumentation)
The performance of the proposed systems		
Single CNN (baseline)	-	98.63%
Hierarchical CNNs	-	98.81%
MPPs CNN	-	**99.58%**
MLP with features from RGB colours	-	92.51%
MLP with features from DCGANs	-	95.77%
MLP with features from WGANs	-	94.58%
10-layer Multi-loss CNN	-	97.61%
18-layer Multi-loss CNN	-	97.74%
34-layer Multi-loss CNN	-	96.30%
The performance of the famous systems		
Committee of CNNs [20]	**99.46%**	98.64%
Human (best individual) [22]	99.22%	99.22%
Human (average) [22]	98.84%	98.84%
Multi-Scale CNN [19]	98.31%	-
Random Forests (HOG 2) [18]	96.14%	-
LDA (HOG 2) [22]	95.68%	-

Table 6.3 The performance of the proposed systems, followed by the performance of the famous systems.

6.4 Summary

In this chapter, traffic signs are recognised with four proposed CNNs, which are trained by new strategies or constructed by new network architectures. First, different from the most existing approaches that apply single level classifiers for TSR, a hierarchical classification system is applied for improving the performance of TSR. Second, unsupervised feature learning is a challenge in computer vision. By introducing Generative Adversarial Networks (GANs), a TSR system is proposed based on features that are learnt without labels. Third, in comparison with previous CNNs that only apply max pooling for reducing feature dimension, a novel layer is proposed, which applies Max Pooling Positions (MPPs) of max pooling as information for sparse feature learning. Finally, in order to overcome gradient vanishing and degradation problems, a novel CNN architecture design that learns with multiple loss functions is developed. The performance of the proposed methods are verified on the German Traffic Sign Recognition Benchmark (GTSRB), and excellent results have been achieved compared with the famous systems.

Chapter 7

Chinese and English License Plate Detection Based on Extremal Regions

In this chapter, a License Plate Detection (LPD) system is proposed for detecting Chinese and English license plates with large skew angles. The proposed system consists of three main stages: (i) a character proposal stage to find candidate characters based on Extremal Regions (ERs); (ii) feature extraction and classification relies on Convolutional Neural Network (CNN); (iii) LPD by linking individual characters based on a new region linking method. By leaving out character segmentation, the proposed method has better robustness compared with existing methods. The performance of the proposed LPD system is evaluated on five datasets, including a large field-captured dataset, a skew and tilt dataset, a 12 countries dataset and two benchmark datasets. For Chinese civilian vehicles, the accuracy of LPD is 98.3%.

7.1 Introduction

Automatic detection and recognition of license plates of vehicles via digital imaging is a key component in Intelligent Transportation System (ITS), such as car park access control, electronic toll collection, suspect vehicle analysis and tracking, and automatic identification of expired registrations. In order to develop a robust LPD system, there are still some problems, for example, the detection of license plates with different layouts, characters and skew angles.

There is a great deal of variability between license plates in different countries with regards to the design, colours, characters used, and layout. Previously published LPD methods have been mainly centered upon a small number of the countries, including: mainland China,

Taiwan, American, Korea and certain European countries. On the other hand, most of the reported LPD systems are based on a common structure, the consecutive composition of: (i) LPD; (ii) character segmentation; (iii) character recognition. Effective segmentation is a critical step for LPD as the last step character classification depends largely on the quality of the segmentation output. However, segmentation is difficult in practice, especially when dealing with images with inherently noisy, low resolution and poor weather conditions.

Therefore, there are still challenges to be solved in order to produce a robust LPD system that is able adapt to the variability of the environments and different demands. For example, a captured vehicle image could be at a poor level of resolution due to the large distance between camera and the vehicle, or the poor lighting and low contrast which may in turn come from overexposure, reflections or shadows. Part of the plate can be obscured due to discoloration or dirt. Variations in the angle between a camera and the vehicle can produce perspective projection distortion of a license plate. Compared with most western countries, Chinese license plates are more challenging because of the complex combination of different characters on a license plate, including Chinese characters, English letters and digits, and many sub-categories of license plates with distinct colours and layouts.

The main motivation of this research is to develop a Chinese and English LPD system, and three main contributions are included as follows.

(i) A framework for LPD which simultaneously completes detection and segmentation.

(ii) A LPD system that is able to detect both Chinese and English license plate.

(iii) A LPD system that is able to detect extremely skewed license plate.

The rest of this chapter is organised as follows: Section 7.2 presents the detailed approach of the proposed LPD system; experimental results will be provided in Section 7.3, followed by summary in Section 7.4.

7.2 Approach

The overall LPD detection system is demonstrated in Fig. 7.1. In the first stage, a large number of candidate regions that may contain license plate characters are generated by region generators. In the second stage, all the generated regions are passed to a CNN for feature extraction and classification. In the third stage, license plates are detected based on a region linking strategy, which can link individual characters into license plate with large angles.

Fig. 7.1 System overview of the proposed LPD system.

7.2.1 Region Proposal

ERs [151] are connected components of an image binarised at a certain threshold. In this system, an input image (RGB) is converted into a set of binary images by applying pre-set thresholds, followed by connected component analysis for generating ERs. In order to reduce computational cost, regions with small areas (< 20 pixels) are ignored. The main motivation of generating different binary images and subsequent regions is to compensate for the negative effects of varying brightness and contrast. Therefore, high recall rate can be achieved in this stage.

7.2.2 Classification

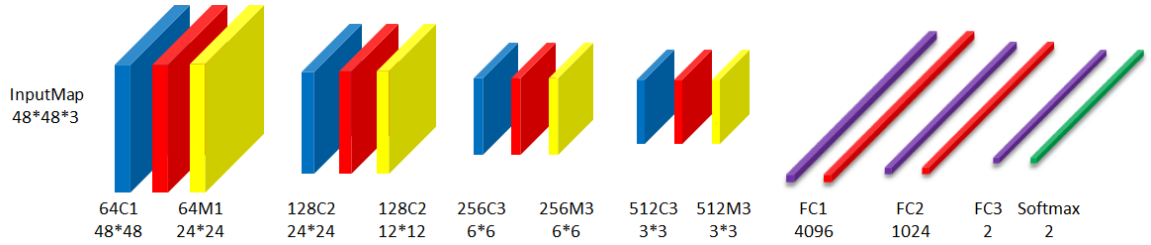


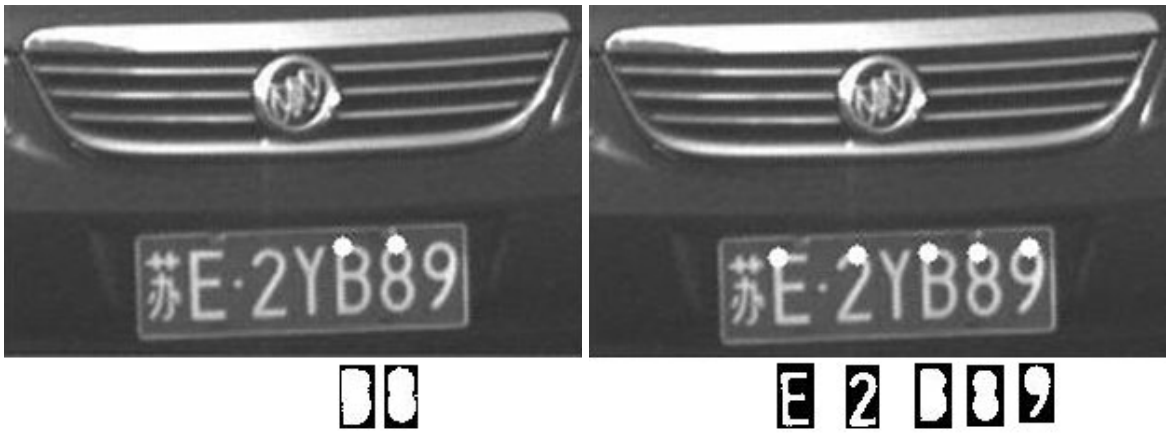
Fig. 7.2 Network architecture of the applied CNN.

The network applied is similar to the famous VGGNets [30], which is illustrated in Fig. 7.2. The network consists of four convolution layers followed by fully-connected layers and softmax layer. Each convolution layer followed by non-linear activation layer and max pooling layer. ReLU [173] is employed as the activation function for convolutional layers and full connection layers. Dropout [174] is adopted for preventing overfitting. The final softmax layer has 2 outputs, corresponding to character and background. All the input regions are firstly resized to $48 \times 48 \times 3$. After the classification, the label of each region is achieved, and then all the positive and negative regions are passed to next stage.

7.2.3 Region Linking

Individual characters should be linked together to format a license plate. The layout of license plates are specially designed in most countries. For example, all the Chinese license plates are composed of Chinese characters, digits and English letters. For the majority of Chinese civilian vehicles, the first character is Chinese, representing provincial level divisions. To the right of the Chinese character is Latin alphabet character representing the municipality or county. The remaining part is a combination of five digits or English letters. Therefore, in order to detect a Chinese license plate, it is sufficient to only consider digits and English letters on the plate, and then infer the complete plate location based on the prior knowledge about the layout of Chinese license plates.

For each binary threshold used in region proposal stage, a set of bounding boxes are obtained and denoted as $\text{Rect}_n^r = \{x_n^r, y_n^r, w_n^r, h_n^r\}$. The positive regions from previous stage can also generate a set of bounding boxes $\text{Rect}_m^c = \{x_m^c, y_m^c, w_m^c, h_m^c\}$. For example, there are 2 filtered regions in Fig. 7.3a and 5 filtered regions in Fig. 7.3b respectively. The region linking procedure can be described in the following steps.



(a) (b)

Fig. 7.3 Examples of filtered regions.

(i) Assuming the position and size of current region (hypothesis character) defined by a rectangle $(x_m^c, y_m^c, w_m^c, h_m^c)$, where (x, y) is the coordinate of top left corner of the rectangle, w and h are corresponding width and height. The next region, similarly denoted as $(x_n^r, y_n^r, w_n^r, h_n^r)$, is localised as follows.

   (a) Search towards the right for a matched region with similar height: $|h_n^r - h_m^c| < 0.5h_m^c$, and the horizontal and vertical differences between the two regions: $|x_m^c -$

$x_n^r|$, $|y_m^c - y_n^r|$, being less than the height of current hypothesis character. If such a region is found, the searching process will continue. Otherwise, move to next step.

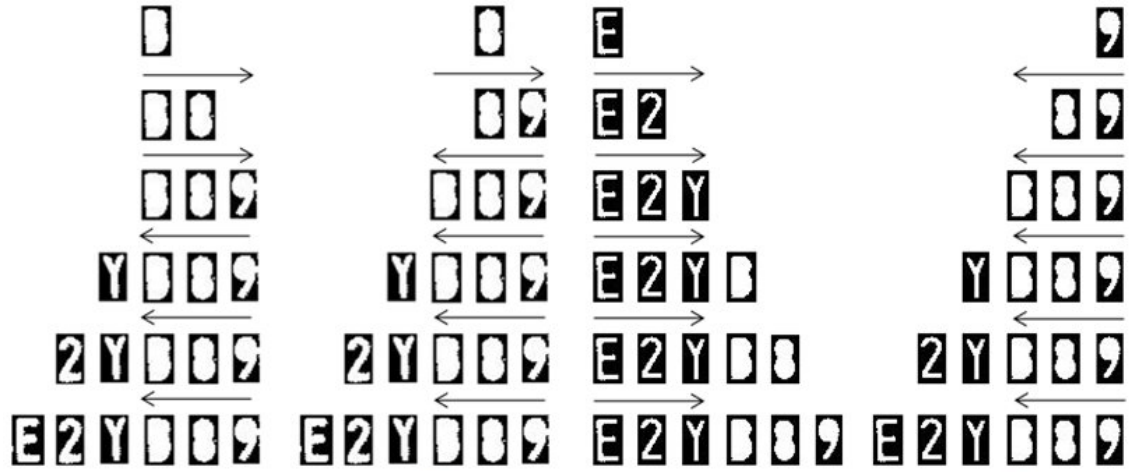
(b) Search towards the left for a matched region with similar height: $|h_n^r - h_m^c| < 0.5h_m^c$, and the horizontal and vertical differences between the two regions: $|x_m^c - x_n^r|$, $|y_m^c - y_n^r|$, being less than the height of current hypothesis character. If such a region is found, the searching process will continue. Otherwise, move to next step.

Some examples are provided in following Fig. 7.4 to further explain the above two steps. In Fig. 7.4, four examples are used to demonstrate the detection processes. In Fig 7.4a, the algorithm first searches towards the right with two similar regions found, and then searches towards the left finding three similar regions. In Fig. 7.4b, the algorithm first searches towards the right with one similar region found, and then searches towards the left with four similar regions located. In Fig. 7.4c, the algorithm first searches towards right finding five similar regions, and then search towards the left without detecting any similar regions. In Fig. 7.4d, the algorithm first searches towards right without finding any similar region, and then searches towards the left with five similar regions detected.

(ii) Count the number of regions. If the number is less than 6, then the current region is not a character on a license plate. If the number equals to 6, and the spacing between the leftmost character and its right neighbor conforms to the standard, it can be concluded that the regions are from a license plate. If the number is larger than 6, then attempt to find a region such that the spacing between it and its right neighbor conforms to the standard. If such a region exists, the region together with its right-side 5 consecutive regions will be deemed as the characters on a license plate. Otherwise the current region is not a character on a license plate.

(iii) The position of the last 6 consecutive characters will be used to infer the position of Chinese character on the license plate.

(iv) Since a same license plate may be detected several times by using different regions, it is necessary to remove those duplicate license plates. In the algorithm, this is simply implemented by removing the redundant license plates with the same position.



(a) (b) (c) (d)

Fig. 7.4 Explanation of the inference process for a hypothesis license plate. (a) and (b): license plate detection starts from B and 8. (c) and (d): license plate detection starts from E and 9.

7.3 Experiments

7.3.1 Implementation Details

In order to evaluate the performance of the proposed LPD system, a workstation with an Intel Xeon 3.3GHz CPU, 48GB memory and a NVIDIA GTX Titan GPU is employed. The programs run on a 64-bit Windows 7 operating system with CUDA 7.5, CUDNNv5, Matlab 2015b and MatConvNet 1.0-beta23 [172] deep learning toolbox.

7.3.2 Field-captured Dataset

The first set is collected from surveillance cameras at different traffic intersections in Suzhou, China. This image set consists of 1039 images with size 1360×1024 pixels. The distribution between the daytime and nighttime capture time of the images are 85% and 15%, as shown in Table 7.1.

	Car	Bus	Truck	Van
Daytime	244	233	172	230
Nighttime	18	80	42	20

Table 7.1 Data distribution of the field-captured dataset.

As the LPD system depends on two main parameters, namely the step of thresholds and the confidence scores of CNN. In the following, the system performance based on the change of the two parameters is evaluated.

The main motivation for generating multiple binary images from a single original image with different thresholds is to compensate for the large illumination variation, particularly the change from day to night. This procedure is also able to minimise the negative effect from different background colours of the license plates (white, yellow and blue). According to the experiments, a range from 10 to 240 is sufficient for threshold adaptation. In other words, the minimum and maximum threshold values are 10 and 240 respectively, and the internal thresholds are recursively calculated by a step. More specifically, a step of 5 means 46 thresholds: 10, 15, 20, $\cdots$, 230, 235, 240 will be applied to generate 46 binary images for the subsequence detection, while a step of 70 means only four thresholds: 10, 80, 170, 220 will be applied. With the 1039 sample images, the detection rates with the different steps are illustrated in Fig. 7.5.

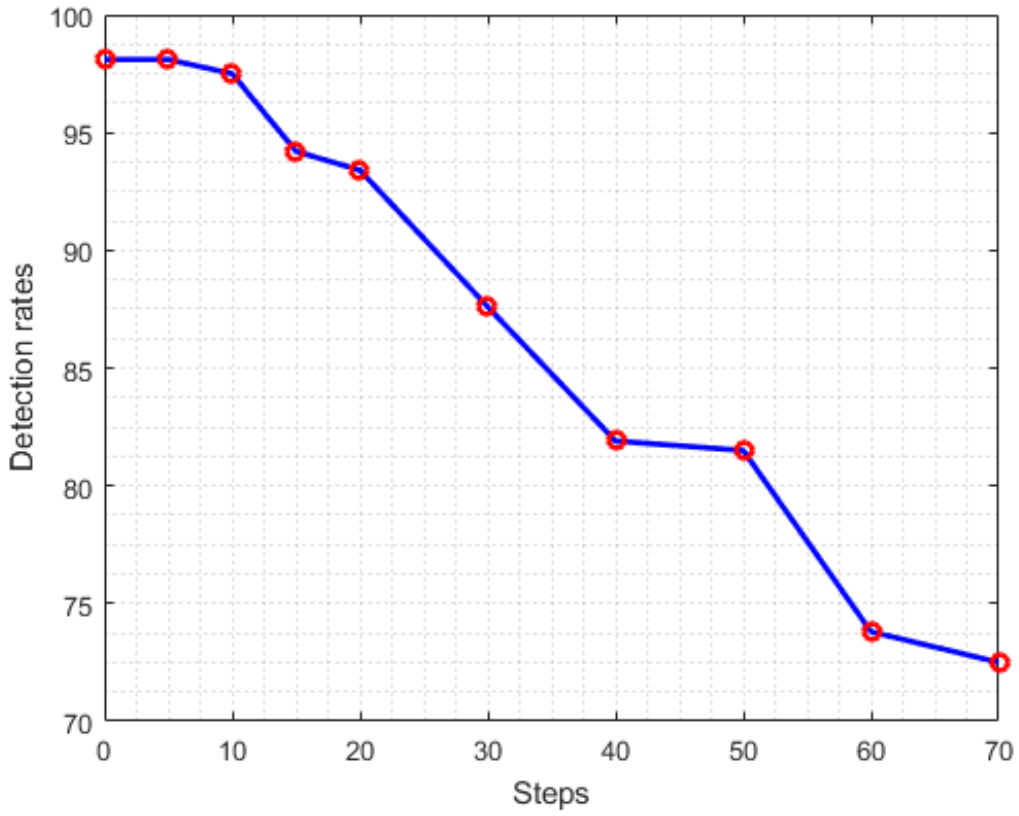


Fig. 7.5 Illustration of the detection rates with the change of steps in the range $[10, 240]$.

As Fig. 7.5 indicates, with the increasing of the step, less binary images are generated, resulting in reduced detection rates. The relatively small steps $(1 \sim 10)$ bring high detection rates ($> 98\%$), and once the step becomes too large (> 10), the detection rate drops rapidly. On the other hand, the computational cost is also decreasing with the increasing of step, in particular in the range $(1 \sim 5)$. Therefore, in order to achieve the best trade off between speed and performance, a step value of 10 is chosen in the system.

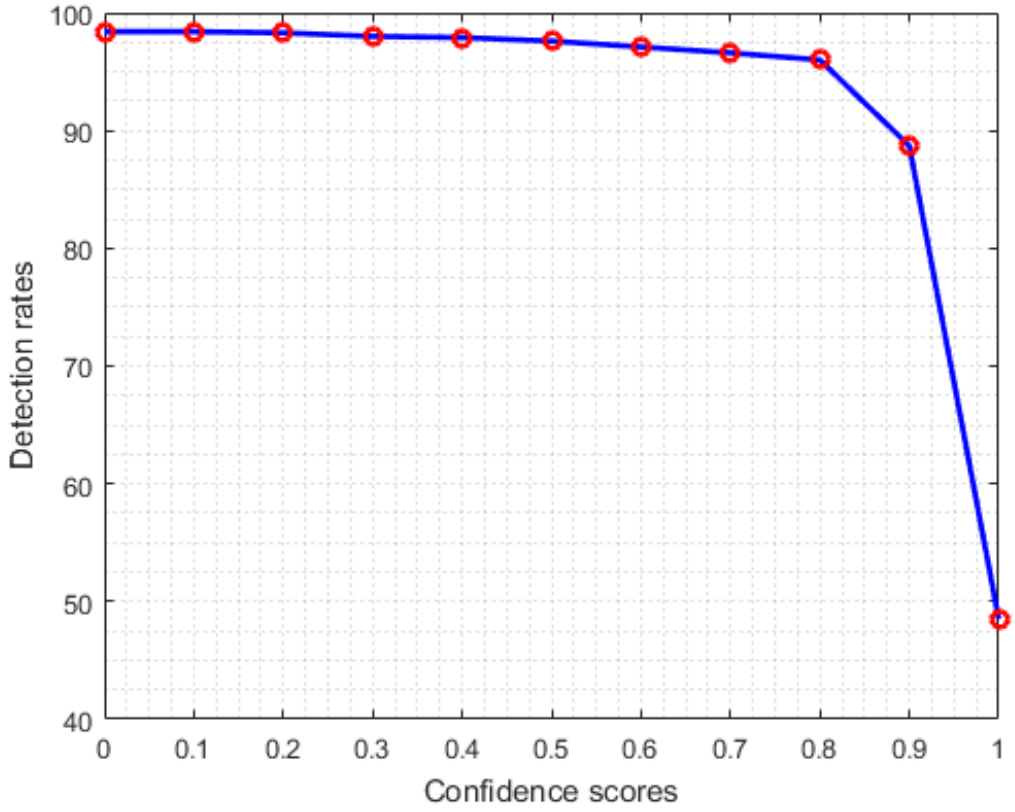


Fig. 7.6 Illustration of the detection rates with the change of confidence scores of CNN.

Confidence score is another important factor that critically influences the performance. The detection rates under different confidence scores are demonstrated in Fig. 7.6. The results show that 0.8 is a reasonable threshold of confidence score. When score larger than 0.8 is adopted, many characters are filtered out, thus giving low detection rate. On the contrary, score smaller than 0.8 also decrease the detection speed because regions that do not correspond to any characters would be treated as positive.

With the step of 10 for image binarisation and the confidence score of 0.8 for CNN, the detection results corresponding on field-captured dataset are given in Table 7.2. The accuracies of LPD are about 98.3% and 98.1% for daytime images and nighttime images

respectively. This demonstrates the advantage of the proposed algorithm, such as changes in the illumination do not apparently affect the performance.

The overall localisation rate of over 98.3% is indeed good. The result also indicates that the performance does not change much for different types of vehicles. The main reasons for the 1.8% of images which are failed in the experiments are: (i) the two neighboring characters become connected after binarisation; (ii) part of a character is missed; or (iii) the boundary of the characters is obscured. These problems are generally from vehicles with rusty plate or dirty plate.

