THE UNIVERSITY
of ADELAIDE

Dynamic Scene Understanding with Applications to Traffic Monitoring

Qichang Hu

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

November 2017

Abstract

Many breakthroughs have been witnessed in the computer vision community in recent years, largely due to deep Convolutional Neural Networks (CNN) and large-scale datasets. This thesis aims to investigate dynamic scene understanding from images. The problem of dynamic scene understanding involves simultaneously solving several sub-tasks including object detection, object recognition, and segmentation. Successfully completing these tasks will enable us to interpret the objects of interest within a scene.

Vision-based traffic monitoring is one of many fast-emerging areas in the intelligent transportation system (ITS). In the thesis, we focus on the following problems in traffic scene understanding. They are 1) How to detect and recognize all the objects of interest in street view images? 2) How to employ CNN features and semantic pixel labelling to boost the performance of pedestrian detection? 3) How to enhance the discriminative power of CNN representations for improving the performance of fine-grained car recognition? 4) How to learn an adaptive color space to represent vehicle images for vehicle color recognition?

For the first task, we propose a single learning based detection framework to detect three important classes of objects (traffic signs, cars, and cyclists). The proposed framework consists of a dense feature extractor and detectors of these three classes. The advantage of using one common framework is that the detection speed is much faster, since all dense features need only to be evaluated once and then are shared with all detectors. The proposed framework introduces spatially pooled features as a part of aggregated channel features to enhance the robustness to noises and image deformations. We also propose an object subcategorization scheme as a means of capturing the intra-class variation of objects.

To address the second problem, we show that by re-using the convolutional feature maps (CFMs) of a deep CNN model as visual features to train an ensemble of boosted decision forests, we are able to remarkably improve the performance of pedestrian detection without using specially designed learning algorithms. We also show that semantic pixel labelling can be simply combined with a pedestrian detector to further boost the detection performance.

Fine-grained details of objects usually contain very discriminative information which are crucial for fine-grained object recognition. Conventional pooling strategies (e.g. max-pooling, average-pooling) may discard these fine-grained details and hurt the

recognition performance. To remedy this problem, we propose a spatially weighted pooling (swp) strategy which considerably improves the discriminative power of CNN representations. The swp pools the CNN features with the guidance of its learnt masks, which measures the importance of the spatial units in terms of discriminative power.

In image color recognition, visual features are extracted from image pixels represented in one color space. The choice of the color space may influence the quality of extracted features and impact the recognition performance. We propose a color transformation method that converts image pixels from the RGB space to a learnt space for improving the recognition performance. Moreover, we propose a ColorNet which optimizes the architecture of AlexNet and embeds a mini-CNN of color transformation for vehicle color recognition.

Declaration

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

I give consent to this copy of my thesis, when deposited in the University Library, being made available for loan and photocopying, subject to the provisions of the Copyright Act 1968.

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

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

Signed: **Date**:

Publications

This thesis is based on the content of the following journal papers:

- **Qichang Hu**, Sakrapee Paisitkriangkrai, Chunhua Shen, Anton van den Hengel, Fatih Porikli. "Fast Detection of Multiple Objects in Traffic Scenes With a Common Detection Framework", IEEE Transactions on Intelligent Transportation Systems (TITS), 2015, DOI: 10.1109/TITS.2015.2496795. (incorporated as Chapter 3)

- **Qichang Hu**, Peng Wang, Chunhua Shen, Anton van den Hengel, Fatih Porikli. "Pushing the Limits of Deep CNNs for Pedestrian Detection", IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), 2017, DOI: 10.1109/TCSVT.2017.2648850. (incorporated as Chapter 4)

- **Qichang Hu**, Huibing Wang, Teng Li, Chunhua Shen. "Deep CNNs with Spatially Weighted Pooling for Fine-grained Car Recognition", IEEE Transactions on Intelligent Transportation Systems (TITS), 2017, DOI: 10.1109/TITS.2017.2679114. (incorporated as Chapter 5)

In addition, I have co-authored the below papers:

- Biyun Sheng, **Qichang Hu**, Jun Li, Wankou Yang, Baochang Zhang, Changyin Sun. "Filtered Shallow-Deep Feature Channels for Pedestrian Detection", Journal of Neurocomputing, 2017, DOI: 10.1016/j.neucom.2017.03.007

- Teng Li, **Qichang Hu**, Chunhua Shen, Yidong Li. "Spatially Weighted Pooling in Deep Convolutional Neural Networks for Fine-grained Visual Recognition", IEEE Transactions on Multimedia (TMM). (under review)

Acknowledgments

First of all, I would like to express the deepest gratitude to my principle supervisor, Prof. Chunhua Shen, for his in-depth guidance on my research and generous support for my life. He has been very supportive since the days I contacted him for a postgraduate study. He always made himself available to me for discussing any research problems and potential directions that I brought. I believe that this thesis would not have been possible without his supervision and constant help.

I would like to thank the other members in my advisory team, Dr. Peng Wang and Prof. Anton van den Hengel, for their constructive advice and insightful feedback. I would also like to thank my former co-supervisor Dr. Sakrapee Paisitkriangkrai for his valuable support and insightful advice over the first year of my Ph.D. study.

I thank my supportive friends and colleagues at the University of Adelaide. I would like to thank Yuanzhouhan Cao, Ruizhi Qiao, Hui Li, Yao Li, Teng Li, Bohan Zhuang, and Peng Wang, for not only the research discussions but also the happy time we enjoyed together.

Last but not at least, I would like to express my great appreciation to my family for their unconditional love and endless support throughout the past three years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

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

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

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

networks (CNN) (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015; He et al., 2016), which have significantly outperformed comparable methods on a wide variety of vision problems. Compared to hand-crafted features, deep CNNs are able to learn semantically meaningful features within the end-to-end training process, when given a large-scale annotated dataset. Recent studies (Donahue et al., 2014; Azizpour et al., 2016) have shown that features learned by CNNs pre-trained on the ImageNet dataset (Deng et al., 2009) tend to be generic, hence are beneficial to many other vision tasks. For example, a region-based convolutional neural network (R-CNN) (Girshick et al., 2014b) has shown that fine-tuned CNN features achieved excellent performance for generic object detection. For the task of fine-grained object classification, the performance can be significantly boosted by employing the fine-tuned CNN features (Liu et al., 2015; Lin et al., 2015c). Moreover, the emergence of the CNN features and CNN architectures do introduce some new directions to solve existing vision problems.

1.2 Problem formulations

This thesis focuses on the following applications of traffic scene understanding: a common detection framework to fastly detect multiple objects of interest in traffic scenes, pedestrian detection, fine-grained car recognition, and vehicle color recognition.

1.2.1 A common detection framework to fastly detect multiple objects of interest in traffic scenes

Object detection is an important application of traffic scene understanding which aims to extract accurate real-time on-road environment information. The ability to detect multiple objects (traffic signs, cars, and cyclists) of interest effectively plays a crucial role in the task. One challenge is that the detector need to detect objects of different classes simultaneously. Most previous methods have designed specific detectors using different features for each of these three classes. None of them can detect all the objects of three classes at the same time.

We address this problem by proposing a single learning based detection framework. The proposed framework consists of a dense feature extractor and detectors of these three classes. Once the dense features have been extracted, these features are shared with all detectors. The advantage of using one common framework is that the detection speed is much faster, since all dense features need only to be evaluated once. The proposed framework introduces spatially pooled features (Paisitkriangkrai

et al., 2014b) as a part of aggregated channel features (Dollár et al., 2010) to enhance the feature robustness to noises and image deformations. In order to further improve the generalization performance, we propose an object subcategorization scheme as a means of capturing the intra-class variation of objects.

1.2.2 Pedestrian detection

Pedestrian detection is an important problem due to its practical use in many computer vision applications. The problem is made challenging by the inevitable variation in target appearance, illumination, pose, and by occlusion. Prior to the very recent work in deep convolutional neural networks (DCNNs) based methods (Cai et al., 2015; Tian et al., 2015), the top performing pedestrian detectors are boosted decision forests with carefully hand-crafted features.

Recently, DCNNs have significantly outperformed comparable methods on a wide variety of vision problems. A region-based convolutional neural network (R-CNN) (Girshick et al., 2014a) achieved excellent performance for generic object detection. Later, R-CNN was extended to the Fast R-CNN (Girshick, 2015) which significantly increases the detection speed and accuracy. The performance of pedestrian detection is improved over hand-crafted features by a large margin by two very recent approaches relying on DCNNs: CompACT-Deep (Cai et al., 2015) combines hand-crafted features and fine-tuned DCNNs into a complexity-aware cascade. DeepParts (Tian et al., 2015) fine-tunes a pool of part detectors using a pre-trained GoogLeNet (Szegedy et al., 2015) and delivers similar results as CompACT-Deep. Both approaches are much more sophisticated than the standard R-CNN framework: CompACT-Deep involves the use of a variety of hand-crafted features, a small CNN and a large VGG16 model (Simonyan and Zisserman, 2015). DeepParts contains 45 fine-tuned DCNN models and needs a set of strategies (including bounding-box shifting handling and part selection) to arrive at the reported result. Note that the high complexity of DCNNs can lead to practical difficulties.

We propose alternative methods for pedestrian detection, which are simpler in design, with comparable or even better performance. Firstly, we extensively evaluate the convolutional feature maps (CFMs) extracted from multiple convolutional layers of a fine-tuned VGG16 model for pedestrian detection. Using only a CFM of a single convolutional layer, we train a boosted-tree-based detector and the resulting model already significantly outperforms all conventional methods. Next, we show that the CFMs from multiple convolutional layers can be used for training effective boosted decision forests. These boosted decision forests can be combined altogether simply by score averaging. The resulting ensemble model beats all competing CNN-based

methods. The detection performance can be further improved by incorporating a semantic pixel labelling model.

1.2.3 Fine-grained car recognition

Compared to generic objects, cars have a unique hierarchical structure, which contains three levels from top to bottom: car make, car model, and year of manufacture. Fine-grained car recognition aims to distinguish subcategories within the same car category. Car model classification is a intra-class classification task which is made difficult by the small visual differences between subcategories, unconstrained poses, different illuminations, and cluttered backgrounds. In this thesis, we mainly focus on car model classification.

A common approach for fine-grained object classification tasks is the parts-based pooling strategy (Zhang et al., 2012). Another line of research focuses on the robust feature representations of images, such as the VLAD (Jégou et al., 2010), Fisher vector (Perronnin et al., 2010) with SIFT features (Lowe, 1999). A breakthrough was made recently by the cross-convolutional-layer pooling method (Liu et al., 2015). This method extracts subarrarys of convolutional feature maps (CFMs) as local features and uses the CFMs of the successive convolutional layer as pooling channels. Then, the extracted features are pooled with these pooling channels to generate more robust image representations. Although this method achieves good results, it does not support the end-to-end training.

Considering the promising performance of deep CNNs in image classification, we propose a spatially weighted pooling (swp) strategy which considerably enhances the discriminative power of feature representations of CNNs for fine-grained car recognition. The swp is a novel pooling layer which contains a predefined number of spatially weighted masks or pooling channels. The swp pools CNN features with the guidance of its learnt masks, which measures the importance of the spatial units in terms of discriminative power. In addition, the swp layer is compatible with most dominant CNNs and the parameters of the swp layer can be learned in the end-to-end training process.

1.2.4 Vehicle color recognition

Vehicle color is an important property for vehicle identification and provides visual cues for many applications, such as vehicle detection, vehicle tracking, law enforcement, etc.. One challenging is that vehicle color can be easily influenced by illumination changes and uncontrolled environments. The quality of input images or videos also limits the recognition performance.

Most previous methods handle this problem by designing hand-crafted color histograms. Since color images are usually represented in RGB space, RGB histogram is a natural representation for color recognition. However, the RGB space is very sensitive to illumination change since all three channels include a representation of brightness. The normalized RGB histogram (Kender, 1976) is proposed to normalize raw RGB pixels to increase the illumination invariance. The Hue histogram (Van de Weijer et al., 2006) represents color information without illumination, thus it is partially invariant to illumination change. Although these methods obtain reasonable performance gains, they are still far from being satisfactory in real-world scenarios.

In image color recognition, visual features are usually extracted from image pixels represented in one color space. The choice of the color space may influence the quality of extracted features and impact the recognition performance. This observation motivates us to propose a color transformation method that converts input images from the RGB space to a learnt space. Moreover, we propose a small CNN architecture, called ColorNet, which optimizes the architecture of AlexNet (Krizhevsky et al., 2012) and embeds a mini-CNN of color transformation for vehicle color recognition.

1.3 Main contributions

The main contributions of this thesis include a set of algorithms for several applications of traffic scene understanding (i.e. traffic sign detection, car detection, pedestrian detection, fine-grained car recognition, and vehicle color recognition). More specifically, they are:

- We propose a single learning based detection framework to effectively and efficiently detect three important classes of objects (traffic signs, cars, cyclists) in traffic scenes. The advantage of this framework is that the detection speed is much faster, since all dense features need only to be evaluated once and then are shared with all detectors. We also employ spatially pooled features as a part of aggregated channel features to enhance the robustness to noises and image deformations.

- We propose a subcategorization scheme for capturing the intra-class variation of objects. This method applies spectral clustering to geometrical features of training samples to generate multiple subcategories. The subcategorization scheme can simplify the original learning problem by dividing it into multiple sub-problems.

- We propose an ensemble of boost decision forests with CNN features for pedes-

trian detection. The convolutional feature maps (CFMs) extracted from multiple convolutional layers of a fine-tuned VGG16 model can be used for training effective boosted decision forests. Our key insight is that an ensemble of these boosted decision forests further improves the detection performance and outperforms all previous methods.

- We propose to use a semantic pixel labelling model to enhance the predictions of a pedestrian detector. We use the weighted sum of pixel-labelling scores within a proposal region to represent the score of the proposal. This score can be used to enhance the detector confidence of the same proposal region. Our method shows that the detection performance of our ensemble model can be further improved by incorporating the pixel labelling model.

- We propose a spatially weighted pooling (swp) strategy which considerably enhances the discriminative power of feature representations of most dominant deep CNNs for fine-grained car recognition. We also collect two challenging fine-grained car datasets: CarFlag-563 and CarFlag-1532. These two datasets are more challenging than existing datasets due to the large number of car objects and the rich diversity of car models.

- We propose a color transformation method that converts image pixels from the RGB space to a learnt space. We show that the recognition performance can be boosted by employing this color transformation. Moreover, we propose a ColorNet which optimizes the architecture of AlexNet and embeds a mini-CNN of color transformation for vehicle color recognition.

1.4 Thesis outline

The structure of this thesis is organized as follows.

In chapter 2, we first review some common hand-crafted features and boosting algorithms which have been widely used in many visual problems. Also, we give a detailed literature review on convolutional neural networks and their applications to visual classification and detection.

In chapter 3, a common detection framework for fast detection of multiple objects of interest is given. The proposed framework can detect three important classes of objects (traffic signs, cars, cyclists) in traffic scenes. An object subcategorization scheme is also proposed as a means of capturing the intra-class variation of objects.

In chapter 4, we explore how to apply deep CNNs in the context of pedestrian detection. We design a simple but effective algorithm for training a pedestrian detector.

A pixel labelling method is also proposed to cooperate with a pedestrian detector for improving the detection performance. The proposed algorithm fills in the gaps in chapter 3.

In chapter 5, we focus on the problem of fine-grained car model classification. We propose a spatially weighted pooling (swp) strategy which considerably improves the discriminative poser of feature representations of most dominant deep CNNs. Two challenging fine-grained car datasets are given in this chapter. This approach can be used to post-process outputs of the detection framework in chapter 3 to get detailed vehicle information.

In chapter 6, we concentrate on the vehicle color recognition that is an extension of the problem in chapter 5. A color transformation method is proposed to convert image pixels from the RGB space to one learnt space. We also propose a light-weight ColorNet which is specifically designed for vehicle color recognition.

In chapter 7, the conclusion and the potential research directions are discussed.

Chapter 2

Literature Review

In this chapter, we will firstly review some of the conventional hand-crafted features to object recognition and detection. Then, we will introduce the adaptive boosting algorithm which is the most commonly used classification technique. Finally, some background information on deep convolutional neural networks will be introduced.

2.1 Hand-crafted features

Real-world data such as images and videos are usually complex, redundant, and highly variable. However, computer vision tasks such as recognition and detection often require input that is mathematically and computationally convenient to process. Thus, it is necessary to discover some useful feature representations from raw data. This motivates the design of hand-crafted features based on the prior domain-specific knowledge. Many different hand-crafted features (Lowe, 2004; Dalal and Triggs, 2005; Dalal et al., 2006; Ahonen et al., 2004; Tuzel et al., 2006; Dollár et al., 2014; Paisitkriangkrai et al., 2014b; Nam et al., 2014; Zhang et al., 2015) are proposed in the literature. We briefly introduce some of them used in this thesis.

2.1.1 Scale invariant feature transform

The scale invariant feature transform (SIFT) method (Lowe, 2004) are used to detect and describe local features in images. Keypoints of a image can be extracted by the SIFT detector using the appropriate level of the gaussian pyramid of the image. Therefore, the SIFT features are generated by computing the gradient at each pixel in a 16×16 patch around the detected keypoints. In each 4×4 quadrant, a histogram of gradient orientations is computed by adding the weighted gradient value to one of eight orientation histogram bins. Histogram bins of each quadrant within the 16×16 patch are concatenated as a 128-D vector. The 128-D vector represents the SIFT features for the patch. To reduce the negative effects of contrast or gain, the 128-D vector is normalized to unit length. Another popular variant of SIFT is PCA-SIFT (Ke

and Sukthankar, 2004), which computes gradient derivatives over a 39×39 patch and reduces the resulting 3042-D vector to 36 using principal component analysis (PCA).

2.1.2 Histogram of oriented gradients

The histogram of oriented gradients (HOG) (Dalal and Triggs, 2005) are visual descriptors used in computer vision for the purpose of object detection and recognition. HOG is computed by evaluating well-normalized local histograms of image gradient orientations in a dense grid. The key idea is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. The implementation of HOG can be achieved by dividing the image window into small spatial regions called cells, and for each cell accumulating a histogram of gradient orientations or edge directions over all pixels within the cell. The combination of these histograms represents the descriptor for the image. To improve the robustness of HOG, the local histograms can be contrast-normalized by accumulating a measure of the intensity over larger spatial regions called blocks, and then using this value to normalize all cells within this block. The normalized features are partially invariant to change in illumination and shadowing.

2.1.3 Local binary patterns

The local binary patterns (LBP) (Ojala et al., 1996) is a powerful means of texture description for object recognition. The original LBP labels the pixels of an image by thresholding the 3×3-neighbourhood of each pixel with the center value and considering the result as a binary number. All binary results are concatenated to form an 8-bit length binary sequence with 2^8 different labels. The histogram of these 256 different labels can represent a texture descriptor. Later, the original LBP was extended by using neighbourhoods of different sizes (Ojala et al., 2002). The neighbourhood can be any radius and number of pixels by using circular neighbourhoods. Another extension to the original LBP is the uniform LBP (Ojala et al., 2002), which can be used to reduce the length of the binary sequence and implement a simple rotation invariant descriptor. This idea is motivated by the fact that some binary patterns occur more frequently in images than others.

2.1.4 Region covariance features

The region covariance features (Tuzel et al., 2006) employs covariance matrices as region descriptors for object detection. Let I be a one dimensional grayscale image or three dimensional color image. Let F be the $W \times H \times d$ dimensional features

extracted from I

$$F(x, y) = \phi(I, x, y) \tag{2.1}$$

where the function ϕ can be any transformation. Given a rectangular region $R \subset F$, we use $\{z_k\}_{k=1,\ldots,n}$ to represent the d-dimensional feature points in R. The descriptor of the region R can be represented by the $d \times d$ covariance matrix of the feature points

$$\mathbf{C}_R = \frac{1}{n-1} \sum_{k=1}^{n} (z_k - \mu)(z_k - \mu)^T \tag{2.2}$$

where μ is the mean of the points.

There are several advantages of using these covariance matrices as region descriptors. A covariance matrix extracted from a region usually contains enough information to match the region in different viewpoints and poses. Compared to other visual descriptors, the covariance matrices are low-dimensional due to the symmetry property of $\mathbf{C}_R$. Moreover, the covariance features are partially invariant to change in scale and rotation with the appropriate choice of ϕ.

2.1.5 Aggregated channel features

Aggregated channel features (ACF) (Dollár et al., 2014) combine various features that are extracted from multiple image channels using pixel lookups method. Many image channels are available for extracting features. A trivial channel of a grayscale image is the image itself. For a color image, each color channel can be used as a channel. Other channels can be computed using various transformations of the input image. In order to accelerate the speed of feature extraction, all transformations are required to be translationally invariant. It means that the transformation need only to be evaluated once on the entire image rather than separately for each overlapping detection window. ACF consists of 10 feature channels: LUV color channels (3 channels), histogram of oriented gradients (6 channels), and normalized gradient magnitude (1 channel). ACF achieves state-of-the-art performance in the pedestrian detection problem by employing carefully designed channel features.

2.2 AdaBoost

Boosting is a supervised method for improving the accuracy of any given learning algorithm. Adaptive Boosting (AdaBoost) (Freund and Schapire, 1999) is one of the most popular boosting methods. We briefly introduce the training algorithm of AdaBoost and a variant of AdaBoost in this section.

2.2.1 Introduction

AdaBoost is a classification technique which combines multiple weak classifiers into a single strong classifier, where the weak classifiers perform only slightly better than random guesses. The principle of the algorithm is to learn a global binary decision function by iteratively adding and training weak classifiers. AdaBoost has been applied to many classification and recognition tasks. It has become a widely used machine learning technique due to its simplicity and performance in terms of accuracy and speed.

2.2.2 Training AdaBoost

AdaBoost takes as input a training set (x_1,y_1), ..., (x_N,y_N) where the $x_i \in X$ are the training samples, and $y_i \in Y$ are the corresponding labels. We focus here on the AdaBoost for binary classification, where $Y = \{-1, +1\}$. AdaBoost calls a given weak learning algorithm repeatedly in a series of rounds $t = 1, \ldots, T$. It maintains a distribution of weights over the training set. Initially, all weights are set equally to $1/N$. The weight of this distribution on the training sample x_i on round t is denoted as $D_t(i)$.

The training process is as follows: at each iteration $t = 1, \ldots, T$, a weak classifier $h_t : X \to \{-1, +1\}$ is trained using the training samples weighted by a set of weights $D_t(i)$, $i = 1, \ldots, N$. The performance of a weak classifier is measured by its error

$$\epsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(x_i) \neq y_i} D_t(i) \tag{2.3}$$

Once the weak classifier h_t has been trained, AdaBoost chooses a parameter a_t to determine the importance of the weak classifier in the final hypothesis.

$$a_t = \frac{1}{2} ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right) \tag{2.4}$$

Note that $a_t \geq 0$ if $\epsilon_t \leq 1/2$, and that a_t gets larger as ϵ_t gets smaller. Then, we update the distribution D_t. The weights of misclassified samples are increased and the weights of correctly classified samples are decreased. Thus, the weak classifier h_t is forced to focus on the hard examples in the training set. The final hypothesis $H(x)$ is a weighted majority vote of the T weak classifiers.

$$H(x) = sign\left(\sum_{t=1}^{T} a_t h_t(x)\right) \tag{2.5}$$

Input: The training set $S = \{(x_1, y_1), \cdots, (x_i, y_i), \cdots, (x_N, y_N)\}$, $x_i \in X$, $y_i \in \{-1, +1\}$, $i = 1, 2, \cdots, N$.

Initialize: The weighted distribution D of training set in the 1st round, $D_1 = (w_{1,1}, \cdots, w_{1,i}, \cdots, w_{i,N}), w_{1,i} = 1/N$, $i = 1, 2, \cdots, N$.

for $t = 1 \cdots T$ **do**

· Train the weak classifier h_t using the weighted distribution D_t,

$$h_t : X \to \{-1, +1\}$$

· Compute the error rate ϵ_t of h_t in training set S.

$$\epsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$$

· Compute the coefficient a_t of h_t.

$$a_t = \frac{1}{2} ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right)$$

· Update the weighted distribution of the training set

$$D_{t+1} = (w_{t+1,1}, \cdots, w_{t+1,i}, \cdots, w_{t+1,N})$$
$$w_{t+1,i} = \frac{w_{t,i}}{Z_t} \exp\left(-a_t y_i h_t(\vec{x}_i)\right), i = 1, 2, \cdots, N$$

where Z_t is a normalization factor,

$$Z_t = \sum_{i=1}^{N} w_{t,i} \exp\left(-a_t y_i h_t(\vec{x}_i)\right)$$

end

Output: Final classifier

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} a_t h_t(\vec{x})\right).$$

Algorithm 1: The boosting algorithm AdaBoost.

where a_t is the weight assigned to h_t. Algorithm 1 describes the basic Adaboost algorithm. The variable Z_t is a normalization factor in order to make D_{t+1} a distribution.

2.2.3 Shrinkage version of AdaBoost

The accuracy of AdaBoost can be further improved by applying a weighting coefficient known as shrinkage (Hastie et al., 2005). The shrinkage version of AdaBoost can be viewed as a form of regularization for boosting. At each iteration, the coefficient of weak learner is updated by

$$H_t(\vec{x}) = H_{t-1}(\vec{x}) + \nu \cdot w_t h_t(\vec{x}). \tag{2.6}$$

Input: The training set $S = \{(\vec{x}_1, y_1), \cdots, (\vec{x}_i, y_i), \cdots, (\vec{x}_N, y_N)\}$, $\vec{x}_i \subseteq \mathbb{R}^n$, $y_i \in \{-1, +1\}$, $i = 1, 2, \cdots, N$.