	Daytime	Night	Overall Rates (%)
Car	98.0	94.4	97.7
Bus	99.1	98.9	99.0
Truck	95.9	100	96.7
Van	99.6	95.0	99.2
Overall Rates (%)	98.3	98.1	98.3

Table 7.2 Results of the field-captured dataset.

7.3.3 LP and Caltech Cars 1999 Datasets

To further verify the advantages of the proposed LPD system, the comparison experiments are conducted with four published LPD methods on two benchmark datasets, which are called the LP dataset [85] and Caltech Cars 1999 [194]. The first dataset contains 410 Chinese vehicle images and bears varied imaging conditions such as resolution, illumination and viewing angles. The second dataset has 126 images, each containing a American license plate with a cluttered background.

The first compared method is the Principal Visual Word (PVW) [85], which locate license plates by principal visual word, discovery and local feature matching. Other three methods are cited and compared in [85], including hybrid license plate extraction based on line detection and the construction of weighted edge map, denoted as HLPE, license plate detection in coarse-to-fine based on vertical edge detection and mathematical morphology, denoted as LPE , and license plate detection based on edge statistics and morphology, denoted as ESM.

In [85], the metric of evaluation for the two benchmark datasets is defined by (i) *high level-true*: license plate is totally encompassed by the bounding box and $A \cap B/A \cup B \geq 0.5$, where A is the detected region and B is the ground truth region; (ii) *low level-false*: the license plate is totally missed by the bounding box; (iii) *middle level-partial*: the remaining results excluded by the above two types.

For proposed system, the above metric needs to be revised because the proposed LPD system aims at simultaneously detecting individual characters and the whole plates. The metric is redefined as (i) *high level-true*: which means license plate is completely localised, and all of the characters are segmented; (ii) *low level-false*: the license plate is totally missed by the bounding box; (iii) *middle level-partial*: which refers to the situation that only part of the license plate has been localised and the segmented characters are incomplete.

The accuracy definition [85] of $True$, $Partial$, $False$ and $FalsePositiveRate(FPR)$ are formulated as:

$$True = \frac{TP}{TP + Partial_{TP} + FN + FP} \tag{7.1}$$

$$Partial = \frac{Partial_{TP}}{TP + Partial_{TP} + FN + FP} \tag{7.2}$$

$$False = \frac{FP + FN}{TP + Partial_{TP} + FN + FP} \tag{7.3}$$

$$FPR = \frac{FP}{TP + Partial_{TP} + FP} \tag{7.4}$$

where the TP is the true positive number, $Partial_{TP}$ stands for partial true positive number, FP denotes false positive number, and FN is false negative number (miss detection).

With above metric of evaluation, the comparison results are summarised in Tables 7.3 and 7.4, from the LP dataset and Caltech Cars 1999 datasets respectively. For the LP dataset, it is obvious that the proposed system performs the best in terms of the accuracy. Both the *Partial* and *False* detection rates are lower than all the other four methods. However, the FP 3.4% is higher than the result 1.0% of PVW method. This is mainly because many vehicle images in the LP dataset have advertisement or telephone numbers painted on the vehicles, which are prone to be detected as license plates. For the Caltech Cars 1999 dataset, the $True$ detection rate of proposed approach is 88.4%, which is higher than all the other methods. The *Partial* detection rate 6.2% is a little high, because of the low resolution of images in Caltech Cars 1999 dataset.

Some images in the LP dataset are captured with skew and tilt angles. As shown in Fig. 7.7 (top), the proposed system can successfully detect the license plates. Shadow is another negative factor that may affect performance. For weak shadow, there will be no influence on detection result as demonstrated by the examples in Fig. 7.7 (middle). However, when the shadow increases, it may causes detection failure. As illustrated in Fig. 7.7 (bottom), a plate with strong shadow result in incomplete characters.

Fig. 7.7 Examples of the LP dataset. Top: skew license plates, middle: weak shadow license plates, bottom: strong shadow license plates.

Approach	Accuracy			*FPR*
	True	*Partial*	*False*	
HLPE	80.8%	6.6%	12.6%	12.6%
LPE	84.6%	1.5%	13.9%	7.6%
ESM	74.6%	7.3%	18.1%	17.9%
PVW	93.2%	0.3%	6.5%	1.0%
Proposed	94.3%	0%	5.7%	3.4%

Table 7.3 Comparison results of the LP dataset.

Approach	Accuracy			*FPR*
	True	*Partial*	*False*	
HLPE	61.6%	9.8%	28.6%	28.6%
LPE	58.0%	1.8%	40.2%	29.5%
ESM	68.7%	2.7%	28.6%	25.9%
PVW	84.8%	1.8%	13.4%	4.5%
Proposed	88.4%	6.2%	5.4%	2.4%

Table 7.4 Comparison results of the Caltech cars 1999 dataset.

7.3.4 Skew and Tilt Dataset

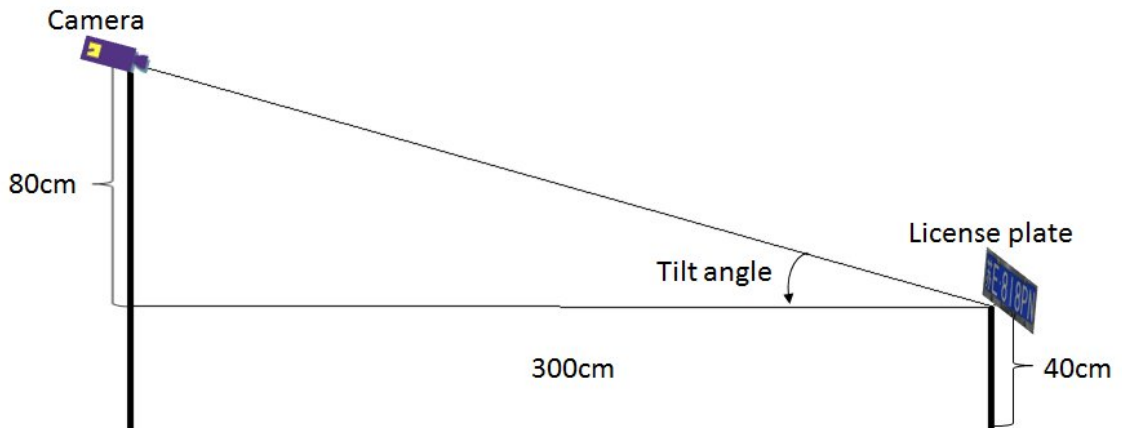


Fig. 7.8 Explanation of capturing system for collecting images with skewed and tilted license plates.

In many situations, there exists considerable variation in the capturing angles between a camera and the target license plate, which results in distorted license plate images. The license plates obtained from such images are far from perfect rectangles. Most of the

previously proposed LPD methods failed to detect such distorted plates. A possible solution is perspective rectification as discussed in many computer vision topics, which is however computational expensive in finding a transformation matrix that may rectify the perspective projection distortion.

To evaluate the performance of the proposed algorithm for skewed and tilted license plates, a set of vehicle images are collected with controlled capturing angles, as illustrated in Fig 7.8.

Fig. 7.9 Examples of the skew and tilt dataset. Top: skew degree $\pm 70^o$, bottom: skew degree $\pm 75^o$.

To change the tilt degrees, the camera is fixed on a tripod and the distance to ground is around $120cm$. As the perpendicular distance between the license plate and ground is $40cm$, the horizontal distance between camera and plate is $300cm$, the tilting angle between the camera and plate is around 15^o. To obtain vehicle images with different skew angles, the angles towards the left are regarded as negative and the viewing angles towards the right are regarded as positive. The sampling angles are -75^o, -70^o, -65^o, -60^o, -50^o, -40^o,

-30^o, -20^o, -10^o, 0^o, 10^o, 20^o, 30^o, 40^o, 50^o, 60^o, 65^o, 70^o and 75^o. For each angle, three images are shot with tilting degrees 14^o, 15^o and 16^o.

In the 57 skew and tilt images, 54 of license plates are correctly detected, which means a 94.7% detection rate. The system fails to detect license plate when the skew degree raised to 75^o (1 failed) and -75^o (2 failed) respectively. This indicates that the proposed algorithm is tolerant to skew degree up to $\pm 75^o$ when the tilting degree is around 15^o. Some of the examples are displayed in Fig. 7.9.

7.3.5 12 Countries Dataset

Fig. 7.10 Examples of the 12 different countries, including Australia, Austria, Canada, Croatia, France, Germany, Israel, Italy, Spain, Portugal, UK, and USA.

The majority of the published methods about LPD are proposed for a particular type of license plate in a specific country or region. This lack of extendibility is well-known problem in LPD. The method proposed in this chapter is able to overcome this problem due to the flexible framework of the region linking stage.

In this section, the proposed LPD system is tested on a dataset that is collected from the Internet image search engine. This dataset contains 171 licensed vehicle images, which belong to 12 different countries as illustrated in Fig. 7.10. The detection results are given in Table 7.5, the total detection rate is 96% on this 12 countries dataset.

	Australia	Austria	Canada	Croatia
#Images	10	11	10	13
#Detected	9	10	10	13
	France	Germany	Israel	Italy
#Images	11	20	7	11
#Detected	10	20	7	10
	Spain	Portugal	UK	USA
#Images	10	10	11	47
#Detected	10	10	10	45

Table 7.5 Results of the 12 countries dataset.

7.4 Summary

In this chapter, a LPD system is proposed for detecting Chinese and English license plates with large skew angles. The main contributions include: (i) a framework for LPD which simultaneously completes detection and segmentation; (ii) a LPD system that is able to detect both Chinese and English license plate; (iii) a LPD system that is able to detect extremely skewed license plate. Extensive experiments have been presented with different kind of datasets, and excellent results have been achieved.

Chapter 8

Context-aware Traffic Text Detection with Convolutional Neural Networks and Directed Acyclic Graph

Automatic acquisition of traffic information, including traffic signs, traffic texts, vehicles and pedestrians, plays an important role in high-level tasks of Intelligent Transportation System (ITS), such as Advanced Driver Assistance Systems (ADAS) and autonomous cars. The detection of traffic signs, vehicles and pedestrians has been widely discussed over the past two decades. Comparatively, Traffic Text Detection (TTD) has received less attention. Due to the various appearance and layout of texts, TTD in the wild still faces challenges. Moreover, there exist more problems for the development of TTD system in China, because of the large number and complex layout of Chinese characters. In this chapter, a context-aware Chinese TTD system is proposed by applying Convolutional Neural Network (CNN)s and Directed Acyclic Graph (DAG). First, the proposed system jointly applies Maximally Stable Extremal Regions (MSERs) and a regressor as region generator to ensure high recall. Second, all of the generated regions are not only assigned with confidence scores for character/non-character classification, but also inter-similarities for context-aware detection. Finally, characters are refined and linked into texts by applying an unified character linking method, which is based on DAG. With the help of these improvements, the proposed system can handle non-horizontal text lines and crossed text lines, while some existing text detection systems only are able to detect horizontal or near horizontal text lines. The error accumulation problem that usually occurs in the systems that use individual and sequential stages is also avoided. Experiments have been conducted using the proposed dataset, demonstrating excellent performance with regard to high recall and precision rate. The overall system $F_{measure}$ is 70.3% on the Internet-collected dataset.

8.1 Introduction

ADAS are systems that developed to enhance vehicles for automated/adaptive/safety driving, such as automatic parking, automotive navigation system, adaptive cruise control, lane departure warning system, pedestrian protection system and traffic sign recognition. Over the past two decades, due to the untiring investigation and development, ADAS become the fastest growing parts in automotive electronics [195, 5, 7, 6]. ADAS work rely on multiple data that captured by a variety of sensors, including lidar, radar, camera and vehicle communication system. Among all the sensors, camera sensors have the advantages such as cheap, small and flexible deployment. Moreover, due to the development of computer vision and hardware, camera sensors have shown excellent growth rate and market share in recent years.

Fig. 8.1 Examples of traffic texts in China. Green rectangles: the characters are composed of isolated components. Yellow rectangles: stand-alone characters. Red rectangles: non-horizontal and crossed text lines.

While the detection of traffic signs [53, 52], vehicles [6] and pedestrians [5] has been widely discussed, there is much less research focused on the detection of traffic text. Comparatively, TTD is sometimes more important due to the rich and clear high level semantic information for drivers. Nevertheless, TTD imposes more challenges [101, 102], such as the

wide variety of text appearance due to different fonts, colours, sizes and layouts. Therefore, there are fairly less works [95–103] that specifically focus on the detection of traffic text.

Among all the developed TTD systems, three systems [95, 102, 103] are designed for English texts and the other systems [96–101] are focused on Spain texts, remaining a blank for the detection of Chinese traffic texts. Generally, texts from different countries vary greatly in appearance, which means the applications of specially designed systems are geographically limited. Some examples of Chinese texts are illustrated in Fig. 8.1. In comparison to English and Spain texts, detection of Chinese texts is confronted with more challenges: (i) there is a huge number of Chinese characters included in traffic environment; (ii) different from English and Spain words that are composed of one or several consistent letters, Chinese characters are usually constructed of several inconsistent strokes, it is more difficult to detect Chinese characters as a whole; (iii) the inconsistent and isolated strokes of Chinese characters also make Chinese TTD lack of intra-character and inter-character contextual information; (iv) Chinese texts are usually shorter than English and Spain texts, the intervals between Chinese characters may larger, resulting in harder of linking characters into text lines; (v) except horizontal and near horizontal text lines, non-horizontal and crossed text lines widely exist in traffic environment. Therefore, Chinese TTD is more complex and difficult, in order to develop a high performance Chinese TTD system, the following problems should be solved.

(i) To establish a benchmark for Chinese TTD.

(ii) To use a region generator which can detect Chinese characters as a whole.

(iii) To apply a classifier that can discriminate characters/non-characters.

(iv) To develop a character linking method that can handle non-horizontal and crossed text lines.

To address these problems, discriminative features and powerful classifiers are the keys. Recently, the deep learning [24–28] has attracted much attention in computer vision and machine learning fields. In particular, CNNs have achieved outstanding performance due to the generalisation ability and learning capacity, and CNNs have been widely applied in various computer vision tasks, such as image recognition [29–32] and object detection [33–36]. Therefore, CNNs are applied in the proposed system to solve the problems listed above.

In this chapter, a context-aware Chinese TTD system is proposed, with a number of key contributions as follows.

(i) The first contribution is the construction of a new Chinese traffic text dataset, which is purely collected from Internet. This dataset contains 1825 images with about $20k$ individual Chinese characters and $7k$ text lines labelled. With the help of this dataset, the gap of Chinese traffic text detection and recognition will be filled.

(ii) The second contribution is a region proposal method that jointly uses of MSERs and a CNN based regressor to hypothesise the character locations by taking into account the complementary information from both modules. This method boosts performance while characters are constructed of isolated components, blurred or partially covered.

(iii) The third contribution is a classification method that uses both information from a local model and a contextual model. The local model is a CNN based classifier that is applied for assigning character/non-character confidence scores of candidate regions. The contextual model is a Siamese network that is introduced for measuring inter-similarities between pairs of candidate regions. The combination of local and contextual model allows more accuracy discrimination, especially in the detection of stand-alone and distant characters that are contained in Chinese traffic texts.

(iv) The fourth contribution is an unified character linking method based on DAG, which detects text lines by finding shortest path based on local weight, contextual weight and spatial weight. The local weight considers the confidence scores of candidate regions. The contextual weight measures the similarities of candidate regions. The spatial weight controls the angle, distance and size likelihood of candidate regions. Therefore, the problem of text detection is formulated into a task of finding the shortest path. This method is able to detect non-horizontal and crossed text lines. The error accumulation problem that usually occurs in the systems that use individual and sequential models is also avoided.

The rest of this paper is organised as follows: Section 8.2 presents the proposed Chinese traffic text dataset; Section 8.3 gives a detailed introduction of the proposed Chinese TTD system; The experimental results will be provided in Section 8.4, followed by summary in Section 8.5.

8.2 Chinese Traffic Text Dataset

8.2.1 Introduction of Proposed Dataset

The proposed Chinese traffic text dataset is established as follows.

Fig. 8.2 Examples from the proposed dataset. Red rectangles: characters, yellow rectangles: text lines. The top row: horizontal traffic texts and stand-alone characters, the second row: vertical traffic texts, the third row: non-horizontal traffic texts, the bottom row: crossed traffic texts.

(i) 1825 images with Chinese traffic texts are downloaded from the Internet, the resolutions of collected images vary from 240×180 pixels to 5184×3456 pixels.

(ii) In order to protect privacy, all the privacy related contents, such as faces and license plates, are blurred.

(iii) The ground-truth of all the characters are firstly annotated, and then the characters are manually linked into text lines accord to their semantic information.

(iv) The images are randomly divided into three sets, namely training, validation and testing, the allocation weights are 40%, 10% and 50% respectively.

The details of the proposed dataset are provided in Table 8.1. Some examples from the proposed dataset are illustrated in Fig. 8.2, including traffic texts with different layouts and capturing angles.

Resolution of Image	Training	Validation	Testing	Total
$\leq 300 \times 300$	34	11	46	91
$300 \times 300 \sim 600 \times 600$	350	92	399	841
$600 \times 600 \sim 900 \times 900$	242	49	340	631
$900 \times 900 \sim 1200 \times 1200$	59	20	66	145
$\geq 1200 \times 1200$	45	11	61	117
Total	730	183	912	1825

Table 8.1 Details of the proposed Chinese traffic Text dataset

Annotation of Characters

For character level annotation, all the legible character regions are marked with bounding boxes and character classes, whereas illegible tiny character regions are removed for preventing man-made error. Consequently, a total number of 20429 Chinese characters are collected. The details of the collected characters are provided in Table 8.2.

Annotation of Text Lines

For text line level annotation, all the labeled characters are manually linked into text lines accord to their semantic information, and marked with text labels. Different from English traffic texts, the length of Chinese traffic texts are usually shorter. To more comprehensively present proposed dataset, the distribution of the lengths of text lines in the proposed dataset are provided in Table 8.3.

Resolution of Character	Training	Validation	Testing	Total
$\leq 10 \times 10$	227	38	181	446
$10 \times 10 \sim 20 \times 20$	3574	787	3973	8334
$20 \times 20 \sim 30 \times 30$	2510	502	3236	6248
$30 \times 30 \sim 40 \times 40$	871	314	1263	2448
$40 \times 40 \sim 50 \times 50$	471	157	598	1226
$\geq 50 \times 50$	739	173	815	1727
Total	8392	1971	10066	20429

Table 8.2 Details of characters from the proposed Chinese traffic Text dataset

Length of Text Line	Training	Validation	Testing	Total
1	246	59	253	558
2	1069	214	1274	2557
3	683	161	779	1623
4	589	143	709	1441
5	129	37	165	331
≥ 6	137	36	178	351
Total	2853	650	3358	6861

Table 8.3 Details of text lines from the proposed Chinese traffic text dataset

8.2.2 Evaluation Protocols

In this section, the evaluation protocols for the proposed dataset are introduced, including evaluation protocols of character and text detection.

Evaluation Protocol of Character Detection

To evaluate performance of character detection, similar evaluation protocols as PASCAL VOC object detection competition [170] are employed. Four metrics are introduced, namely *Intersection over Union (IoU)*, *Precision*, *Recall* and *Average Precision (AP)*. Multiple detections of same character are considered to be all positives.

The definition of IoU is the overlap ratio between the predicted bounding box B_p and ground-truth bounding box B_{gt} formulated as:

$$IoU = \frac{Area(B_p \bigcap B_{gt})}{Area(B_p \bigcup B_{gt})} \tag{8.1}$$

where $B_p \bigcap B_{gt}$ and $B_p \bigcup B_{gt}$ denote the intersection and union of the predicted and ground-truth bounding boxes respectively. For a predicted sample, its IoU value should be at least 0.5 for considering as a true positive sample.

The definition of *Precision* is the ratio between all true positive samples and all predicted samples, while *Recall* is defined as the ratio between all true positive samples and all ground-truth samples. The formulations of *Precision* and *Recall* are:

$$Precision = \frac{|TP|}{|D|} \tag{8.2}$$

$$Recall = \frac{|TP|}{|G|} \tag{8.3}$$

where TP, D and G are the sets of true positive samples, all detected samples and all ground-truth samples respectively.

The definition of AP is the mean precision that calculated from a set of eleven equally spaced recall levels $[0, 0.1, ..., 1]$:

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, ..., 1\}} P_{interp}(r) \tag{8.4}$$

For each recall level r, all the precisions are collected if the corresponding recalls exceed r, subsequently, the maximum precision is regarded as the output:

$$P_{interp}(r) = \underset{\tilde{r}:\tilde{r} \geq r}{max}(P(\tilde{r})) \tag{8.5}$$

where $P(\tilde{r})$ are all the precisions that have corresponding recalls exceed r.

Evaluation Protocol of Text Detection

To evaluate performance of text detection, based on the widely used STD protocols in ICDAR robust reading competitions [196–198], new character based protocols have been designed for evaluating performance on proposed dataset. Three widely used metrics are introduced, namely, *Precision*, *Recall* and $F_{measure}$.

The definition of *Precision* is the ratio between all true positive text lines and all detected text lines, while *Recall* is defined as the ratio between all true positive text lines and all ground-truth text lines. The formulations of *Precision* and *Recall* are:

$$Precision(G, D) = \frac{\sum Match(G, D)}{|D|} \tag{8.6}$$

$$Recall(G, D) = \frac{\sum Match(G, D)}{|G|} \tag{8.7}$$

$$Match(G,D) = \begin{cases} 1, & if\ pefect\ match \\ 0.5, & if\ partial\ match \\ 0, & otherwise \end{cases} \tag{8.8}$$

where D and G are the sets of all detected text lines and all ground-truth text lines. $Match(G,D)$ is a function that takes different types of matches into consideration. The details are listed as follows.

(i) Perfect match is defined as: characters between a ground-truth text line and a detected text line are exactly matched in number, linking and overlap ($IoU \geq 0.5$).

(ii) Partial match is defined as: characters between a ground-truth text line and a detected text line are partially matched in number and linking. The numbers of characters in a ground-truth (n_{gt}) and a detected (n_d) text lines should satisfy $\frac{min(n_d,n_{gt})}{max(n_d,n_{gt})} \geq 0.5$. The number of correctly detected characters ($n_c\ with\ IoU \geq 0.5$) in a ground-truth text line should satisfy $\frac{n_c}{n_{gt}} \geq 0.5$;

(iii) All the other situations are regarded as mismatch. For each ground-truth text line, the match with the largest value is found, all the rest matches are taken as false positives.