Initialize:The weighted distribution D of training set in 1st round, $D_1 = (w_{1,1}, \cdots, w_{1,i}, \cdots, w_{i,N}), w_{1,i} = 1/N$, $i = 1, 2, \cdots, N$.

for $t = 1 \cdots T$ **do**

· Train the weak learner $h_t(\cdot)$ using the weighted distribution D_t,

$$h_t(\cdot) : \mathbb{R}^n \to \{-1, +1\}$$

· Compute the error rate e_t of $h_t(\cdot)$ in training set S.

$$e_t = \sum_{i=1}^{N} w_{t,i} \cdot \mathbb{1}(h_t(\vec{x}_i) \neq y_i)$$

· Compute the coefficient w_t of $h_t(\cdot)$ and update it by multiplying shrinkage parameter ν.

$$w_t = \frac{1}{2} \log \frac{1 - e_t}{e_t}$$
$$a_t = \nu \cdot w_t$$

· Update the weighted distribution of the training set

$$D_{t+1} = (w_{t+1,1}, \cdots, w_{t+1,i}, \cdots, w_{t+1,N})$$
$$w_{t+1,i} = \frac{w_{t,i}}{Z_t} \exp\left(-a_t y_i h_t(\vec{x}_i)\right), i = 1, 2, \cdots, N$$

where Z_t is a normalization factor,

$$Z_t = \sum_{i=1}^{N} w_{t,i} \exp\left(-a_t y_i h_t(\vec{x}_i)\right)$$

end

Output: Final classifier

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} a_t h_t(\vec{x})\right).$$

Algorithm 2: Shrinkage version of AdaBoost

Here $h_t(\cdot)$ is a weak learner of AdaBoost at the t-th round and w_t is the coefficient of the weak learner. $\nu \in (0, 1]$ is a learning rate which controls the trade-off between overall accuracy and training time. The smaller the value of ν, the higher the overall accuracy as long as the number of weak learners is sufficiently large. Compared to the standard AdaBoost, shrinkage often produces better generalization performance (Friedman et al., 2000). Algorithm 2 describes the shrinkage version of AdaBoost algorithm.

2.3 Convolutional neural networks

Prior to the development of deep learning techniques, multi-layer feed-forward neural networks (NNs) have shown to be a very powerful machine learning technique as they can be trained to approximate complex non-linear functions from high-dimensional input data. NNs can be viewed as a trainable structure consisting of a set of inter-connected neurons, each implementing a simple function, and together performing a complex task. NNs consist of an input layer, a sequence of hidden layers, and an output layer. More specifically, an NN takes a single vector as input and each element of the vector forms one neuron in the input layer. These input data are then propagated to all neurons in the succeeding layer. Each hidden layer consists of a set of neurons and each neuron in the hidden layer connects to all neurons in the previous layer. The output of a neuron is generated by a weighted summation of the inputs from the previous layer followed by an activation function. The last fully-connected layer in an NN is referred to the output layer.

The problem with NNs is that when the dimension of input data is high, the number of inter-connections and the number of parameters is also high because each hidden unit would be fully connected to the previous layer. However, the number of training samples might be relatively small compared to the dimension of input data, which might lead to an over-fitting issue in the training of NNs. Another disadvantage is that NNs do not take into account correlations of neighbouring input data. In fact, there is generally a high amount of local correlation in the object recognition and detection.

Convolutional Neural Networks (CNNs) (LeCun et al., 1998) are an extension of the conventional NN that alleviates the above mentioned drawbacks. Compared to NNs, CNNs propose to implement the principle of weight sharing which remarkably reduces the number of parameters and thus increase the generalization capacity. Moreover, CNNs employ the pooling techniques to improve the robustness of CNN features to small translations and distortions in the input image. Nowadays, many deep CNNs achieve state-of-the-art performance at a wide range of computer vision problems, due to the large-scale datasets and powerful computational capacities.

2.3.1 Architecture

Compared to the conventional NNs, CNNs contain a variety of distinct layers, including convolutional layer, pooling layer, ReLU layer, and so on. We discuss them further below:

- *Convolutional layer*. The convolutional layer is the core building block of CNNs.

The parameters of this layer are a filter bank which consists of a set of learnable filters. Each filter has a small receptive field and the number of channels matches the depth of the input volume. In the forward pass, each filter is convolved with local regions at different spatial locations in the input volume with a fixed stride, computing the dot product between the coefficients of the filter and the input and producing a 2-dimensional feature map of that filter. Feature maps generated by all filters are concatenated in the channel dimension. The resulting feature maps are the output of this convolutional layer and serve as the input of the next layer. Therefore, the CNN learns multiple distinct filters that activate when it detects some specific patterns at some spatial location in the input image.

- *Pooling layer*. The pooling layer is a form of non-linear down-sampling. Max pooling and average pooling are two most commonly used pooling strategies in CNNs. Max-pooling takes the maximum value in a spatial region (e.g. 2×2) to represent the pooled features in the region. It aims to keep the most salient information and discard irrelevant details over the pooling region. Since the pooling layer reduces the spatial size of the feature representation, it can be used to reduce the number of parameters and the amount of computation in the CNN model. The pooling operation also alleviates the over-fitting and provides a form of translation invariance.

- *ReLU layer*. The rectified linear unit (ReLU) layer applies the non-saturating activation function $f(x) = max(0, x)$ to the input feature maps. It aims to increase the non-linearity of the feature representation. Compared to other activation functions (sigmoid, tangent), the ReLU addresses the gradient exploding or vanishing problem during the training phase. Thus, it accelerates the CNN training phase without degrading the performance.

- *Fully-connected layer*. The fully-connected layer in CNNs is the same as it in the conventional NNs. Each neuron in a fully-connected layer has full connections to all activations in the previous layer. Their activations can be computed by a matrix multiplication followed by a bias offset.

- *Loss layer*. The loss layer is used to penalize the deviation between the predicted and true labels during the training phase. Different loss functions are design for different tasks. Softmax loss and sigmoid cross-entropy loss are two commonly used loss functions in object classification and detection. Softmax loss is used for predicting a single class of K mutually exclusive classes. Sigmoid cross-entropy loss is used for predicting K independent probability values in $[0, 1]$.

The loss layer is usually the last layer in CNNs.

2.3.2 Extensions and variants

In this section, we briefly review some dominant CNNs in the computer vision community.

- *AlexNet* (Krizhevsky et al., 2012) is a recently major breakthrough in the field of deep learning. This network employs the ReLU layer and the dropout layer for the first time. It contains five convolutional layers and three fully-connected layers, followed by a 1000-way softmax. AlexNet won the the ImageNet large scale visual recognition challenge (ILSVRC) in 2012, which significantly outperformed other competitors by a large margin.

- *VGGNet* (Simonyan and Zisserman, 2015) obtained excellent performance in the ILSVRC 2014. Compared to AlexNet, VGGNet employs a relatively small (3×3) convolutional filters. The depth of the VGGNet can be increased up to 19 by stacking a sequence of these small filters. It has shown that the importance of deepening the CNN for achieving better performance in many recognition and detection tasks.

- *GoogleNet* (Szegedy et al., 2015) is a deep CNN architecture that employs an inception module to remarkably reduce the number of parameters in the network. By a carefully crafted design, the depth of GoogleNet is increased to 22 while keeping the computational cost constant. GoogleNet achieved the new state-of-the-art for classification and detection in the ILSVRC 2014.

- *ResNet* (He et al., 2016) is the state-of-the-art CNN architecture in the ILSVR 2015. In general, deeper CNNs are more difficult to train. To remedy this problem, ResNet adopted residual learning to every few stacks of layers by adding skip connections. By employing a good weights initialization method (He et al., 2015b) and batch normalization (Ioffe and Szegedy, 2015), ResNet achieved better performance than VGGNet and GoogleNet. Since there are no fully-connected layers, the number of parameters in ResNet is significantly reduced.

- *SqueezeNet* (Iandola et al., 2017) is a small CNN architecture which achieves AlexNet-level accuracy on ImageNet with 50x fewer parameters. By employing some model compression techniques, the SqueezeNet can be compressed to less than 0.5MB. Due to the small size of SqueezeNet, it can be deployed into some mobile devices to implement real-time detection or classification tasks.

2.3.3 Training CNNs

Backpropagation is the most commonly used method to train multi-layer feed-forward networks (e.g. NNs, CNNs). The algorithm repeatedly implement two phases: propagation and weight update. During the forward pass in the propagation phase, the input is fed to the network and is propagated forward through the network, until it reaches the output layer. The residual error of the output layer can be computed by using a loss function which compares the output of the network with the ground-truth. The residual error is then propagated backwards from the output layer to the input layer. Then, backpropagation uses the residual error to compute the gradient of the loss function with respect to the weights in the network using the chain rule. In the update phase, this gradient is fed to an optimization method which uses it to update the weights and attempts to minimize the loss function.

Backpropagation is usually used in conjunction with an optimization method such as stochastic gradient descent (SGD). Compared to gradient descent, SGD computes the gradient of the loss function using only a few training samples instead of the entire training set. The use of SGD is motivated by the high computational cost of implementing the backpropagation over the entire training set. These few training samples are usually organized in a mini-batch. A typical size of the mini-batch is 256 for training ImageNet. The optimal size of the mini-batch is determined by different applications and architectures. The learning rate a in SGD is typically a very small value (e.g. 0.001). A practical way to determine a is to use a small enough constant learning rate that gives stable convergence in initial epochs and then halve the value as convergence slows down.

The depth of CNNs is an important factor to achieve better performance in many applications. However, deeper CNNs are more difficult to train since the huge number of parameters may lead to the over-fitting. In order to remedy this issue, dropout proposed by (Srivastava et al., 2014) is a regularization technique to avoid over-fitting when training CNNs. The key idea is to randomly drop some neurons in the training phase. Batch normalization (Ioffe and Szegedy, 2015) is a useful method to accelerate the training process of CNNs. It addresses the gradient exploding or vanishing problem by reducing the internal covariate shift of each mini-batch. It also acts as a regularizer to avoid the over-fitting.

2.3.4 Applications

Nowadays, CNNs have demonstrated great success in many visual detection and recognition tasks. In this section, we review two fundamental tasks in image understanding, including image classification and object detection.

Image classification is the task of assigning a predefined label to the input image. Based on the subjects in the image, image classification tasks can be divided into different subcategories, such as generic object classification (person, vehicle, dog *etc*.) and fine-grained object classification(e.g. different models of vehicles). For generic object classification, the ImageNet dataset (Deng et al., 2009) is one of the most dominant benchmark datasets, and the ILSVRC competition (Russakovsky et al., 2015) has become the driving-force for deep learning research in the computer vision community. Some state-of-the-art CNNs, such as AlexNet (Krizhevsky et al., 2012), VGGNet (Simonyan and Zisserman, 2015) and ResNet (He et al., 2016), have achieved tremendous success in the ILSVRC classification tasks in different years. The fine-grained object classification aims to distinguish subcategories within the same object category. The best performance of the fine-grained object classification are achieved by recently proposed CNNs based methods (Lin et al., 2015c; Liu et al., 2015).

Object detection is a challenging but important research topic in the computer vision community. It has achieved successful outcomes in many practical applications such as face detection and pedestrian detection. Prior to the very recent work in deep CNNs based methods, the top performing object detectors are built upon hand-craft features (e.g. HOG, LBP, ACF, etc.) and a classifier (e.g. SVM, AdaBoost, etc.). A region-based convolutional neural network (R-CNN) (Girshick et al., 2014a) achieved excellent performance for generic object detection. Later, R-CNN was extended to the fast R-CNN (Girshick, 2015) which significantly increases the detection speed by introducing a Region-of-Interest pooling layer for fast feature extraction. Faster R-CNN (Ren et al., 2015) proposes to employ a region proposal network to replace traditional proposal generation methods (e.g. selective serach, edgeBox, Bings, etc.) Instead of generating object proposals, there are also some recent works (Redmon et al., 2016; Liu et al., 2016) which predicts bounding boxes of objects and class probabilities directly from full images in one evaluation.

Chapter 3

Fast Detection of Multiple Objects in Traffic Scenes with a Common Framework

3.1 Introduction

Vision-based traffic scene perception (TSP) is one of many fast-emerging areas in the intelligent transportation system. This field of research has been actively studied over the past decade (Sivaraman and Trivedi, 2013). TSP involves three phases: detection, recognition and tracking of various objects of interest. Since recognition and tracking often rely on the results from detection, the ability to detect objects of interest effectively plays a crucial role in TSP. In this chapter, we focus on three important classes of objects: traffic signs, cars, and cyclists. Fig. 3.1 shows a typical on-road traffic scene with the detected objects of interest and illustrates some positive examples from the three mentioned classes.

The aim of traffic sign detection is to alert the driver of the changed traffic conditions. The task is to accurately localize and recognize road signs in various traffic environments. Prior approaches (De La Escalera et al., 1997; de la Escalera et al., 2003; Kuo and Lin, 2007) use color and shape information. However, these approaches are not adaptive under severe weather and lighting conditions. Additionally, appearance of traffic signs can physically change over time, due to the weather and damage caused by accidents. Instead of using color and shape features, most recent approaches (Mathias et al., 2013b; Wang et al., 2013a) employ texture or gradient features, such as local binary patterns (LBP) (Ahonen et al., 2004) and histogram of oriented gradients (HOG) (Dalal and Triggs, 2005). These features are partially invariant to image distortion and illumination change, but they are still unable to handle severe deformations.

Car detection is a more challenging problem compared to traffic sign detection

Figure 3.1: Top image: A typical on-road traffic scene with the detected objects of interest. Bottom images: Each block represents one class of objects of interest. From left to right, the first block contains traffic sign examples, the second contains car examples, and the third contains cyclist examples.

due to its large intra-class variation caused by different viewpoints and occlusions. Although sliding-window based methods have shown promising results in face and human detection (Viola and Jones, 2004; Dalal and Triggs, 2005), they often fail to detect cars due to a large variation of viewpoints. Recently the deformable parts model (DPM) (Felzenszwalb et al., 2010), which has gained a lot of attention in generic object detection, has been adapted successfully for car detection (Geiger et al., 2011; Hejrati and Ramanan, 2012; Pepik et al., 2013). In addition to the DPM, visual subcategorization based approaches (Divvala et al., 2012; Kuo and Nevatia, 2009; Ohn-Bar and Trivedi, 2014) have been applied to improve the generalization performance of detection model.

Cyclist detection is a new attractive application in the domain of TSP. At present, only few methods are designed purposely for cyclist detection. Many existing pedestrian detection approaches (Dalal and Triggs, 2005; Dollár et al., 2014; Geiger et al., 2011) can be adapted for cyclist detection because appearances of pedestrians are very similar to appearances of cyclists along the road. Compared to pedestrian detection, the new problem is more difficult because the various appearances and viewpoints increase the diversity of cyclists. Therefore, existing pedestrian detectors hardly achieve the acceptable performance for cyclist detection.

Most previous methods have designed specific detectors using different features for each of these three classes. The approach we claim here differs from these existing

approaches in that we propose a single learning based detection framework to detect all the three important classes of objects. The proposed framework consists of a dense feature extractor and detectors of these three classes. Once the dense features have been extracted, these features are shared with all detectors. The advantage of using one common framework is that the detection speed is much faster, since all dense features need only to be evaluated once in the testing phase. The proposed framework introduces spatially pooled features (Paisitkriangkrai et al., 2014b) as a part of aggregated channel features (Dollár et al., 2009) to enhance the feature robustness to noises and image deformations. In order to further improve the generalization performance, we propose an object subcategorization method as a means of capturing the intra-class variation of objects.

3.2 Background

3.2.1 Generic object detection

Object detection is a challenging but important application in the computer vision community. It has achieved successful outcomes in many practical applications such as face detection and pedestrian detection (Viola and Jones, 2004; Ahonen et al., 2004; Dalal and Triggs, 2005; Wang et al., 2009). Complete survey of object detection can be found in (Viola and Jones, 2004; Dalal and Triggs, 2005; Felzenszwalb et al., 2010; Wang et al., 2013b; Girshick et al., 2014b). This section briefly reviews several generic object detection methods.

One classical object detector is the detection framework of Viola and Jones which uses a sliding-window search with a cascade classifier to achieve accurate location and efficient classification (Viola and Jones, 2004). The other commonly used framework is using a linear support vector machine (SVM) classifier with histogram of oriented gradients (HOG) features, which has been applied successfully in pedestrian detection (Dalal and Triggs, 2005). These frameworks achieve excellent detection results on rigid object classes. However, for object classes with a large intra-class variation, their detection performance falls down dramatically (Paisitkriangkrai et al., 2014b).

In order to deal with appearance variations in object detection, a deformable parts model (DPM) based method has been proposed in (Felzenszwalb et al., 2010). This method relies on a variant of HOG features and window template matching, but explicitly models deformations using a latent SVM classifier. It has been applied successfully in many object detection applications (Geiger et al., 2011; Torres et al., 2014; Yan et al., 2013). In addition to the DPM, visual subcategorization (Divvala

et al., 2012) is another common approach to improve the generalization performance of detection model. It divides the entire object class into multiple subclasses such that objects with similar visual appearance are grouped together. A sub-detector is trained for each subclass and detection results from all sub-detectors are merged to generate the final results. Recently, a new detection framework which uses aggregated channel features (ACF) and an AdaBoost classifier has been proposed in (Dollár et al., 2014). This framework uses exhaustive sliding-window search to detect objects at multi-scales. It has been adapted successfully for many practical applications (Ohn-Bar and Trivedi, 2014; Mathias et al., 2013b; Paisitkriangkrai et al., 2014b).

3.2.2 Traffic sign detection

Many traffic sign detectors have been proposed over the last decade with newly created challenging benchmarks. Interested reader should see (Mogelmose et al., 2012) which provides a detailed analysis on the recent progress in the field of traffic sign detection. Most existing traffic sign detectors are appearance-based detectors. These detectors generally fall into one of four categories, namely, color-based approaches, shape-based approaches, texture-based approaches, and hybrid approaches.

Color-based approaches (De La Escalera et al., 1997; de la Escalera et al., 2003; Kuo and Lin, 2007) usually employ a two-stage strategy. First, segmentation is done by a thresholding operation in one specific color space. Subsequently, shape detection is implemented and is applied only to the segmented regions. Since RGB color space is very sensitive to illumination change, some approaches (Fang et al., 2003; Maldonado-Bascón et al., 2007; Kuo and Lin, 2007) convert the RGB space to the HSI space which is partially invariant to light change. Other approaches (Janssen et al., 1993; De La Escalera et al., 1997) implement segmentation in the normalized RGB space which is shown to outperform the HSI space (Gómez-Moreno et al., 2010). Both the HSI and the normalized RGB space can alleviate the negative effect of illumination change, but still fail on some severe situations.

Shape-based approaches (Houben, 2011; Loy and Barnes, 2004; Timofte et al., 2009) detect edges or corners from raw images using canny edge detector or its variants. Then, edges and corners will be connected to regular polygons or circles by using Hough-like voting scheme. These detectors are invariant to illumination change, but the memory and computational requirement is quite high for large images. In (de la Escalera et al., 2003), a genetic algorithm is adopted to detect circles and is invariant to projective deformation, but the expensive computational requirement limits its application.

Texture-based approaches firstly extract hand-crafted features computed from

texture of images, and then use these extracted features to train a classifier. Popular hand-crafted features include HOG, LBP, ACF, etc (Dalal and Triggs, 2005; Ahonen et al., 2004; Dollár et al., 2014). Some approaches (Liang et al., 2013; Wang et al., 2013a; Pettersson et al., 2008) use the HOG features with a SVM, others (Mathias et al., 2013b) use the ACF features with an Adaboost classifier. Besides the above approaches, a convolutional neural network (CNN) is adopted for traffic sign detection and achieves excellent results in (Sermanet and LeCun, 2011).

Hybrid approaches (Gao et al., 2006; Prisacariu et al., 2010) are a combination of the aforementioned approaches. Usually, the initial step is the segmentation to narrow the search space, which is same as the color-based approaches. Instead of only using edges features or texture-based features, these methods use them together to improve the detection performance.

One standard benchmark for traffic sign detection is the German traffic sign detection benchmark (GTSDB) (Houben et al., 2013) which collects three important categories of road signs (prohibitory, danger, and mandatory) from various traffic scenes. All traffic signs have been fully annotated with the rectangular regions of interest (ROIs). Researchers can conveniently compare their work based on this benchmark.

3.2.3 Car detection

Many existing car detectors are vision-based detectors. Interested reader should see (Sivaraman and Trivedi, 2013) which discusses different approaches for vehicle detection using mono, stereo, and other vision-sensors. We focus on vision-based car detectors using monocular information in this chapter. These detectors can be divided into three categories: DPM-based approaches, subcategorization-based approaches and motion-based approaches.

DPM-based approaches are built on the deformable parts model (DPM) (Felzenszwalb et al., 2010) which has been successfully applied in car detection (Torres et al., 2014). In (Geiger et al., 2011), a variant of DPM discretizes the number of car orientations and each component of the mixture model corresponds to one orientation. The authors of (Hejrati and Ramanan, 2012) train a variant of DPM to detect cars under severe occlusions and clutters. In (Pepik et al., 2013), occlusion patterns are used as training data to train a DPM which can reason the relationships between cars and obstacles for detection.

Visual subcategorization which learns subcategories within an object class is a common approach to improve the model generalization in car detection (Divvala et al., 2012). It usually consists of two phases: feature extraction and clustering.

Samples with similar visual features are grouped together by applying clustering algorithm on extracted feature space. Subcategorization-based methods are commonly used with DPM to detect cars from multiple viewpoints. In (Kuo and Nevatia, 2009), subcategories of cars corresponding to car orientation are learned by using locally linear embedding method with HOG features. In (Ohn-Bar and Trivedi, 2014), cars with similar viewpoints, occlusions, and truncation scenarios are grouped into the same subcategory using a semi-supervised clustering method with ACF features.

Motion-based approaches often use appearance cues in monocular vision since monocular images do not provide any 3D and depth information. In (Broggi et al., 2008), adaptive background model is used to detect cars based on motion that differentiated them from the background. The authors of (Wang et al., 2005) propose an adaptive background model to model the area where overtaking cars tend to appear in the camera's field of view. Optical flow (Martinez et al., 2008), which is a popular tool in machine vision, has been used for monocular car detection. In (Kyo et al., 1999), a combination of optical flow and symmetry tracking is used for car detection. Optical flow is also used in conjunction with appearance-based techniques in (Cui et al., 2010).

The KITTI vision benchmark (KITTI) (Geiger et al., 2013) is a novel challenging benchmark for the tasks of monocular, stereo, optical flow, visual odometry, and 3D object detection. The KITTI dataset provides a wide range of images from various traffic scenes with fully annotated objects. Objects in the KITTI dataset includes pedestrians, cyclists, and vehicles.

3.2.4 Cyclist detection

Many existing cyclist detectors use pedestrian detection techniques since appearances of pedestrians are very similar to appearances of cyclists along the road (Qui et al., 2003; Rogers and Papanikolopoulos, 2000; Wang et al., 2006). In (Qui et al., 2003), corner feature extraction, motion matching, and object classification are combined to detect pedestrians and cyclists simultaneously. In (Wang et al., 2006), a stereo vision based approach is proposed for pedestrian and cyclist detection. It uses the shape features and matching criterion of partial Hausdorff distance to detect targets. The authors of (Rogers and Papanikolopoulos, 2000) propose a cyclist detector to detect two wheels of bicycles on road, but this approach is limited to detect crossing cyclists.

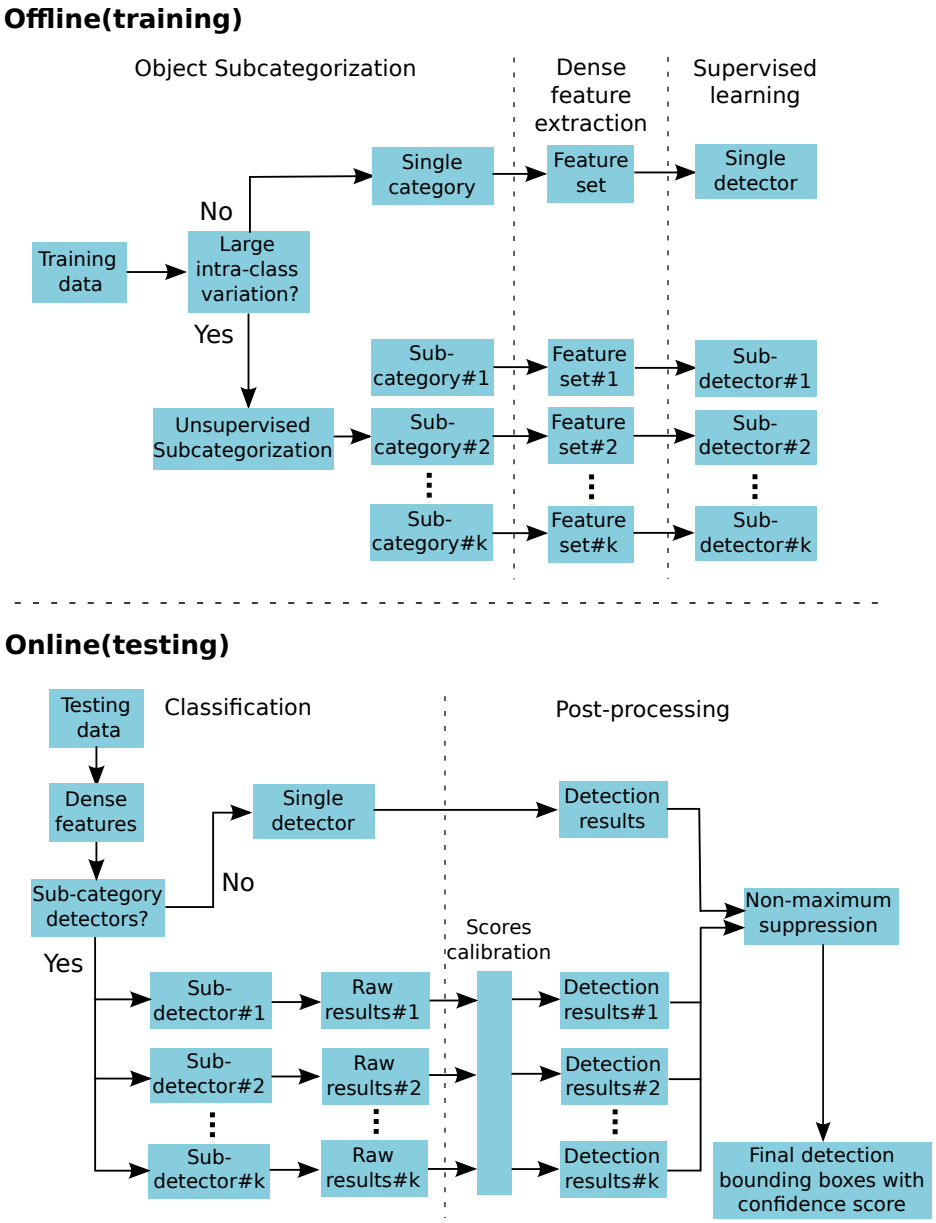

Figure 3.2: Overview of the proposed detection framework. Top diagram is the training phase and bottom diagram is the testing phase.

3.3 Proposed approach

Despite several important techniques have been proposed on object detection, the conventional sliding-window based method of Viola and Jones (Viola and Jones, 2004) is still the most successful and practical object detector. The VJ framework consists of two main components: a dense feature extractor and a cascade classifier. In

this chapter, we build a common object detection framework for traffic scene perception based on the VJ framework, but our framework can employ a number of different classifiers to detect target objects of different classes. Apart from basic components of the VJ framework, we propose an object subcategorization method to improve the generalization performance and employ spatially pooled features (Paisitkriangkrai et al., 2014b) to enhance the robustness and effectiveness.

Fig. 3.2 shows an overview of our framework. In the training phase, we firstly check the intra-class variation of the input object class with respect to object properties, e.g. size, orientation, aspect ratio, and occlusion. If the variation is considerable large, we apply the object subcategorization method to categorize the object class into multiple subcategories and train one sub-detector for each subcategory. Otherwise, we train a single detector for the entire object class. In the testing phase, raw detection results from all sub-detectors need to be calibrated before merging them together. Non-maximum suppression is used to eliminate redundant bounding boxes. If the framework employs detectors of different classes, detection results need to be carefully merged together.

3.3.1 Object Subcategorization

For object classes with a large intra-class variation like cars, the appearances and shapes of cars change significantly as viewpoints change. In order to deal with these variations that cannot be tackled by the conventional VJ framework, we present an object subcategorization method which aims to cluster the object class into visually homogeneous subcategories. The proposed subcategorization method applies an unsupervised clustering method to one specific feature space of the training samples to generate multiple subcategories. This method simplifies the original learning problem by dividing it into multiple sub-problems and improves model generalization performance.

3.3.1.1 Visual features

A variety of hand-designed features can be used to perform the clustering algorithm, such as HOG and ACF (Dalal and Triggs, 2005; Dollár et al., 2014). HOG is successful at capturing the shapes of objects while does not consider color information. ACF combines both color information and gradient information, which is shown to outperform HOG (Dollár et al., 2009). In our experiments, a total of 10 channels of features are used for clustering: LUV color channels (3 channels), histogram of oriented gradients at 6 bins (6 channels), and normalized gradient magnitude (1 channel). To extract features from the training samples, all samples are resized to the median

object size.

3.3.1.2 Geometrical features

Besides the visual features, geometrical information of objects can be extracted from traffic scenes using a variety of sensors and methods. In the KITTI dataset, objects in images from a velodyne laser scanner were annotated with 3D bounding boxes and 3D orientations. Ohn-Bar *et al.* (Ohn-Bar and Trivedi, 2015) propose an analysis of different types of geometrical features, which shows that the geometrical features outperform the visual features for clustering, even for the CNN features. We use the following set of geometrical features to represent the object instances in our experiments.

- **3D orientation**. The appearances and shapes of objects change significantly as viewpoints change. We include the 3D orientation (relative orientation between the object and the camera) in clustering, aiming at grouping objects with similar visual appearance together.
- **Aspect-ratio**. The aspect-ratio (width/height) of objects is strongly correlated with the geometry of objects being detected. We use this feature because learning models at different aspect-ratios significantly improve the generalization performance.
- **Truncation level**. The truncation level refers to the percentage of the object outside of the image boundaries. This feature strongly affects appearances of objects.
- **Occlusion index**. Instead of using subtle occlusion patterns defined in (Ohn-Bar and Trivedi, 2015), we use an occlusion index to indicate whether an object is not occluded, partially occluded, largely occluded or an unknown situation. We simplify the occlusion patterns because some occlusion features cannot be defined for each occluded object, such as occlusion level, relative orientation and relative 3D point between occluded objects and occluders. The above features are only available when the object is occluded by other labelled occluders. However, many occluders are unlabelled in the KITTI dataset.

3.3.1.3 Clustering

A clustering method is used to generate a predefined number of clusters on a specific feature space. Traditional clustering schemes, such as k-means or single linkage, suffer from the cluster degeneration which means that a few clusters claim most

data samples (Jain et al., 1999). The cluster degeneration problem can be alleviated by using spectral clustering. Spectral clustering followed by k-means often outperforms the traditional schemes. We implement the normalized spectral clustering using the algorithm proposed in (Ng et al., 2002). The quality of clustering results is very sensitive to the predefined number of clusters. Unfortunately, how to determine the appropriate number of centroids is still an open question. We experimentally determine the number of clusters for each application.

3.3.2 Feature extraction

The proposed framework introduces spatially pooled features (Paisitkriangkrai et al., 2014b) as a part of the aggregated channel features (Dollár et al., 2009) and employs them as dense features in the training phase. All feature channels are aggregated in 4×4 blocks in order to produce fast pixel lookup features.

3.3.2.1 Aggregated channel features (ACF)

Given an input image I, a channel C of I is a feature map, where the output pixels are computed from corresponding pixels of the input image. Aggregated channel features are extracted from multiple image channels using pixel lookups method. Many image channels are available for extracting features. For example, a trivial channel of a grayscale image is the image itself. For a color image, each color channel can be used as a channel. Other channels can be computed using various transformations of I. In order to accelerate the speed of feature extraction, all transformations are required to be translational invariant. It means that the transformation need only to be evaluated once on the entire image rather than separately for each overlapping detection window.

ACF uses the same channel features as **ChnFtrs** (Dollár et al., 2009): LUV color channels (3 channels), histogram of oriented gradients (6 channels), and normalized gradient magnitude (1 channel). ACF combines the richness and diversity of statistics from these channels, which is shown to outperform HOG (Dollár et al., 2014, 2009). Prior to computing these 10 channels, we smooth the input image I to suppress fine scale structures as well as noises.

- **LUV color channels**. LUV color space contains 3 channels, L channel describes the lightness of the object, U channel and V channel represent the chromaticity of the object. Compared to RGB space, LUV space is able to partially invariant to illumination change. So the proposed detector can work under different light conditions. Images can be converted to LUV space by using a specific transformation.

- **Gradient magnitude channel**. A normalized gradient magnitude is used to measure the edge strength. Gradient magnitude $M(x, y)$ at location (x, y) is computed by $\sqrt{I_x^2 + I_y^2}$, where I_x and I_y are first intensity derivatives along the x-axis and y-axis, respectively. Since the gradient magnitude is computed on 3 LUV channels independently, only the maximum response is used as the gradient magnitude channel.

- **Gradient histogram channels**. A histogram of oriented gradients is a weighted histogram where bin index is determined by gradient orientation and weighted by gradient magnitude (Dollár et al., 2009). The histogram of oriented gradients at location (x, y) is computed by $M(x, y) \cdot \mathbb{1}[\Theta(x, y) = \theta]$, where $\mathbb{1}$ is the indicator function, $M(x, y)$ and $\Theta(x, y)$ are the gradient magnitude and discrete gradient orientation, respectively. ACF quantizes the orientation space to 6 orientations and compute one gradient histogram channel for each orientation.

3.3.2.2 Spatially pooled features

Spatial pooling is used to combine multiple visual descriptors obtained at nearby locations into a lower dimensional descriptor over the pooling region. We follow the work of (Paisitkriangkrai et al., 2014b) which is shown that pooling can enhance the robustness of two hand-crafted low-level features, covariance features (Tuzel et al., 2006) and LBP (Ahonen et al., 2004).

Covariance matrix A covariance matrix is a positive semidefinite matrix which provides a measure of the relationship between multiple sets of variates. The diagonal elements of a covariance matrix represent the variance of each feature and non-diagonal elements represent the correlation between different features. In order to compute the covariance matrix, we use the following variates proposed in (Paisitkriangkrai et al., 2014b):

$$[x, y, |I_x|, |I_y|, |I_{xx}|, |I_{yy}|, M, O_1, O_2]$$

where x and y indicate the pixel location. I_x and I_y are first intensity derivatives along the horizontal-axis and vertical-axis respectively. Similarly, I_{xx} and I_{yy} are second intensity derivatives, respectively. M is the gradient magnitude $\sqrt{I_x^2 + I_y^2}$. O_1 is the edge orientation $\arctan(|I_x|/|I_y|)$ and O_2 is an additional edge orientation in which,

$$O_2 = \begin{cases} \text{atan2}(I_y, I_x) & \text{if atan2}(I_y, I_x) > 0, \\ \text{atan2}(I_y, I_x) + \pi & \text{otherwise.} \end{cases}$$

where the atan2 function is defined in terms of the arctan in the following:

$$\text{atan2}(y,x) = \begin{cases} \arctan\frac{y}{x} & x > 0 \\ \arctan\frac{y}{x} + \pi & y \geq 0, x < 0 \\ \arctan\frac{y}{x} - \pi & y < 0, x < 0 \\ +\frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ \text{undefined} & y = 0, x = 0 \end{cases}$$

The covariance descriptor of a region is a 9×9 covariance matrix which can be computed efficiently because the computational cost is independent of the size of the region. We also exclude the variance of pixel locations (x and y coordinates) and the correlation coefficient between pixel locations (x and y coordinates), since these features do not capture discriminative information. Due to the symmetry, each covariance descriptor finally contains 42 different values.

Spatially pooled covariance The spatial invariance and robustness of the covariance descriptors can be improved by applying pooling method. There are two common pooling methods in this context: average pooling and max pooling. Max pooling is used in our framework as it has been shown to outperform average pooling in image classification (Coates and Ng, 2011). Max-pooling uses the maximum value of a pooling region to represent the pooled features in the region. It aims to retain the most salient information and discard irrelevant details and noises over the pooling region. The image window is divided into multiple dense patches (refer to Fig. 3.3). Covariance features are computed over pixels within each patch. Then, we perform max pooling over a fixed-size pooling region and use the pooled features to represent the covariance features in the pooling region. In fact, multiple covariance matrices within each pooling region are summarized into a single matrix which has better invariance to image deformation and translation. The pooled features extracted from each pooling region is called the spatially pooled covariance (sp-Cov) features in (Paisitkriangkrai et al., 2014b).

Implementation To expand the richness of our feature representation, we extract sp-Cov features using multi-scale patches with the following sizes: 4×4, 8×8 and 16×16 pixels. Each scale will generate an independent set of visual descriptors. In our experiments, the patch step-size is set to be 1 pixel, the pooling region is set to be 4×4 pixels, and the pooling spacing stride is set to be 4 pixels.

Local Binary Pattern (LBP) LBP is a texture descriptor which uses a histogram to represent the binary code of each image patch (Ahonen et al., 2004). The original

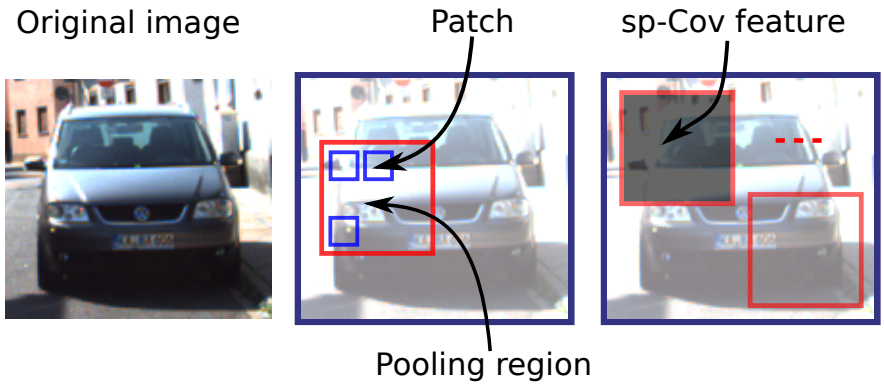

Figure 3.3: Architecture of the spatially pooled covariance features.

LBP is generated by thresholding the 3×3-neighbourhood of each pixel with the value of centre pixel. All binary results are concatenated to form an 8-bit length binary sequence with 2^8 different labels. The histogram of these 256 different labels can represent a texture descriptor. By following the work of (Paisitkriangkrai et al., 2014b), we convert the input image from the RGB space to LUV space, and extract the uniform LBP (Wang et al., 2009) from the luminance (L) channel. The uniform LBP, which is an extension of the original LBP, can better filter out noises.

Spatially pooled LBP Similar to the sp-Cov features, the image window is divided into multiple dense patches and LBP histogram is computed over pixels within each patch. In order to enhance the invariance to image deformation and translation, we perform max pooling over a fixed-size pooling region and use the pooled features to represent the LBP histogram in the pooling region. The pooled features extracted from each pooling region is called the spatially pooled LBP (sp-LBP) features in (Paisitkriangkrai et al., 2014b).

Implementation To extract LBP, we apply the LBP operator on the 33-neighbourhood at each pixel. The LBP histogram is extracted from a 4×4 pixels patch. We extract the 58-dimension LBP histogram using a C-MEX implementation of (Vedaldi and Fulkerson, 2010). In our experiments, the patch step-size, the pooling region, and the pooling spacing stride are set to 1 pixel, 8×8 pixels, and 4 pixels, respectively. Instead of extracting LBP histograms from multi-scale patches, the sp-LBP and LBP are combined as channel features.

3.3.3 Supervised learning

Once dense features have been extracted, we are in a position to train a classifier. Instead of training a standard AdaBoost classifier, we use a shrinkage version of

AdaBoost as the strong classifier and use decision trees as weak learners. To train the classifier, the procedure known as bootstrapping is applied, which collects hard negative samples and re-trains the classifier. If the object subcategorization is applied to the object class, we train one classifier for each subcategory. The learning algorithm is the shrinkage version of AdaBoost presented in Algorithm 2.

Bootstrapping To improve the performance of the learned classifier, we perform three bootstrapping iterations in addition to the original training phase. The initial training phase randomly sample negative samples from training images with positive regions cropped out, and further bootstrapping iterations add more hard negatives to the training set. The learning process consists of 4 training iterations with increasing number of weak learners and the final model consists of 2048 weak learners.

3.3.4 Post-processing

Raw detection results are generated by applying trained detectors to test images, but these results often contain some noises and redundant information. To improve detection performance, some techniques are used to post-process raw detection results.

3.3.4.1 Calibration of confidence scores

If we have multiple sub-detectors and apply them to test data, detection results of each sub-detector are required to merge together to generate the integrated results. However, the classifier of each sub-detector is learned with different training data, confidence scores of raw detection results output by individual classifiers need to be calibrated appropriately to suppress noises before merging them together. We address this problem by transforming the output of each classifier by a sigmoid regression to generate comparable score distributions (Platt, 1999; Lin et al., 2007). For sample i in subcategory k, its confidence score is the output of the ensemble classifier which is defined as

$$s_i^k = \sum_{t=1}^{T} a_t h_t(\vec{x}_i^k), \tag{3.1}$$

its calibrated score is defined as

$$g_i^k = \frac{1}{1 + \exp(A_k \cdot s_i^k + B_k)}, \tag{3.2}$$

where A_k, B_k are the learned parameters for the k-th subcategory of the following regularized maximum likelihood problem:

$$\arg\min_{A_k, B_k} - \sum_{i=1}^{N_k} \left[t_i \log g_i^k + (1 - t_i) \log (1 - g_i^k) \right], \tag{3.3}$$

$$t_i = \begin{cases} \frac{N_+ + 1}{N_+ + 2} & \text{if } y_i = +1 \\ \frac{1}{N_- + 2} & \text{if } y_i = -1 \end{cases}, i = 1, \cdots, N_k. \tag{3.4}$$

The g_i^k in equation 3.3 can be cancelled by reformulation:

$$\arg\min_{A_k, B_k} \sum_{i=1}^{N_k} \left[(t_i - 1)(A_k \cdot s_i^k + B_k) + \log\left(1 + \exp(A_k \cdot s_i^k + B_k)\right) \right]. \tag{3.5}$$

N_k is the total number of training examples for the k-th subcategory-specific classifier, N_+ is the number of positive examples, and N_- is the number of negative examples.

3.3.4.2 Non-maximum suppression (NMS)

NMS aims to suppress redundant bounding boxes among the raw detection results. When multiple bounding boxes overlap, NMS will eliminate the lower-scored detections and retain the highest-scored detection. Pascal overlap score (Everingham et al., 2010) is used to determine the overlap ratio a_0 between two bounding boxes. The overlap ratio a_0 is defined as

$$a_0 = \frac{\text{area}(B_1 \cap B_2)}{\text{area}(B_1 \cup B_2)}, \tag{3.6}$$

where B_1 and B_2 are two different bounding boxes. If the overlap ratio a_0 exceeds a predefined threshold, bounding box with the lower confidence score is discarded.

3.3.4.3 Fusion of detection results

The proposed framework can detect multiple objects simultaneously using detectors or sub-detectors of different classes. We need to consider how to merge detection results from different detectors. Since an object may be detected redundantly using multiple sub-detectors or a single detector at multiple scales, NMS is usually used to eliminate these redundant detections in the merging process. However, NMS is not suitable for merging detections from different classes. Assume that a car is occluded by a cyclist. If their overlap ratio exceeds the threshold, NMS will simply delete the lower-scored detection, and retain the higher-scored detection. It means that one true positive will be removed in this case.

To solve the above problem, we propose a fusion method to merge all detection results in two steps. Instead of applying NMS to detection results from all detectors, we apply NMS to detections of each single class (traffic sign, car, cyclist) separately to filter out redundant bounding boxes generated by either a single detector or multiple

(a) Prohibitory **(b) Danger** **(c) Mandatory**

Figure 3.4: Sample images of three main categories on the GTSDB dataset.

sub-detectors of the class. Next, we directly combine filtered bounding boxes from different classes without using NMS to generate the final detection results. This fusion method can eliminate the overlapped false positives of each single class while it keeps the true positives from different classes as much as possible.

3.4 Experiments

In this section, we demonstrate the effectiveness and robustness of the proposed framework in three detection problems: traffic sign detection, car detection, and cyclist detection.

3.4.1 Traffic sign detection on GTSDB dataset

We firstly conduct an experiment on traffic sign detection and evaluate our detector on the German Traffic Sign Detection Benchmark (GTSDB) (Houben et al., 2013).

3.4.1.1 Dataset

The GTSDB dataset contains 600 images for training and 300 images for testing. Images are captured from various scenes (highway, urban, rural) and various time slots (morning, afternoon, dusk, etc.). The dataset contains more than 1000 traffic signs from different categories. Three main categories of traffic signs (prohibitory, danger, mandatory) are selected as the target classes in the IJCNN 2013 (Houben et al., 2013) competition and in our experiments. The resolutions of traffic signs vary from 16×16 pixels to 128×128 pixels. Fig. 3.4 illustrates some samples of three mentioned categories on the GTSDB dataset.

3.4.1.2 Evaluation criteria

Pascal overlap score (Everingham et al., 2010) is used to find the best match between each predicted bounding box and each ground truth. The minimum overlap ratio a_0 is set to be 60% on the GTSDB. Only the bounding box with the highest confidence score is counted as true positive if multiple bounding boxes satisfy the overlap criterion, the others are ignored. To compare the performance of different detectors, we follow the evaluation metric of the GTSDB which uses the area under the precision-recall curve (AUC) as a final score.

3.4.1.3 Parameter selection

To alleviate the effect of the illumination change, we apply the automatic color equalization algorithm (ACE) (Getreuer, 2012) to globally normalize all images. The resolution of the traffic sign model is set to 20×20 pixels and the dimension of model padding is set to 30×30 pixels. This border provides an additional amount of context that helps improve the detection performance (Dalal and Triggs, 2005; Dollár et al., 2010). Additionally, we increase the number of positive samples by adding jittered versions of the original samples, which significantly improves the detection performance. For prohibitory and danger signs, flipped versions are added to the training set. For mandatory signs, samples are randomly perturbed in translation ($[-2, 2]$ pixels), in scale ($[0.8, 1]$ ratio), in rotation ($[-5, 5]$ degrees), and flipping. We demonstrate the performance gain on the test set in table 3.1. Negative samples are collected from the GTSDB training images with the corresponding traffic sign regions cropped out.

	Prohibitory	Danger	Mandatory	Avg. AUC
Original dataset	98.76%	93.65%	86.86%	93.09%
Jittered dataset	100.00%	98.00%	97.57%	98.52%

Table 3.1: Performance (AUC) difference between training on original training set and jittered training set.

3.4.1.4 Experimental design

We investigate the experimental design of the proposed detector on traffic sign detection. Since traffic signs are divided into three subcategories, we train one sub-detector for each subcategory. We train all detectors on the GTSDB training set and evaluate them on the GTSDB test set. All experiments are carried out using combined fea-

tures (ACF+sp-Cov+sp-LBP) as dense features, Adaboost with shrinkage value of 0.1 as the strong classifier, and depth3-decision trees as weak learners (if not specified otherwise).

Shrinkage We evaluate the performance of AdaBoost with 4 different shrinkage values from $\{0.05, 0.1, 0.2, 0.5\}$. We decrease the reject threshold of soft cascade by a factor of ν as coefficients of weak learners have been diminished by a factor of ν. The area under precision-recall curve of different detectors are shown in Table 3.2. We observe that applying a small shrinkage value often improves the detection performance and the best performance is achieved by setting $\nu = 0.1$. However, without increasing the number of weak learners, setting the shrinkage value to be too small ($\nu = 0.05$) can degrade the performance as the boosting cannot converge within a limited number of boosting iterations.

Shrinkage	Prohibitory	Danger	Mandatory	Avg. AUC
$\nu = 0.5$	98.13%	95.28%	90.32%	94.58%
$\nu = 0.2$	99.38%	96.80%	92.79%	96.32%
$\nu = 0.1$	**100.00%**	**98.00%**	**97.57%**	**98.52%**
$\nu = 0.05$	99.99%	97.81%	95.16%	97.63%
$\nu = 0.05^*$	99.99%	98.00%	96.76%	98.25%

Table 3.2: Performance (AUC) of detectors with different shrinkage values. $*$ The model consists of 4096 weak learners while others consist of 2048 weak learners.

Depth of decision trees We trained 4 different traffic sign detectors with decision trees of depth 1 to depth 4. Table 3.3 shows the detection performance of different detectors. We observe that increase the depth of decision trees provides a performance gain, especially for the mandatory category. However, the depth-3 decision trees achieve better generalization performance and are faster to train than depth-4 decision trees.

Depth	Prohibitory	Danger	Mandatory	Avg. AUC
depth-1	99.98%	97.41%	75.47%	90.95%
depth-2	99.99%	97.98%	95.49%	97.82%
depth-3	**100.00%**	**98.00%**	97.57%	**98.52%**
depth-4	99.99%	96.77%	**98.10%**	98.29%

Table 3.3: Performance (AUC) of detectors with different depths of decision trees.

Combination of features To compare the discriminative power of different feature representations, we evaluate the performance of various feature combinations.

The results are shown in Table 3.4. We observe that a combination of the sp-Cov features and LUV outperforms the ACF features and combining more features can further improve the detection performance. The best result is achieved using a combination of all features (sp-Cov+sp-LBP+ACF).

Feature combination	Prohibitory	Danger	Mandatory	Avg. AUC
ACF (LUV+O+M)	98.72%	94.58%	92.65%	95.32%
sp-LBP+ACF	99.99%	95.07%	96.12%	97.06%
sp-Cov+LUV	99.30%	96.67%	95.56%	97.18%
sp-Cov+ACF	98.73%	95.23%	95.61%	96.52%
sp-Cov+sp-LBP+ACF	**100.00%**	**98.00%**	**97.57%**	**98.52%**

Table 3.4: Performance (AUC) of detectors with various feature combinations.