$F_{measure}$ is an overall harmonic mean that combines precision and recall. The relative weights of these are controlled by α, which is usually set to 0.5 to give equal importance to precision and recall:

$$F_{measure} = \frac{1}{\frac{\alpha}{Precision} + \frac{1-\alpha}{Recall}} \tag{8.9}$$

8.3 Approach

In this section, the proposed context-aware detection system will be introduced in detail. As Fig. 8.3 shows, the proposed system consists of three main stages, which will be presented in Sections 8.3.1, 8.3.2 and 8.3.3 respectively.

8.3.1 Character Proposal

The first stage of proposed detection system depends on the generation of character bounding boxes. In this stage, high recall is required for ensuring the success of entire system, low number of proposals are also expected for decreasing complexity and computational cost. Therefore, MSERs is firstly employed because of its advantages in high recall, high

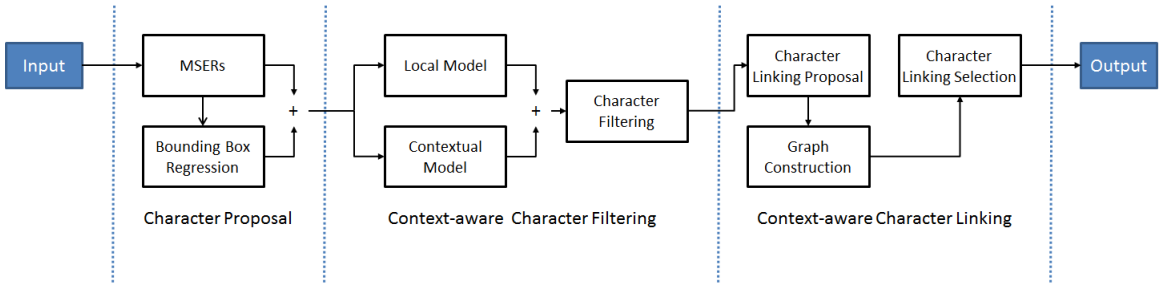


Fig. 8.3 System overview. The proposed system contains three main stages, namely character proposal, context-aware character filtering and context-aware character linking.

computational speed and strong robustness. Moreover, since there are a lot of Chinese characters that are constructed of isolated strokes, it is hard for MSERs to detect them as a whole. In order to address this problem, a bounding box regressor is also applied to adjust all the bounding boxes that generated by MSERs.

MSERs

In most images, there exist Distinguished Regions (DRs) that can be detected based on their distinguishing, invariant and stable properties [150]. As a kind of DRs, Extremal Regions (ERs) have the following desirable properties: (i) pixels in an ERs are contiguous in coordinates; (ii) pixels are closed in intensities. Further, by introducing an extremal property, namely, affinely-invariant stability, MSERs can be generated as subsets of ERs [150]. More specifically, MSERs denote a set of outstanding regions that are extremely stable inside the regions and on their outer boundary. An input image is firstly binarised using different threshold levels, and then connected components analysis is used to find all the connected components at each threshold level. Finally, the regions that maintain their shape and area at several thresholds are selected as MSERs. In this step, let $\mathbf{I}$ be an input image, a set of regions $R_M = (r_1, r_2, ..., r_n)$ are generated based on MSERs method. The bounding boxes of all the regions are denoted as $B_M = (b_1, b_2, ..., b_n)$.

Bounding Box Regression

For each bounding box $b \in B_M$, its coordinates can be defined by the positions of its top-left and bottom-right corners, denoted as $b = (x_1, y_1, x_2, y_2)$. The aim of bounding box regression is to update a new bounding box $b^* = (x_1^*, y_1^*, x_2^*, y_2^*)$ that has higher overlap with targets. Instead of directly using the absolute coordinates of bounding box for regression, the original coordinates are firstly encoded into a set of translation-invariant values in the system. Let $\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2}$ be the centre, w be the weight and h be the height of bounding

box b, then the distances from its centre to four boundaries can be calculated, denoted as (d_l, d_r, d_t, d_b). With the cropped image in b, the regressor predicts a set of scaling factors (z_l, z_r, z_t, z_b). Subsequently, a set of new distances $(d_l^*, d_r^*, d_t^*, d_b^*)$ are obtained, resulting a updated bounding box b^*. The details are illustrated in Fig. 8.4.

Fig. 8.4 The bounding box regressor adjusts the original bounding box (red) to a updated bounding box (green). The blue point is the central point of original bounding box, and the coordinate encoding scheme uses it as reference point.

The regression CNN is trained by minimising the Euclidean loss between the ground-truth scaling factors and predicted scaling factors:

$$\underset{W^R}{argmin} \sum_{i=1}^{N} L^R(z_i, f(r_i; W^R)) + \Psi(W^R) \tag{8.10}$$

$$L^R(z_i, f(r_i; W^R)) = \left\| z_i - f(r_i; W^R) \right\|_2^2 \tag{8.11}$$

where $L^R(.)$ denotes the Euclidean loss, N is the number of training examples, r_i is the input image region, $f(r_i; W^R)$ is the predicted scaling factors that mapped by parameterised function W^R, z_i is the ground-truth scaling factors and $\Psi(W^R)$ is a regularisation term.

The bounding box regressor takes all the bounding boxes $(b_1, b_2, ..., b_n) \in B_M$ as input, and produces a set of adjusted bounding boxes $B_R = (b_{n+1}, b_{n+2}, ..., b_{2n})$. Then, the bounding

boxes from both sets are collected and combined together, resulting in a new set of bounding boxes $B = (b_1, b_2, ..., b_{2n})$ and the corresponding regions $R = (r_1, r_2, ..., r_{2n})$.

8.3.2 Context-aware Character Filtering

In the second stage, two models are applied, namely local model and contextual model. The local model is a CNN based classifier that is employed for assigning character/non-character confidence scores of candidate regions. The contextual model is a Siamese network that is introduced for measuring inter-similarities between pairs of candidate regions. Based on the two models, candidate regions are filtered for removing partial non-character regions while keeping all character regions.

Local Model

The local model is a CNN that follows the architecture in [30]. For each region in set $R = (r_1, r_2, ..., r_{2n})$, the local model assigns a confidence score c, forming a set of confidence scores $C = (c_1, c_2, ..., c_{2n})$.

The local model is trained by minimising Softmax loss between the ground-truth labels and predicted labels:

$$\underset{W^L}{argmin} \sum_{i=1}^{N} L^L(l_i, f(r_i; W^L)) + \Psi(W^L) \tag{8.12}$$

$$L^L(l_i, f(r_i; W^L)) = -l_i \; log(f(r_i; W^L)) \tag{8.13}$$

where $L^L(.)$ denotes the Softmax loss, N is the number of training examples, r_i is the input image region, $f(r_i; W^L)$ is the predicted label that mapped by parameterised function W^L, l_i is the ground-truth label and $\Psi(W^L)$ is a regularisation term.

Contextual Model

The contextual model aims to jointly measure and reason about multiple regions. Following the work in [27], the contextual model is formulated as a Siamese network. This type of network has two branches that share exactly the same architecture and parameters. For a pair of image patches, each branch takes one of them as input and passes the patch through the entire network. Finally, the outputs from both branches are concatenated and measured by a similarity function. The details of a Siamese network are illustrated in Fig. 8.5.

Based on the regions in set $R = (r_1, r_2, ..., r_{2n})$, a number of m different pairs of regions can be collected, denoted as $R_S = ((r_1^1, r_1^2), (r_2^1, r_2^2), ..., (r_m^1, r_m^2))$. Let (r_i^1, r_i^2) be a paired

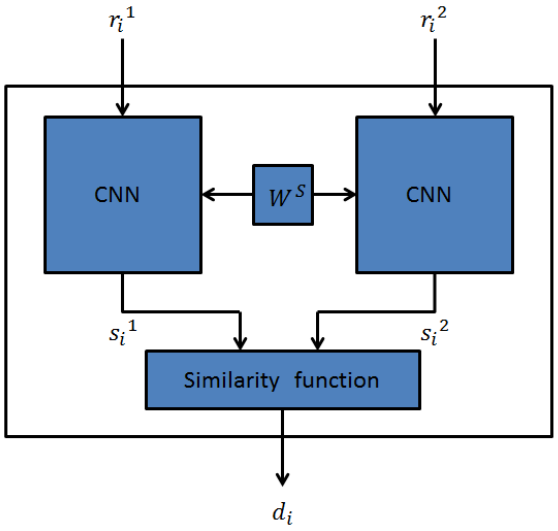


Fig. 8.5 The architecture of Siamese network

training examples, W^S be the network parameters. The employed similarity function can be formulated as:

$$d_i = \left\| f(r_i^1; W^S) - f(r_i^2; W^S) \right\|_2^2 \tag{8.14}$$

where $f(r_i^1; W^S)$ and $f(r_i^2; W^S)$ are the output values of network, and d_i is the predicted similarity. Then, the contextual model is trained by minimising Contrastive loss based on the similarity of input pairs:

$$\underset{W^S}{argmin} \sum_{i=1}^{N} L^S(y_i, d_i) + \Psi(W^S) \tag{8.15}$$

$$L^S(y_i, d_i) = (1 - y_i) d_i + y_i \, max(0, margin - d_i) \tag{8.16}$$

where $L^S(.)$ denotes the Contrastive loss, N is the number of training examples, y_i is a binary flag of the pair (0 for similar pairs, 1 for dissimilar pairs), *margin* is the minimum distance for dissimilar pairs and $\Psi(W^S)$ is a regularisation term.

For each region in set $R = (r_1, r_2, ..., r_{2n})$, the contextual model measures a value s (output of Siamese network, as indicated in Fig. 8.5), forming a set of similarity coordinates in contextual model, denoted as $S = (s_1, s_2, ..., s_{2n})$.

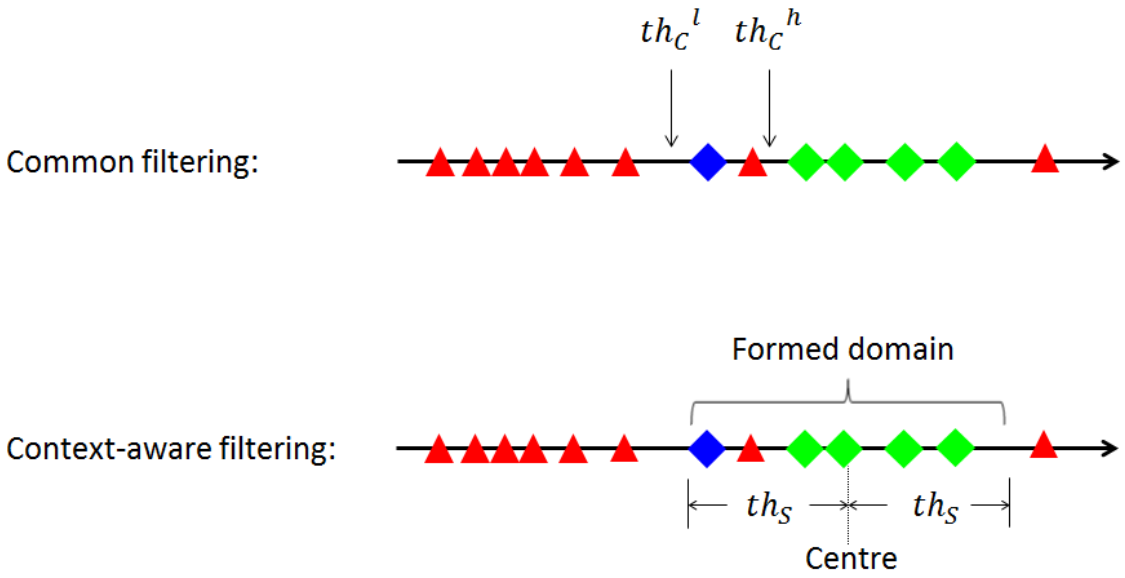


Fig. 8.6 Green diamond: positive sample with high confidence score. Blue diamond: positive sample with low confidence score. Red triangle: negative sample. There are 2 negative samples recalled if confidence scores are solely used. However, if similarity coordinates are applied, only 1 negative sample is recalled.

Character Filtering

After the previous stages, a set of bounding boxes $B = (b_1, b_2, ..., b_{2n})$, the corresponding confidence scores $C = (c_1, c_2, ..., c_{2n})$ and similarity coordinates $S = (s_1, s_2, ..., s_{2n})$ are achieved. All the essential information for character linking have been already obtained. However, due to the large number of candidate regions, a character filtering step should be performed for decreasing computational cost and complexity of the next stage.

To tackle this problem, the regions are not simply filtered based on a threshold th_C^l of C, but rather jointly use C and S. More specifically, the regions that have confidence scores higher than th_C^h are firstly selected. And then, the selected regions form a domain in the output space of contextual network. The centre is calculated from the average value of selected s, and the range th_S is determined based on the standard deviation of selected s. Finally, all the unselected regions inside this domain are also picked. The details are illustrated in Fig. 8.6.

8.3.3 Context-aware Character Linking

In the final stage, characters are linked into text line with an unified character linking method based on DAG, which detects text line by finding shortest path between characters. Three steps are involved in this stage: (i) character lining proposal for detecting all the possible

combinations of text lines; (ii) DAG for finding the shortest path of the text lines; (iii) character lining selection for selecting the final text lines.

Character Lining Proposal

Based on the assumption that all text lines are straight, all the selected regions from last stage are sorted according to their confidence scores. For each regions B, and two nearby regions A and C, their spatial and geometrical relationships of regions have been illustrated in Fig. 8.7.

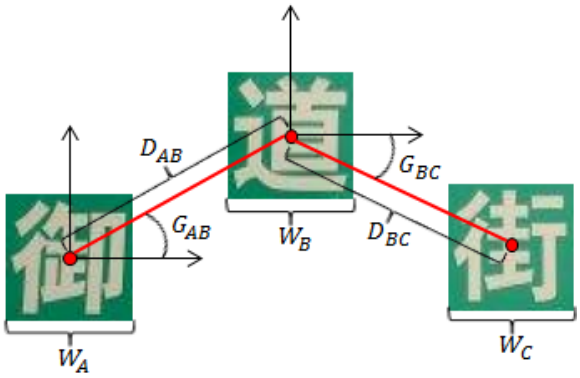


Fig. 8.7 The spatial and geometrical relationships of three regions.

In order to reduce errors and searching space, the connection of B and C is restricted by certain constraints.

(i) The regions B and C have not been connected in previous character lining.

(ii) The distance between B and C should satisfy the condition: $\frac{D(B,C)}{\min(W_B,W_C)} \leq Th_D$.

(iii) The width difference between B and C should satisfy the condition: $\frac{|W_B-W_C|}{\min(W_B,W_C)} \leq Th_W$.

(iv) The IoU difference between B and C should satisfy the condition: $IoU(B,C) \leq Th_I$.

(v) The angle difference between A, B and C should satisfy the condition: $|G(A,B) - G(B,C)| \leq Th_G$.

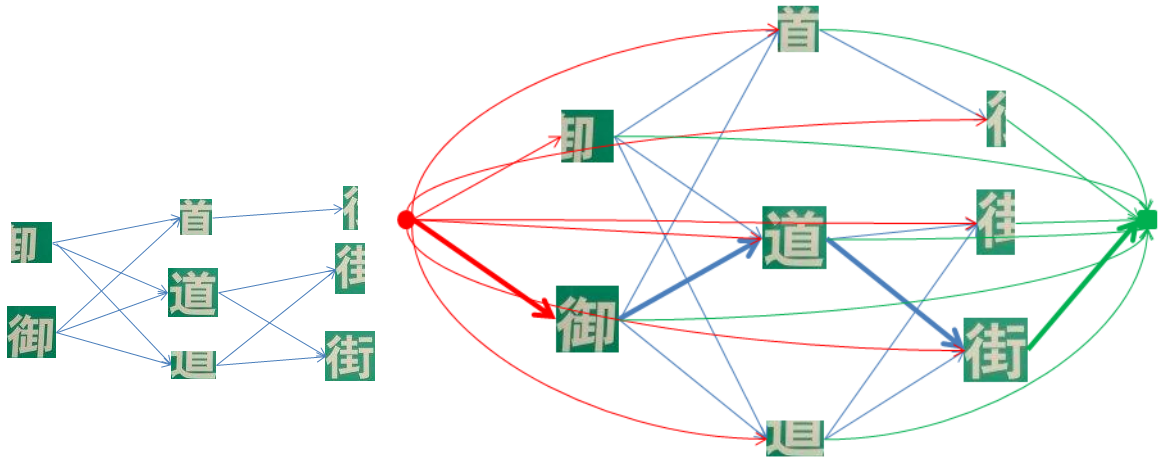
For each text line, the character lining is performed until no further character can be connected into text line. Extensive experiments on validation set show that the best trade-off between performance and searching space is under the values: $Th_D = 3, Th_W = 0.3, Th_I = 0.1, Th_G = 15°$.

Based on the above constraints, all the potential text lines or stand-alone characters can be proposed. However, background or overlapped regions are also included in the results.

In order to remove the background and overlapped regions, the following steps should be performed. In fact, with a classifier for text/no-text discrimination and Non-Maximal Suppression (NMS), the text detection is already finished for typical approaches that apply common evaluation protocol of text detection.

Directed Acyclic Graph Construction

In graph theory, the shortest path problem is the problem of finding a path between two nodes in a graph such that the sum of the weights of its constituent edges is minimised. Due to the structure of characters in straight text lines, the paths between them are directed and weighted with no directed cycles. Therefore, character linking problem is actually a shortest path problem that can be solved with weighted DAG. An example of the constructed graph is shown in Fig. 8.8.



(a) (b)

Fig. 8.8 Illustration of the directed acyclic graph construction: (a) shows the eight detected character regions (nodes) with connections (edges) in blue lines. (b) shows the constructed directed acyclic graph. The red circle is the entry node and the green rectangle is the exit node. For each region in (a), entry (red lines) and exit (green lines) edges are created with corresponding weights. For each pair of connected regions, the weights are also assigned. The final output path is shown in bold line.

The weights that have been applied in the DAGs are explained in the following. For edges between two candidate regions, the weight is composed of local, contextual and spatial weights.

The local weight is formulated as:

$$Weight_{local}(A) = -\alpha * C(A) \tag{8.17}$$

where A is a candidate region and $C(A)$ is the confidence score of A being a character region. The local weight is negatively correlated the confidence score. Therefore, higher confidence score means more negative weight, which decreases the distance of the entire path. α is a parameter that used to adjust the importance of local weight.

The contextual weight is formulated as:

$$Weight_{contextual}(A,B) = -(1+\beta * S(A,B)) * (C(A)+C(B)) \tag{8.18}$$

where A and B are two candidate regions with the similarity measured in $S(A,B)$. The contextual weight is also negative, so that both high confidence score and high similarity can result in more negative weight, which decreases the distance of the entire path. β is a parameter that used to adjust the importance of similarity.

The spatial weight is formulated as:

$$Weight_{spatial}(A,B) = \mu * D(A,B)) + \nu * W(A,B)) \tag{8.19}$$

where A and B are two candidate regions with distance $D(A,B)$ and width difference $W(A,B))$. The spatial weight is non-negative, therefore, it is large when the candidate regions are spatially far away or have very large width difference. In other words, candidate regions with large spatial weight are less likely to belong to a same text line.

Since every regions has the chance to be the starting/ending of a text line, the entry weight and exist weight are formulated as:

$$Weight_{entry}(A) = -\alpha * C(A) \tag{8.20}$$

$$Weight_{exit}(A) = 0 \tag{8.21}$$

where entry weight is the same as local weight, while exit weight equals to 0.

Finally, the text lines can be selected by finding the shortest path from entry to exit nodes. Extensive experiments on validation set show that the best performance is achieved when: $\alpha = 0.01, \beta = 0.1, \mu = 0.5, \nu = 0.2$.

Character Lining Selection

In this step, the text lines obtained in last step are further selected for generating final outputs that without overlap. Two simple rules are applied: (i) the text lines with small horizontal angles ($\leq 45^{\circ}$) and small weights are firstly selected; (ii) if two text lines are overlapping, the text line with small weight is kept, the other one is split up or removed.

8.4 Experiments

In this section, first, implementation details will be introduced, including system environment, regression network, classification network and Siamese network. Second, the results of the experiments will be presented. Finally, error evaluation will be discussed.

8.4.1 Implementation Details

System Environment

To implement the Chinese TTD system, a workstation with a Intel Xeon 3.3GHz CPU, 48GB memory and a NVIDIA GTX Titan X GPU is employed. The programs run on a 64-bit Windows 7 operating system with CUDA 7.5, CUDNNv5, Matlab 2015b and MatConvNet 1.0-beta20 [172] deep learning toolbox.

Regression Network

The bounding box regression network is constructed of 4 convolutional layers, followed by 3 fully-connected layers with 4096, 1024 and 4 neurons respectively. All the convolutional layers are normalised by BN [158], activated by ReLU, and down-sampled by a max pooling layer with kernel size 2×2 and stride 2. The inputs to all the convolutional layers are zero-padded for preserving dimensionality. Dropout layers [174] are applied after the fully-connected layers for preventing over-fitting. The details of the regression network are shown in Table 8.4.

For the training set, all of the candidate regions generated by MSERs are collected, and the candidate regions have ≥ 0.2 IoU overlap with ground-truth are selected as training samples. And then, all the samples are resized to $48 \times 48 \times 3$ RGB images. Finally, the images are pre-processed by subtracting the mean and dividing by the standard deviation. The validation set is built follow the same steps.

For the training stage, the network parameters are initialised by applying *Improved Xavier* [157]. Then, the parameters of the network are optimised by minimising the Euclidean loss between ground-truth and estimated values with SGD algorithm. The size of mini-batch B is set to 128, momentum 0.9, weight decay 0.0005 and training epoch 100. The learning rate η is initialised at 0.01, and divided by 10 if the recall on validation set maintains for 5 epoches. After the training is finished, the network with the highest recall on validation set is selected.

Classification Network

The details of the classification network are presented in Table 8.4. Different with regression network, the number of final output neurons are 2, corresponding to characters or backgrounds.

Network configurations			
Layer Index	Regression network	Classification network	Siamese network
1	Input: $48 \times 48, 3$		
2	Convolution: $3 \times 3, 64$		
3	BN		
4	ReLU		
5	Max pooling: 2×2 with stride 2		
6	Convolution: $3 \times 3, 128$		
7	BN		
8	ReLU		
9	Max pooling: 2×2 with stride 2		
10	Convolution: $3 \times 3, 256$		
11	BN		
12	ReLU		
13	Max pooling: 2×2 with stride 2		
14	Convolution: $3 \times 3, 512$		
15	BN		
16	ReLU		
17	Max pooling: 2×2 with stride 2		
18	FC: 4096		
19	BN		
20	ReLU		
21	Dropout		
22	FC: 1024		
23	BN		
24	ReLU		
25	Dropout		
26	FC: 4	FC: 2	FC: 2
27	Euclidean loss	Softmax loss	Contrastive loss

Table 8.4 Network configurations

For the training set, first, all the candidate regions from MSERs and regressor are collected. Second, all the candidate regions have ≥ 0.5 IoU overlap with ground-truth and ground-truth regions are selected as positive samples. And then, from the rest samples, two times number of negative samples to positive samples are uniformly selected. Finally, all the samples are

resized to $48 \times 48 \times 3$ RGB images, followed by subtracting the mean and dividing by the standard deviation. The validation set is built follow the same steps.

For the training stage, the network parameters are initialised by applying *Improved Xavier* [157]. Then, the parameters of the network are optimised by minimising the softmax loss between target labels and predicted labels with SGD algorithm. The size of mini-batch B is set to 128, momentum 0.9, weight decay 0.0005 and training epoch 100. The learning rate η is initialised at 0.1. Early stopping is employed for selecting the best model.

Siamese Network

The details of the Siamese network are shown in Table 8.4. The network is similar to classification network, except the loss function is contrastive loss instead of softmax loss.

For the training set, in each iteration, similar pairs (two positive samples which have ≥ 0.5 IoU overlap with ground-truth) and dissimilar pairs (one positive sample and one negative sample) are randomly and uniformly sampled from the training images. And then, all the samples are resized to $48 \times 48 \times 3$ RGB images, followed by subtracting the mean and dividing by the standard deviation. The validation set is built follow the same steps.

For the training stage, the Siamese network comprises two identical sub-networks, one sub-network is firstly initialised by applying *Improved Xavier* [157], and then all the parameters are copied to the other one. Subsequently, pairs of images are passed through the sub-networks, yielding pairs of outputs which are further passed to the contrastive loss. The gradients of both sub-networks are computed using back-propagation. The network parameters are updated with SGD based on the average gradients of the two sub-networks. The size of mini-batch B is set to 128, momentum 0.9, weight decay 0.0005, training iteration 100000, and margin value 1. The learning rate η is initialised at 0.01. After the training is finished, the network with best performance on validation set is selected.

8.4.2 Performance Evaluation of Character Proposal

The performance of the proposed character proposal method is presented in Fig. 8.9. The recall rates for characters are illustrated with the changes of IoU between the range $[0.1, 1]$. With an IoU value of 0.5, the recall rates for MSERs, regression and the combination of the two methods are 93.8%, 97.3% and 97.7% respectively. Apparently, the regression method indeed improves the performance of MSERs, and the combination of the two methods shows further boosted performance.

The performance the proposed method are compared with the five widely used methods, including MSERs, Edge Boxes ($\alpha = 0.65$ and $\beta = 0.75$), Selective Search, Stroke Width

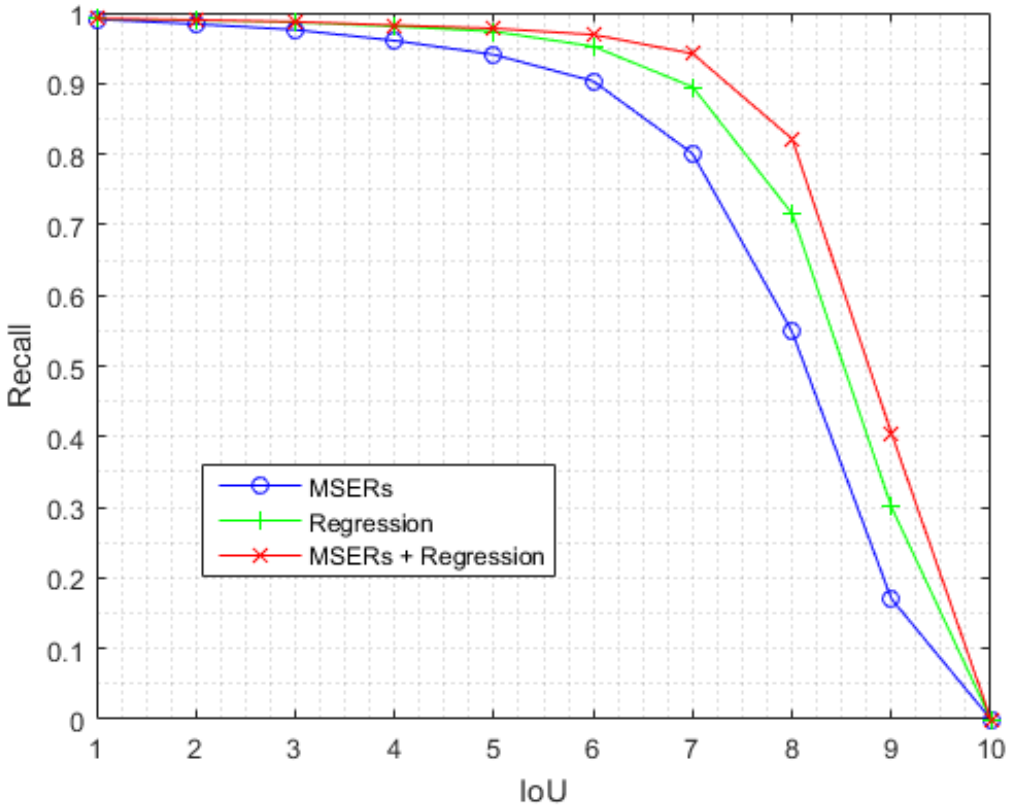


Fig. 8.9 Performance of the proposed character proposal method. The recall rates are illustrated with the changes of IoU between the range $[0.1, 1]$.

Type	Average number of proposals / image
Ours	964
MSERs	482
Edge Boxes	4000
Selective Search	3574
SWT	340
AdaBoosting	5000

Table 8.5 Average number of proposals per image

Transform (SWT) and AdaBoosting, with an IoU value of 0.5, the recall rates are 97.7%, 93.8%, 52.7%, 57.3%, 60.3% and 72.3% respectively. The curve of recall rates versus IoUs are illustrated in Fig. 8.10. The proposed method outperforms these widely used methods. The average numbers of proposals of all the methods are presented in Table 8.5. The results in Table 8.5 indicate that the number of proposals generated by proposed method is reasonable while it outperforms other methods in recall rate.

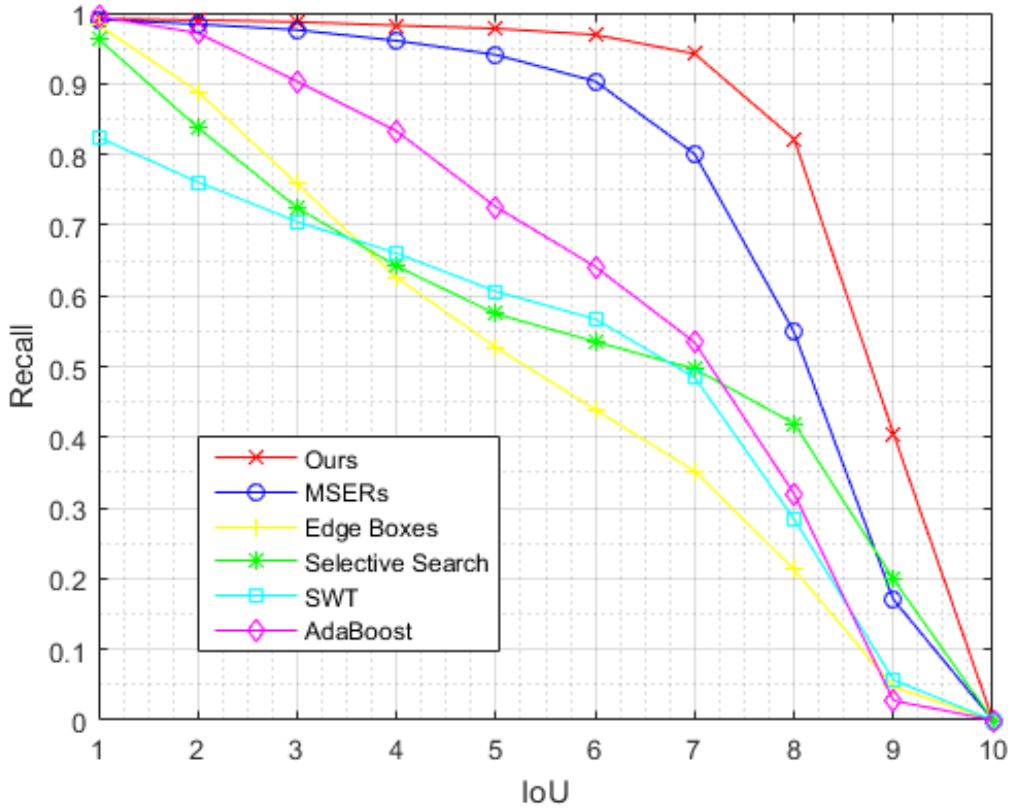


Fig. 8.10 Performance comparison of the character proposal method with the five widely used methods, including MSERs, Edge Boxes, Selective Search, Stroke Width Transform and AdaBoosting. The recall rates are illustrated with the changes of IoU between the range $[0.1, 1]$.

8.4.3 Performance Evaluation of Character Filtering

The Precision - Recall (PR) curves of the classifiers are illustrated in Fig. 8.11. Two experiments are set up for providing a comprehensive evaluation. The Fig. 8.11 shows that the sole use of local model presents good performance when high confidence scores are used for prediction. However, with the decreasing of the confidence score, the performance

of local model is decreased due to the factors such as: (i) there are thousands of different Chinese characters contained in traffic text, lots of them are not involved in training set; (ii) capturing angle, colour fading and partial covering make the classification very difficult. On the other hand, the contextual model directly measures the similarities of two inputs, the robustness to unseen and broken characters is better than local model. Therefore, the local and contextual models are jointly used in the proposed system. The APs of the sole model and the joint models are 93.9% and 93.4% respectively.

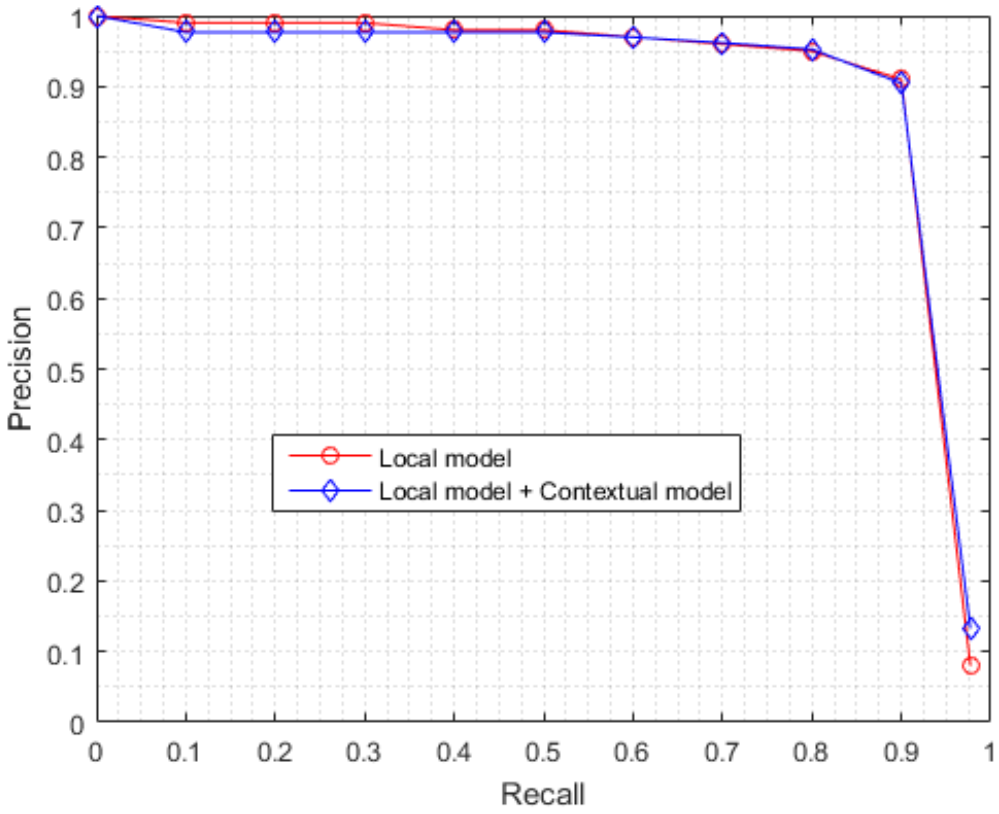


Fig. 8.11 Illustration of the precision - recall curves. Red line: sole use of local model. Blue line: joint use of local and contextual models.

Type	Recall	Precision
Input	97.7%	6.7%
Local model filtering	97.7%	8.0%
Context-aware filtering	97.7%	12.4%

Table 8.6 Performance of character filtering

The recall and precision rates for input and filtered sets are shown in Table 8.6. Without missing character regions, the proposed method increase the precision from 6.7% to 12.4%. In other words, the size of filtered set is only about 54.1% of input set. Therefore, the computational cost and complexity of next stage will be largely reduced.

8.4.4 Performance Evaluation of Character Linking

The performance of the DAG based character linking method is presented in Table 8.7, followed by a set of experiments that are implemented based on different character linking methods: (i) simple perceptual similarity rules [102]; (ii) pairing candidates and aggregating pairs with similar orientations [115, 116, 135]; (iii) text flow [149].

Type	Recall	Precision	$F_{measure}$
Character linking proposal (Ours)	95.5%	10.3%	18.7%
Directed acyclic graph (Ours)	90.4%	9.78%	17.7%
Final results (Ours)	89.2%	58.2%	70.3%
Text Flow [149]	82.9%	60.1%	69.7%
Simple similarity rules [102]	73.3%	53.5%	61.8%
Orientation pairing [115, 116, 135]	78.2%	51.4%	62.1%

Table 8.7 Performance of character linking

The results in Table 8.7 indicates that the proposed approach is indeed effective. In character linking proposal stage, the applied method achieves 95.5% recall rate. After region refinement based on directed acyclic graph, the recall rate reduced to 90.4%, the reduced recalls are mainly partial matches, which are caused by inaccurate path selection. Finally, the character linking method achieves 89.2% recall and 58.2% precision rates, resulting in 70.3% $F_{measure}$, which outperforms the existing character linking method.

8.4.5 Error Evaluation

Some examples of successful or failed detection are illustrated in Fig. 8.12. The results in Fig. 8.12 indicate that the proposed system is able to detect horizontal, non-horizontal, vertical traffic or even crossed traffic texts. However, failed detection exists in some cases, such as the characters from several texts are very close or characters from one text are far from each other. In order to address these problems, traffic text recognition is essential. With the help of semantic meanings of the characters, it will be much easy to link them together.

Fig. 8.12 Illustration of examples of detection results. Blue rectangles: successfully detected characters, red rectangle: missed characters, green lines: successfully linked text lines, red lines: incorrectly lined text lines.

8.5 Summary

In this chapter, a context-aware Chinese TTD system is proposed, the contributions include: (i) the construction of a new Chinese traffic text dataset. With the help of this dataset, the gap of Chinese traffic text detection and recognition will be filled; (ii) a region proposal method that jointly uses of MSERs and a CNN based regressor to hypothesise the character locations. This method boosts performance while characters are constructed of isolated components, blurred or partially covered; (iii) a classification method that uses both information from a local model and a contextual model. The combination of local and contextual model allows more accuracy discrimination, especially in the detection of stand-alone and distant characters that are contained in Chinese traffic texts; (iv) an unified character linking method based on DAG. The problem of text detection is then formulated into a task of finding the shortest path, and this method also allows to detect non-horizontal and crossed text lines. Extensive experiments have been performed, and good results have been achieved on the stages, including character proposal, character filter and character linking.

Chapter 9

Pedestrian Attribute Classification with Convolutional Neural Networks and Feature Selection

Visual attribute classification has been widely discussed due to its impact on lots of applications, such as face recognition, action recognition and scene representation. Recently, Convolutional Neural Network (CNN)s have demonstrated promising performance in image recognition and object detection. Such networks are able to automatically learn a hierarchy of discriminate features that richly describe image content. However, dimensions of features of CNNs are usually very large. In this chapter, a pedestrian attribute classification system is proposed based on CNNs and a novel feature selection algorithm. Experiments have been conducted using the Berkeley Attributes of People dataset. The best overall Mean Average Precision (MAP) is about 89.2%.

9.1 Introduction

Visual attributes are human-nameable properties (e.g., *is male*, *wear hat* and *wear glasses*) that are discernible in images or videos, and they are crucially important to solving various vision problems. For example, scene representation proposed in [199] characterises target scene by a series of attributes rather than a single label which is too restrictive to describe the properties of a scene. In [200], the face verification problem is reformulated as the recognition of the presence or absence of describable aspects of visual appearance. Different formulations of attributes have been proposed, including binary attributes and relative attributes [201]. Binary attributes are used to represent the presence of certain properties. Relative attributes

describe the relative strength of each attribute, which are closer to the ways that human describe and compare objects in real world.

For computer vision tasks, feature representation is a critical factor that affects system performance. The problem of extracting discriminative and representative features has been profoundly researched in the past decades. Recently, deep learning have achieved huge popularity. In particular, the success achieved by Krizhevsky et al. [29] on the ILSVRC-2012 image classification benchmark, leading a new way of applying CNNs to tasks like image recognition and object detection. The works in [29, 30] show that CNNs are able to automatically learn discriminative feature representation. With the help of region proposals, CNN based object detection [33, 34] is significantly developed. And CNNs are also widely applied in action recognition [202] and scene representation [199].

However, dimensions of features of CNNs are usually very large, and there are irrelevant elements included in features of CNNs. Therefore, feature selection could be exploited to remove the irrelevant or redundant features of CNNs. Feature selection is an important method in many computer vision tasks. The motivations include: (i) reducing feature dimension; (ii) improving classification performance; (iii) decreasing computational cost. Feature selection methods can be divided into two categories, which are supervised feature selection method and unsupervised feature selection method. Supervised feature selection methods [203, 204] reduce feature dimensions based on correlation information between features and labels. On the other hand, unsupervised feature selection methods [205, 206] select features mainly based on similarity preserving or clustering. However, due to the lack of label information, feature selection should be performed without guide of classification accuracy, leading particular challenge in unsupervised feature selection.

In this chapter, a pedestrian attribute classification system will be presented. Inspired by R*CNN [202] that aims to improve classification accuracy by introducing secondary regions, the proposed system focus on exploring the relationships between different portions of features with regard to visual attribute tasks, two main contributions are included as follows.

(i) A pre-trained CNN model to extract discriminative feature representation.

(ii) A novel feature selection algorithm to remove irrelevant or redundant features of CNNs, resulting in reduced feature dimension, improved classification performance and decreased computational cost.

The rest of this chapter is organised as follow: Section 9.2 gives a detailed introduction of the proposed visual attribute classification system. Experimental results will be introduced in Section 9.3, followed by summary in Section 9.4.

9.2 Approach

The proposed visual attribute classification system mainly includes three stages: (i) feature extraction; (ii) feature selection; (iii) classification, as Fig. 9.1 illustrates. The modules will be detailed in the following sections.

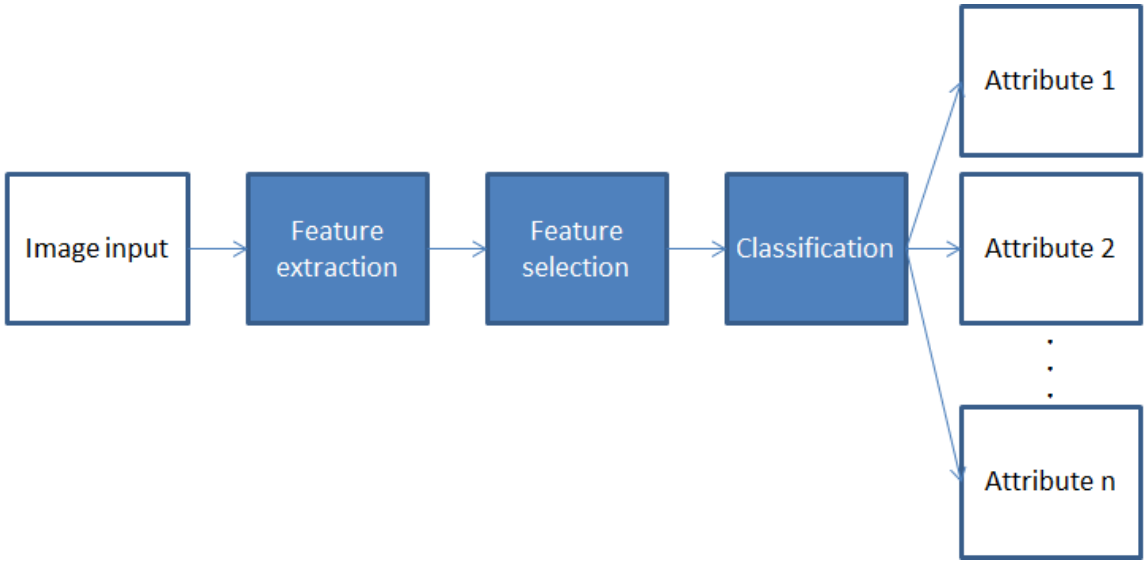


Fig. 9.1 System overview.

9.2.1 Feature Extraction

In this stage, a CNN is applied for feature extraction, and the applied model is based on the networks in Fast R-CNN [34] and R*CNN [202]. The network is built based on the 16-layer architecture (VGG16) [30], which have demonstrated outstanding performance in image classification and object detection. Since only the features before fully-connected layers are used, the last layer of the network is the Region of Interest (RoI) pooling layer [34] in proposed system. The detailed network configurations are illustrated in Table 9.1.