3.4.1.5 Comparison with state-of-the-art detectors

Detection performance of various detectors on the GTSDB test set are shown in Table 3.5. The proposed detector achieves the comparable results with state-of-the-art detectors despite its simplicity. These detectors (Wang et al., 2013a; Mathias et al., 2013b) that offer better performance employ multi-scale models in detection. The authors of (Wang et al., 2013a) train multiple subcategory-specific classifiers for each type of mandatory signs to achieve the best performance.

Method	Prohibitory	Danger	Mandatory	Avg. AUC
Ours	**100.00%**	98.00%	**97.57%**	**98.52%**
Wang *et al.* (Wang et al., 2013a)	**100.00%**	**99.91%**	**100.00%**	**99.97%**
Mathias *et al.* (Mathias et al., 2013b)	**100.00%**	**100.00%**	**96.98%**	**98.99%**
BolognaCVLab (Houben et al., 2013)	99.98%	**98.72%**	95.76%	98.15%
Liang *et al.* (Liang et al., 2013)	100.00%	98.85%	92.00%	96.95%
Timofte *et al.* (Timofte et al., 2009)	61.12%	79.43%	72.60%	71.05%
Viola-Jones (Viola and Jones, 2004)	90.81%	46.26%	44.87%	60.65%

Table 3.5: Detection performance (AUC) of various detectors on GTSDB test set with 60% overlap ratio.

3.4.2 Car detection on UIUC dataset

Next, we conduct an experiment on car detection and compare detection performance of different detectors on the UIUC dataset (Agarwal et al., 2004). The UIUC dataset captures images of side views of cars with a resolution 40×100 pixels. The

training set contains 550 positive samples and 500 negative samples. The test set is divided into two subsets: 170 single-scale test images, containing 200 cars at roughly the same scale as in the training set, and 108 multi-scale test images, containing 139 cars at various scales.

We follow the evaluation protocol provided along with the UIUC dataset. A bounding box is counted as true positive if it lies within 25% of the ground truth dimension in each direction. Only the bounding box with the highest confidence score is counted as true positive if multiple bounding boxes satisfy the criterion, the others are counted as false positives. In the dataset, three criteria are used to evaluate the performance: F_1-score, detection rate, and the number of false positives. F_1-score is the weighted harmonic mean of precision and recall.

The dimension of UIUC car model is set to 40×100 pixels without marginal padding as the car images are clipped to the same size. We expand the positive samples by flipping car images along the vertical axis. Since viewpoints of cars in the UIUC dataset are limited to side-views, we train a single detector without applying subcategorization and bootstrapping. Table 3.6 shows the results of different detectors on the multi-scale test images. We observe that our detector achieves the best detection rate with slight more false positives on this dataset.

Method	F-Measure	Det. rate	No. false pos.
Ours	98.6%	99.28%	3
Pruning (Paisitkriangkrai et al., 2014a)	98.6%	97.8%	1
AdaBoost (Viola and Jones, 2004)	98.6%	98.6%	2
AdaBoost+LDA (Wu et al., 2008)	98.6%	97.8%	1
CS-AdaBoost (Masnadi-Shirazi and Vasconcelos, 2011)	95.3%	95.5%	9

Table 3.6: Detection performance of various detectors on UIUC multi-scale test set.

3.4.3 Car detection on KITTI dataset

To further demonstrate the effectiveness and robustness of the proposed detector on car detection, we evaluate our detector on a more challenging object detection benchmark, KITTI dataset (Geiger et al., 2013)

3.4.3.1 Dataset

The KITTI dataset is a recently proposed challenging dataset which consists of 7481 training images and 7518 test images, comprising more than 80 thousands of annotated objects in traffic scenes. Table 3.7 provides a summary of existing car datasets. We observe that the KITTI dataset provides a large number of cars with different

	Training		Testing		Properties				
	# cars	# images	# cars	# images	color	Annotations	multi-views	occ. labels	trunc. labels
UIUC Car	550	1050	139	108					
MIT Car	516	516	-	-	✓				
Street Parking	-	881	-	-	✓	✓	✓	✓	
Pascal VOC	1250	713	1201	721	✓	✓	✓	✓	✓
KITTI Car	27k	7481	-	7518	✓	✓	✓	✓	✓

Table 3.7: Comparison of car datasets. The first four columns indicate the amount of training/testing data in each dataset. Note that KITTI dataset is two orders of magnitude larger than other existing datasets. The next five columns provide additional properties of each dataset.

sizes, viewpoints, occlusion patterns, and truncation scenarios. Due to the diversity of these objects, the dataset has three subsets (Easy, Moderate, Hard) with respect to the difficulty of object size, occlusion and truncation. Since KITTI evaluates the detection performance on the moderate subset, the moderate subset is used as the training data in our experiments. The moderate subset contains 15710 cars, with the heights of the cars vary from 25 pixels to 270 pixels and the aspect ratios vary between 0.9 and 4.0. Since annotations of test data are not provided by the KITTI benchmark, we split the KITTI training images into training set (first 4000 images) and validation set (remaining 3481 images).

3.4.3.2 Evaluation criteria

We follow the provided protocol for evaluation. Pascal overlap score is used to find the best match and the minimum overlap ratio a_0 is set to be 70%. Only the bounding box with the highest confidence score is kept if multiple bounding boxes satisfy the overlap criterion, the others are counted as false positives. Instead of using AUC, average precision (AP) (Everingham et al., 2010) is used to evaluate the detection performance. The AP summaries the shape of the precision-recall curve, and is defined as the mean precision at a set of evenly spaced recall levels.

3.4.3.3 Parameter selection

We apply the proposed subcategorization method to categorize the training data into multiple subcategories. To find the model dimensions of each subcategory, we set the base height of each model to 52 pixels. From the base height, the width of each model

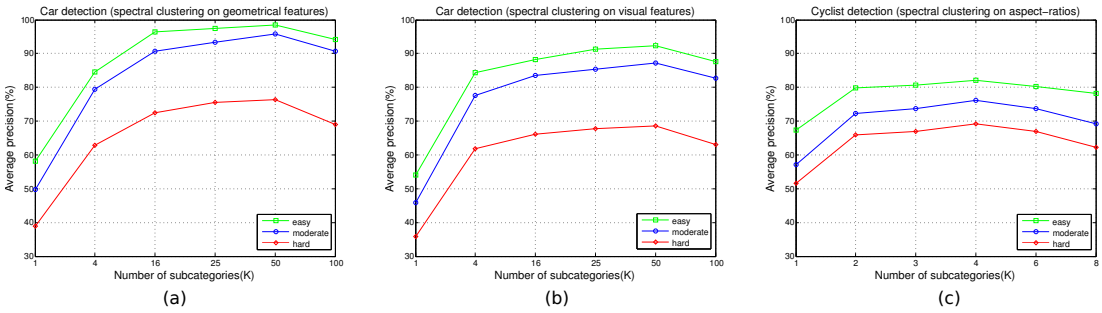

Figure 3.5: Detection performance (AP) of various detectors with different number of subcategories on the KITTI validation set. (a) Car detector (spectral clustering + geometrical features). (b) Car detector (spectral clustering + visual features). (c) Cyclist detector (spectral clustering + aspect-ratios).

can be obtained by taking the median aspect ratios of cars in the corresponding subcategory. Each model includes additional 4 pixels of marginal padding on all sides. Using a model with suitable aspect ratio can significantly improves the detection performance due to better localization. We expand the positive training samples by randomly perturbing original car samples in translation ($[-2, 2]$ pixels), and in rotation ($[-2, 2]$ degrees). Negative samples are collected from the KITTI training images with vehicle regions cropped out.

3.4.3.4 Experimental design

We investigate the experimental design of the proposed detector on car detection. We train car detectors on the training set and evaluate them on the validation set. All experiments are carried out using ACF as dense features, Adaboost with shrinkage value of 0.1 as the strong classifier, depth4-decision trees as weak learners, and $K = 25$ in the subcategorization method (if not specified otherwise).

Number of subcategories To investigate the effect of different number of clusters in our subcategorization method, we set the number from {1, 4, 16, 25, 50, 100}. Fig. 3.5(a) and Fig. 3.5(b) shows the effect of increasing the number of subcategories on geometrical features and visual features, respectively. We observe that geometrical features outperform visual features in spectral clustering. We also observe that the detection performance improves as we increase the number of subcategories up to 50. However, setting the number of subcategories to be too large ($K = 100$) can hurt the performance as the average number of samples in each subcategory is not enough to train an effective model. For the rest of our experiments, we set the number of subcategories to be 25 as it gives a better trade-off between the performance and the complexity.

Depth of decision trees We evaluate the performance for different decision tree depths. As can be observed in Table 3.8, the depth-4 decision trees perform the best as they can provide the best generalization performance.

Depth	Easy	Moderate	Hard
depth-2	96.38%	89.18%	70.87%
depth-3	97.17%	91.21%	74.44%
depth-4	**97.41%**	**93.37%**	**75.60%**
depth-5	96.67%	92.08%	72.77%

Table 3.8: Performance (AP) of detectors with different depths of decision trees.

Combination of features We evaluate the performance of various feature combinations on car detection. The results are shown in table 3.9. We observe that the detection performance improves as we add more features and the best performance is achieved using a combination of all features (sp-Cov+sp-LBP+ACF). A combination of sp-LBP features and ACF features also achieves the similar performance and is five times faster than the combination of all features. We use the combination of sp-LBP features and ACF features as dense features in the testing phase since it gives a better trade-off between detection performance and runtime.

Feature combination	Easy	Moderate	Hard	Runtime
ACF (LUV+O+M)	97.41%	93.37%	75.60%	0.5s
sp-LBP+ACF	97.74%	94.38%	76.50%	1.5s
sp-Cov+LUV	97.76%	93.68%	75.68%	6.8s
sp-Cov+ACF	97.98%	93.48%	75.61%	6.8s
sp-Cov+sp-LBP+ACF	**98.42%**	**94.55%**	**76.66%**	7.5s

Table 3.9: Performance (AP) of detectors with various feature combinations.

3.4.3.5 Comparison with state-of-the-art detectors

Table 3.10 shows the performance comparison of state-of-the-art detectors on the KITTI testing set. Experimental results show that the proposed detector is of not only better performance than all DPM-based methods (Pepik et al., 2013; Torres et al., 2014; Felzenszwalb et al., 2010) but also less runtime. More significantly, our detector outperforms the SubCat (Ohn-Bar and Trivedi, 2014) which employs a similar object subcategorization method and the Regionlets (Wang et al., 2013b; Long et al., 2014) which employs a similar pooling strategy. We conjecture that the additional performance gain is provided by the spatially pooled features.

Method	Easy	Moderate	Hard	Runtime
Ours	**87.19%**	**77.40%**	**60.60%**	1.5s
Regionlets (Wang et al., 2013b)	84.75%	76.54%	59.70%	1s
SubCat (Ohn-Bar and Trivedi, 2014)	81.94%	66.32%	51.10%	0.3s
AOG (Li et al., 2014a)	80.26%	67.03%	55.60%	3s
OC-DPM (Pepik et al., 2013)	74.94%	65.95%	53.86%	10s
DPM-C8B1 (Torres et al., 2014)	74.33%	60.99%	47.16%	15s
MDPM-un-BB (Felzenszwalb et al., 2010)	71.19%	62.16%	48.43%	60s
mBoW (Behley et al., 2013)	36.02%	23.76%	18.44%	10s

Table 3.10: Detection performance (AP) of various detectors on KITTI car test set with 70% overlap ratio.

3.4.4 Cyclist detection on KITTI dataset

In this section, we conduct an experiment on cyclist detection and evaluate our detector on the KITTI dataset.

3.4.4.1 Dataset

The KITTI dataset contains annotated cyclist objects which are captured from various traffic scenes. Similar to cars, cyclists are divided into three subsets (Easy, Moderate, Hard) and the moderate subset is used as the training data in our experiments. The moderate subset contains 1098 cyclists, with the heights of the cyclists vary from 25 pixels to 275 pixels and the aspect ratios vary between 0.3 and 1.5.

3.4.4.2 Evaluation criteria

The KITTI cyclist detection uses the same evaluation protocol with the car detection except that the minimum overlap ratio is relaxed to 50%.

3.4.4.3 Parameter selection

The proposed subcategorization method is applied to cyclist detection. We define the dimensions of cyclist models using the similar method in car detection. We set the base height of each model to 56 pixels, and the width of each model is derived from the median aspect ratios of cyclists in the corresponding subcategory. Each model includes additional 4 pixels of marginal padding on all sides. We expand the positive training samples by randomly perturbing the original cyclists in translation ($[-2, 2]$ pixels), in rotation ($[-2, 2]$ degrees). Negative samples are collected from the KITTI training images with cyclist regions cropped out.

3.4.4.4 Experimental design

We investigate the experimental design of our detector on cyclist detection. We train cyclist detectors on the training set and evaluate them on the validation set. All experiments are carried out using ACF as dense features, Adaboost with shrinkage value of 0.1 as the strong classifier, depth4-decision trees as weak learners, and $K = 4$ in the subcategorization method (if not specified otherwise).

Number of subcategories We set the number of clusters from {1, 2, 3, 4, 6, 8} in our subcategorization method. Since only the minority of cyclists are occluded and truncated, clustering on all geometrical features leads to a cluster degeneration problem. We carefully select the aspect-ratios of cyclists as the feature space to avoid the above problem. Fig. 3.5(c) shows the effect of increasing the number of subcategories. We observe that the detection performance improves as we increase the number of subcategories up to 4. Since the number of cyclists is much less than cars, the average number of cyclists in each subcategory becomes very small when we have a large number of subcategories, which results in an imbalanced learning problem and degrades the detection performance.

Depth of decision trees We trained 4 cyclist detectors with decision trees of depth 2 to depth 5. Average precisions of different detectors are shown in Table 3.11. We observe that depth-4 decision trees offer the best generalization performance, as similar in the car detection.

Depth	Easy	Moderate	Hard
depth-2	80.92%	75.47%	69.46%
depth-3	89.83%	82.67%	76.65%
depth-4	**92.15%**	**86.18%**	**79.28%**
depth-5	90.98%	85.21%	78.26%

Table 3.11: Performance (AP) of detectors with different depths of decision trees.

Combination of features We evaluate the performance of various feature combinations on cyclist detection. The results are shown in Table 3.12. We observe that the best performance is achieved using a combination of sp-LBP features and ACF features. The performance declines when we add the sp-Cov features as a part of aggregated channel features. The reason may be due to the lack of enough cyclist training samples. We use the combination of sp-LBP features and ACF features as the dense features in the testing phase.

Feature combination	Easy	Moderate	Hard	Runtime
ACF (LUV+O+M)	92.15%	86.18%	79.28%	0.2s
sp-LBP+ACF	**92.56%**	**87.40%**	**80.01%**	0.6s
sp-Cov+LUV	85.48%	79.17%	72.20%	5.8s
sp-Cov+ACF	85.16%	80.58%	73.64%	5.8s
sp-Cov+sp-LBP+ACF	90.08%	83.80%	76.89%	6.1s

Table 3.12: Performance (AP) of detectors with various feature combinations.

3.4.4.5 Comparison with state-of-the-art detectors

Table 3.13 shows the performance comparison with state-of-the-art approaches. As shown in Table 3.13, our detector outperforms all other methods on the test set. Specifically, our detector outperforms the best DPM-based method DPM-VOC+VP (Pepik et al., 2015) on all the three subsets by 16.29%, 14.95%, and 12.35%, respectively. Our detector also performs slightly better than the Regionlets (Wang et al., 2013b; Long et al., 2014).

Method	Easy	Moderate	Hard	Runtime
Our method	**58.72%**	**46.03%**	**40.58%**	0.6s
Regionlets (Wang et al., 2013b)	56.96%	44.65%	39.05%	1s
MV-RGBD-RF(González et al., 2015)	52.97%	42.61%	37.42%	4s
DPM-VOC+VP (Pepik et al., 2015)	42.43%	31.08%	28.23%	8s
LSVM-MDPM-us (Felzenszwalb et al., 2010)	38.84%	29.88%	27.31%	10s
DPM-C8B1 (Torres et al., 2014)	43.49%	29.04%	26.20%	15s
mBoW (Behley et al., 2013)	28.00%	21.62%	20.93%	10s

Table 3.13: Detection performance (AP) of various detectors on KITTI cyclist test set with 50% overlap ratio.

3.4.5 An evaluation of the overall runtime

We conduct an experiment on the evaluation of the overall runtime of the proposed detection framework on the KITTI dataset. All experiments are carried out on a computer with an octa-core Intel Xeon 2.50GHz processor. The average runtime of each component of the framework can be seen in Table 3.14. For feature extraction, we observe that the ACF features can be extracted very quickly within 0.1s. When we add the sp-LBP features, the runtime increases moderately, but these features provide an obvious performance gain in all three applications. When the sp-Cov features are employed, the runtime of feature extraction increases rapidly and dominates the

Feature combination	Feature extraction	Cars(25) detection	Cyclists(4) detection	Signs(3) detection	Total Runtime
ACF (LUV+O+M)	0.10s	0.40s	0.10s	0.05s	0.65s
sp-LBP+ACF	0.35s	1.20s	0.30s	0.10s	1.95s
sp-Cov+ACF	5.50s	1.30s	0.30s	0.10s	7.20s
sp-Cov+sp-LBP+ACF	5.75s	1.75s	0.35s	0.15s	8.00s

Table 3.14: An evaluation of the overall runtime of the proposed framework with various feature combinations.

total runtime of the system. For object detection, we observe that the car detector costs the most time in this framework since it has 25 sub-detectors. The traffic sign detector uses the least time since it has only 3 sub-detectors. We also observe that the runtime of detection increases as we add more complicated features in the framework. According to observe the detection results of three applications, we conjecture that using a combination of ACF features and sp-LBP features can provide a better trade-off between detection performance and system runtime.

3.5 Conclusion

In this chapter, we propose a common detection framework for detecting three important classes of objects in traffic scenes. The proposed framework introduces spatially pooled features as a part of aggregated channel features to enhance the feature robustness and employs detectors of three important classes to detect multiple objects. The detection speed of the framework is fast since dense features need only to be evaluated once rather than individually for each detector. To overcome the weakness of the VJ framework for object classes with a large intra-class variation, we propose an object subcategorization method to improve the generalization performance by capturing the variation. We demonstrated that our detector achieves the competitive results with state-of-the-art detectors in traffic sign detection, car detection, and cyclist detection. Future work could include that contextual information can be used to facilitate object detection in traffic scenes and convolutional neural network can be used to generate more discriminative feature representations.

Chapter 4

Pushing the Limits of Deep CNNs for Pedestrian Detection

4.1 Introduction

The problem of pedestrian detection has been intensively studied in recent years. Prior to the very recent work in deep convolutional neural networks (DCNNs) based methods (Cai et al., 2015; Tian et al., 2015), the top performing pedestrian detectors are boosted decision forests with carefully hand-crafted features, such as histogram of gradients (HOG) (Dalal and Triggs, 2005), self-similarity (SS) (Shechtman and Irani, 2007), aggregate channel features (ACF) (Dollár et al., 2014), filtered channel features (Zhang et al., 2015) and optical flow (Paisitkriangkrai et al., 2016).

Recently, DCNNs have significantly outperformed comparable methods on a wide variety of vision problems (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015; Szegedy et al., 2015; Girshick et al., 2014a; Tompson et al., 2014; Hariharan et al., 2014; Simonyan and Zisserman, 2014; Branson et al., 2014). A region-based convolutional neural network (R-CNN) (Girshick et al., 2014a) achieved excellent performance for *generic* object detection, for example, in which a set of potential detections (object proposals) are evaluated by a DCNN model. Later, R-CNN was extended to the Fast R-CNN (Girshick, 2015) which significantly increases the detection speed. CifarNet (Krizhevsky and Hinton, 2009) and AlexNet (Krizhevsky et al., 2012) have been extensively evaluated in the R-CNN detection framework in (Hosang et al., 2015) for pedestrian detection. In their work, the best performance is 23.3% log-average miss rate (MR) on Caltech dataset, which was achieved by AlexNet pre-trained on the ImageNet (Deng et al., 2009) classification dataset. Note that this result is still inferior to conventional pedestrian detectors such as (Zhang et al., 2015) and (Paisitkriangkrai et al., 2016). The DCNN models in (Hosang et al., 2015) underperform mainly because the network design is not optimal for pedestrian detection. The performance of R-CNNs for pedestrian detection has further improved to 16.43%

MR in (Tian et al., 2015) through the use of a deeper GoogLeNet model which is fine-tuned using Caltech pedestrian dataset.

To explicitly model the deformation and occlusion, another line of research for object detection is part-based models (Enzweiler et al., 2010; Felzenszwalb et al., 2010; Lin et al., 2015b; Girshick et al., 2015) and explicit occlusion handling (Mathias et al., 2013a; Ouyang and Wang, 2013b; Tang et al., 2014). DCNNs have also been incorporated along this stream of work for pedestrian detection (Ouyang and Wang, 2012, 2013a; Luo et al., 2014), but none of these approaches has achieved better results than the best hand-crafted features based method of (Zhang et al., 2015) on the Caltech dataset.

The performance of pedestrian detection is improved over hand-crafted features by a large margin (a $\sim$ 5% MR gain on Caltech), by two very recent approaches relying on DCNNs: CompACT-Deep (Cai et al., 2015) combines hand-crafted features and fine-tuned DCNNs into a complexity-aware cascade. Tian *et al.* (Tian et al., 2015) fine-tuned a pool of part detectors using a pre-trained GoogLeNet, and the resulting ensemble model (refer to as DeepParts) delivers similar results as CompACT-Deep. Both approaches are much more sophisticated than the standard R-CNN framework: CompACT-Deep involves the use of a variety of hand-crafted features, a small CNN model and a large VGG16 model (Simonyan and Zisserman, 2015). DeepParts contains 45 fine-tuned DCNN models and needs a set of strategies (including bounding-box shifting handling and part selection) to arrive at the reported result. Note that the high complexity of DCNN models can lead to practical difficulties. For example, it can be too costly to load all 45 DCNN models into a GPU card.

Here we ask a question: Is a complex DCNN based learning approach really a must for achieving the state-of-the-art performance? Our answer to this question is negative. In this work, we propose alternative methods for pedestrian detection, which are simpler in design, with comparable or even better performance. Firstly, we extensively evaluate the CFMs extracted from multiple convolutional layers of a fine-tuned VGG16 model for pedestrian detection. Using only a CFM of a single convolutional layer, we train a boosted-tree-based detector and the resulting model already significantly outperforms all previous methods except the above two sophisticated DCNN frameworks. This model can be seen as a strong baseline for pedestrian detection as it is very simple in terms of implementation.

We show that the CFMs from multiple convolutional layers can be used for training effective boosted decision forests. These boosted decision forests are combined altogether simply by score averaging. The resulting ensemble model beats all competing methods on the Caltech dataset. We further improve the detection performance by incorporating a semantic pixel labelling model. Next we review some

related work.

4.2 Background

4.2.1 Convolutional feature maps (CFMs)

It has been shown in (Ren et al., 2016; Hariharan et al., 2015; Yang et al., 2015a) that CFMs have strong representation abilities for many tasks. Long *et al.* (Long et al., 2015) adapt predominant DCNNs into fully convolutional networks and transfer their learned representations by fine-tuning to the semantic segmentation domain. In (Hariharan et al., 2015), the CFMs from multiple layers are stacked into one vector and used for segmentation and localization. Ren *et al.* (Ren et al., 2016) learn a network on the CFMs (pooled to a fixed size) of a pre-trained model.

The work by Yang *et al.* (Yang et al., 2015a) is close to ours, which trains a boosted decision forest for pedestrian detection with the CFM features from the Conv3-3 layer of the VGG16 model (Simonyan and Zisserman, 2015), and the performance (17.32% MR) on Caltech is comparable to checkerboards (Zhang et al., 2015). It seems that there is no significant superiority of the CFM used in (Yang et al., 2015a) over hand-crafted features on the task of pedestrian detection. The reason may be two-fold. First, the CFM used in (Yang et al., 2015a) are extract from the pre-trained VGG16 model which is *not fine-tuned on a pedestrian dataset*; Second, CFM features are extracted from only one layer and the multi-layer structure of DCNNs is not fully exploited. We show in this work that both of these two issues are critically important in achieving good performance.

4.2.2 Segmentation for object detection

The cues used by segmentation approaches are typically complementary to those exploited by top-down methods. Recently, Yan *et al.* (Yan et al., 2015) propose to perform generic object detection by labelling super-pixels, which results in an energy minimization problem with data term learned by DCNN models. In (Fidler et al., 2013; Hariharan et al., 2014), segmented image regions (not bounding boxes) are generated as object proposals and then used for object detection.

In contrast to the above region (or super-pixel) based methods, we here exploit at an even finer level of information, that is, pixel labelling. In particular, in this work we demonstrate that we can improve the detection performance by simply re-scoring the proposals generated by a detector, using pixel-level scores.

4.3 Datasets, evaluation metric and models

Before we present our methods, we briefly describe the datasets, evaluation metric and boosting models in our experiments.