9.2.2 Feature Selection

In feature selection stage, first, the features of all images at the RoI pooling layer are collected. The RoI pooling layer is a kind of adaptive max pooling layer, the size (7×7 in the system) of its output feature maps are fixed whatever the size of inputs. Therefore, the size of extracted features for each input image is $7 \times 7 \times 512$ (25088 in total). Then, feature selection is performed using proposed algorithm. More specifically, for each attribute classifier, the detailed procedure are listed as follows.

(i) Data collection. For the training attribute, all related images in training set are divided into two classes (positive and negative) based on labels.

(ii) Data processing. In order to measure the similarities of features belonging to the same class, all of the features are transformed into binary arrays using a threshold value. Since the activation function of the last convolutional layer (weight layer 13) is ReLU [173], only values larger than 0 are able to pass to next layer. In other words, feature of each position (25088 in total) is activated if the value is larger than 0. Therefore, 0 is adopted as the threshold value in this step. And then, all of the binary arrays from same class are accumulated together and normalised by dividing by the number of images. In this manner, for each class, a array that indicate the probability of activation of each feature position can be obtained. Finally, two probability arrays are achieved, namely, $p_{positive}$ and $p_{negative}$.

(iii) Feature selection. In this step, feature selection is performed by comparing the magnitude of the probability of each position in $p_{positive}$ and $p_{negative}$. First, a distance vector can be computed based on $|p_{positive} - p_{negative}|$. Second, the vector is sorted according to its magnitude. Finally, given a desired dimension n, the original 25088 feature can be reduced to n by simply select the positions that contain top n largest values in vector.

9.2.3 Classification

In classification stage, linear SVMs are employed [207]. Classification of SVMs are performed by constructing maximum-margin hyper-plane. With the selected features extracted from the previous stage, linear SVMs are trained to discriminate between presence or absence for each attribute.

9.3 Experiments

In this section, first, the implementation details will be introduced. Second, the Berkeley Attributes of People Dataset will be illustrated. Then, the experimental results of proposed system will be presented and discussed followed by the performance comparison. Finally, error evaluation will be presented.

9.3.1 Implementation Details

System Setup

To implement the system, a laptop with an Intel Xeon E3-1231 V3 CPU, 32GB memory and a NVIDIA GTX 970m GPU is employed. The program runs on a 64-bit Open-source Linux operating system with CUDA 7.5, Python 2.7.3, Matlab 2014b and Caffe deep learning platform installed.

ConvNet Configurations		
Weight layer	VGG16	Applied
Input		
1	Convolution $3 \times 3, 64$	Convolution $3 \times 3, 64$
2	Convolution $3 \times 3, 64$	Convolution $3 \times 3, 64$
Max pooling		
3	Convolution $3 \times 3, 128$	Convolution $3 \times 3, 128$
4	Convolution $3 \times 3, 128$	Convolution $3 \times 3, 128$
Max pooling		
5	Convolution $3 \times 3, 256$	Convolution $3 \times 3, 256$
6	Convolution $3 \times 3, 256$	Convolution $3 \times 3, 256$
7	Convolution $3 \times 3, 256$	Convolution $3 \times 3, 256$
Max pooling		
8	Convolution $3 \times 3, 512$	Convolution $3 \times 3, 512$
9	Convolution $3 \times 3, 512$	Convolution $3 \times 3, 512$
10	Convolution $3 \times 3, 512$	Convolution $3 \times 3, 512$
Max pooling		
11	Convolution $3 \times 3, 512$	Convolution $3 \times 3, 512$
12	Convolution $3 \times 3, 512$	Convolution $3 \times 3, 512$
13	Convolution $3 \times 3, 512$	Convolution $3 \times 3, 512$
	Max pooling	RoI pooling
14	FC 4096	
15	FC 4096	
16	FC 1000	
	Soft-max	

Table 9.1 Network configurations of CNNs

Network Configurations

The network details are illustrated in Table 9.1. In the training stage, the CNN started from a model [202] initialised with discriminative pre-training for the Berkeley Attributes of People dataset [208], and fine-tuning is not performed for the CNN. For each sample in the dataset,

only the information provided from the ground-truth region is used for the tasks of attributes classification.

9.3.2 Berkeley Attributes of People Dataset

The Berkeley Attributes of People dataset [208] contains 8035 images with at least a full body of a person included. 9 attributes are provided, and the detail distribution of labels are illustrated in Table 9.2. Some examples from the dataset have been shown in Fig. 9.2.

	Positive	Negative
Is Male	3395	2365
Has Long Hair	1456	3361
Has Glasses	1238	4083
Has Hat	1096	5532
Has T-Shirt	1019	3350
Has Long Sleeves	3045	3099
Has Shorts	477	2020
Has Jeans	771	1612
Has Long Pants	2020	760

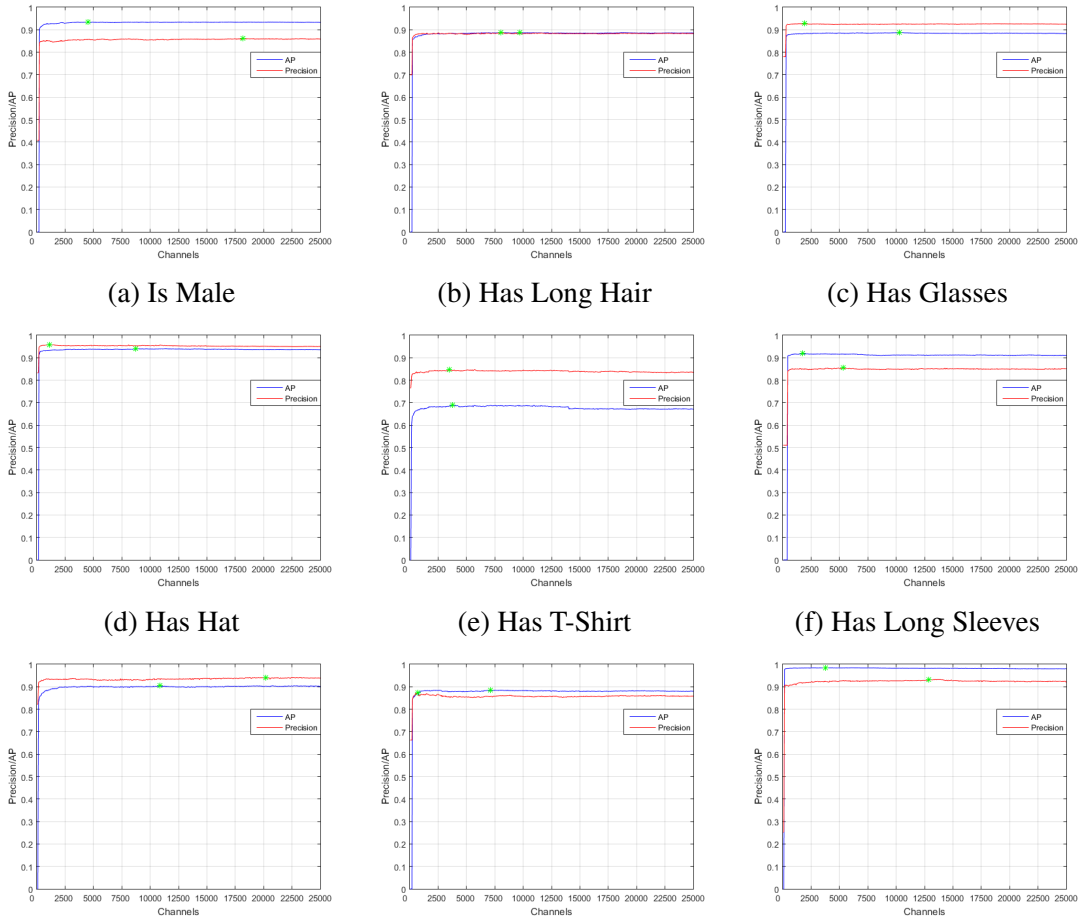
Table 9.2 Number of positive and negative labels of Berkeley Attributes of People Dataset.

9.3.3 Experimental Results

Performance Evaluation of Feature Selection

The objective of proposed feature selection method is to remove the redundant or irrelevant parts of the features. Thus, the curves of classifying performance versus the numbers of selected features (channels) are presented, as illustrated in Fig. 9.3. All the attributes in the test set of the Berkeley Attributes of People dataset are included. The sizes of selected features are set from 50 to 25088 with a step size of 50. Two measurement parameters are employed for evaluating the system performance, namely Average Precision (AP) and *precision*. The highest values of *AP* and *precision* for each attribute task are highlighted with green stars. As Fig. 9.3 indicates, the classifying performance increases as the selected feature dimension (< 2500) increases, which means the discriminant part of the features have been selected. Subsequently, the classifying performance becomes nearly stable regardless of the increasing of feature dimension, which means the rest features are not such relevant to the corresponding tasks. Therefore, the experiments show that the proposed methods can reduced the feature dimension effectively.

Fig. 9.2 Examples from the Berkeley Attributes of People Dataset. The persons in question are highlighted with a red box.



(a) Is Male (b) Has Long Hair (c) Has Glasses

(d) Has Hat (e) Has T-Shirt (f) Has Long Sleeves

(g) Has Shorts (h) Has Jeans (i) Has Long Pants

Fig. 9.3 Illustration of *AP − channel* and *precision − channel* curves of the attribute classifiers on the test set. The sizes of selected features are set from 50 to 25088 with a step size of 50, and the highest values of *AP* and *precision* for each attribute task are highlighted with green stars.

Performance Comparison

	Is Male	Has Long Hair	Has Glasses	Has Hat	Has T-Shirt	Has Long Sleeves	Has Shorts	Has Jeans	Has Long Pants	MAP
Fast R-CNN	91.8	88.9	81.0	90.4	73.1	90.4	88.6	88.9	97.6	87.8
Ours (500)	91.8	86.9	87.9	93.0	66.2	91.0	87.2	86.4	98.1	87.6
Ours (1500)	92.7	88.2	88.2	93.5	67.5	91.7	89.0	88.1	98.3	88.6
Ours (2500)	92.9	88.2	88.4	93.6	68.3	91.6	89.9	88.3	98.4	88.8
Ours (3500)	93.3	88.4	88.6	93.8	68.5	91.7	89.9	87.8	98.4	88.9
Ours (4500)	93.4	88.3	88.5	93.8	68.2	91.6	90.2	87.9	98.4	88.9
Ours (highest)	**93.4**	88.7	**88.7**	**94.0**	68.9	91.9	90.5	88.5	98.4	89.2
Poselet [208]	82.4	72.5	55.6	60.1	51.2	74.2	45.5	54.7	90.3	65.2
PANDA [202]	91.7	82.7	70.0	74.2	49.8	86.0	79.1	81.0	96.4	79.0
Gkioxari et al.[202]	92.9	**90.1**	77.7	93.6	72.6	**93.2**	**93.9**	**92.1**	**98.8**	**89.5**
R*CNN [202]	92.8	88.9	82.4	92.2	**74.8**	91.2	92.9	89.4	97.9	89.2

Table 9.3 Performance of the Berkeley Attributes of People test set.

Table 9.3 shows the comparative results of all the pedestrian attributes in the Berkeley Attributes of People dataset. Since the pre-trained model from [202] is applied in proposed system, the performance of Fast R-CNN is shown as the baseline, followed by the results of proposed method with feature selection dimensions (500, 1500, 2500, 3500 and 4500). Comparing with the baseline method, the proposed method achieves better performance on the most of tasks. In particular, the tasks *Is Male*, *Has Glasses*, *Has Hat* and *Has Long Sleeves* perform obviously better. The MAP seems to stable at about 88.8% after the feature selection dimension larger than 2500, which means the proposed method can effectively reduce the feature dimension from 25088 to 2500 without loss of performance.

The proposed method is also compared to other approaches. As Table 9.3 indicates, the highest results are collected for each task under different feature selection dimensions, and the proposed method obtains the best performance in the tasks *Is Male*, *Has Glasses* and *Has Hat*. The maximum MAP is about 89.2%, which is the same as the result obtained by R*CNN.

Error Evaluation

Although the proposed pedestrian attribute classification system has demonstrated boosted performance on most attributes, some of the attributes are still hard to discriminate, such as *Has Shorts* and *Has T-Shirt*. As Fig. 9.3e elaborated, the classification performance of the attribute *Has T-Shirt* is especially low, the AP value is stabled at about 68.5%, which is much lower than the corresponding precision value 84.3%. Some of false examples are given in Fig. 9.4.

The main reasons cause false predictions include: (i) low resolution of images; (ii) partial occlusions; (iii) huge appearance variations; (iv) the representative ability of features extracted with the applied CNN are not such discriminative. Even though the proposed feature selection method is able to improve classification performance by removing irrelevant or redundant features, the system performance will be influenced if the features are extracted using under-fitting models. Comparing to the baseline method, some of the attributes show decreased performance, which indicates that MLP could achieve better performance than SVM for these tasks.

9.4 Summary

This chapter proposes a visual attribute classification system based on CNNs and feature selection, with main contributions including: (i) a pre-trained CNN model to extract discriminative feature representation; (ii) a novel feature selection algorithm to remove irrelevant

Fig. 9.4 Illustration of examples where the prediction failed.

or redundant features, resulting in reduced feature dimension, improved classification performance and decreased computational cost. By introducing the proposed method, feature dimensions can be reduced, and system achieves higher accuracy and lower computational cost. Experiments have been conducted on the Berkeley Attributes of People test set, and competitive results have been achieved.

Chapter 10

Conclusions and Future Work

In this chapter, the conclusions of this thesis will be presented in the first, followed by possible research directions for future works.

10.1 Conclusions

The objectives of this thesis is to evaluate, develop and apply computer vision and deep learning methods, techniques and algorithms for traffic scene objects detection and recognition. Three objects are mainly focused, including traffic sign, traffic text and pedestrian. For traffic sign, CNNs with the most influencing CNN architectures are trained based on different training algorithms, such as weight initialisation, regularisation and optimisation. And then, CNNs are trained based on proposed methods, and improved performance is achieved. For traffic text, Chinese and English license plates are firstly detected with proposed method. And then, the Chinese texts on traffic panels are detected based on CNNs and DAG. For pedestrian, the attributes of pedestrian are classified based on CNNs and a novel feature selection algorithm. The detail discussion will be presented in the following.

In **Chapter 3**, a real-time system for TSD is proposed by template matching with features based a new feature MCCH). Comparing with classical features such as HOG, the feature dimension of MCCH is less while the representation ability is comparative. Therefore, MCCH has lower computational cost, and it is very suitable in real-time applications. Followed by the detection of traffic signs over roads, traffic signs painted on road surface are detected in **Chapter 4**, the proposed TSD system applies a hybrid region proposal method and a CNN. The features of CNN are more discriminative than the simple features such as SIFT, HOG and MCCH. Therefore, the system accuracy of CNN based methods are higher than traditional methods. Moreover, in order to improve detection speed and accuracy, the famous Fast R-CNN framework is applied. Fast R-CNN takes whole images as input, and predicts outputs

based on multi-task networks, which improve system accuracy and decrease computational cost.

In **Chapter 5**, traffic signs are recognised with the most influencing CNN architectures, including the very deep networks, residual networks and dense networks. Extensive experiments are presented and discussed based on different training algorithms, such as weight initialisation, regularisation and optimisation. Then, in **Chapter 6**, four TSR systems are proposed based on new strategies or constructed by new network architectures. First, different from the most existing approaches that apply single level classifiers for TSR, a hierarchical classification system is applied for improving the performance of TSR. Second, unsupervised feature learning is a challenge in computer vision. By introducing GANs, a TSR system is proposed based on features that are learnt without labels. Third, in comparison with previous CNNs that only apply max pooling for reducing feature dimension, a novel layer is proposed, which applies MPPs of max pooling as information for sparse feature learning. The sparse layer is able to extract the important part of features based on different tasks, so that the classification performance of CNNs is improved. Finally, in order to overcome gradient vanishing and degradation problems, a novel CNN architecture design that learns with multiple loss functions is developed. With the help of this design, the gradient vanishing and degradation problems are eased. Comparing with residual learning, the proposed design has better performance in shallow networks.

In **Chapter 7**, a system for detecting Chinese and English license plates is proposed. Comparing with existing methods, it is more robust and able to detect Chinese and English license plate with large skew angles. Then, a context-aware Chinese TTD system is proposed in **Chapter 8** with three advantages. First, the proposed system jointly applies MSERs and a regressor as region generator to ensure high recall. Second, all of the generated regions are not only assigned with confidence scores for character/non-character classification, but also inter-similarities for context-aware detection. Finally, characters are refined and linked into texts by applying an unified character linking method, which is based on DAG. With the help of these improvements, the proposed system can handle non-horizontal text lines and crossed text lines, while some existing text detection systems only are able to detect horizontal or near horizontal text lines. The error accumulation problem that usually occurs in the systems that use individual and sequential stages is also avoided.

In **Chapter 9**, a pedestrian attribute classification system is proposed based on CNNs and a novel feature selection algorithm. CNNs have demonstrated promising performance in image recognition and object detection. Such networks are able to automatically learn a hierarchy of discriminate features that richly describe image content. However, dimensions of features of CNNs are usually very large, and there are irrelevant elements included in

features of CNNs. Comparing with R*CNN that aims to improve classification accuracy by introducing secondary regions, the proposed system focus on exploring the relationships between different portions of features with regard to visual attribute tasks.

10.2 Future Works

In this thesis, there are different directions for future work based on existing outcomes. Some of the directions are listed as follows.

(i) In **Chapter 6**, a novel layer is proposed by applying MPPs of max pooling as information for sparse feature learning. Although excellent results have been achieved, the existing sparse layer is not able to train with entire network together. It would be interesting to improve the sparse layer for training the entire network as a whole.

(ii) In **Chapter 6**, a novel CNN architecture design that learns with multiple loss functions is developed. Although improved results have been achieved, the problems of gradient vanishing and degradation are not totally addressed. It would be interesting to further improve the architecture of CNNs.

(iii) In **Chapter 8**, a context-aware Chinese TTD system is proposed by applying CNNs and DAG. Although excellent results have been achieved, the task of traffic text recognition is not addressed. Traffic text could be recognised by Recurrent Neural Network (RNN) or Long Short-Term Memory (LSTM). With the result of traffic text recognition, the system performance can be further improved. However, due to the large number of Chinese characters, more than 5000 classes of characters are involved. It will be interesting and challenging.

References

[1] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[2] K. Bengler, K. Dietmayer, B. Farber, M. Maurer, C. Stiller, and H. Winner, "Three decades of driver assistance systems: Review and future perspectives," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, pp. 6–22, winter 2014.

[3] M. M. Trivedi, T. Gandhi, and J. McCall, "Looking-in and looking-out of a vehicle: Computer-vision-based enhanced vehicle safety," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, pp. 108–120, March 2007.

[4] J. C. McCall and M. M. Trivedi, "Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, pp. 20–37, March 2006.

[5] D. Geronimo, A. M. Lopez, A. D. Sappa, and T. Graf, "Survey of pedestrian detection for advanced driver assistance systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 1239–1258, July 2010.

[6] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, pp. 1773–1795, Dec 2013.

[7] L. Li, D. Wen, N. N. Zheng, and L. C. Shen, "Cognitive cars: A new frontier for adas research," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, pp. 395–407, March 2012.

[8] E. D. Dickmanns, B. Mysliwetz, and T. Christians, "An integrated spatio-temporal approach to automatic visual guidance of autonomous vehicles," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, pp. 1273–1284, Nov 1990.

[9] D. Pomerleau and T. Jochem, "Rapidly adapting machine vision for automated vehicle steering," *IEEE Expert*, vol. 11, pp. 19–27, Apr 1996.

[10] U. Franke, D. Gavrila, S. Gorzig, F. Lindner, F. Puetzold, and C. Wohler, "Autonomous driving goes downtown," *IEEE Intelligent Systems and their Applications*, vol. 13, pp. 40–48, Nov 1998.

[11] M. Bertozzi, A. Broggi, and A. Fascioli, "Vision-based intelligent vehicles: State of the art and perspectives," *Robotics and Autonomous Systems*, vol. 32, no. 1, pp. 1–16, 2000.

[12] M. Bertozzi, A. Broggi, M. Cellario, A. Fascioli, P. Lombardi, and M. Porta, "Artificial vision in road vehicles," *Proceedings of the IEEE*, vol. 90, pp. 1258–1271, July 2002.

[13] F. Heimes and H.-H. Nagel, "Towards active machine-vision-based driver assistance for urban areas," *International Journal of Computer Vision*, vol. 50, pp. 5–34, Oct 2002.

[14] W. Enkelmann, "Video-based driver assistance–from basic functions to applications," *International Journal of Computer Vision*, vol. 45, pp. 201–221, Dec 2001.

[15] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 743–761, April 2012.

[16] C. Guo, J. Meguro, Y. Kojima, and T. Naito, "A multimodal adas system for unmarked urban scenarios based on road context understanding," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, pp. 1690–1704, Aug 2015.

[17] S. Di, H. Zhang, C. G. Li, X. Mei, D. Prokhorov, and H. Ling, "Cross-domain traffic scene understanding: A dense correspondence-based transfer learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, no. 99, pp. 1–13, 2017.

[18] F. Zaklouta, B. Stanciulescu, and O. Hamdoun, "Traffic sign classification using k-d trees and random forests," in *IEEE International Joint Conference on Neural Networks*, pp. 2151–2155, July 2011.

[19] P. Sermanet and Y. LeCun, "Traffic sign recognition with multi-scale convolutional networks," in *IEEE International Joint Conference on Neural Networks*, pp. 2809–2813, July 2011.

[20] D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber, "A committee of neural networks for traffic sign classification," in *IEEE International Joint Conference on Neural Networks*, pp. 1918–1921, July 2011.

[21] B. Gecer, G. Azzopardi, and N. Petkov, "Color-blob-based cosfire filters for object recognition," *Image and Vision Computing*, vol. 57, pp. 165–174, 2017.

[22] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, vol. 32, pp. 323–332, 2012. Selected Papers from {IJCNN} 2011.

[23] J. Jin, K. Fu, and C. Zhang, "Traffic sign recognition with hinge loss trained convolutional neural networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, pp. 1991–2000, Oct 2014.

[24] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.

[25] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 5786, p. 504, 2006.

[26] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, pp. 1527–1554, July 2006.

[27] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 539–546, June 2005.

[28] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 1735–1742, 2006.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, June 2015.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016.

[33] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 580–587, June 2014.

[34] R. Girshick, "Fast r-cnn," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, Dec 2015.