4.3.1 Caltech pedestrian dataset

The Caltech dataset (Dollar et al., 2012) is one of the most dominant datasets for pedestrian detection. It contains 250k frames captured from 10 hours of urban traffic videos. There are in total 350k annotated bounding boxes with 2300 unique pedestrians. The standard training set and test set consider one out of each 30 frames. In our experiments, the training images are expanded to one out of each 4 frames. The test set is fixed and contains 4024 images. The dataset has different test settings with respect to the difficulty of pedestrian height, visibility and aspect ratio. The proposed framework is evaluated on five test settings (Reasonable, Near scale, Medium scale, Partial occlusion, and Overall). Definitions of these test settings are given in the following:

- Reasonable The minimum height of pedestrians is 50 pixels, the occlusion level is either no or partial occlusion.
- Near scale The minimum height of pedestrians is 80 pixels, there is no occlusion on pedestrians.
- Medium scale The height of pedestrians is greater than or equal to 30 pixels and is less than or equal to 80 pixels, there is no occlusion on pedestrians.
- Partial occlusion The minimum height of pedestrians is 50 pixels, the occlusion level is partial occlusion.
- Overall The minimum height of pedestrians is 20 pixels, the occlusion level can be one of no occlusion, partial occlusion, or heavy occlusion.

Note that many competing methods (Zhang et al., 2015; Yang et al., 2015a; Hosang et al., 2015) have used the same extended training set or even more data (every third frame). We evaluate the performance of various detectors using the log-average miss rate (MR) which is computed by averaging the miss rate at false positive rates spaced evenly between 0.01 to 1 false-positive-per-image (FPPI) range. The dataset has different test settings with respect to the difficulty of pedestrian height, visibility and aspect ratio. Unless otherwise specified, the detection performance on our experiments shown in the remainder of the chapter is the MR on the Caltech Reasonable test setting.

4.3.2 Inria pedestrian dataset

The Inria dataset (Dalal and Triggs, 2005) contains 614 positive training images and 288 positive test images. Images of Inria are captured from multiple different scenes, including scenic spot, forest, grassland, snow mountain, etc.. We use the log-average miss rate to evaluate the detection performance as same as the Caltech. All results are reported on the 288 positive test images (negative images are not used).

4.3.3 KITTI pedestrian dataset

The KITTI dataset (Geiger et al., 2012) consists of 7481 training images and 7518 test images, comprising more than 80 thousands of annotated objects in traffic scenes. The KITTI dataset provides a large number of pedestrians with different sizes, viewpoints, occlusions, and truncations. Due to the diversity of these objects, the dataset has three subsets (Easy, Moderate, Hard) with respect to the difficulty of object size, occlusion and truncation. Definitions of these subsets are provided in the following:

- Easy The minimum height of bounding boxes is 40 pixels, the maximum occlusion level is fully visible, the maximum truncation level is 15% .
- Moderate The minimum height of bounding boxes is 25 pixels, the maximum occlusion level is partial occlusion, the maximum truncation level is 30% .
- Hard The minimum height of bounding boxes is 25 pixels, the maximum occlusion level is heavy occlusion, the maximum truncation level is 50%.

We use the Moderate training subset as the training data in our experiments. Average precision (AP) is used to evaluate the detection performance for KITTI dataset. The average precision summaries the shape of the precision-recall curve, and is defined as the mean precision at a set of evenly spaced recall levels. All methods are ranked based on the Moderate difficult results.

4.3.4 Boosted decision forest

For supervised classification tasks, boosting is a popular method to select features for improving the performance of any given learning algorithm (Freund and Schapire, 1999; Demiriz et al., 2002; Paisitkriangkrai et al., 2016; Zhang et al., 2015). In this chapter, we use the boosted decision forest as a strong classifier which is a convex linear combination of a set of given weak decision trees. The final classification is based on the weighted vote of these decision trees. Unless otherwise specified, we train all our boosted decision forests using the following parameters. The boosted decision forest consists of 4096 depth-5 decision trees, trained via the shrinkage version

of real-Adaboost (Hastie et al., 2005). The size of detection model is set to 128×64 pixels for Caltech and Inria, 64×32 pixels for KITTI. One bootstrapping iteration is implemented to collect hard negatives and re-train the model. The sliding window stride is set to 4 pixels.

4.4 Boosted decision forests with multi-layer CFMs

In this section, we firstly introduce the general layout of VGG16 model. Then, we show that the performance of boosted decision forests with CFMs can be significantly improved by simply fine-tuning DCNNs with hard negative data extracted through bootstrapping. Next, boosted decision forests are trained with different layers of CFMs, and the resulting ensemble model is able to achieve the best reported result on Caltech dataset.

4.4.1 Architecture of the VGG16 model

VGG16 (Simonyan and Zisserman, 2015) is one of the most commonly used deep CNN model. In this work, VGG16 is used to extract CFMs. The input of the model is a fixed-size 224×224 RGB image. The image is passed through a sequence of 13 convolutional (Conv) layers. Each Conv layer contains a predefined number of kernels with a 3×3 receptive field. Both the stride of convolution and the spatial padding is set to 1 pixel. All 13 Conv layers are organized into five Conv stacks. The first two Conv stacks have two Conv layers, respectively. Each of the succeeding three Conv stacks contains three Conv layers. Conv layers in the same stack have the same down-sampling ratio. Each Conv stack is followed by a max-pooling layer which is performed over a 2×2 pooling region with stride 2. These Conv stacks are followed by three fully-connected (FC) layers: the first two have 4096 neurons each, the last one contains 1000 neurons and performs 1000-way classification. The final layer is the soft-max layer. To increase the non-linearity of the model, all Conv layers and FC layers are equipped with the rectification (ReLU) layers. Fig. 4.1 shows the architecture of VGG16 model. We use Conv**Y**-**x** to denote a specific Conv layer, where **Y** indicates the **Y**th Conv stack and **x** indicates the **x**th Conv layer in this stack. FC-6, FC-7, and FC-8 are used to denote three FC layers, respectively.

4.4.2 Fine-tuning DCNNs with Bootstrapped Data

As we know, the VGG16 model was originally pre-trained on the ImageNet data with image-level annotations and was not trained specifically for the pedestrian detection task. The CCF framework of (Yang et al., 2015a) extracts CFMs from a single

Figure 4.1: The architecture of the VGG16 model.

convolutional layer (Conv3-3) of the pre-trained VGG16 model to train the boosted decision forest for diverse detection tasks. To maintain a good generalization ability, the method dose not fine-tune the VGG16 model on any domain-specific datasets. It is expected that the detection performance of boosted decision forests trained with CFMs ought to be improved by fine-tuning the VGG16 model with Caltech pedestrian data. Moreover, We extract CFMs from multiple convolutional layers to train effective boosted decision forests. These boosted decision forests are combined into an ensemble model which further improves the detection performance.

To adapt the pre-trained VGG16 model to the pedestrian detection task, we modify the structure of the model. We replace the 1000-way classification layer with a randomly initialized binary classification layer and change the input size from 224×224 to 128×64 pixels. We also reduce the number of neurons in fully connected layers from 4096 to 2048. We fine-tune all layers of this modified VGG16, except the first 4 convolutional layers since they correspond to low-level features which are largely universal for most visual objects. The initial learning rate is set to 0.001 for convolutional layers and 0.01 for fully connected layers. The learning rate is divided by 10 at every 10000 iterations. For fine-tuning, 30k positive and 90k negative examples are collected by different approaches. The positive samples are those overlapping with a ground-truth bounding box by $[0.5, 1]$, and the negative samples by $[0, 0.25]$. At each stochastic gradient descent (SGD) iteration, we uniformly sample 32 positive samples and 96 negative samples to construct a mini-batch of size 128.

Shallow convolutional layers of the VGG16 contain low-level features which are precise in localization. On the contrary, deep convolutional layers contain discriminative information which are good in classification. According to the evaluation of different CFMs of the VGG16 model in (Yang et al., 2015a), we find that features of Conv3-3 layer provide the best trade-off between the localization information and the discriminative information. It means that these features can achieve the reasonable detection performance and provide effective region proposals simultaneously.

Model	Fine-tuning data	Shrinkage	Avg. miss rate (%)
CFM3a	No fine-tuning	–	18.71
CFM3b	Collected by ACF	–	16.42
CFM3c	Bootstrapping with CFM3b	–	14.54
CFM3	Bootstrapping with CFM3b	0.5	**13.49**

Table 4.1: Performance improvements with different fine-tuning strategies and shrinkage (on Reasonable). All boosted decision forests are trained with the CFM extracted from the Conv3-3 layer of VGG16. CFM3a: the original VGG16 model pre-trained on ImageNet is used to extract features. CFM3b: the VGG16 model is fine-tuned with the data collected by an ACF (Dollár et al., 2014) detector. CFM3c and CFM3: the fine-tuning data is obtained by bootstrapping with CFM3b. With the same fine-tuning data, setting the shrinkage parameter of Adaboost to 0.5 brings an additional 1% reduction on the MR

We train boosted decision forests with the CFM extracted from the Conv3-3 layer of differently fine-tuned VGG16 models and the results are shown in Table 4.1. Note that all the VGG16 models in this table are fine-tuned from the original model pre-trained on ImageNet data. It can be observed that the log-average miss rate is reduced from 18.71% to 16.42% by replacing the pre-trained VGG16 model with the one fine-tuned on data collected by applying an ACF (Dollár et al., 2014) detector on Caltech training dataset. The detection performance is further improved to 14.54% MR if it is fine-tuned on the bootstrapped data using the previous trained model CFM3b. Another 1% performance gain is obtained by applying shrinkage to the coefficients of weak learners, with shrinkage parameter being 0.5 (see (Paisitkriangkrai et al., 2014b)). The last model (corresponding to row 4 in Table 4.1) is referred to as CFM3 from now on.

4.4.3 Ensemble of Boosted Decision Forests

In the last experiment, we only use a CFM from a single layer of the VGG16 model. In this section, we intensively explore the deep structure of the VGG16 model. We ignore the CFMs of the first two convolutional stacks since they are universal for most visual objects.

We train boosted decision forests with CFMs from individual convolutional layers of the VGG16 model which is the one fine-tuned with bootstrapped data (same as row 4 in Table 4.1). All boosted decision forests are trained with the same data as CFM3. For models with Conv3-**x** features, the input image are directly applied on the convolutional layers and resulting in a feature map with the down-sampling ratio of 4. The corresponding boosted decision forests work as a sliding-window detector with step-size of 4. In detection, we upsample the image by a factor of 2

Convolutional layer	# Channels	Down-sampling ratio	Avg. miss rate (%)
Conv3-1	256	4	19.15
Conv3-2	256	4	16.25
Conv3-3 (CFM3)	256	4	**13.49**
Conv4-1	512	8	12.95
Conv4-2	512	8	12.68
Conv4-3 (CFM4)	512	8	**12.21**
Conv5-1 (CFM5)	512	16	**14.17**
Conv5-2	512	16	14.56
Conv5-3	512	16	18.24

Table 4.2: Comparison of detection performance (on Reasonable) of boosted decision forests trained on individual CFMs. Note that models with Conv3-**x** features works as sliding-window detectors, and models with Conv4-**x** and Conv5-**x** features are applied to the proposals generated by CFM3. The top performing layers in each convolutional stack are Conv3-3, Conv4-3 and Conv5-1 respectively. The models trained with these three layers are denoted as CFM3, CFM4, and CFM5 respectively

as in (Zhang et al., 2015) and the minimum size of the shortest image edge is 72 pixels. The number of scales per each octave is set to 8. For models with Conv4-**x** and Conv5-**x** features, they are applied to proposals generated by CFM3 model. This is due to the large downsampling ratio of Conv4-**x** and Conv5-**x**. If the step-size of the sliding-window detector is too large, it will hurt the detection performance.

Table 4.2 shows the comparison of detection performance of these boosted decision forests on Caltech Reasonable setting. We can observe that the MR is relatively high for the Conv3-1 layer and the Conv5-3 layer. We conjecture that the Conv3-1 layer provides relatively low-level features which result in an under-fitting training. In contrast, the semantic information in the Conv5-3 layer may be too coarse to precisely localize small pedestrians. We also note that Conv5-3 layer performs much worse than Conv5-1 layer. This may be caused by that Conv5-3 has a larger receptive field than Conv5-1, more localization information is lost. The large receptive field of Conv5-3 layer degrades its final detection performance. According to Table 4.2, the best performing layer in each convolutional stack, are from inner layers of Conv3-3 (CFM3), Conv4-3 (CFM4), and Conv5-1 (CFM5) respectively. Fig. 4.2 shows the spatial distribution of regions of different CFMs selected by boosting algorithms. Features within the warm color area are frequently selected by above three CFM models. We observe that most active regions correspond to the contours of human-body. The head-shoulder area shows to be more discriminative than other body parts.

The boosted decision forests trained with CFMs of these three layers are further

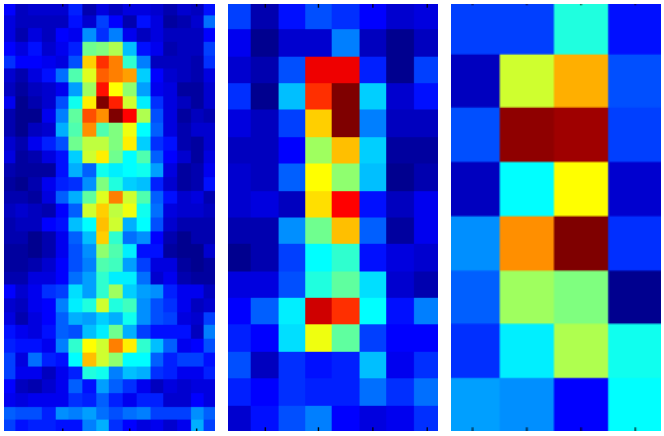

Figure 4.2: The spatial distribution of regions of CFMs selected by boosting algorithms. For a 128×64 input image, the size of feature maps are 32×16, 16×8, 8×4 respectively. Red pixels indicate that a large number of features are selected in those regions and blue pixels correspond to low frequency regions. The most discriminative regions correspond to the head, shoulder, waist and feet of a human.

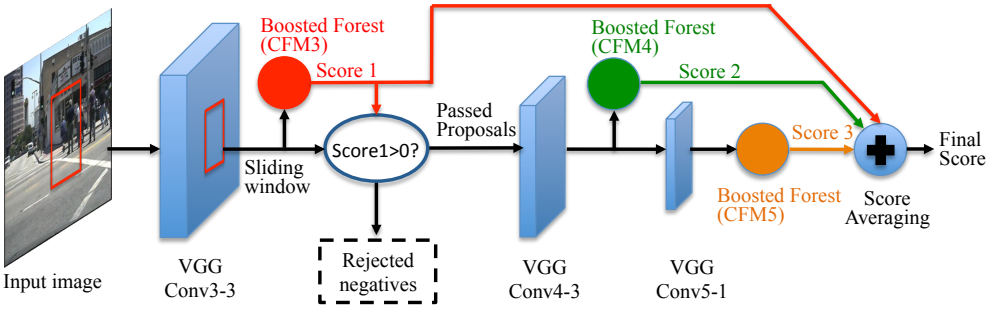

Figure 4.3: The framework of an ensemble of boosted decision forests with multi-layer CFMs (CFM3+CFM4+CFM5), which obtain a 10.46% MR on the Caltech Reasonable test setting.

combined together simply through score averaging. Fig. 4.3 shows the framework of the resulting ensemble model. Firstly, CFM3 model works as a sliding-window detector, which rejects the majority of negative examples and pass region proposals to CFM4 and CFM5. Both CFM4 and CFM5 generate the confidence score for each incoming proposal. The CFM3 features are reused in the computation of CFM4 and CFM5 features. A subregion of CFM3 feature map is cropped and fed into the 4/5-th convolutional layers of the VGG16 model to compute CFM4 and CFM5 features. The final score is computed by averaging over the scores output by these three boosted decision forests. *This model delivers the best reported log-average miss rate (10.46%) on Caltech* Reasonable *setting without using any sophisticatedly designed algorithms.*

Model combination	Avg. miss rate (%)
CFM3+CFM4	10.68
CFM3+CFM5	10.88
CFM3+CFM4+CFM5	10.46
CFM3+CFM4+CFM5+DCNN	**10.07**

Table 4.3: The comparison of performance (on Reasonable) of different ensemble models. DCNN: the entire VGG16 model fine-tuned by data collected by CFM3. The combination of multi-layer CFM models improves the detection performance of single-layer CFM models significantly (3%)

We also evaluate other combinations of the ensemble models. Furthermore, a VGG16 model is fine-tuned with another round of bootstrapping (using CFM3) and its final output is also combined to improve the detection performance. The corresponding results can be found in Table 4.3. We can see that combining two layers already beats all existing approaches on Caltech, and adding the entire large VGG16 model also gives a small improvement.

4.5 Pixel labelling improves pedestrian detection

In this section, the sliding-window based detectors are enhanced by semantic pixel labelling. By incorporating DCNNs, the performance of pixel labelling (semantic image segmentation) methods have been recently improved significantly (Long et al., 2015; Chen et al., 2015a; Hariharan et al., 2015; Zheng et al., 2015; Lin et al., 2015a). In general, we argue that pixel labelling models encode information complementary to the sliding-window based detectors. Empirically, we show that consistent improvements are achieved over different types of detectors.

The segmentation method proposed in (Chen et al., 2015a) is used here for pixel labelling, in which a DCNN model (VGG16) is trained on the Cityscapes dataset (Cordts et al., 2015). The prediction map is refined by a fully-connected conditional random field (CRF) (Krähenbühl and Koltun, 2011) with DCNN responses as unary terms. The Cityscapes dataset that we use for training is similar to the KITTI dataset which contains dense pixel annotations of 19 semantic classes such as road, building, car, pedestrian, sky, etc. Note that our models that exploiting pixel labelling have used extra data for training on top of the Caltech dataset. However, most deep learning based methods (Cai et al., 2015; Tian et al., 2015) have used extra data, at least the ImageNet dataset for pre-training the deep model. Pedestrian detection may benefit from the semantic pixel labelling in the following aspects:

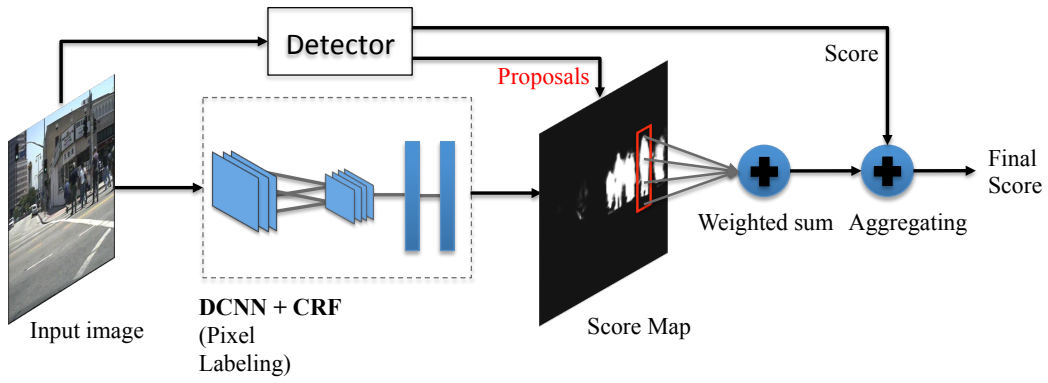

Figure 4.4: The framework for pedestrian detection with pixel-labelling. The region proposals and pixel-level score maps are obtained by individually applying the sliding-window detector and the pixel labelling model. Next, the weighted sum of pixel scores within a proposal region is aggregated with the detector score of the same proposal region.

– *Multi-class information:* Learning from multiple classes, in contrast to the object detectors typically trained with two-class data, the pixel labelling model carries richer object-level information.

– *Long-range context:* Using CRFs (especially fully-connected CRFs) as post-processing procedure, many models (for example, (Chen et al., 2015a; Lin et al., 2015a; Zheng et al., 2015)) have the ability to capture long-range context information. In contrast, sliding-window detectors only extract features from fixed-sized bounding boxes.

– *Object parts:* The trained pixel labelling model may cater for more fine-grained details, such that they are more insensitive to deformation and occlusion to some extent.

However, it is not straightforward to apply pixel labelling models to pedestrian detection problems. One of the main impediments is that it is difficult to estimate the object bounding boxes from the pixel score map, especially for people in crowds.

To this end, we propose to bring the pedestrian detector and pixel labelling model together. In our framework (see Fig. 4.4), a sliding-window detector is responsible for providing region proposals and a pixel labelling model is applied to the input image to generate a score map for the "person" class. Next, a spatially weighted mask $\mathbf{M}$ is applied to the proposal region $\mathbf{x}$ of the "person" score map to generate the weighted sum of pixel scores. The weighted sum of the kth region proposal, denoted as $\mathbf{S_k}$, can be calculated by the following equation:

$$\mathbf{S}_k = \sum_{i}^{H \times W} m_i x_i^k \tag{4.1}$$

Method	Avg. miss rate (%)	Improve. (%)
ACF (Dollár et al., 2014)	22.23	4.50
ACF+Pixel label.	17.73	
Checkerboards (Zhang et al., 2015)	18.25	3.61
Checkerboards+Pixel label.	14.64	
CFM3 (ours)	13.49	1.91
CFM3+Pixel label.	11.58	

Table 4.4: Performance improvements by aggregating pixel labelling models with sliding-window detectors (on Reasonable). All the three detectors achieve performance gains, which shows that pixel labelling can be used to help detection. Note that the performance of our model 'CFM3 with Pixel labelling' already outperforms the previously best reported result of (Cai et al., 2015)

where H and W denote the height and width of the mask $\mathbf{M}$, x_i^k denotes the ith local value of the kth region proposal on the "person" score map, and m_i is the corresponding coefficient on the mask. Note that the dimension of each cropped proposal region $\mathbf{x}$ need to be resized to match the dimension of the mask $\mathbf{M}$. Finally, the weighted sum and the detector score for the same proposal are aggregated together as the final score.

To learn the spatially weighted mask, the pixel labelling model is firstly applied to all training images to generate the "person" score maps. Then, ground truth regions are cropped from these score maps and all cropped patches are resized to the dimension of the detection model without padding area (e.g. 100×41 pixels for Caltech). The mask is learned by averaging these cropped patches.

Note that, there are more sophisticated methods for exploiting the labelling scores. For example, one can use the pixel labelling scores as the image features, similar to 'object bank' (Li et al., 2014b), and train a linear model. In this work, we show that even simply weighted sum of the pixel scores considerably improves the results.

Table 4.4 shows the detection performance of different sliding-window detectors enhanced by pixel labelling. Boosted decision forests are trained here with three types of features, which are ACF (Dollár et al., 2014), checkerboards features (Zhang et al., 2015) and the CFM from the Conv3-3 layer of VGG16 model (CFM3). We can see that the performances of all the three detectors are improved by aggregating pixel labelling models. Fig. 4.5 presents some region proposals on the original images and the corresponding pixel score maps. Some of false proposals generated by pedestrian detectors (CFM3) can be eliminated by considering the context of a larger region (the largest bounding box in the first column in Fig. 4.5). Some occluded pedestrians have responses on the pixel score map (the rightmost bounding box in the last column in

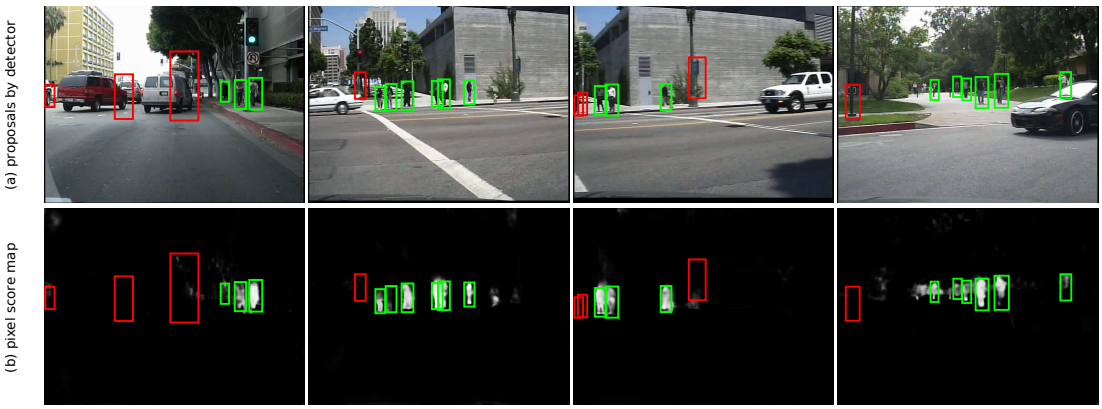

Figure 4.5: Examples of some region proposals on the original images and the corresponding pixel score maps. A strong complementary relationship can be found in the generated proposals and the pixel score maps.

Fig. 4.5). This clearly illustrates why this combination works.

4.6 Overview of the proposed framework

Fig. 4.6 shows an overview of the proposed pedestrian detection framework. The framework consists of two components: a pedestrian detector and a semantic pixel labelling model. Our pedestrian detector is an ensemble detection model which takes as input an image and outputs a number of proposals with detection scores. The pixel labelling model takes as input an image and proposals within the image. It generates the weighted sum of pixel scores for each proposal. Finally, the confidence score of one proposal is computed by averaging outputs of multiple components. To accelerate the detection speed, the CFM3 detection model can be replaced by a light-weight proposal method, which is described in section 4.7.4.