[35] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, June 2017.

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

[37] M. Sonka, V. Hlavac, and R. Boyle, *Image processing, analysis, and machine vision*. Cengage Learning, 2014.

[38] B. Jahne, H. Haussecker, and P. Geissler, *Handbook of computer vision and applications*, vol. 2. Academic Press San Diego, 1999.

[39] D. Forsyth and J. Ponce, *Computer vision: a modern approach*. Upper Saddle River, NJ; London: Prentice Hall, 2011.

[40] R. Jain, R. Kasturi, and B. G. Schunck, *Machine vision*, vol. 5. McGraw-Hill New York, 1995.

[41] R. C. Gonzalez and R. E. Woods, *Digital image processing*. Prentice-Hall, 2008.

[42] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[43] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 1798–1828, Aug. 2013.

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

[45] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 886–893, June 2005.

[46] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, Nov 2004.

[47] H. Bay, T. Tuytelaars, and L. Van Gool, *SURF: Speeded Up Robust Features*, pp. 404–417. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[48] P. Dollar, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, pp. 1532–1545, Aug 2014.

[49] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[50] P. Viola and M. Jones, "Fast and robust classification using asymmetric adaboost and a detector cascade," vol. 14, pp. 1311–1318, 2002.

[51] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, Oct 2001.

[52] A. Mogelmose, M. M. Trivedi, and T. B. Moeslund, "Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, pp. 1484–1497, Dec 2012.

[53] H. Gomez-Moreno, S. Maldonado-Bascon, P. Gil-Jimenez, and S. Lafuente-Arroyo, "Goal evaluation of segmentation algorithms for traffic sign recognition," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, pp. 917–930, Dec 2010.

[54] S. Maldonado-Bascon, S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno, and F. Lopez-Ferreras, "Road-sign detection and recognition based on support vector machines," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, pp. 264–278, June 2007.

[55] J. F. Khan, S. M. A. Bhuiyan, and R. R. Adhami, "Image segmentation and shape analysis for road-sign detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, pp. 83–96, March 2011.

[56] J. Greenhalgh and M. Mirmehdi, "Real-time detection and recognition of road traffic signs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, pp. 1498–1506, Dec 2012.

[57] X. Yuan, J. Guo, X. Hao, and H. Chen, "Traffic sign detection via graph-based ranking and segmentation algorithms," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, pp. 1509–1521, Dec 2015.

[58] Y. Yang, H. Luo, H. Xu, and F. Wu, "Towards real-time traffic sign detection and classification," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, pp. 2022–2031, July 2016.

[59] S. Salti, A. Petrelli, F. Tombari, N. Fioraio, and L. D. Stefano, "Traffic sign detection via interest region extraction," *Pattern Recognition*, vol. 48, no. 4, pp. 1039–1049, 2015.

[60] N. Barnes, A. Zelinsky, and L. S. Fletcher, "Real-time speed sign detection using the radial symmetry detector," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, pp. 322–332, June 2008.

[61] R. Belaroussi and J. P. Tarel, "Angle vertex and bisector geometric model for triangular road sign detection," in *Workshop on Applications of Computer Vision (WACV)*, pp. 1–7, Dec 2009.

[62] H. Li, F. Sun, L. Liu, and L. Wang, "A novel traffic sign detection method via color segmentation and robust shape matching," *Neurocomputing*, vol. 169, pp. 77–88, 2015.

[63] H. H. Aghdam, E. J. Heravi, and D. Puig, "A practical approach for detection and classification of traffic signs using convolutional neural networks," *Robotics and Autonomous Systems*, vol. 84, pp. 97–112, 2016.

[64] X. Baro, S. Escalera, J. Vitria, O. Pujol, and P. Radeva, "Traffic sign recognition using evolutionary adaboost detection and forest-ecoc classification," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, pp. 113–126, March 2009.

[65] C. Liu, F. Chang, and Z. Chen, "Rapid multiclass traffic sign detection in high-resolution images," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, pp. 2394–2403, Dec 2014.

[66] A. Mogelmose, D. Liu, and M. M. Trivedi, "Detection of u.s. traffic signs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, pp. 3116–3125, Dec 2015.

[67] T. Chen and S. Lu, "Accurate and efficient traffic sign detection using discriminative adaboost and support vector regression," *IEEE Transactions on Vehicular Technology*, vol. 65, pp. 4006–4015, June 2016.

[68] Y. Zhu, C. Zhang, D. Zhou, X. Wang, X. Bai, and W. Liu, "Traffic sign detection and recognition using fully convolutional network guided proposals," *Neurocomputing*, vol. 214, pp. 758–766, 2016.

[69] M. Shi, H. Wu, and H. Fleyeh, "Support vector machines for traffic signs recognition," in *IEEE International Joint Conference on Neural Networks*, pp. 3820–3827, June 2008.

[70] B. Hoferlin and K. Zimmermann, "Towards reliable traffic sign recognition," in *IEEE Intelligent Vehicles Symposium*, pp. 324–329, June 2009.

[71] W. J. Kuo and C. C. Lin, "Two-stage road sign detection and recognition," in *IEEE International Conference on Multimedia and Expo*, pp. 1427–1430, July 2007.

[72] A. Ruta, Y. Li, and X. Liu, "Robust class similarity measure for traffic sign recognition," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, pp. 846–855, Dec 2010.

[73] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 210–227, Feb 2009.

[74] K. Lu, Z. Ding, and S. Ge, "Sparse-representation-based graph embedding for traffic sign recognition," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, pp. 1515–1524, Dec 2012.

[75] X. Lu, Y. Wang, X. Zhou, Z. Zhang, and Z. Ling, "Traffic sign recognition via multi-modal tree-structure embedded multi-task learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, pp. 960–972, April 2017.

[76] P. Comelli, P. Ferragina, M. N. Granieri, and F. Stabile, "Optical recognition of motor vehicle license plates," *IEEE Transactions on Vehicular Technology*, vol. 44, pp. 790–799, Nov 1995.

[77] R. Zunino and S. Rovetta, "Vector quantization for license-plate location and image coding," *IEEE Transactions on Industrial Electronics*, vol. 47, pp. 159–167, Feb 2000.

[78] T. Naito, T. Tsukada, K. Yamada, K. Kozuka, and S. Yamamoto, "Robust license-plate recognition method for passing vehicles under outside environment," *IEEE Transactions on Vehicular Technology*, vol. 49, pp. 2309–2319, Nov 2000.

[79] C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, V. Loumos, and E. Kayafas, "A license plate-recognition algorithm for intelligent transportation system applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, pp. 377–392, Sept 2006.

[80] C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, I. D. Psoroulas, V. Loumos, and E. Kayafas, "License plate recognition from still images and video sequences: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, pp. 377–391, Sept 2008.

[81] C. N. E. Anagnostopoulos, "License plate recognition: A brief tutorial," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, pp. 59–67, Spring 2014.

[82] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, "Automatic license plate recognition (alpr): A state-of-the-art review," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, pp. 311–325, Feb 2013.

[83] Z. X. Chen, C. Y. Liu, F. L. Chang, and G. Y. Wang, "Automatic license-plate location and recognition based on feature salience," *IEEE Transactions on Vehicular Technology*, vol. 58, pp. 3781–3785, Sept 2009.

[84] Y. Wen, Y. Lu, J. Yan, Z. Zhou, K. M. von Deneen, and P. Shi, "An algorithm for license plate recognition applied to intelligent transportation system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, pp. 830–845, Sept 2011.

[85] W. Zhou, H. Li, Y. Lu, and Q. Tian, "Principal visual word discovery for automatic license plate detection," *IEEE Transactions on Image Processing*, vol. 21, pp. 4269–4279, Sept 2012.

[86] S.-L. Chang, L.-S. Chen, Y.-C. Chung, and S.-W. Chen, "Automatic license plate recognition," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, pp. 42–53, March 2004.

[87] H.-J. Lee, S.-Y. Chen, and S.-Z. Wang, "Extraction and recognition of license plates of motorcycles and vehicles on highways," in *International Conference on Pattern Recognition*, vol. 4, pp. 356–359, Aug 2004.

[88] B. F. Wu, S. P. Lin, and C. C. Chiu, "Extracting characters from real vehicle licence plates out-of-doors," *IET Computer Vision*, vol. 1, pp. 2–10, March 2007.

[89] J. M. Guo and Y. F. Liu, "License plate localization and character segmentation with feedback self-learning and hybrid binarization techniques," *IEEE Transactions on Vehicular Technology*, vol. 57, pp. 1417–1424, May 2008.

[90] W. Jia, H. Zhang, X. He, and M. Piccardi, "Mean shift for accurate license plate localization," in *IEEE Intelligent Transportation Systems*, pp. 566–571, Sept 2005.

[91] S. K. Kim, D. W. Kim, and H. J. Kim, "A recognition of vehicle license plate using a genetic algorithm based segmentation," in *IEEE International Conference on Image Processing*, vol. 1, pp. 661–664, Sep 1996.

[92] H. Mahini, S. Kasaei, F. Dorri, and F. Dorri, "An efficient features - based license plate localization method," in *International Conference on Pattern Recognition*, vol. 2, pp. 841–844, 2006.

[93] B. Hongliang and L. Changping, "A hybrid license plate extraction method based on edge statistics and morphology," in *International Conference on Pattern Recognition*, vol. 2, pp. 831–834 Vol.2, Aug 2004.

[94] Y. Peng, M. Xu, J. S. Jin, S. Luo, and G. Zhao, "Cascade-based license plate localization with line segment features and haar-like features," in *International Conference on Image and Graphics*, pp. 1023–1028, Aug 2011.

[95] W. Wu, X. Chen, and J. Yang, "Detection of text on road signs from video," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, pp. 378–390, Dec 2005.

[96] A. Reina, R. Sastre, S. Arroyo, and P. Jimenez, "Adaptive traffic road sign panels text extraction," in *International Conference on Signal Processing, Robotics and Automation*, pp. 295–300, 2006.

[97] A. Gonzalez, L. M. Bergasa, M. Gavilan, M. A. Sotelo, F. Herranz, and C. Fernandez, "Automatic information extraction of traffic panels based on computer vision," in *IEEE International Conference on Intelligent Transportation Systems*, pp. 1–6, Oct 2009.

[98] A. Gonzalez, L. M. Bergasa, J. J. Yebes, and M. A. Sotelo, "Automatic information recognition of traffic panels using sift descriptors and hmms," in *IEEE International Conference on Intelligent Transportation Systems*, pp. 1289–1294, Sept 2010.

[99] A. Gonzalez, M. A. Garrido, D. F. Llorca, M. Gavilan, J. P. Fernandez, P. F. Alcantarilla, I. Parra, F. Herranz, L. M. Bergasa, M. a. Sotelo, and P. R. de Toro, "Automatic traffic signs and panels inspection system using computer vision," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, pp. 485–499, June 2011.

[100] A. Gonzalez, L. M. Bergasa, J. J. Yebes, and J. Almazan, "Text recognition on traffic panels from street-level imagery," in *IEEE Intelligent Vehicles Symposium*, pp. 340–345, June 2012.

[101] A. Gonzalez, L. M. Bergasa, and J. J. Yebes, "Text detection and recognition on traffic panels from street-level imagery using visual appearance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, pp. 228–238, Feb 2014.

[102] J. Greenhalgh and M. Mirmehdi, "Recognizing text-based traffic signs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, pp. 1360–1369, June 2015.

[103] A. Mammeri, A. Boukerche, and E. H. Khiari, "Mser-based text detection and communication algorithm for autonomous vehicles," in *IEEE Symposium on Computers and Communication (ISCC)*, pp. 1218–1223, June 2016.

[104] J. Shi and C. Tomasi, "Good features to track," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 593–600, Jun 1994.

[105] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," pp. 1–22, 2004.

[106] H. Zhang, K. Zhao, Y.-Z. Song, and J. Guo, "Text extraction from natural scene image: A survey," *Neurocomputing*, vol. 122, pp. 310–323, 2013. Advances in cognitive and ubiquitous computingSelected papers from the Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2012).

[107] Y. Zhu, C. Yao, and X. Bai, "Scene text detection and recognition: recent advances and future trends," *Frontiers of Computer Science*, vol. 10, no. 1, pp. 19–36, 2016.

[108] Q. Ye and D. Doermann, "Text detection and recognition in imagery: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, pp. 1480–1500, July 2015.

[109] L. Neumann and J. Matas, "A method for text localization and recognition in real-world images," in *Asian Conference on Computer Vision (ACCV)* (R. Kimmel, R. Klette, and A. Sugimoto, eds.), (Berlin, Heidelberg), pp. 770–783, Springer Berlin Heidelberg.

[110] L. Neumann and J. Matas, "Text localization in real-world images using efficiently pruned exhaustive search," in *International Conference on Document Analysis and Recognition*, pp. 687–691, Sept 2011.

[111] L. Neumann and J. Matas, "Real-time scene text localization and recognition," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3538–3545, June 2012.

[112] L. Neumann and J. Matas, "Scene text localization and recognition with oriented stroke detection," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 97–104, Dec 2013.

[113] L. Neumann and J. Matas, "On combining multiple segmentations in scene text recognition," in *International Conference on Document Analysis and Recognition*, pp. 523–527, Aug 2013.

[114] L. Neumann and J. Matas, "Real-time lexicon-free scene text localization and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 1872–1885, Sept 2016.

[115] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu, "Detecting texts of arbitrary orientations in natural images," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1083–1090, June 2012.

[116] C. Yao, X. Bai, and W. Liu, "A unified framework for multioriented text detection and recognition," *IEEE Transactions on Image Processing*, vol. 23, pp. 4737–4749, Nov 2014.

[117] X. C. Yin, X. Yin, K. Huang, and H. W. Hao, "Robust text detection in natural scene images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, pp. 970–983, May 2014.

[118] X. C. Yin, W. Y. Pei, J. Zhang, and H. W. Hao, "Multi-orientation scene text detection with adaptive clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, pp. 1930–1937, Sept 2015.

[119] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting text in natural scenes with stroke width transform," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2963–2970, June 2010.

[120] H. Chen, S. S. Tsai, G. Schroth, D. M. Chen, R. Grzeszczuk, and B. Girod, "Robust text detection in natural images with edge-enhanced maximally stable extremal regions," in *IEEE International Conference on Image Processing*, pp. 2609–2612, Sept 2011.

[121] C. Yi and Y. Tian, "Text string detection from natural scenes by structure-based partition and grouping," *IEEE Transactions on Image Processing*, vol. 20, pp. 2594–2605, Sept 2011.

[122] T. Novikova, O. Barinova, P. Kohli, and V. Lempitsky, "Large-lexicon attribute-consistent text recognition in natural images," in *European Conference on Computer Vision (ECCV)* (A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, eds.), (Berlin, Heidelberg), pp. 752–765, Springer Berlin Heidelberg.

[123] A. R. Chowdhury, U. Bhattacharya, and S. K. Parui, "Scene text detection using sparse stroke information and mlp," in *International Conference on Pattern Recognition*, pp. 294–297, Nov 2012.

[124] C. Yi and Y. Tian, "Localizing text in scene images by boundary clustering, stroke segmentation, and string fragment classification," *IEEE Transactions on Image Processing*, vol. 21, pp. 4256–4268, Sept 2012.

[125] T. Q. Phan, P. Shivakumara, S. Tian, and C. L. Tan, "Recognizing text with perspective distortion in natural scenes," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 569–576, Dec 2013.

[126] A. Gonzalez and L. M. Bergasa, "A text reading algorithm for natural images," *Image and Vision Computing*, vol. 31, no. 3, pp. 255–274, 2013.

[127] H. I. Koo and D. H. Kim, "Scene text detection via connected component clustering and nontext filtering," *IEEE Transactions on Image Processing*, vol. 22, pp. 2296–2305, June 2013.

[128] R. Minetto, N. Thome, M. Cord, N. J. Leite, and J. Stolfi, "Snoopertext: A text detection system for automatic indexing of urban scenes," *Computer Vision and Image Understanding*, vol. 122, pp. 92–104, 2014.

[129] J. Fabrizio, B. Marcotegui, and M. Cord, "Text detection in street level images," *Pattern Analysis and Applications*, vol. 16, no. 4, pp. 519–533, 2013.

[130] W. Huang, Z. Lin, J. Yang, and J. Wang, "Text localization in natural images using stroke feature transform and text covariance descriptors," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 1241–1248, Dec 2013.

[131] L. Kang, Y. Li, and D. Doermann, "Orientation robust text line detection in natural images," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4034–4041, June 2014.

[132] W. Huang, Y. Qiao, and X. Tang, "Robust scene text detection with convolution neural network induced mser trees," in *European Conference on Computer Vision (ECCV)* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 497–511, Springer International Publishing.

[133] L. Sun, Q. Huo, W. Jia, and K. Chen, "Robust text detection in natural scene images by generalized color-enhanced contrasting extremal region and neural networks," in *International Conference on Pattern Recognition*, pp. 2715–2720, Aug 2014.

[134] C. Yu, Y. Song, Q. Meng, Y. Zhang, and Y. Liu, "Text detection and recognition in natural scene with edge analysis," *IET Computer Vision*, vol. 9, no. 4, pp. 603–613, 2015.

[135] T. He, W. Huang, Y. Qiao, and J. Yao, "Text-attentional convolutional neural network for scene text detection," *IEEE Transactions on Image Processing*, vol. 25, pp. 2529–2541, June 2016.

[136] Y. Tang and X. WU, "Scene text detection and segmentation based on cascaded convolution neural networks," *IEEE Transactions on Image Processing*, vol. PP, no. 99, pp. 1–1, 2017.

[137] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Deep features for text spotting," in *European Conference on Computer Vision (ECCV)* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 512–528, Springer International Publishing.

[138] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading text in the wild with convolutional neural networks," *International Journal of Computer Vision*, vol. 116, no. 1, pp. 1–20, 2016.

[139] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2315–2324, June 2016.

[140] Z. Zhang, W. Shen, C. Yao, and X. Bai, "Symmetry-based text line detection in natural scenes," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2558–2567, June 2015.

[141] Z. Zhang, C. Zhang, W. Shen, C. Yao, W. Liu, and X. Bai, "Multi-oriented text detection with fully convolutional networks," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4159–4167, June 2016.

[142] Y. F. Pan, X. Hou, and C. L. Liu, "A hybrid approach to detect and localize texts in natural scene images," *IEEE Transactions on Image Processing*, vol. 20, pp. 800–813, March 2011.

[143] X. Chen and A. L. Yuille, "Detecting and reading text in natural scenes," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 366–373, June 2004.

[144] J. J. Lee, P. H. Lee, S. W. Lee, A. Yuille, and C. Koch, "Adaboost for text detection in natural scene," in *International Conference on Document Analysis and Recognition*, pp. 429–434, Sept 2011.

[145] K. Wang, B. Babenko, and S. Belongie, "End-to-end scene text recognition," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 1457–1464, Nov 2011.

[146] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng, "End-to-end text recognition with convolutional neural networks," in *International Conference on Pattern Recognition (ICPR)*, pp. 3304–3308, Nov 2012.

[147] A. Mishra, K. Alahari, and C. V. Jawahar, "Top-down and bottom-up cues for scene text recognition," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2687–2694, June 2012.

[148] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven, "Photoocr: Reading text in uncontrolled conditions," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 785–792, Dec 2013.

[149] S. Tian, Y. Pan, C. Huang, S. Lu, K. Yu, and C. L. Tan, "Text flow: A unified text detection system in natural scene images," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 4651–4659, Dec 2015.

[150] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and Vision Computing*, vol. 22, no. 10, pp. 761–767, 2004. British Machine Vision Computing 2002.

[151] J. Matas and K. Zimmermann, "A new class of learnable detectors for categorisation," in *Image Analysis: 14th Scandinavian Conference* (H. Kalviainen, J. Parkkinen, and A. Kaarna, eds.), (Berlin, Heidelberg), pp. 541–550, Springer Berlin Heidelberg, 2005.

[152] J. Sochman and J. Matas, "Waldboost - learning for time constrained sequential detection," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 150–156, June 2005.

[153] C. L. Zitnick and P. Dollár, *Edge Boxes: Locating Object Proposals from Edges*, pp. 391–405. Cham: Springer International Publishing, 2014.

[154] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2014.

[155] M. D. Zeiler and R. Fergus, *Visualizing and Understanding Convolutional Networks*, pp. 818–833. Cham: Springer International Publishing, 2014.

[156] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International Conference on Artificial Intelligence and Statistics* (Y. W. Teh and M. Titterington, eds.), vol. 9 of *Proceedings of Machine Learning Research*, (Chia Laguna Resort, Sardinia, Italy), pp. 249–256, PMLR, 13–15 May 2010.

[157] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, Dec 2015.

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

[159] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *CoRR*, vol. abs/1505.00387, 2015.

[160] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5353–5360, June 2015.

[161] K. He, X. Zhang, S. Ren, and J. Sun, *Identity Mappings in Deep Residual Networks*, pp. 630–645. Cham: Springer International Publishing, 2016.

[162] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, *Deep Networks with Stochastic Depth*, pp. 646–661. Cham: Springer International Publishing, 2016.

[163] G. Larsson, M. Maire, and G. Shakhnarovich, "Fractalnet: Ultra-deep neural networks without residuals," *CoRR*, vol. abs/1605.07648, 2016.

[164] G. Huang, Z. Liu, K. Q. Weinberger, and V. D. M. Laurens, "Densely connected convolutional networks," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[165] Y. I. Ohta, T. Kanade, and T. Sakai, "Color information for region segmentation," *Computer Graphics and Image Processing*, vol. 13, no. 3, pp. 222–241, 1980.

[166] K. Grauman and T. Darrell, "The pyramid match kernel: discriminative classification with sets of image features," in *IEEE International Conference on Computer Vision (ICCV)*, vol. 2, pp. 1458–1465, Oct 2005.

[167] K. Ikeuchi, ed., *Color Appearance Models*, pp. 109–109. Boston, MA: Springer US, 2014.

[168] J. Iivarinen and A. J. E. Visa, "Shape recognition of irregular objects," *The International Society for Optical Engineering*, vol. 2904, pp. 25–32, 1996.

[169] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, pp. 154–171, Sep 2013.

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

[171] K. He, X. Zhang, S. Ren, and J. Sun, *Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition*, pp. 346–361. Cham: Springer International Publishing, 2014.