4.7 Fusing models and evaluations

In this section, we integrate all individual components into a combined model and evaluate the performance of the model.

4.7.1 Using complementary hand-crafted features

The detection performance of the CFM3 model is critical in the proposed ensemble model, since later components often rely on the detection results of this model. In order to enhance the detection performance of the CFM3 model, we make two vari-

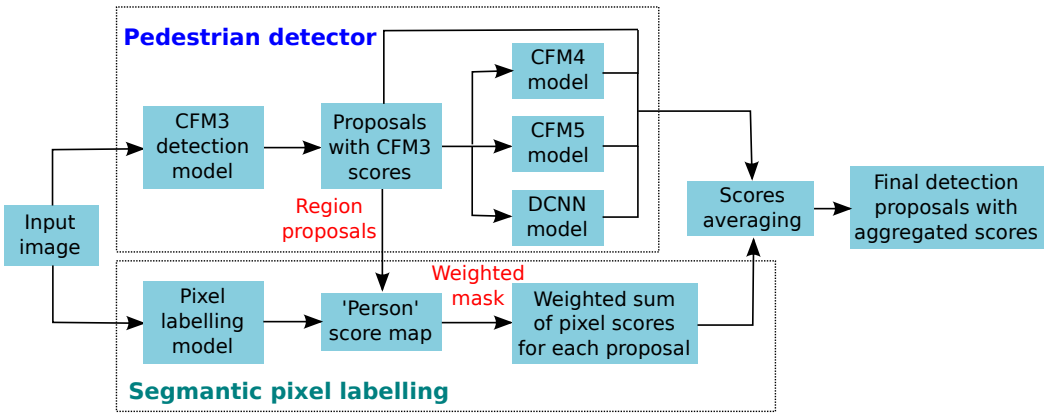

Figure 4.6: Overview of our pedestrian detection framework. The framework consists of one pedestrian detector and one pixel labelling model. The final confidence score of one proposal is computed by averaging outputs of multiple components.

Method	Avg. miss rate (%)
CFM3 only	13.49
CFM3+ACF	12.38
CFM3+ACF+Flow	11.11
(CFM3+ACF)+CFM4+CFM5+DCNN	9.37
(CFM3+ACF+Flow)+CFM4+CFM5+DCNN	**9.32**

Table 4.5: Comparison of detection results of different variants of the CFM3 detector (on Reasonable). The convolutional features of the Conv3-3 layer are combined with different types of hand-crafted features, and used to train a boosted decision forest. Both the performance of the variants and the ensemble models is improved with these additional features. Flow: optical flow features. DCNN: the entire VGG16 model fine-tuned by data collected by CFM3

ants of it by combining two hand-crafted features: the ACF and optical flow. We augment the CFM3 features with the ACF and optical flow features to train an ensemble of boosted decision forests. ACF and optical flow features are directly concatenated with the CFM3 features. Optical flow features are extracted the same way as in (Paisitkriangkrai et al., 2016).

Table 4.5 shows the detection results of different variants of CFM3 model. With adding the ACF features, the MR of CFM3 detector is reduce by 1.11%. With the extra optical flow features, the MR is further reduced to 11.11%. These experimental results demonstrate that hand-crafted features carry complementary information which can further improve the DCNN convolutional features. Fig. 4.7 shows the visualization of some intermediate features. We can observe that the ACF features may be viewed

Figure 4.7: Visualization of some intermediate features.

Figure 4.8: Visualization of detection results of different variants of the CFM3 detector. Yellow bounding boxes are ground truth, green bounding boxes are true positives, and red bounding boxes are false positives.

as lower-level features, compared with the middle-level features in CFM3. The optical flow clearly encodes motion information which is not in CFM3 features. By adding the other components of the proposed ensemble model, our detector can achieve 9.32% MR. The MR is slightly increased to 9.37% by removing motion information. Fig. 4.8 shows the visualization of detection results of different variants of the CFM3 detector. By involving these hand-crafted features, more hard false positives can be eliminated by the proposed detector.

Method	Avg. miss rate (%)
CFM3+Pixel label.	11.58
CFM3+CFM4+CFM5+Pixel label.	9.94
CFM3+CFM4+CFM5+DCNN+Pixel label.	9.53
(CFM3+ACF)+CFM4+CFM5+ DCNN+Pixel label.	9.06
(CFM3+ACF+Flow)+CFM4+CFM5+ DCNN+Pixel label. (All-in-one)	**8.93**

Table 4.6: Comparison of detection performance (on Reasonable) of different ensemble models with pixel labelling. DCNN: the entire VGG16 model fine-tuned by hard negative data collected by CFM3; Pixel label.: pixel labelling model; Flow: optical flow. The pixel labelling model consistently improves all the considered models in this table. The All-in-one model set a new record on the Caltech pedestrian benchmark

4.7.2 Pixel labelling

As shown in Section 4.5, the pixel labelling model is also complementary to convolutional features. Table 4.6 shows the detection performance of different ensemble models enhanced by pixel labelling model. The best result is achieved by combining the most number of different types of models (which is referred to as All-in-one), which reduces the MR on the Caltech Reasonable setting from the previous best 11.7% to 8.9%. Note that the combination rule used by our methods is simple, which implies a potential for further improvement.

4.7.3 Ablation studies

We investigate the overall pipeline of the All-in-one model by adding each component step by step, which is shown in Table 4.7. As the start point, the CFM3a model with the original VGG16 model pre-trained on ImageNet data achieves a miss rate of 18.71%. A 5.22% performance gain can be obtained by fine-tuning the VGG16 model with bootstrapped data. The detection results can be improved to 10.46% (better than all previous methods) by adding CFM4 and CFM5 models to construct an ensemble model. We obtain 0.39% performance improvement if we use the entire VGG16 model (fine-tuned by bootstrapped data with CFM3) as a component of our ensemble model. Combining the pixel labelling information to detected bounding boxes can further reduce the MR by 0.54%. By replacing the CFM3 model to CFM3+ACF+Flow model, the MR of our ensemble model can eventually achieve 8.93% on the Caltech Reasonable test setting.

Model	CFM3a	CFM3	CFM3+CFM4	CFM3+CFM4 +CFM5	CFM3+CFM4 +CFM5+DCNN	CFM3+CFM4+CFM5 +DCNN+Label.	All-in-one
Pipeline	CFM3a	fine-tuning	Add CFM4	Add CFM5	Add DCNN	Add Pixel Label.	Use (CFM3+ACF+Flow)
Miss rate (%)	18.71	13.49	10.68	10.46	10.07	9.53	8.93
Improve. (%)	–	+5.22	+2.81	+0.22	+0.39	+0.54	+0.6

Table 4.7: Ablation studies of the All-in-one model on the Caltech Reasonable test setting

4.7.4 Fast ensemble models

In this section, we investigate the speed issue of the proposed detector. Our All-in-one model takes about 8s for processing one 640×480 image on a workstation with one octa-core Intel Xeon 2.30GHz processor and one Nvidia Tesla K40c GPU. Most of time (about 7s) is spent on the extraction of the CFMs on a multi-scale image pyramid. The remaining components of the ensemble model take less than 1s to process the passed region proposals. The pixel labelling model only uses about 0.25s to process one image since it only need to be applied to one scale. It can be easily observed that the current bottleneck of the proposed detector is the CFM3 which is used to extract region proposals with associated detection scores. The speed of our detector can be accelerated using a light-weight proposal method at the start of the pipeline in Fig. 4.3.

We use two pedestrian detectors ACF (Dollár et al., 2014) and checkerboards (Zhang et al., 2015) as the proposal methods. Our ACF detector consists of 4096 depth-4 decision trees, trained via real-Adaboost. The model has size 128×64 pixels, and is trained via four rounds of bootstrapping. The sliding window stride is 4 pixels. The checkerboards detector is trained using almost identical parameters as for ACF. The only difference is that the feature channels are the results of convolving the ACF channels with a set of checkerboards filters. In our implementation, we adopt a set of 12 binary 2×2 filters to generate checkerboards feature channels. To limit the number of region proposals, we set a threshold of the above two detectors to generate about 20 proposals per image in average.

Table 4.8 shows the detection performance of the original ensemble model and

Method	Avg. miss rate (%)	runtime (s)
CFM3 (proposals)+CFM4+ CFM5+DCNN+Pixel label.	9.53	8.0
ACF (proposals)+CFM3+CFM4+ CFM5+DCNN+Pixel label.	12.20	0.85
Checkerboards (proposals)+CFM3+ CFM4+CFM5+DCNN+Pixel label.	10.65	1.25

Table 4.8: Comparison of detection performance (on Reasonable) between the original ensemble model and fast ensemble models

fast ensemble models on Caltech Reasonable test setting. We can observe that the quality of proposals are enhanced by a large margin using both ensemble models and the pixel labelling model. The best result of fast ensemble models is achieved by using proposals generated by the checkerboards detector. This method uses the data collected by checkerboards detector as the initial fine-tuning data. With a negotiable performance loss (e.g., 1.12%), it's about 6 times faster than the original method. Note that the fast ensemble model (with checkerboards proposals) also achieves the state-of-the-art results.

4.7.5 Comparison to state-of-the-art approaches

In this section, we demonstrate the effectiveness and robustness of the proposed model in three dominant datasets: Caltech, Inria, and KITTI.

4.7.5.1 Caltech

We compare the detection performance of our models with existing state-of-the-art approaches on the Caltech dataset. Table 4.9 and Fig. 4.9 compares our models with a wide range of detectors, including boosted decision forests trained on hand-crafted features, RCNN-based methods and the state-of-the-art methods on the Caltech Reasonable test setting. The performance of the first two types are quite close to each other. Using only one single layer of convolutional feature map, our CFM3 model has outperformed all other methods except the two sophisticated methods (Tian et al., 2015; Cai et al., 2015). Note that the RCNN based methods are based on larger models than CFM3. As feature representation, the CFM from the Conv3-3 layer of our fine-tuned model significantly outperforms all other hand-crafted features. The CFM3+Pixel labelling model already outperforms the state-of-the-art performance achieved by sophisticated methods (Tian et al., 2015; Cai et al., 2015). Our

Type	Method	Miss Rate (%)
Hand-crafted Features	SpatialPooling (Paisitkriangkrai et al., 2014b)	29.24
	SpatialPooling+ (Paisitkriangkrai et al., 2016)†	21.89
	LDCF (Nam et al., 2014)	24.80
	Checkerboards (Zhang et al., 2015)	18.47
	Checkerboards+ (Zhang et al., 2015)†	17.10
RCNN based	AlexNet (Hosang et al., 2015)	23.32
	GoogLeNet (Tian et al., 2015)	16.43
State-of-the-arts	DeepParts (Tian et al., 2015)	11.89
	CompACT-Deep (Cai et al., 2015)	11.75
Ours	CFM3	13.49
	CFM3+Label.	11.58
	CFM3+CFM4+CFM5	10.46
	CFM3+CFM4+CFM5+DCNN+Label.	9.53
	All-in-one†	**8.93**

Table 4.9: Detection performance of different types of detectors on the Caltech Reasonable test setting. Three types of approaches are compared in this table, including boosted decision trees trained on hand-crafted features, RCNN-based methods and the state-of-the-art sophisticated methods. All of our models outperform the first three types of models, and our All-in-one set a new recorded MR on Caltech pedestrian benchmark. † indicates the methods trained with optical flow features

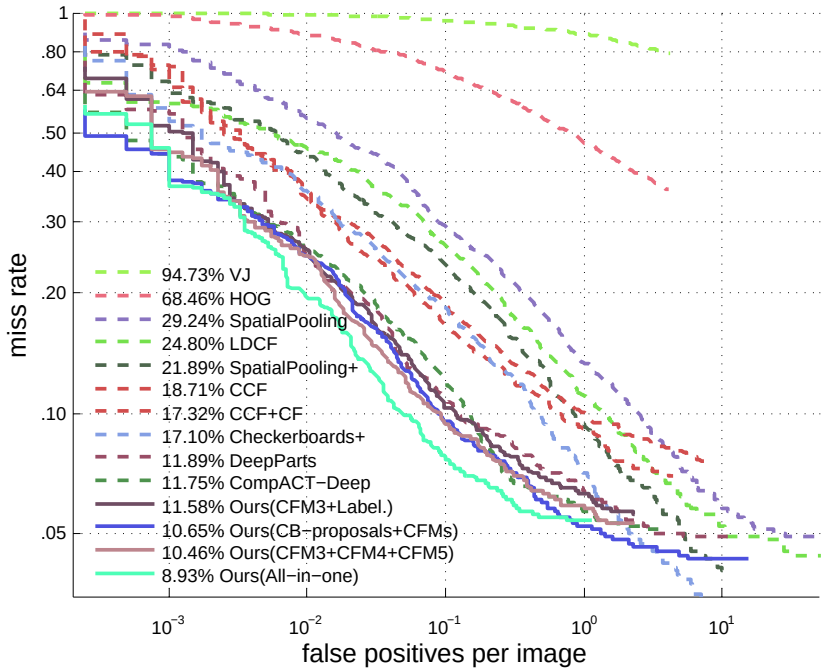

Figure 4.9: Comparison to state-of-the-art approaches on the Caltech Reasonable test setting.

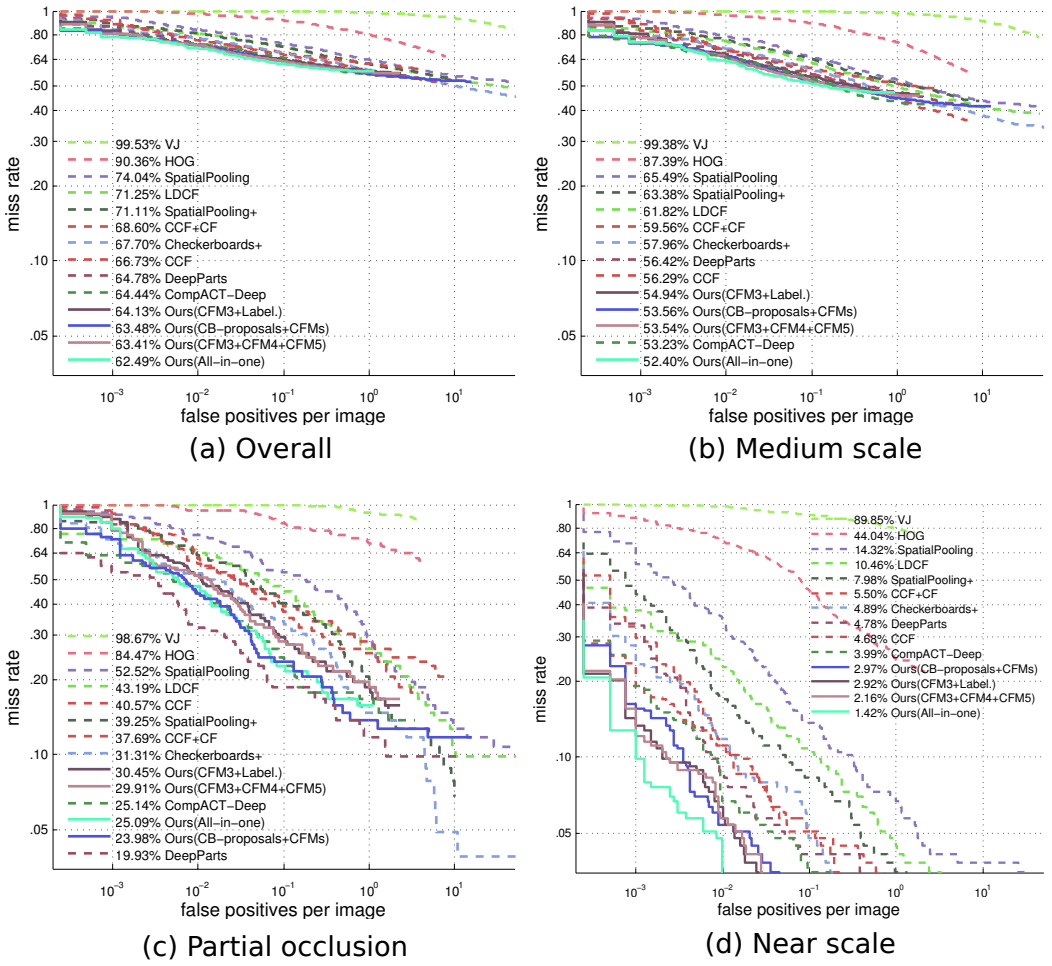

Figure 4.10: Comparison to state-of-the-art approaches on various Caltech test settings.

CFM3+CFM4+CFM5 model performs even better. Without using hand-crafted features, our model can achieve 9.53% MR. The best result is achieved by the All-in-one model which combines a number of hand-crafted features and CFM models.

Fig. 4.10 shows a more complete evaluation of the proposed detection framework on various Caltech test settings, including Overall, Near scale, Medium scale, Partial occlusion. We can observe that our ensemble models achieve the best results on most test settings (including Reasonable). On the Partial occlusion test setting, our models are only outperformed by DeepParts which is specifically trained for handling occlusions.

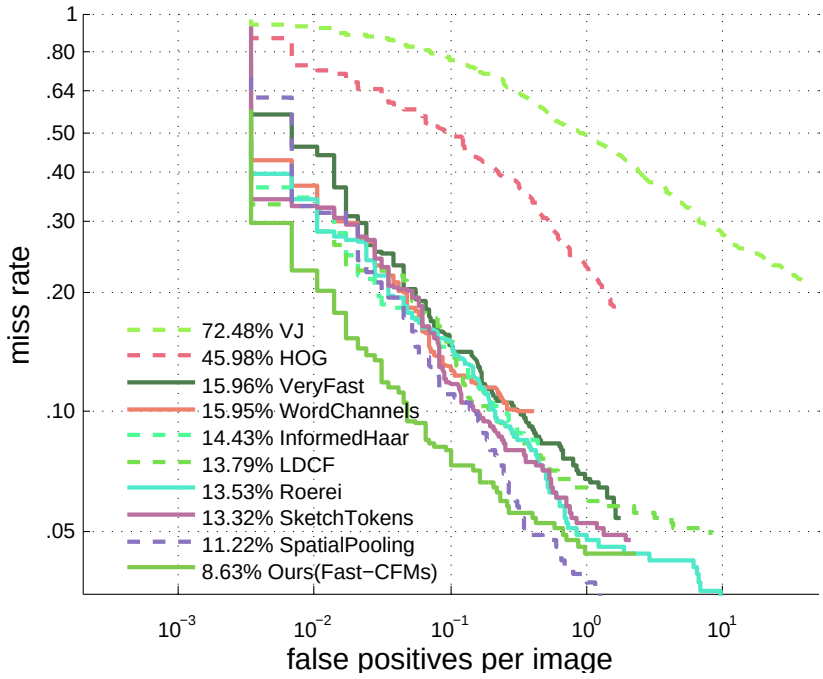

Figure 4.11: Comparison to state-of-the-art approaches on the Inria positive test set.

4.7.5.2 Inria

Fig. 4.11 represents the detection results on the Inria dataset. In our experiments, we only apply the fast ensemble model without using the pixel labelling method. Since our pixel labelling model is trained on the Cityscapes dataset which has totally different scenes from the Inria dataset, the improvement of pixel labelling is limited for this dataset. It can be observed that our method achieves the lowest MR of 8.63% outperforming all previously-reported results.

4.7.5.3 KITTI

Table 4.10 shows the detection results on the KITTI dataset. Since images of KITTI are larger than in Caltech, the feature extraction of `CFM3` model is time-consuming. In our experiments, only the fast ensemble model with Checkerboards proposals is used for testing on KITTI. Our model achieves competitive results, 74.22%, 63.26%, and 56.44% AP on Easy, Moderate, and Hard subsets respectively. Fig. 4.12 presents the comparison of detection performance on the KITTI Moderate test subset. It can be observed that the proposed detector outperforms all published monocular-based methods. Note that the 3DOP (Chen et al., 2015b) is based on stereo images. The proposed ensemble model is the best-performing detector based on DCNN, and sur-

Method	Moderate(%)	Easy(%)	Hard(%)
3DOP* (Chen et al., 2015b)	67.47	81.78	64.70
Fast-CFMs (Ours)	63.26	74.22	56.44
Reionlets (Wang et al., 2013b)	61.15	73.14	55.21
CompACT-Deep (Cai et al., 2015)	58.74	70.69	52.71
DeepParts (Tian et al., 2015)	58.67	70.49	52.78
FilteredICF (Zhang et al., 2015)	56.75	67.65	51.12
pAUCEnsT (Paisitkriangkrai et al., 2016)	54.49	65.26	48.60
R-CNN (Hosang et al., 2015)	50.13	61.61	44.79

Table 4.10: Detection results (AP) on three KITTI test subsets. Note: * indicates the methods trained with stereo images

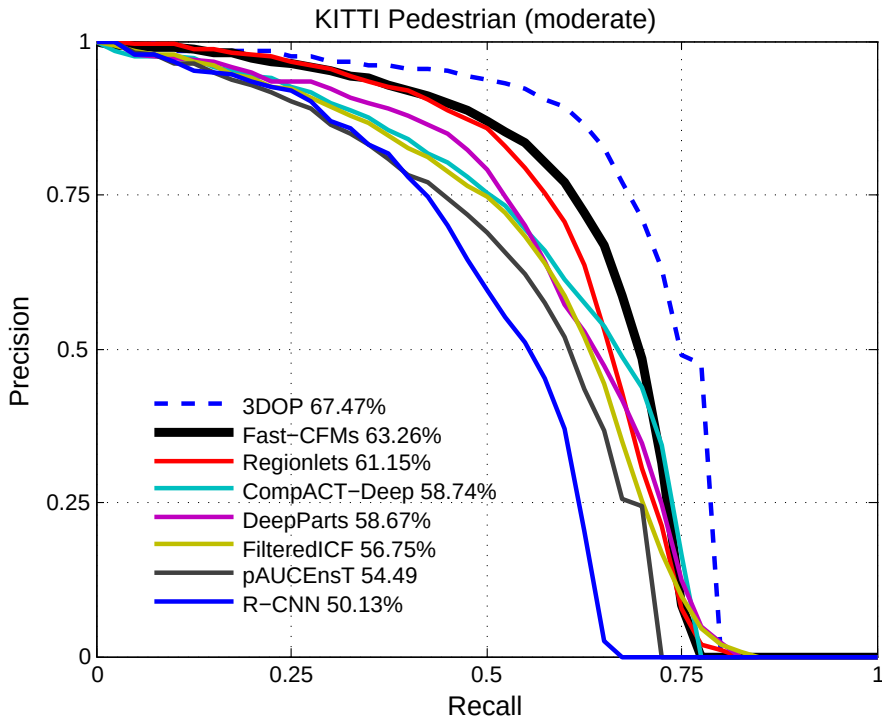

Figure 4.12: Comparison to state-of-the-art approaches on the KITTI Moderate test set.

passes CompACT-Deep (Cai et al., 2015) and DeepParts (Tian et al., 2015) by 4.52% and 4.59% respectively.

4.8 Conclusion

In this chapter, we have built a simple-yet-powerful pedestrian detector, which reuses inner layers of convolutional features extracted by a properly fine-tuned VGG16 model. This 'vanilla' model has already achieved the best reported results on the Caltech dataset, using the same training data as previous DCNN approaches. With a few simple modifications, its variants have achieved even more significant results.

We have presented extensive and systematic empirical evaluations on the effectiveness of DCNN features for pedestrian detection. We show that it is possible to build the best pedestrian detector, yet avoiding complex custom designs. We also show that a pixel labelling model can be used to improve performance by simply incorporating the labelling scores with the detection scores of a standard pedestrian detector. Note that simple combination rules are used here, which leaves potentials for further improvement. For example the ROI pooling for further speed and performance improvement.

Chapter 5

Deep CNNs with Spatially Weighted Pooling for Fine-Grained Car Recognition

5.1 Introduction

Fine-grained object recognition, exemplified by fine-grained car recognition, has attracted much attention recently. Many works and datasets have been proposed in this research field (Krause et al., 2015; Lin et al., 2015c; Krause et al., 2013; Yang et al., 2015b). Compared to other objects, cars have some unique properties, which provides a range of challenging research topics in object recognition. The enormous number of car models makes car a rich object class. Moreover, cars have a large intra-class variation due to unconstrained poses and multiple viewpoints. Cars also have a unique hierarchical structure, which contains three levels from top to bottom: car make, car model, and year of manufacture. This structure presents a direction to address the fine-grained car recognition in a hierarchical way which targets at recognizing the identity of a car, such as car make, car model, even the year of manufacture. In contrast to generic object classification (Felzenszwalb et al., 2010; Wang et al., 2013b; Hu et al., 2016), the fine-grained car classification aims to distinguish subcategories within the same car category. Car model classification is a intra-class classification task which is made difficult by the small visual differences between subcategories, unconstrained poses, different illuminations, and cluttered backgrounds. In this chapter, we mainly focus on the car model classification.