[172] A. Vedaldi and K. Lenc, "Matconvnet: Convolutional neural networks for matlab," in *ACM International Conference on Multimedia*, MM '15, (New York, NY, USA), pp. 689–692, ACM, 2015.

[173] R. H. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *Nature*, vol. 405, no. 6789, pp. 947–951, 2000.

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

[175] K. N. Plataniotis, "Color image processing and applications," *Measurement Science and Technology*, vol. 12, no. 2, p. 222, 2001.

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

[177] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, pp. 211–252, Dec 2015.

[178] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, June 2015.

[179] Y. LeCun *et al.*, "Generalization and network design strategies," *Connectionism in perspective*, pp. 143–155, 1989.

[180] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, pp. 541–551, Dec 1989.

[181] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[182] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[183] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pp. III–1139–III–1147, JMLR.org, 2013.

[184] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, July 2011.

[185] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[186] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *CoRR*, vol. abs/1212.5701, 2012.

[187] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[188] J. T. Springenberg, "Unsupervised and semi-supervised learning with categorical generative adversarial networks," in *International Conference on Learning Representations (ICLR)*, 2016.

[189] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *ICML*, 2017.

[190] M. Simon, , E. Rodner, , and J. Denzler, *Asian Conference on Computer Vision (ACCV)*, ch. Part Detector Discovery in Deep Convolutional Neural Networks, pp. 162–177. Cham: Springer International Publishing, 2015.

[191] M. Simon and E. Rodner, "Neural activation constellations: Unsupervised part model discovery with convolutional networks," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 1143–1151, Dec 2015.

[192] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *CoRR*, vol. abs/1312.6034, 2013.

[193] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2672–2680, Curran Associates, Inc., 2014.

[194] Online, "Caltech plate dataset," 1999.

[195] J. F. Ziomek, "Intelligent vehicle technology and trends - [book review]," *IEEE Instrumentation Measurement Magazine*, vol. 9, pp. 54–54, Feb 2006.

[196] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young, "Icdar 2003 robust reading competitions," in *International Conference on Document Analysis and Recognition*, pp. 682–687, Aug 2003.

[197] A. Shahab, F. Shafait, and A. Dengel, "Icdar 2011 robust reading competition challenge 2: Reading text in scene images," in *International Conference on Document Analysis and Recognition*, pp. 1491–1496, Sept 2011.

[198] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i. Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. A. Almazan, and L. P. de las Heras, "Icdar 2013 robust reading competition," in *International Conference on Document Analysis and Recognition*, pp. 1484–1493, Aug 2013.

[199] J. Shao, K. Kang, C. C. Loy, and X. Wang, "Deeply learned attributes for crowded scene understanding," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4657–4666, June 2015.

[200] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar, "Attribute and simile classifiers for face verification," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 365–372, Sept 2009.

[201] D. Parikh and K. Grauman, "Relative attributes," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 503–510, Nov 2011.

[202] G. Gkioxari, R. Girshick, and J. Malik, "Contextual action recognition with r*cnn," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 1080–1088, Dec 2015.

[203] F. Nie, H. Huang, X. Cai, and C. H. Ding, "Efficient and robust feature selection via joint l2, 1-norms minimization," in *Advances in Neural Information Processing Systems*, pp. 1813–1821, 2010.

[204] F. Nie, S. Xiang, Y. Jia, C. Zhang, and S. Yan, "Trace ratio criterion for feature selection.," in *AAAI Conference on Artificial Intelligence*, vol. 2, pp. 671–676, 2008.

[205] S. Boutemedjet, D. Ziou, and N. Bouguila, "Unsupervised feature selection for accurate recommendation of high-dimensional image data.," in *Advances in Neural Information Processing Systems*, pp. 177–184, 2007.

[206] C. Maung and H. Schweitzer, "Pass-efficient unsupervised feature selection," in *Advances in Neural Information Processing Systems*, pp. 1628–1636, 2013.

[207] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, June 2008.

[208] L. Bourdev, S. Maji, and J. Malik, "Describing people: A poselet-based approach to attribute classification," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 1543–1550, Nov 2011.

[209] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 2018–2025, Nov 2011.

[210] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia, "Incorporating second-order functional knowledge for better option pricing," in *Advances in Neural Information Processing Systems*, pp. 472–478, 2001.

[211] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *International Conference on Machine Learning*, 2013.

[212] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 2146–2153, Sept 2009.

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

Appendix A

Convolutional Neural Networks

A.1 Network Blocks

A.1.1 Weight Layers

Convolution

Generally, convolution is an operation about two real valued functions. Suppose there is a sensor that measures a signal $x(t)$ at time t, however, the measured data is noisy. In order to estimate a new data with less noise, an effective choice is to use average data between several continuous measurements. Since more recent measurements are more relevant, higher weights should be assigned to these measurements, oppositely, dated measurements should have lower weights. The estimated signal $y(t)$ can be achieved by convolution between the signal $x(t)$ and a weighting function $w(t)$, formulated as:

$$y(t) = \int_{-\infty}^{+\infty} x(a)w(t-a)da \tag{A.1}$$

Normally, the convolution operation is denoted as an asterisk:

$$y(t) = x(t) * w(t) \tag{A.2}$$

In real world applications, it is impossible to measure data at every instant, the data are discretely collected with selected frequencies. The discrete convolution is defined as:

$$y(t) = \sum_{a=-\infty}^{+\infty} x(a)w(t-a) \tag{A.3}$$

And in computer vision field, the inputs are usually multi-dimensional array data, such as image and video. Therefore, convolutions are performed over more than one axis at a time. The convolution of 3D input and output is formulated as:

$$y^j = \sum_i f^{ij} * x^i + b^j \tag{A.4}$$

where x^i denotes the i-th input data in depth, y^j denotes the j-th output data in depth. f^{ij} is the convolution kernels and b^j is the bias. $*$ represents 2D convolution operation. $\sum$ represents summation operation.

Convolution operation is effective and powerful in machine learning systems, because of three aspects as follows. (i) Sparse interactions or local connectivity, which is accomplished by connecting neurons to only a local region of input data. The spatial extent of this local region is also called local receptive field. Normally, the sparse intersections are local in width and height, but dense in depth. This property ensures convolution operation needs much less parameters and much fewer computational cost. (ii) Parameter sharing refers to repeatedly use same parameters in a network. In traditional neural networks, each parameter only contributes to output once. However, in CNNs, each parameter in kernels is repeatedly used in every local operation. This parameter sharing strategy means that only a set of parameters are learnt, rather than different sets of parameters for every location. Therefore, memory requirement and computational cost are further reduced. (iii) Equivalent representations is another important property that benefits from parameter sharing. The equivalent property of a function means that outputs change in the same way as inputs change. In CNNs, outputs and inputs are equivalent in translation, and this property has been widely used in lots of CNN architectures. However, CNNs are not naturally equivalent in rotation and scale.

Convolution Transpose (Deconvolution)

Mathematically, deconvolution is an algorithm that reverses the effects of convolution operation on data. However, in CNNs, deconvolution [209] and convolution transpose are synonymous, both of them perform transpose operation of convolution. The convolution transpose of 3D input and output can be formulated as:

$$\tilde{x}^j = \sum_i y^i * f^{ij} + b^j \tag{A.5}$$

where y^i denotes the $i-th$ input data in depth and $\tilde{x}^j$ denotes the $j-th$ output data in depth. f^{ij} is the convolution transpose kernels and b^j is the bias. $*$ represents 2D convolution transpose operation. $\sum$ represents summation operation.

Fully-connected

Fully-connected layers have been used for a long time in traditional neural network, which apply matrix multiplication between inputs and weights. The intersections between each input unit and output unit are fully connected, which means every input unit is connected to every output with separate weight. Matrix multiplication in fully-connected layers can be formulated as:

$$y^j = \sum_i w^{ij} x^i + b^j \tag{A.6}$$

where x^i and y^j denotes the $i-th$ input data and the $j-th$ output data respectively. w^{ij} is the weight matrix and b^j is the bias.

Parameter Initialisation

In order to train deep CNNs, initial points should be generated to optimisation algorithms for iterating. However, the convergence of most optimisation algorithms are strongly depended on the choice of initialisation. Bad initialisation may cause unstable and numerical problems, sometimes models can not converge at all. On the other hand, good initialisation is able to speed up convergence and improve generalisation.

Currently, the initialisation methods are simple and heuristic. It is very difficult to design improved initialisation methods, because optimisation of deep models has not been totally understood. Parameters in weight layers are composed of weights and biases. Normally, weights are initialised randomly, and biases are set to constants. There are three widely applied methods, which will be detailed as follows.

(i) *Gaussian distribution* is the most common initialisation method. The scale of initial distribution has significant effect on both convergence and generalisation. In order to propagate information forward and backward successfully, initial scales should be large enough. However, large weights are harmful to regularisation, resulting in bad generalisation. Denoting a scale value s, gaussian distribution initialisation can be formulated as:

$$w \sim (-s, s) \tag{A.7}$$

(ii) *Xavier* [156] is a kind of normalised initialisation. The weights are normalised by the formulation:

$$w \sim (-\sqrt{\frac{3}{hwd_{in}}}, \sqrt{\frac{3}{hwd_{in}}}) \tag{A.8}$$

where h and w are the height and the width of kernels respectively. d_{in} is the depth of the input.

(iii) *Improved Xavier* [157] further improves *Xavier* initialisation and demonstrates excellent performance in the training of CNNs. The main difference is that *Improved Xavier* considers the influences from nonlinear blocks. The weights are normalised by the formulation:

$$w \sim (-\sqrt{\frac{2}{hwd_{out}}}, \sqrt{\frac{2}{hwd_{out}}}) \tag{A.9}$$

where h and w are the height and the width of kernels respectively. d_{out} is the depth of the output.

A.1.2 Nonlinear Activation Layers

The sigmoid and Hyperbolic Tangent (Tanh) functions are formulated as follows:

$$Sigmoid(x) = \frac{1}{1+e^{-x}} \tag{A.10}$$

$$Tanh(x) = \frac{e^{2x}-1}{e^{2x}+1} \tag{A.11}$$

where x is the input and e is the exponential function. As Fig. A.1 illustrates, sigmoid and tanh are bounded by minimum and maximum values, which may cause saturation problem that makes gradient based learning very difficult.

The ReLU [173], Softplus [210] and Leaky ReLU [211] functions are formulated as following:

$$ReLU(x) = max(x, 0) \tag{A.12}$$

$$Softplus(x) = log(1+e^{x}) \tag{A.13}$$

$$Leaky\ ReLU(x) = max(x, 0.01x) \tag{A.14}$$

where x is the input and e is the exponential function. The curve of each function is illustrated in Fig. A.2.

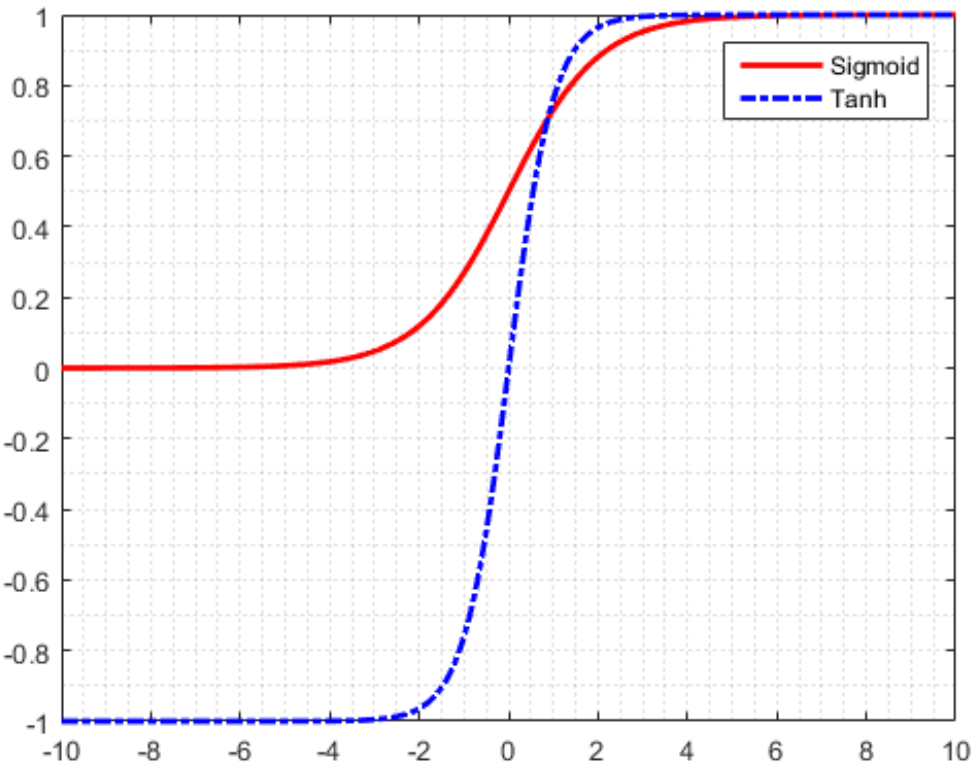


Fig. A.1 Sigmoid and Tanh activation functions.

A.1.3 Pooling Layers

Generally, two kinds of pooling layers are usually applied, which are max pooling and mean pooling. Max pooling finds the largest values in selected extents, while meaning pooling calculates the mean values. The formulations are denoted as:

$$y_{i,j}^{k} = \max_{0 \leqslant m < h, 0 \leqslant n < w} x_{i+m,j+n}^{k} \tag{A.15}$$

$$y_{i,j}^{k} = \frac{1}{hw} \sum_{0 \leqslant m < h, 0 \leqslant n < w} x_{i+m,j+n}^{k} \tag{A.16}$$

where x^k and y^k are the $k-th$ input and output feature map. h and w are the height and width of selected window.

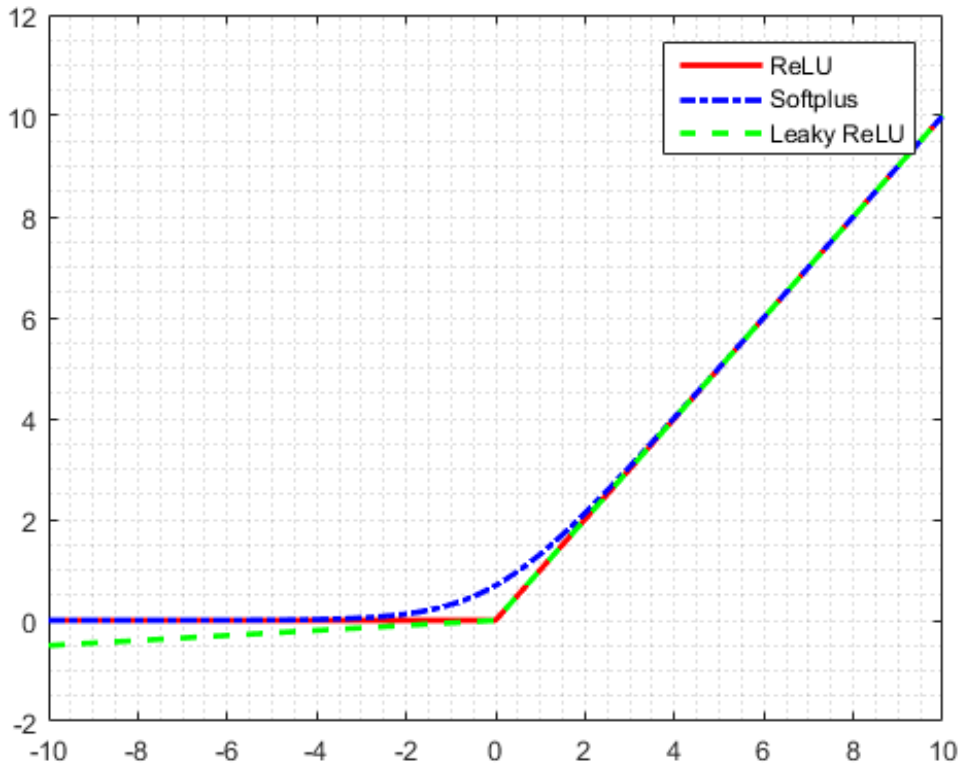


Fig. A.2 ReLU, Softplus and Leaky ReLU activation functions.

A.1.4 Normalisation Layers

Local Response Normalisation

The LRN is originally inspired by Local Contrast Normalisation (LCN) [212], which normalise features based on nearby adjacent in same feature maps or same spatial location in different feature maps by subtraction and division. Comparing to LCN, LRN does not subtract the mean values, therefore, the brightness information in LRN is abundant. The response-normalised features can be computed by:

$$y_{i,j}^{k} = x_{i,j}^{k} / \left(\gamma + \alpha \sum_{l=max(0,k-n/2)}^{min(N-1,k+n/2)} (x_{i,j}^{l})^2 \right)^{\beta} \tag{A.17}$$

where $x_{i,j}^{k}$ and $y_{i,j}^{k}$ are the input and output feature maps respectively. N is the total depth of input feature maps, and the normalisation runs over n adjacent feature maps at the same

spatial positions. The constants α, β and γ are hyper-parameters whose values should be determined on a validation set.

Batch Normalisation

Due to the varying distribution of inputs of each layer, the training of CNNs is complicate and sensitive. Generally, low learning rates and careful parameter initialisation are required for tackling this problem. This phenomenon is referred as internal covariate shift [158], which has been significantly addressed by BN. Different from LCN and LRN that perform normalisation for each individual input, BN normalising inputs for each training mini-batch. Networks with BN layers can be trained with higher learning rates and less careful initialisation, and the networks also demonstrate better generalisation ability.

Denoting a mini-batch B of size m, for each spatial location of the inputs, there are m features in the mini-batch: $B = \{x_1, x_2 ... x_k\}$. Let the normalised features be $\hat{x}_{1...m}$ and the linear transformed features be $y_{1...m}$. BN performs the transformation $BN : x_{1...m} \rightarrow \hat{x}_{1...m} \rightarrow y_{1...m}$. More specifically, BN transformation is illustrated as follows:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{A.18}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2 \tag{A.19}$$

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \tag{A.20}$$

$$y_i = \alpha \hat{x}_i + \beta \tag{A.21}$$

where μ_B is mini-batch mean and σ_B^2 is mini-batch variance. α and β are scale and shift values of linear transformation for improving feature representation. ε is a constant for ensuring numerical stability.

A.1.5 Loss Layers

In most cases, CNNs are required to describe a distribution $p(y|x;\theta)$, and the principle of maximum likelihood is usually used. This means that the loss function is simply the negative log-likelihood, or equivalently denoted as the cross-entropy between the model distribution and the training data distribution. This loss function is given by:

$$J(\theta) = -\mathbb{E}_{x,y\sim\hat{p}_{data}} log_{p_{model}}(y|x) \tag{A.22}$$

The choice of output layers should be tightly designed with the tasks. In a lot of tasks, the prediction value y is a binary variable, and then the sigmoid function is used in output layer, which is formulated as:

$$Sigmoid(x) = \frac{1}{1+e^{-x}} \tag{A.23}$$

On the other hand, in order to train a n-categories classifier, the model distribution is over n different classes. Softmax function is usually adopted, which can be formulated as:

$$Softmax(x)_i = \frac{e^{x_i}}{\sum_i e^{x_i}} \tag{A.24}$$

A.2 Network Architectures

A.2.1 Traditional Networks

These CNNs are simple shallow networks that consist of about 3 convolutional and 2 fully-connected layers, followed by nonlinear activation layers. The sizes of the convolutional kernels are usually larger than 4×4. The pooling layers may also introduced for reducing feature dimension. An example of a 5-layer network is illustrated in Fig. A.3.

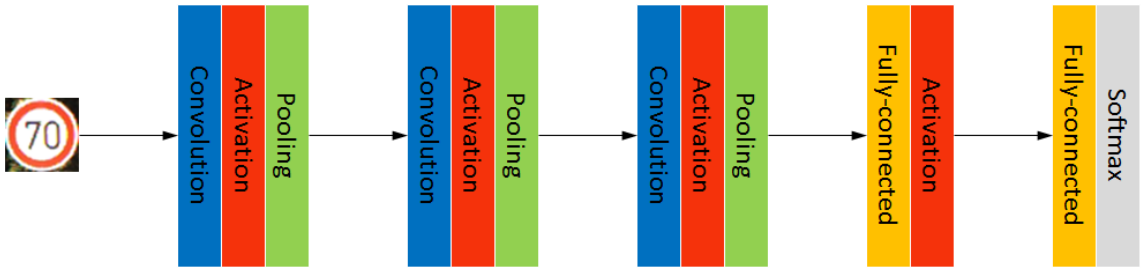


Fig. A.3 Illustration of a 5-layer traditional network.

A.2.2 Very Deep Networks

In the VGGNets, the sizes of the convolutional kernels are all 3×3, therefore, the computational cost of convolution is reduced, and it becomes feasible to build deeper CNNs. An example of a 11-layer VGGNet is illustrated in Fig. A.4.

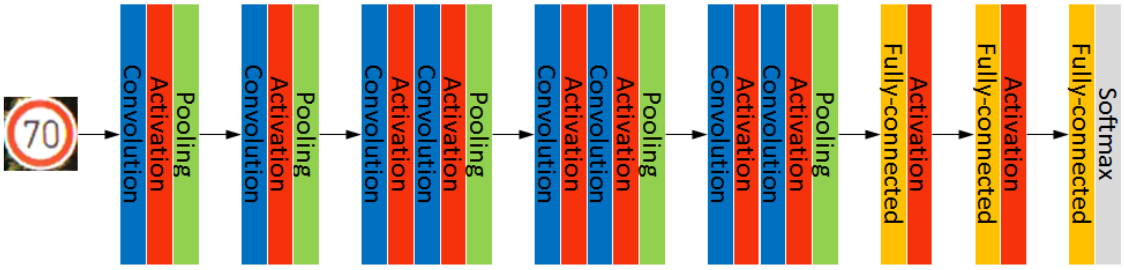


Fig. A.4 Illustration of a 11-layer VGGNet.

A.2.3 Deep Residual Networks

In ResNets, the entire network are not trained through individual forward or backward path, each few stacked layers are able to learn residual mapping by shortcut connections. The identity shortcut connection for residual learning has been illustrated in Fig. A.5. Formally, denoting a desired direct mapping $H(x)$ and a residual mapping $F(x) = H(x) - x$, then the original desired mapping can be represented as $H(x) = F(x) + x$.