A common approach for fine-grained classification tasks is the parts-based pooling strategy (Zhang et al., 2012). In this approach, various discriminative parts of the object are firstly localized, each corresponding to a human-specified object part. Then local features falling into each part are pooled together to obtain a pooled fea-

ture vector used for classification. Those parts are often defined manually based on domain-specific knowledge and part-based detectors are trained in a supervised method. However, there is usually no human-specified parts annotation in many fine-grained classification tasks. Annotating parts is significantly more challenging than collecting image labels. Furthermore, these human-specified parts may not be optimal for the specific task. Another line of research focuses on the robust feature representation of images, such as the VLAD (Jégou et al., 2010), Fisher vector (Perronnin et al., 2010) with SIFT features (Lowe, 1999). Recently, deep convolutional neural networks (DCNNs) have been shown to significantly outperform comparable methods on a wide variety of vision problems (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015; He et al., 2016). By replacing the SIFT with features extracted from convolutional layers of a DCNN pre-trained on ImageNet (Deng et al., 2009), the fisher vector with DCNN features (Cimpoi et al., 2015) achieve state-of-the-art results on a number of classification tasks.

Although DCNNs achieve good results in generic object classification, their performances are still below the aforementioned methods in fine-grained classification tasks. These DCNNs under-perform mainly because their architectures are not optimal for fine-grained objects, especially when objects are small and appear in clutter. A breakthrough was made recently by a cross-convolutional-layer pooling method (Liu et al., 2015). This method extracts subarrarys of convolutional feature maps (CFMs) of a convolutional layer as local features and uses the CFMs of the successive convolutional layer as pooling channels. Then, the extracted features are pooled with these pooling channels to generate more robust image representations. This method achieves the state-of-the-art results on several popular visual classification tasks. Later, a bilinear CNN framework (Lin et al., 2015c) has been proposed for the fine-grained visual recognition. This framework consists of two feature extractors based on DCNNs, one is used for extracting features and the other is used for generating pooling channels. The outputs of these two DCNNs are multiplied using outer product at each location of the image and pooled to obtain an image descriptor. The bilinear framework can capture pairwise correlations between CFMs of different DCNNs in a translationally invariant manner which is particularly useful for the fine-grained object classification.

We propose a spatially weighted pooling (swp) strategy which considerably enhances the robustness and effectiveness of the feature representation of most dominant DCNNs for the fine-grained car classification. More specifically, the swp is a novel pooling layer which contains a predefined number of spatially weighted masks or pooling channels. The swp pools the extracted features of DCNNs with the guidance of its learnt masks. In addition, the swp layer is compatible with most DC-

NNs, such as AlexNet (Krizhevsky et al., 2012), VGGNets (Simonyan and Zisserman, 2015), and ResNets (He et al., 2016). A swp layer can be embedded into those DCNNs with slight modifications. The proposed method also addresses several drawbacks of aforementioned approaches. Firstly, the parameters of the swp layer can be learned in the end-to-end training process of the DCNN. Moreover, the extracted features and the learnt masks are provided from the same DCNN, which is simpler than the bilinear CNN framework.

Experiments are conducted on four fine-grained car datasets. The experimental results show that the swp method can improve the accuracies of fine-grained car classification considerably over the baselines. For example, the VGG16 and ResNet101 achieve 85.4% and 90.9% accuracy on the Cars-196 dataset respectively. The proposed method improves the performance to 90.7% and 93.1%, respectively. More importantly, the ResNet101 with the swp layer outperforms other state-of-the-art approaches and achieves the best reported results on the Cars-196 dataset and CompCars dataset.

5.2 Background

Many existing approaches focus on the car model classification which aims to classify the model type of car objects. These approaches can be divided into three categories: texture feature-based approaches, 3D representation-based approaches and DCNN-based approaches.

The texture feature-based approaches are designed for traffic monitoring using fixed surveillance cameras. These approaches are limited to the fine-grained classification of frontal car images. He *et al.* (He et al., 2015a) propose a framework to correctly detect license plates, headlamps and entire car objects. Features of each component are extracted, normalized, and classified using an ensemble classifier. Liao *et al.* (Liao et al., 2015) propose a DPM-based method for car parts localization and classification. This method uses supervised DPM to categorize frontal car images and integrates discriminative powers of different parts into the classification. Based on the speeded-up robust features (SURF), Hsieh *et al.* (Liao et al., 2015) develop a new symmetrical SURF descriptor to enhance the discriminative power of SURF to detect and segment frontal car images into several grids for the fine-grained classification.

3D representation-based approaches are able to handle car images with unconstrained poses and multiple viewpoints. Krause *et al.* (Krause et al., 2013) propose to extract 3D car representations to train geometry classifiers which outperform their corresponding 2D counterparts and state-of-the-arts in the fine-grained categoriza-

tion. Gu *et al.* (Gu and Lee, 2013) introduce an effective pose and center estimation approach to initialize the active shape model which is used to deal with pose variation and normalization. Lin *et al.* (Lin et al., 2014) propose to optimize the 3D active shape model and the fine-grained classification jointly. 3D model fitting is used to obtain positions of landmarks. Hsiao *et al.* (Hsiao et al., 2014) propose an approach which constructs 3D space curves by back-projecting image curves onto silhouette-based visual hulls. Then, refining them using three-view based alignment technique. BoxCars (Sochor et al., 2016) propose to use the automatically calibrated camera to extract 3D information for improving the recognition system.

Recently, deep convolutional neural networks achieve impressive results at a range of image classification tasks. DCNN-based approaches use dominant DCNNs as the main framework for the fine-grained classification. Yang *et al.* (Yang et al., 2015b) directly apply several DCNN baselines to the car model classification and achieve comparable results with state-of-the-art methods. Krause *et al.* (Krause et al., 2015) propose to use co-segmentation and alignment in combination with R-CNN (Girshick et al., 2014a) to generate parts annotations which are useful for fine-grained classification tasks. Liu *et al.* (Liu et al., 2015) propose to use the sub-arrays of convolutional layer activations as local features and pool extracted local features by using convolutional feature maps from two successive convolutional layers. Lin *et al.* (Lin et al., 2015c) propose a bilinear CNN framework that consists of two DCNNs whose convolutional feature maps are multiplied using outer product at each location of the image and pooled to obtain an image descriptor.

5.3 Properties of fine-grained car datasets

Existing fine-grained car recognition datasets (Krause et al., 2013; Yang et al., 2015b; Lin et al., 2014) generally fall into one of three categories, namely, web-image datasets, surveillance-image datasets, and hybrid datasets. The web-image dataset collected car images from public websites and search engines. These images contain a variety of car models with unconstrained poses. Fig. 5.1 (left block) shows some examples of web images. Note that cars have large appearance variations due to the unconstrained poses and multiple viewpoints. The surveillance-image dataset collected car images from videos taken by fixed surveillance cameras on various traffic scenes. Fig. 5.1 (right block) illustrates some examples of surveillance images, which are captured by cameras from the front view. Hybrid datasets are a combination of those web images and surveillance images.

The fine-grained car datasets own a unique structure. Car objects can be organized into a hierarchical structure. This structure consists of three levels, namely

Figure 5.1: Example images from the web-image dataset (left block). These images have large appearance variations due to the unconstrained poses and multiple viewpoints.Toyota Camry Example images from the surveillance-image dataset (right block). These frontal car images are captured by fixed surveillance cameras from the front view.

car make, car model, and the year of manufacture, from top to bottom as illustrated on Fig. 5.2. The fine-grained car classification can be implemented in a hierarchical way. Following the evaluation protocol in (Krause et al., 2013; Yang et al., 2015b), we mainly focus on the car model classification in this chapter. The complexity of the classification task can be further increased by the fact that there are small visual differences between car models of the same series produced at adjacent years. For instance, four versions of Toyota Camry were produced from 2008 to 2011, respectively. However, they share very similar appearance design.

Two dominant datasets are used to evaluate our method in this chapter, Cars-196 (Krause et al., 2013) and CompCars (Yang et al., 2015b). The Cars-196 dataset contains 16,185 images of 196 car models. Car images are captured from multiple viewpoints in different scenes. The CompCars dataset contains 52,083 images of 431 car models. Car images are captured from multiple viewpoints in various scenes.

In addition to existing datasets, we collected two challenging fine-grained car datasets: CarFlag-563 and CarFlag-1532. The CarFlag-563 is a large-scale surveillance-image dataset which consists of 113,867 frontal car images captured by fixed surveillance cameras on various traffic scenes. The CarFlag-563 contains 100 car makes with 563 car models, and 1152 subcategories on the level of year of manufacture. The CarFlag-1532 is a large-scale web-image dataset which contains 165 car makes with 1532 car models. It contains a total of 156,098 car images with unconstrained poses. The CarFlag-1532 is more challenging due to the large number of car objects and the rich diversity of car models. Table 5.1 provides a summary of existing fine-grained car datasets.

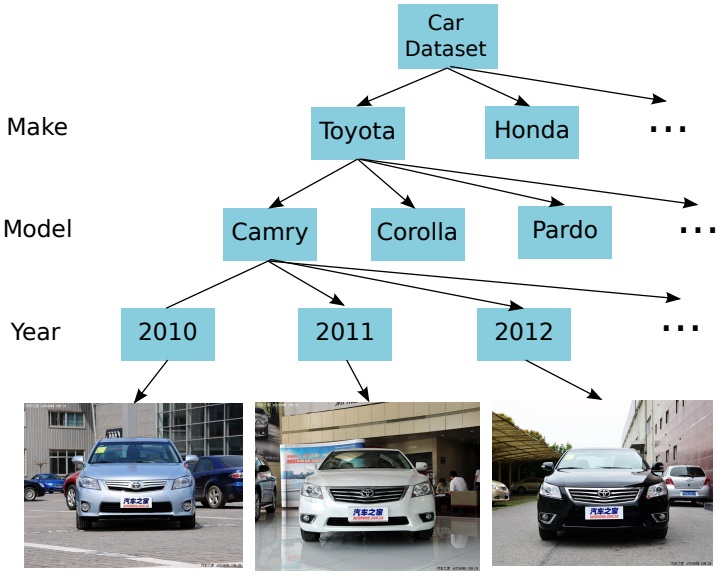

Figure 5.2: The structure of the car model hierarchy. Several car models of Toyota Camry produced in different years are also displayed.

	Images		Structures			Properties			
	# training images	# test images	# car makes	# car models	# year of prod.	web-image	surveillance	frontal view	multi-views
Cars-196 (Krause et al., 2013)	8,144	8,041	-	196	-	✓			✓
CompCars (Yang et al., 2015b)	36,456	15,627	75	431	1631	✓	✓	✓	✓
CarFlag-563	80,210	33,657	100	563	1152		✓	✓	
CarFlag-1532	109,938	46,160	165	1532	5320	✓			✓

Table 5.1: Comparison of existing fine-grained car datasets. The first two columns indicate the amount of training/test data in each dataset. Note that both the CarFlag-563 and CarFlag-1532 datasets are larger than other existing datasets. The middle three columns present the diversity of each dataset. The next four columns provide additional properties of each dataset.

5.4 Proposed approach

Traditional pooling methods aim to reduce the spatial size of convolutional feature maps and discard irrelevant details over the pooling region. In this section, we propose an alternative pooling strategy which can significantly improve the performance of the fine-grained car classification.

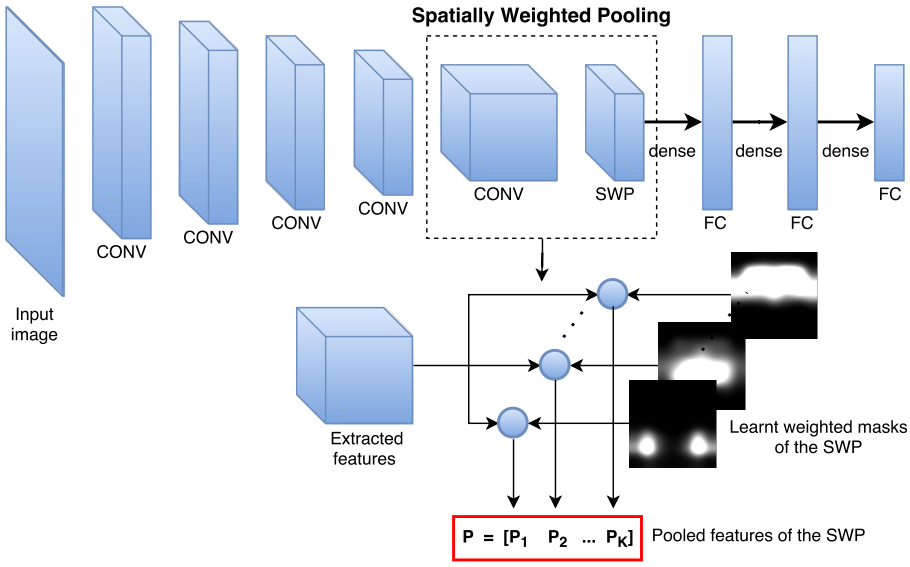

Figure 5.3: Overview of the proposed method. The convolutional feature maps are generated by the last convolutional layer of a pre-trained deep CNN model. The swp layer contains a predefined number of spatially weighted masks and pools the extracted feature maps with the guidance of learnt masks. The image representation is the concatenation of pooled features from multiple pooling channels.

5.4.1 Spatially weighted pooling

After extracting convolutional feature maps (CFMs) from a convolutional layer of a DCNN, one can directly perform max-pooling or average pooling to obtain the image representation. Many previous studies have explored the usage of the mid-level convolutional features in classification tasks (Liu et al., 2014; Li et al., 2015, 2016b). In (Liu et al., 2015), extracted local features are pooled by using CFMs from two successive convolutional layers. The bilinear CNN framework consists of two DCNNs whose CFMs are multiplied using outer product at each location of the image and pooled to obtain an image descriptor. The use of CFMs as pooling channels is motivated by the observation that a feature map of a deep convolutional layer is usually sparse and indicates some semantically meaningful regions. Moreover, the discriminative power of the feature representation can benefit from using a larger number of pooling channels. It has been shown that both methods achieve significantly better performance than previous methods.

Although the aforementioned methods achieve good results, there are still some drawbacks on those methods. The cross-convolutional-layer pooling employs an additional SVM as the classifier. Since the DCNN and the SVM are trained separately, this method does not support the end-to-end training. The bilinear CNN framework

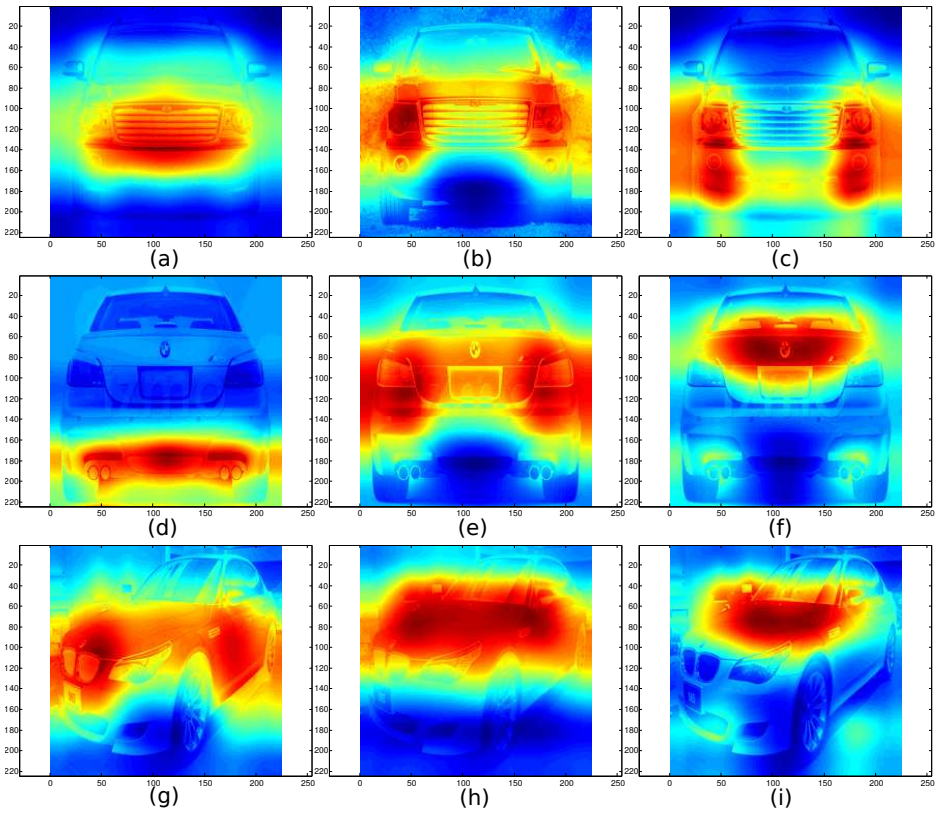

Figure 5.4: Visualization of some spatially weighted masks learned from the VGG16 with the swp layer.

consists of two DCNNs, which increases the complexity and computational cost of the system. To handle the above weaknesses, we introduce the spatially weighted pooling (swp) method to enhance the robustness and effectiveness of the feature representation of DCNNs. Compared to traditional pooling methods, the swp contains a predefined number of spatially weighted masks and pools the features extracted from the previous layer with the guidance of learnt masks. Moreover, the parameters of the swp layer can be learned in the end-to-end training process of DCNNs. The overview of the swp method is illustrated in Fig. 5.3.

Spatially weighted masks of a swp layer can be treated as a set of various pooling channels. These masks can indicate some discriminative image regions used by DCNNs to identify the category of the car object. Features extracted from these discriminative regions contribute significantly to the pooled features. This observation is showed in Fig. 5.4. We choose three typical car images taken from the Cars-196 dataset, including the frontal car image, the rear car image, and the side car image. We show several spatially weighted masks learned from the swp layer of the

VGG16 and overlay them on the original images for better visualization. As can be observed from Fig. 5.4, the activated regions of the learnt masks are actually semantically meaningful. For example, the activated regions of Fig. 5.4(a) correspond to the front face of the car. Fig. 5.4(c) indicates that both the front lights and fog lights are discriminative parts of the car. Fig. 5.4(d) shows that the activated regions are exhaust pipes of the car. Hence, these masks serve as a similar role as the part-region indicator maps. Compared to the human-specified part annotations, parameters of spatially weighted masks can be optimized in the process of training without any domain-specific knowledge.

Suppose the extracted feature maps can be formulated as a tensor of the size $H \times W \times D$, where H, W denote the height and width of each feature map and D denotes the number of features maps. The spatially weighted masks of a swp layer can be formulated as K pooling channels of the size $H \times W$. The height and width of these masks are the same as the size of the input feature maps. The pooled feature of the kth learnt mask, denoted as $\mathbf{P_k}$, can be calculated by the following equation:

$$\mathbf{P}_k = \sum_{i}^{H \times W} m_i^k x_i \tag{5.1}$$

where x_i denotes the ith local feature of the input feature maps and m_i^k is the corresponding coefficient in the kth learnt mask. The pooled feature $\mathbf{P_k}$ has a dimension of $1 \times 1 \times D$. Essentially, each learnt mask defines a pooling channel and the image representation is the concatenation of pooled results from multiple pooling channels. Formally, the image representation generated by the swp layer can be expressed as follows:

$$\begin{aligned} &\mathbf{P} = [\mathbf{P}_1{}^\top, \mathbf{P}_2{}^\top, \cdots, \mathbf{P}_k{}^\top, \cdots, \mathbf{P}_K{}^\top]^\top \\ &\text{where}, \mathbf{P}_k = \sum_{i}^{H \times W} m_i^k x_i \end{aligned} \tag{5.2}$$

where $\mathbf{P}$ denotes the output of the swp layer, which is calculated by concatenating the pooled feature of each learnt mask $\mathbf{P_k}$, $k = 1, \cdots, K$ and K refers to the total number of learnt masks. Concatenating all K pooled D-dimensional features generates $K \times D$ feature representation for the input image.

5.4.2 Using swp with DCNNs

The swp layer is compatible with most dominant DCNNs, including AlexNet (Krizhevsky et al., 2012), VGG16 (Simonyan and Zisserman, 2015), and various ResNets (He et al., 2016). Suppose all above DCNNs retain their default architectures and the size of in-

put image is set to 224×224 (227×227 for AlexNet). We can apply the swp layer to those DCNNs with minimal modifications to their structures.

To apply the swp layer to the AlexNet and VGG16, we insert the swp layer between the last max-pooling layer and the first fully-connected layer. The output of the max-pooling layer has a dimension of $6 \times 6 \times 256$ for the AlexNet and $7 \times 7 \times 512$ for the VGG16, respectively. Through the processing of the swp layer, the dimension of pooled results is changed to $K \times D$, where K is the number of spatially weighted masks and D is the number of feature maps. So the dimension of the swp features changes as the K varies. The number of neurons of two succeeding fully-connected (FC) layers needs to be adjusted appropriately to avoid over-fitting or under-fitting problems. The related experiments are shown on the Section 5.5.3.

ResNets use the global average pooling (GAP) layer as the last pooling layer. The output of the GAP layer has a dimension of $1 \times 1 \times 2048$. In fact, the GAP layer can be treated as a special case of the swp layer which contains only one spatially weighted mask with equally distributed weights. Therefore, we straightforwardly replace the GAP layer with the swp layer for ResNets. Note that the output of the swp layer is K times larger than the original output of the GAP layer, an over-fitting issue may occur due to the increase of the dimension of the swp features. To overcome this problem, we insert one additional FC layer between the swp layer and the last FC layer. Both the swp layer and the inserted FC layer are followed by a Batch Normalization (BN) layer to speed-up the training process.

5.4.3 End-to-end training of swp

To train a DCNN with the swp layer, the DCNN needs to back-propagate derivatives from the loss function to the input image. This means that the parameters of spatially weighted masks can be trained by back-propagating the gradients of the classification loss. Back-propagation routes gradients through the swp layer, as described below.

Suppose M is the set of learnt masks in the swp layer and has size of $H \times W \times K$, K denotes to the number of masks. X is the input feature maps of the swp layer and has a dimension of $H \times W \times D$. The output of the swp layer is P of size $K \times D$. Let $d\ell/dP$ be the gradient of the loss function ℓ with respect to pooled features P in the equation 5.2, then by the chain rule of gradients we have:

$$\frac{\partial \ell}{\partial M} = \frac{\partial \ell}{\partial P}\frac{\partial P}{\partial M} = \frac{\partial \ell}{\partial P}X \tag{5.3}$$

$$\frac{\partial \ell}{\partial X} = \frac{\partial \ell}{\partial P}\frac{\partial P}{\partial X} = \frac{\partial \ell}{\partial P}M^T \tag{5.4}$$

where the partial derivatives $\partial \ell / \partial P$ are already computed by the backwards function of the layer on top of the swp layer. The derivatives of the layers below the swp layer can be computed using chain rule. We fine-tune our models using stochastic gradient descent with mini-batches with weight decay and momentum as described on the Section 5.5.2.

5.5 Experiments

In this section, we comprehensively evaluate the swp method on the Cars-196 (Krause et al., 2013) which is the currently most predominant dataset. Then, we show the robustness and effectiveness of the proposed method on three recently proposed fine-grained car datasets: CompCars (Yang et al., 2015b), Carflag-1532, and Carflag-563. These four datasets cover a large number of cars of various models from different scenes. Previous studies (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015; He et al., 2016) have shown that using some pre-trained DCNNs lead to good performance on generic object classification tasks. In our experiments, we apply the swp method to those DCNNs to further improve the performance of the fine-grained car classification.

5.5.1 Methods

We consider three kinds of widely used pre-trained DCNNs: AlexNet, VGG16, and ResNets as baselines to evaluate the performance improvement caused by using the swp method. For ResNets, we conduct experiments on models with different depths, such as ResNet50 and ResNet101. In addition to ResNets with default structures, we introduce different variants of ResNets as the improved baselines.

5.5.1.1 AlexNet with swp

As described on Section 5.4.2, we apply the swp layer to the AlexNet by inserting the swp layer between the last max-pooling layer and the first fully-connected layer. Since the dimension of the pooled features is changed from $6 \times 6 \times 256$ to $K \times 256$, the number of neurons of FC6 and FC7 layer also need to be adjusted appropriately. All other components remain unchanged. We refer to this model as the AlexNet-swp.

5.5.1.2 VGG16 with swp

We apply the swp layer to the VGG16 in a similar way to the AlexNet. The swp layer changes the dimension of the pooled features from $7 \times 7 \times 512$ to $K \times 512$, we

also need to modify the number of neurons of FC6 and FC7 layer to maximize the performance of the variant of VGG16. All other components keep unchanged. We refer to this variant as VGG16-swp.

5.5.1.3 ResNets with local max-pooling

Instead of using the local max-pooling layer, ResNets use the global averaging pooling (GAP) layer as the last pooling layer. However, the pooled results of the GAP lose all spatial information about the object and may be too spatially coarse ($1 \times 1 \times 2048$ via a 7×7 average pooling kernel) to capture the fine-grained information. To remedy this problem, we change the receptive field of pooling kernel from 7×7 to 3×3 and change the stride from 1 to 2, the pooling operator is also changed to the max-pooling which is partially translation invariant. So the last GAP layer is replaced with a local max-pooling (LMP) layer. Since the dimension of pooled features increases by decreasing the size of receptive field, we need to add one additional FC layer followed by a drop-out layer with probability of 0.5 and a ReLU layer to prevent over-fitting during training. All other layers remain unchanged. The ResNets with the LMP layer are referred as ResNet-LMP.

5.5.1.4 ResNets with swp

Since the dimension of output of the GAP layer is $1 \times 1 \times 2048$ which is spatially coarse, the swp cannot extract meaningful representations from such a low dimensional features. Instead of inserting the swp layer between the GAP layer and the FC layer, we straightforwardly replace the GAP layer with the swp layer followed by a batch normalization (BN) layer and a ReLU layer. The swp layer increases the dimension of pooled features from $1 \times 1 \times 2048$ to $K \times 2048$ where K is the number of learnt masks. One additional FC layer is inserted between the swp layer and the last FC layer. ResNets employ the batch normalization to accelerate the training process and get better generalization performance. We similarly add a BN layer after the newly added FC layer to keep the fast training. We refer the ResNet with the swp layer as ResNet-swp.