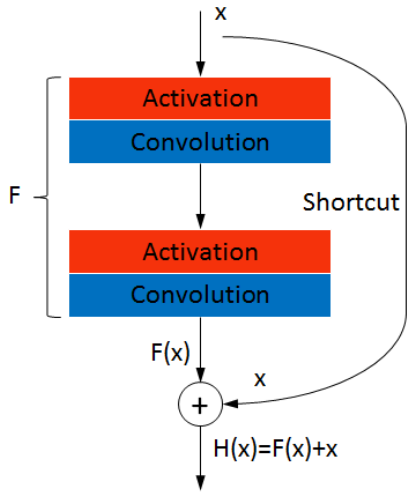


Fig. A.5 Residual learning with identity shortcut connection.

Inspired by the VGGNets, two plain networks with 18 and 34 layers are firstly developed. The architecture of a 18-layer plain network (PlainNet-18) is illustrated in Fig. A.6. In this PlainNet-18, besides the first convolutional layer, the rest 16 convolutional layers are grouped into 8 stacked layers blocks, which are also basic blocks in residual networks. The design of these plain networks follows the following rules: (i) the sizes of all the convolutional kernels are 3×3; (except the first convolutional layer) (ii) if the size of output feature maps are held, the channel of output feature maps are held; (iii) if the size of output feature maps are halved, the channel of output feature maps are doubled; (iv) pooling operation should be performed

by convolutional layers with stride 2. Finally, global average pooling is applied to pool the final feature maps into 1×1 size, followed by fully-connected and softmax layers.

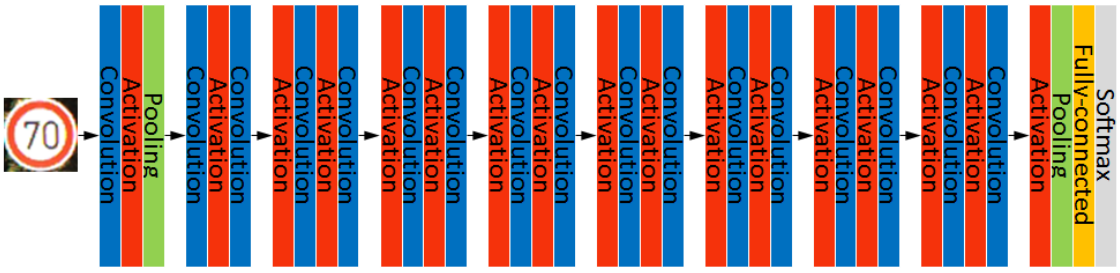


Fig. A.6 Illustration of the architecture of a PlainNet-18.

Based on the above plain network, the corresponding 18-layer residual network (ResNet-18) is simply developed by adding shortcut connections between each stacked layers. The architecture of a ResNet-18 is illustrated in Fig. A.7. Due to the destination of some shortcut connections have increased feature map channels, three designs are developed for dealing with this problem: (i) zero padding for these shortcuts; (ii) projection for these shortcuts; (iii) projection for all the shortcuts. The results indicate the third design has the best performance and the highest computational cost.

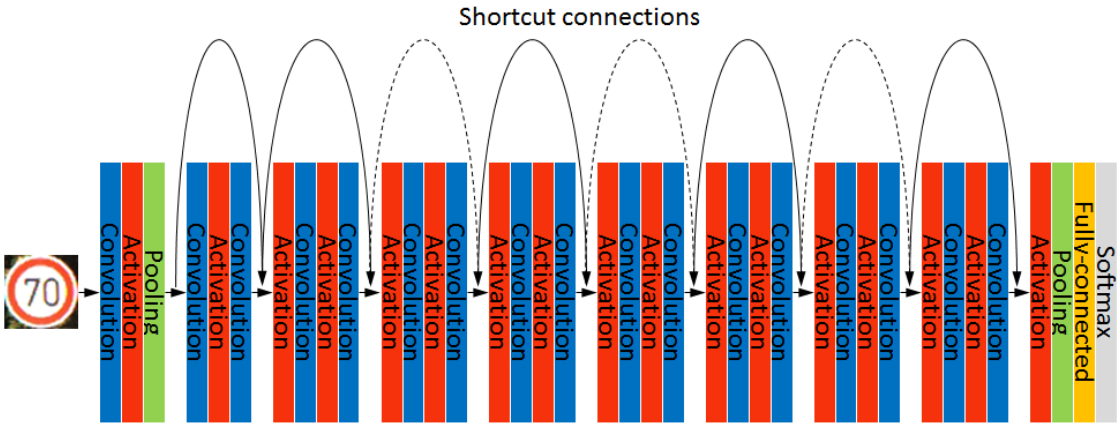


Fig. A.7 Illustration of the architecture of a ResNet-18. The solid shortcut connections hold the feature map channels, and the dotted shortcut connections increase feature map channels.

With the help of the shortcut connections, residual networks can be much deeper than 34 layers. Three much deeper residual networks are subsequently developed, namely ResNet-50, ResNet-101 and ResNet-152. Moreover, in order to improve computational efficiency, the original stacked layers are replaced with bottleneck layers. The detail of these two blocks have been presented in Fig. A.8. In stacked layers, the sizes of all convolutional kernels are

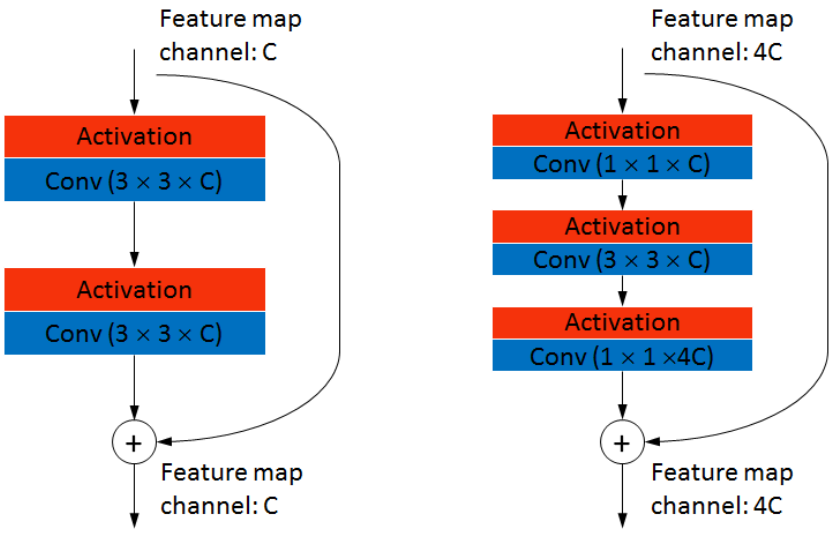


Fig. A.8 Left: stacked layers for ResNet-18 and ResNet-34. Right: bottleneck layers for ResNet-50, ResNet-101 and ResNet-152.

3×3 and the channels of the feature maps are held. On the other hand, in bottleneck layers, the first convolutional layer is applied to reduce the channel of the input feature map into one fourth, followed by the second convolutional layer with 3×3 kernel size. Finally, the channel of output feature map is increased to four times by the third convolutional layer with 1×1 kernel size.

A.2.4 Densely Connected Networks

Different from ResNets, features from preceding layers are combined by concatenation instead of summation. Formally, denoting a desired direct mapping $H(x)$ and an identity shortcut connection x, the mapping of ResNets can be represented as $H(x) = F(x) + x$, and the mapping of DenseNets is $H(x) = [F(x), x]$.

Inspired by [32, 213], DenseNets also employ a basic block called composite layers, which is composed of three layers: *BN-ReLU-Conv*(3×3). In order to improve the efficiency of the networks, this block can be further improved to *bottleneck layers*: *BN-ReLU-Conv*(1×1)*-BN-ReLU-Conv*(3×3). The detail of these two blocks have been presented in Fig. A.9. Subsequently, the dense block can be built by cascading several these blocks. Since the input and output feature maps of each basic block is concatenated in channel, the channel of feature maps inside a dense block grows by a rate G. Due to the concatenation operation inside each dense block, the size of the feature maps are unchangeable. In order to reduce the size of feature maps, transition blocks are developed and used between two dense blocks. This block

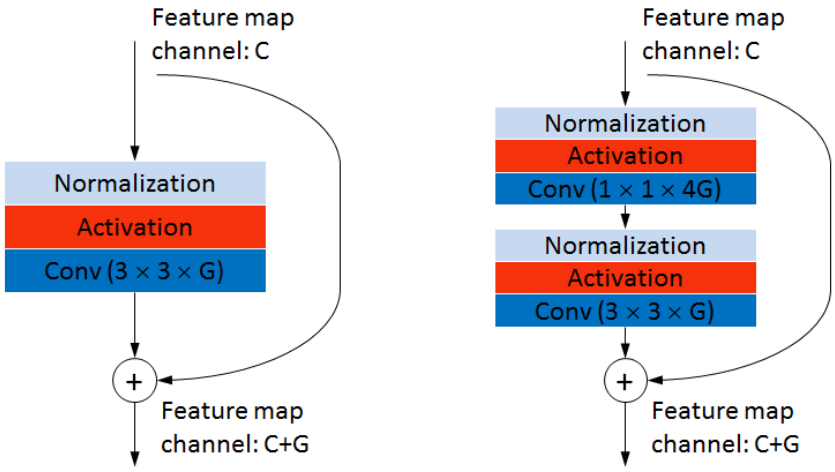


Fig. A.9 Left: composite layers. Right: bottleneck layers. The G stands for the growth rate in DenseNets.

is composed of four layers: *BN-ReLU-Conv*(1×1)*-Pool*(2×2). Furthermore, inside transition blocks, model compactness can be further enhanced by employing a hyperparameter θ that controls the channels of output feature maps. Assuming the channel of the input feature maps of a transition block is C, the channel of the output feature maps can be reduced to $C\theta(0 < \theta \leqslant 1)$ within the internal convolution operation. A DenseNet with three dense blocks and two transition blocks has been presented in Fig. A.10.

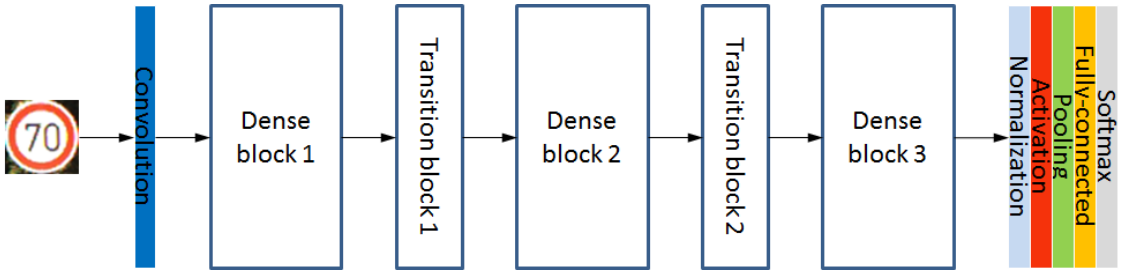


Fig. A.10 Illustration of the architecture of a DenseNet with three dense blocks and two transition blocks.

A.3 Network Regularisation

A.3.1 Data Argumentation

The best and most direct way to achieve better model generalisation is to use more training data. However, the number of training is usually limited in real applications. Therefore,

additional data should be generated based on existing data. In classification problem, such as image recognition, data argumentation has been proven to be an effective technique. Giving an image, it is very easy to create a set of similar images with a variety of variations, such as translation, flipping, scaling and rotation. Moreover, injecting noise to training data is also a form of data augmentation. Nevertheless, data augmentation should be carefully applied in some tasks. For example, in scene text recognition tasks, digits *6* and *9* are both included, so it is not appropriate to augment data with too large rotation angles.

A.3.2 Dropout

Deep CNNs with a mass of parameters have very strong representation ability that can learn very complicated relationships between inputs and outputs. However, many of the learnt relationships are noise, which only exist in training data but never in testing data. Therefore, CNNs usually face serious overfitting. To address this problem, a good choice is to combine the predictions of many CNNs that trained under different settings. However, CNNs are large and slow, the computational cost of averaging the outputs of many separately trained CNNs are extremely high. Moreover, CNNs usually require a large amounts of training data, and it may very difficult to collect enough data to train different CNNs on different subsets of the data.

Dropout [174] as a powerful regularisation methods, is able to prevent overfitting and provide an inexpensive way of combining exponentially many CNNs. The term dropout means to randomly remove output units from original network. The connection is dropped temporarily in training stage during both forward and backward propagation. For each unit, its independent dropping probability is p, which can be simply set to 0.5 or selected based on a validation set. Considering a layer with n outputs and a dropout rate of 0.5, the layer can be regarded as a collection of 2^n possible thinner layers that share parameters. For each iteration in training, a new thinner layer is selected and trained. Therefore, the training of CNNs with dropout is actually the training of a collection of CNNs with extensive parameter sharing. Subsequently, in testing stage, the CNN works like the bagging ensemble of many CNNs.

A.3.3 Parameter Norm Penalty

Parameter norm penalty includes a series of classical regularisation methods that rely on penalising the parameters of models. Denoting a parameter norm penalty term $\Omega(\theta)$, the parameters are penalised by the regularised loss function given by:

$$\hat{J}(\theta;x,y) = J(\theta;x,y) + \alpha\Omega(\theta) \tag{A.25}$$

where $\hat{J}$ and J are regularised and original loss functions respectively. α is a hyperparameter that controls the weight of the parameter norm penalty term. For example, $\alpha = 0$ equals to no regularisation, and large α values result in strong regularisation. In order to achieve different solutions, different choice of penalty terms Ω are adopted. In this section, the wide used L^1 and L^2 penalty terms are detailed.

L^1 Parameter Regularisation

L^1 parameter regularisation is a kind of weight decay method that penalise the size of the parameters of models. The penalisation trends to optimise some of the parameters to have 0 values, resulting in sparse parameters and features. This sparse property has been extensively used by feature selection methods that aim to select subsets from existing features.

Formally, L^1 penalty term can be defined as the sum of the absolute values of all parameters: $\Omega(\theta) = \|w\|_1 = \sum_i |w_i|$. Thus, the regularised loss function with L^1 parameter regularisation is formulated as:

$$\hat{J}(w;x,y) = J(w;x,y) + \alpha \|w\|_1 , \tag{A.26}$$

and the corresponding gradient function:

$$\nabla_w \hat{J}(w;x,y) = \nabla_w J(w;x,y) + \alpha sign(w), \tag{A.27}$$

where $sign(w)$ is the sign of every element in w, and ∇ represents gradients.

L^2 Parameter Regularisation

L^2 parameter regularisation is one of the most common parameter norm penalty, which is also called weight decay or ridge regression. This penalisation trends to drive all the weights closer to the origin. More specifically, it can regularise the parameters and make them close to any specific point, which is usually 0 since it is hard to verify the best point.

Formally, L^2 penalty term can be defined as the sum of the square values of all parameters: $\Omega(\theta) = \frac{1}{2}\|w\|_2^2 = \frac{1}{2}w^T w$. Thus, the regularised loss function with L^2 parameter regularisation is formulated as:

$$\hat{J}(w;x,y) = J(w;x,y) + \frac{\alpha}{2}\|w\|_2^2, \tag{A.28}$$

and the corresponding gradient function:

$$\nabla_w \hat{J}(w;x,y) = \nabla_w J(w;x,y) + \alpha w, \tag{A.29}$$

A.3.4 Early Stopping

Early stopping is a simple and effective regularisation method in deep learning. Giving a validation set, early stopping checks the validation error of each iteration, and stores a copy of model parameters with the lowest validation error. If the validation error cannot be reduced for a preset amount of time, the training is stopped and the parameters with the lowest validation error are returned. Early stopping is a parameters selection algorithm that does not influence the underlying training setups, such as network architecture, loss function, learning rate and optimisation algorithm. Therefore, early stopping can be easily used in conjunction with other regularisation methods.

A.4 Network Optimisation

A.4.1 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is the most widely used optimisation algorithm in machine learning and deep learning. For each iteration, SGD samples a mini-batch dataset from entire dataset, and then updates parameters of model with the average gradients of the mini-batch samples. The detailed update rule of SGD is given by:

$$g = \frac{1}{m}\nabla_\theta \sum_i L(f(x^i;\theta), y^i)$$
$$\theta = \theta - \eta g$$

where m is the batch size of a mini-batch B with inputs $\{x^1,...,x^m\}$ and target outputs $\{y^1,...,y^m\}$. g is the computed gradients from loss function and θ is model parameters. η is the learning rate that is a crucial parameter in SGD. In practice, η should be gradually decreased during training. The strategies of choosing learning rates are an art more than a science. Normally, the learning rates are chosen by iterative trail and careful observation of loss value and error.

A.4.2 Momentum

SGD is a very popular optimisation algorithm. However, the training speed of model with SGD is quite slow sometimes. In order to improve training speed, momentum [182] is developed to accelerate training by using accumulated gradients of past updates. As its name indicates, momentum optimisation algorithm adopts the concept of *momentum* from

physics. Momentum in physics can be calculated by mass and velocity, denoted as $p = mv$. In momentum optimisation algorithm, mass is set to unit and velocity v is regarded as gradient. The detailed update rule of momentum is given by:

$$
\begin{aligned}
g &= \frac{1}{m}\nabla_\theta \sum_i L(f(x^i;\theta), y^i) \\
v &= \alpha v - \eta g \\
\theta &= \theta + v
\end{aligned}
$$

where v accumulates the gradients g in each iteration controlled by a factor α. And in each iteration, the final updated gradients in momentum are composed of two parts, one is gradients g of the loss function and the other is moving average gradients v accumulated over past iterations. The training speed is boosted at the beginning of training. And the training is more stable when the gradient direction changes. Furthermore, sometimes the model may jump into a local minimum and g trend to 0, then v can help model jump out from the local minimum.

A.4.3 Nesterov

Nesterov [183] is a variant optimisation algorithm that is developed based on momentum [182]. The detailed update rule of Nesterov is given by:

$$
\begin{aligned}
\hat{\theta} &= \theta + \alpha v \\
g &= \frac{1}{m}\nabla_{\hat{\theta}} \sum_i L(f(x^i;\hat{\theta}), y^i) \\
v &= \alpha v - \eta g \\
\theta &= \theta + v
\end{aligned}
$$

where v and α works similarly as in momentum optimisation algorithm. The only difference between Nesterov and momentum is the update sequence of gradients. In momentum, first, g is calculated by loss function, and then accumulated to v by $v = \alpha v - \eta g$, then, model parameters θ are updated with v. However, if fact, the term of αv for the next iteration is already known in current iteration. Therefore, in Nesterov, a correction factor αv is added to θ before g is calculated. Nesterov shows improved performance in some cases.

A.4.4 AdaGrad

Learning rate of training a neural network is considered as one of the most important hyperparameters that has significant impact to model performance. In the AdaGrad [184] optimisation algorithm, the individual learning rate of each model parameter is automatically adapted to historical gradients. The detailed update rule of AdaGrad is given by:

$$
\begin{aligned}
g &= \frac{1}{m}\nabla_\theta \sum_i L(f(x^i;\theta),y^i) \\
r &= r + g \odot g \\
\Delta\theta &= -\frac{\eta}{\varepsilon+\sqrt{r}} \odot g \\
\theta &= \theta + \Delta\theta
\end{aligned}
$$

where r is the sum of historical squared gradients. $\odot$ stands for element-wise multiplication. Δ presents the difference of θ. ε is a constant for numerical stability, a common value perhaps be 10^{-7}. At the beginning of training, the term r of AdaGrad is able to enlarge gradients. In the middle of training, r restricts gradients. However, once r becomes too large, the gradients will be excessively restricted, and the training is stopped prematurely.

A.4.5 RMSProp

The RMSProp [185] optimisation algorithm further improved the AdaGrad by using an exponentially decaying average to historical gradients. The AdaGrad optimisation algorithm is originally designed to achieve fast convergence rate for convex functions. In contrast, the learning trajectory of non-convex functions are usually longer, and the learning rate in AdaGrad may have been decayed too much before the training has reaches a locally convex bowl. However, RMSProp applies an exponentially decaying average, ensuring it can successfully converge after reaching such a convex bowl. The detailed update rule of RMSProp is given by:

$$
\begin{aligned}
g &= \frac{1}{m}\nabla_\theta \sum_i L(f(x^i;\theta),y^i) \\
r &= \alpha r + (1-\alpha) g \odot g \\
\Delta\theta &= -\frac{\eta}{\sqrt{\varepsilon + r}} \odot g \\
\theta &= \theta + \Delta\theta
\end{aligned}
$$

where r is the exponentially decaying average of historical squared gradients. The rest terms are similar with the AdaGrad.

A.4.6 AdaDelta

The AdaDelta [186] optimisation algorithm is fully adaptive method that do not rely a global learning rate. The detailed update rule of AdaDelta is given by:

$$
\begin{aligned}
g &= \frac{1}{m}\nabla_\theta \sum_i L(f(x^i;\theta),y^i) \\
r &= \alpha r + (1-\alpha) g \odot g \\
\Delta\theta &= -\sqrt{\frac{s+\varepsilon}{r+\varepsilon}} \odot g \\
s &= \alpha s + (1-\alpha)\Delta\theta \odot \Delta\theta \\
\theta &= \theta + \Delta\theta
\end{aligned}
$$

where r is the exponentially decaying average of historical squared gradients, and s is exponentially decaying average of historical updated gradients. As the term of learning rate η is not included, a global learning rate is not necessary in AdaDelta.

A.4.7 Adam

The Adam [187] optimisation algorithm is also able to adapt learning rate. It can be regarded as the combination of momentum and RMSProp, but with several important improvements. Generally, Adam is the most robust optimisation algorithm that works very stable with various hyperparameters. The detailed update rule of Adam is given by:

$$
\begin{aligned}
t &= t+1 \\
g &= \frac{1}{m}\nabla_\theta \sum_i L(f(x^i;\theta),y^i) \\
v &= \alpha v + (1-\alpha)g \\
r &= \beta r + (1-\beta)g \odot g \\
\hat{v} &= \frac{v}{1-\alpha^t} \\
\hat{r} &= \frac{r}{1-\beta^t} \\
\Delta\theta &= -\frac{\eta}{\varepsilon+\sqrt{\hat{r}}} \odot \hat{v} \\
\theta &= \theta + \Delta\theta
\end{aligned}
$$

where t counts the iteration. v and r are the biased first order and second order moments of gradients respectively, $\hat{v}$ and $\hat{r}$ are the corresponding unbiased moments.