5.5.2 Implementation details

To adapt the aforementioned DCNNs to the fine-grained car classification, we fine-tune these DCNNs on each car dataset individually. We replace the 1000-way classification layer trained on ImageNet dataset with a randomly initialized C-way soft-max layer, where C is the number of classes in the specific dataset. We resize the image

dimension to 256×256. Then a 224×224 (227×227 for AlexNet) crop is randomly sampled from the image or its horizontal flip, with the per-pixel mean subtracted. Note that, the implementation of ResNets is slightly different from the AlexNet and VGG16. We follow the training and testing procedure proposed in (He et al., 2016). Once we generate crops from the images, we do color jittering that randomly perturbs the brightness, contrast and saturation of these crops. The standard color augmentation in (Krizhevsky et al., 2012) is applied to these crops. We also adopt the batch normalization right after the swp layer and the newly added FC layer before the ReLU activation. The swp layer and the FC layers are initialized by random weights drawn from Gaussian distributions with a fixed standard deviation of 0.005. We use the stochastic gradient descent (SGD) with a mini-batch size of 64. The initial learning rate is set to 0.001 for convolutional layers and 0.01 for the swp layer and the FC layers. The learning rate is divided by 10 after each 40 training epochs. These models are trained for up to 90 epochs. We use a weight decay of 0.0005 and a momentum of 0.9.

5.5.3 Evaluation on the Stanford Cars-196 dataset

The Cars-196 (Krause et al., 2013) dataset contains 16,185 images of 196 car models. This dataset provides ground-truth annotations of bounding boxes, on both the training set and test set. In our experiments, we evaluate the proposed method on the Cars-196 with bounding box annotations. We train proposed models on classes of 196 car models.

Experimental design We investigate the experimental design of the proposed method, especially the number of spatially weighted masks in the swp layer and the number of neurons of the FC layers. The number of masks affects the quality of the swp method and the setting of succeeding FC layers also affects the overall performance of the proposed method. We determine these parameters in a heuristic way for the fine-grained car classification. We set the number of masks K from {1, 4, 9, 16, 25, 36}. The number of neurons N of the FC layers is correlated to the dimension of the swp features. For the AlexNet-swp and VGG16-swp, we set N of the FC6 and FC7 layer from {512, 1024, 2048}. For the ResNets-swp, we set N of the inserted FC layer from {1024, 2048}.

Table 5.2 represents the comparison of classification results of various DCNNs with different parameter settings. For the AlexNet-swp and VGG16-swp, we observe that the accuracies generally improve as we increase the number of masks up to 9. However, setting the number of masks to be too large (*e.g.*, $K = 36$) may hurt the performance. By observing the 36 learnt masks of the AlexNet-swp, we find that

# weighted masks	AlexNet-swp			VGG16-swp		
	$N = 512$	$N = 1024$	$N = 2048$	$N = 512$	$N = 1024$	$N = 2048$
$K = 1$	81.9%	81.2%	80.7%	89.6%	89.2%	88.9%
$K = 4$	83.1%	82.8%	81.2%	90.3%	90.1%	89.7%
$K = 9$	**83.6**%	83.5%	82.0%	**90.7**%	90.4%	90.3%
$K = 16$	**83.6**%	83.4%	81.9%	90.4%	90.2%	90.1%
$K = 25$	83.3%	83.2%	81.8%	90.5%	90.3%	90.1%
$K = 36$	83.2%	83.0%	81.5%	90.3%	90.0%	89.8%
Baselines	80.7%	80.5%	79.8%	87.2%	86.7%	86.0%

ResNet50-swp		ResNet101-swp	
$N = 1024$	$N = 2048$	$N = 1024$	$N = 2048$
91.9%	91.6%	92.7%	92.6%
92.1%	91.7%	92.9%	92.7%
92.3%	92.1%	92.7%	**93.1**%
92.2%	91.9%	92.6%	92.7%
92.0%	91.7%	92.8%	92.9%
92.1%	91.9%	92.8%	92.8%
89.7% with GAP		90.9% with GAP	

Table 5.2: Comparison of classification results of various DCNNs with different parameter settings on the Stanford Cars-196 dataset. For AlexNet-swp and VGG16-swp, N is the number of neurons of FC6 and FC7 layers. For ResNets-swp, N is the number of neurons of the inserted FC layer.

some masks have similar weight distributions. This means that these masks focus on the similar parts of cars and generate redundant pooled features. For ResNets-swp, the performance is not very sensitive to the number of learnt masks. We conjecture that their deep architectures can learn more semantically meaningful information for each mask. We also observe that accuracies decline with the increase of N in the AlexNet-swp and VGG16-swp. Since the swp layer generates a low dimensional output, the large N may lead to an under-fitting issue. The highest accuracy of 93.1% is achieved by the ResNet101-swp with setting $K = 9$ and $N = 2048$. For the rest of our experiments, we set the number of spatially weighted masks K to 9 as it gives the best trade-off between the performance and the size of the trained model. The number of neurons N is set by considering the number of categories of a specific dataset. N is usually greater than the number of categories.

Comparison of results Table 5.3 shows the comparison of classification results on the Stanford Cars-196 dataset with bounding box annotations. The AlexNet achieves accuracy of 78.9%. The VGG16 significantly improves the accuracy to 85.4%. The recently proposed ResNet50 and ResNet101 remarkably outperform the VGG16 by a large margin about 5%. This outcome is consistent with the result in (He et al., 2016), which shows that the depth of DCNNs is very important for improving the

Type	Methods	Acc. on Model
Baselines	AlexNet	78.9%
	VGG16	85.4%
	ResNet50	89.7%
	ResNet101	90.9%
	ResNet50-LMP	91.6%
	ResNet101-LMP	92.9%
Previous	Chai *et al.* (Chai et al., 2013)	78.0%
	FV-CNN(Gosselin et al., 2014)	82.7%
	B-CNN(Lin et al., 2015c)	91.3%
	Krause *et al.* (Krause et al., 2015)	92.6%
Ours	AlexNet-swp	83.6%
	VGG16-swp	90.7%
	ResNet50-swp	92.3%
	ResNet101-swp	93.1%

Table 5.3: Comparison of classification results on the Stanford Cars-196 dataset with bounding box annotations. 'LMP' means the local max-pooling and 'swp' indicates the spatially weighted pooling.

performance of the classification. Moreover, by replacing the GAP layer with the LMP layer and inserting one additional FC layer with 512 neurons for ResNets, we get some performance gains, *e.g.*, 89.7% to 91.6% for ResNet50-LMP and 90.9% to 92.9% for ResNet101-LMP. These results confirm our hypothesis that the GAP may discard some fine-grained information and hurt the performance of the classification. Furthermore, by applying the swp layer to baselines, all their accuracies are improved considerably, *e.g.*, 89.7% to 92.3% for ResNet50-swp and 90.9% to 93.1% for ResNet101-swp. This shows that the swp method does improve the performance of the car model classification. The comparison with state-of-the-art results is also illustrated on Table 5.3. Two recently proposed methods that perform well on this dataset are 91.3% of the B-CNN framework and 92.6% in (Krause et al., 2015). Our ResNet101-swp method achieves the best accuracy of 93.1% outperforming all previously-reported results.

5.5.4 Evaluation on the CompCars dataset

The CompCars (Yang et al., 2015b) is a hybrid dataset which contains 52,083 images of 431 car models. In our experiments, we train proposed models on classes of 431 car models.

Experimental settings To prove the robustness of the swp method on this dataset,

Type	Methods	Acc. on Model	Acc. on Make
Baselines	AlexNet (Yang et al., 2015b)	81.9%	-
	OverFeat (Yang et al., 2015b)	87.9%	-
	GoogLeNet (Yang et al., 2015b)	91.2%	-
	VGG16	92.4%	95.6%
	ResNet50	93.3%	96.1%
	ResNet101	93.7%	96.4%
	ResNet50-LMP	93.6%	96.2%
	ResNet101-LMP	94.2%	96.7%
Previous	BoxCars (Sochor et al., 2016)†	84.8%	-
Ours	AlexNet-swp	88.0%	91.9%
	VGG16-swp	95.3%	97.8%
	ResNet50-swp	97.5%	99.3%
	ResNet101-swp	97.6%	99.3%

Table 5.4: Comparison of classification results on the CompCars dataset. † is quoted from the baseline in (Sochor et al., 2016).

we use the same parameter settings with the best performance on the Cars-196 to the CompCars. We set the number of spatially weighted masks in the swp layer to 9. Since the number of subcategories on the CompCars is greater than the Cars-196, we set the number of neurons of the FC6 and FC7 layer to 1024 for the AlexNet-swp and VGG16-swp. For ResNets-swp, we set the number of neurons of the inserted FC layer to 1024.

Comparison of results Table 5.4 illustrates the comparison of results on the CompCars dataset. We can observe that the trends among the baselines are similar to those on the Stanford Cars-196 dataset. ResNets once again outperform previously proposed DCNNs, including AlexNet, OverFeat, GoogLeNet, and VGG16. Both the ResNet50 and ResNet101 achieve remarkably good results of 93.3% and 93.7%, respectively. Moreover, we can get a slight performance gain by replacing the GAP layer with the LMP layer. For instance, the ResNet101-LMP improve the accuracy from 93.7% to 94.2%. When the swp method is applied to all baselines, their accuracies are further increased by a large margin. The ResNet101-swp model achieves the best accuracy of 97.6% outperforming all previously reported results. We also evaluate above models in the car make classification task (75 makes) on this dataset and the ResNet101-swp achieves the best accuracy of 99.3%.

Type	Methods	Acc. on Model	Acc. on Make
Baselines	AlexNet	67.64%	75.27%
	VGG16	88.62%	92.54%
	ResNet50	88.72%	92.70%
	ResNet101	89.26%	92.91%
	Resnet50-LMP	90.00%	93.26%
	Resnet101-LMP	90.34%	93.58%
Ours	AlexNet-swp	81.80%	86.96%
	VGG16-swp	92.34%	95.09%
	Resnet50-swp	96.22%	98.43%
	Resnet101-swp	96.70%	98.67%

Table 5.5: Comparison of classification results on the CarFlag-1532 dataset.

5.5.5 Evaluation on our CarFlag-1532 dataset

The CarFlag-1532 is a large-scale web-image dataset which contains 156,098 images of 1,532 car models. In our experiments, we train proposed models on classes of 1,532 car models. Fig. 5.5 illustrates some samples from the CarFlag-1532 dataset.

Experimental settings We set the number of spatially weighted masks in the swp layer to 9 as same as the CompCars. Since the number of car models on the CarFlag-1532 is greater than the CompCars, we set the number of neurons of the FC6 and FC7 layer to 2048 for the AlexNet-swp and VGG16-swp. For ResNets-swp, the number of neurons of the inserted FC layer is also set to 2048.

Comparison of results According to table 5.5, we can observe that ResNets once again outperform the VGG16. The ResNet50 and ResNet101 achieve accuracy of 88.72% and 89.26%, respectively. The AlexNet performs extremely worse on this dataset, we conjecture that this model cannot handle such a huge number of car models effectively. The performance of ResNets can be further improved by replacing the GAP layer with the LMP layer, *e.g.*, ResNet50-LMP improves to 90.00% and ResNet101-LMP improves to 90.34%. The swp method takes additional performance gains to each baseline, especially for the AlexNet. The AlexNet-swp remarkably improves the accuracy from 67.64% to 81.80%. The best performance of 96.70% is achieved by the ResNet101-swp. The swp leads to about 7.44% improvement in its accuracy. All above models are evaluated in the car make classification task (165 makes) on this dataset and the ResNet101-swp achieves the best accuracy of 98.5%.

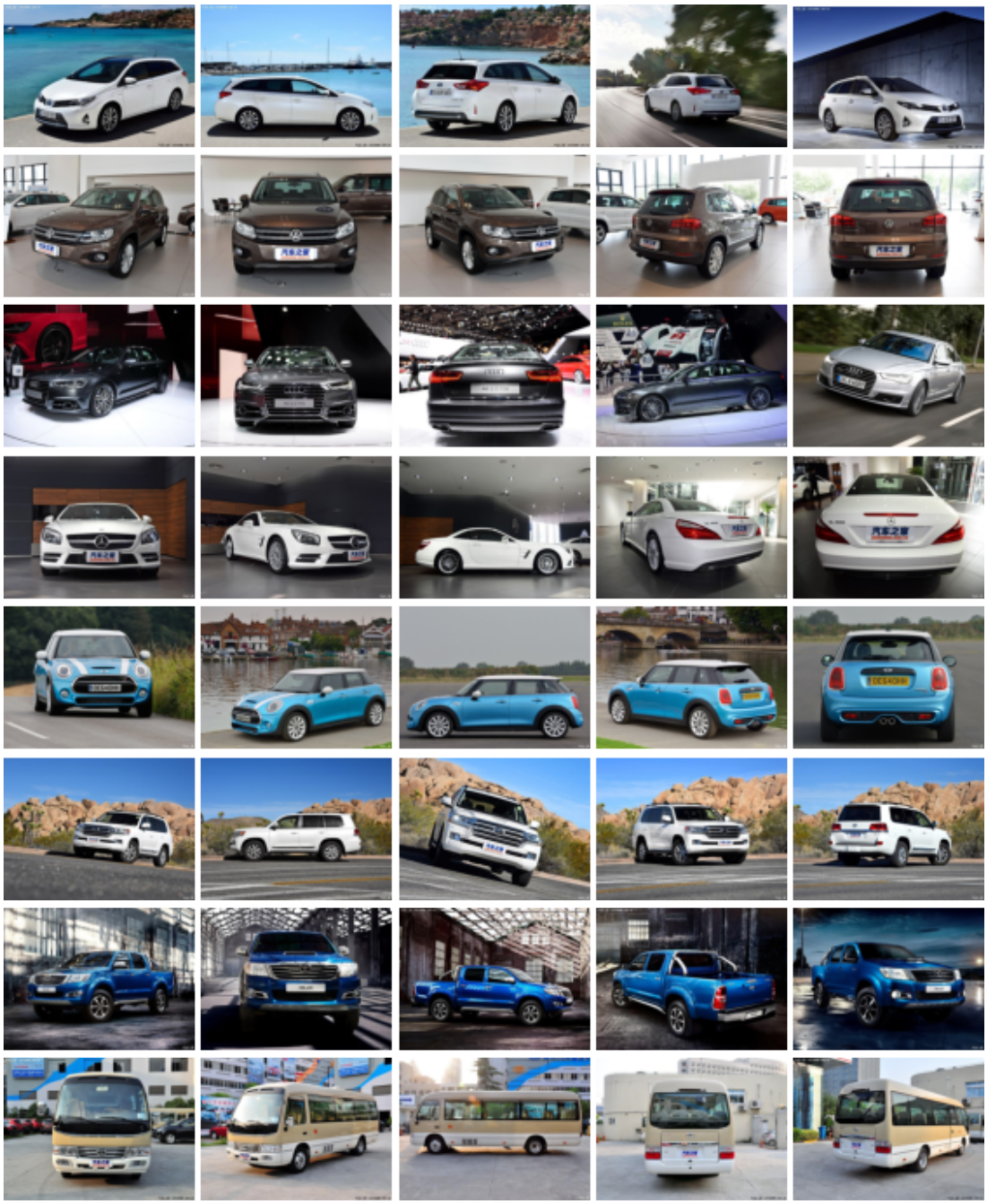

Figure 5.5: Sample images with unconstrained poses from the CarFlag-1532 dataset captured from various scenes.

Methods	Acc. on Year	Acc. on Model	Acc. on Make
AlexNet	85.78%	95.38%	99.38%
VGG16	86.48%	95.59%	99.43%
ResNet50	85.40%	95.28%	99.34%
ResNet101	85.51%	95.35%	99.35%
ResNet50-LMP	86.37%	95.54%	99.42%
ResNet101-LMP	86.52%	95.62%	99.43%
ResNet50-LMP†	86.67%	95.69%	99.46%
ResNet101-LMP†	86.74%	95.72%	99.47%
AlexNet-swp	85.86%	95.39%	99.38%
VGG16-swp	86.70%	95.70%	99.46%
ResNet50-swp	87.22%	96.27%	99.69%
ResNet101-swp	87.46%	96.42%	99.71%

Table 5.6: Comparison of classification results on the CarFlag-563 dataset. † indicates the model is fine-tuned on large-size images (400×400 pixels).

5.5.6 Evaluation on the CarFlag-563 dataset

Compared to aforementioned datasets, the complexity of the CarFlag-563 is relatively simple since it only contains frontal car images. We decide to train proposed models on the subcategories based on the year of manufacture (1152 classes) rather than the car model. Some samples from the CarFlag-563 dataset are shown in Fig. 5.6.

Data preprocessing Car images captured by surveillance cameras are often raw images, which means that car objects are not aligned well in the center of images. There are also some noises on these images, such as parts of other vehicles. Direct using these images leads to poor performance for the fine-grained classification. To remedy this issue, we apply the faster-rcnn model (Ren et al., 2015) pre-trained on the VOC dataset to detect cars from these raw images. All detected cars are cropped with an appropriate padding area and organized as the CarFlag-563 dataset. Fig. 5.7 shows the effect of this preprocessing.

Experimental settings Since viewpoints of car images are constrained to the front view, we conjecture that 9 spatially weighted masks in the swp layer are capable to capture the discriminative regions of car objects. The number of subcategories on the CarFlag-563 is greater than the CompCars, we set the number of neurons of the FC6 and FC7 layer to 2048 for the AlexNet-swp and VGG16-swp. For ResNets-swp, we set the number of neurons of the inserted FC layer to 2048 as same as the CarFlag-1532.

Comparison of results Table 5.6 shows comparison of classification results of various DCNNs on the CarFlag-563 dataset. We surprisingly observe that the trends among the baselines are different from those on previous three datasets. The AlexNet

Figure 5.6: Sample images from the CarFlag-563 dataset captured by surveillance cameras in various weather and illumination conditions. Car types from top to bottom row: sedan, SUV, MPV, bus and truck.

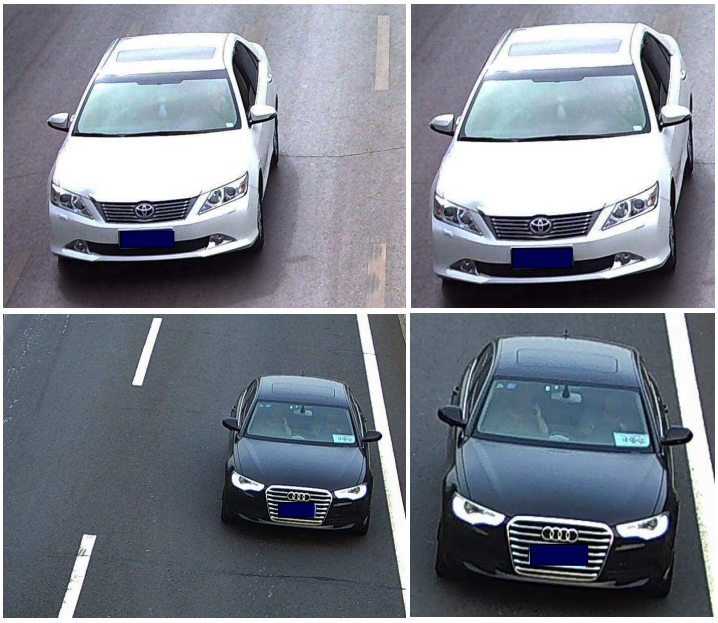

Figure 5.7: Images of left column are raw images. Right column are car images detected by the faster-rcnn model.

slightly outperforms both the ResNet50 and ResNet101, and the VGG16 achieves the best performance of 86.48% among all baselines. We conjecture that this phenomenon is caused by the GAP layer of ResNets. For cars belong to the same model, they share very similar appearance design if they are produced in adjacent years. These cars can only be distinguished by some fine-grained details, such as front lights, fog lights, and the front face. Both the ResNet50 and ResNet101 are inferior to the AlexNet because the GAP layer may discard such fine-grained discriminative information. By replacing the GAP layer with the LMP layer, both the ResNet50-LMP and ResNet101-LMP perform on par with VGG16, *e.g.*, 86.37% for Resnet50-LMP and 86.52% for ResNet101-LMP. Since the fine-grained information is significant to classify the frontal car images, we conduct one experiment to explore the effect of image resolution. We resize car images to 400×400 pixels and train the ResNet50-LMP and ResNet101-LMP on these larger images. We get some additional performance gains on both two models. The swp method further improves all baselines and the ResNet101-swp model achieves the best accuracy of 87.46%. However, the performance gains of the swp method are quite minor on this dataset especially for the AlexNet-swp and VGG16-swp. We conjecture that this was due to the simplicity of car images. Unlike other three datasets, car images of this dataset have only one pose since they are all captured from the frontal viewpoint. Fig. 5.8 shows the visualization of 9 spatially weighted masks learned from the AlexNet-swp. We can observe that most discriminative regions are located on unmeaningful regions rather than

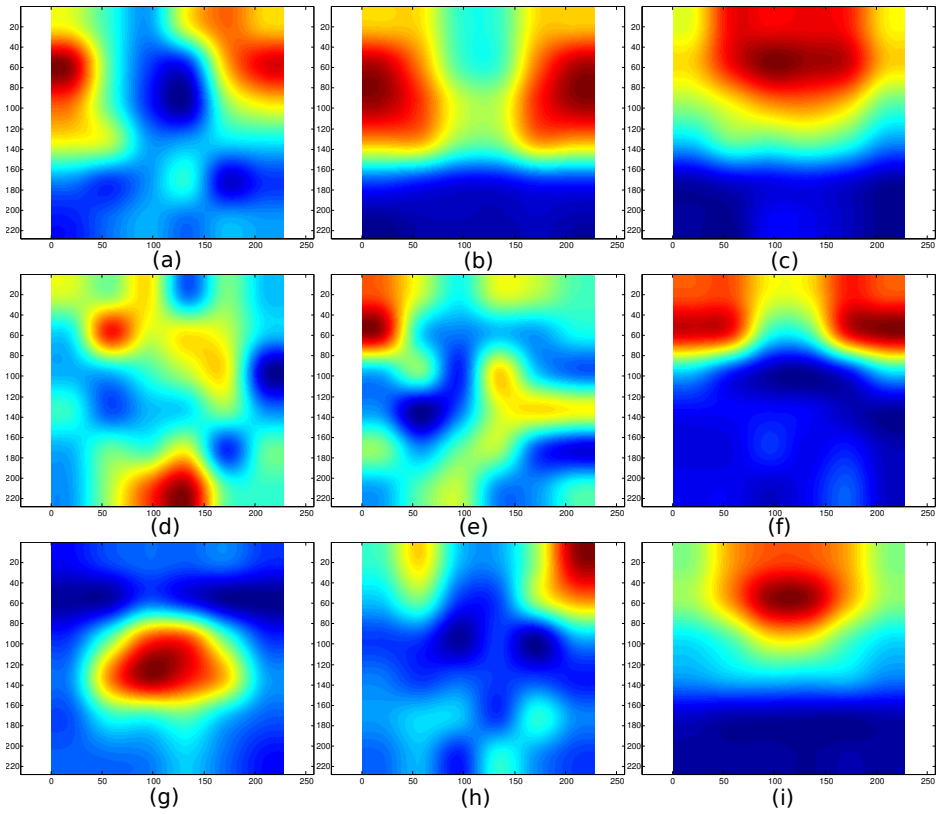

Figure 5.8: Visualization of 9 spatially weighted masks learned from the AlexNet-swp on the CarFlag-563 dataset.

fine-grained parts of the car. It means that AlexNet-swp cannot extract semantically meaningful regions well. VGG16-swp behaves similarly as the AlexNet-swp on this dataset.

We also evaluate trained models in the car model classification task (563 models) and the car make classification task (100 makes) on this dataset. The ResNet101-swp achieves the best accuracy of 96.42% for the car model classification and 99.71% for the car make classification.

5.6 Conclusion

In this work, we have proposed a spatially weighted pooling strategy, which indicates the discriminative regions of the input image used by DCNNs to classify the subcategory of the object. This proposed method has already achieved the best reported results on the public Stanford Cars-196 and CompCars datasets. We also collect two large-scale datasets to prove the robustness of the proposed method. We have presented comprehensive evaluations on the effectiveness of the swp method for the

fine-grained car classification. We show that both the local features and the spatial pooling channels can be extracted from a single DCNN and these pooling channels can be learned in the end-to-end training process of the DCNN. In our future work, we plan to further explore this idea by applying multiple swp layers appropriately into the existing DCNNs.

Chapter 6

ColorNet: A Small CNN with Color Transformation for Vehicle Color Recognition

6.1 Introduction

As an important component of the intelligent transportation system (ITS), automatic vehicle recognition has received a lot of attention in recent years. Vehicles contain a variety of information, including vehicle model, body color, license plate and so on. Vehicle color is an important property for vehicle identification and provides visual cues for many applications of ITS, such as vehicle detection, vehicle tracking, criminal detection, and law enforcement. However, vehicle color recognition is made difficult by the following three aspects:

1. Different illumination conditions and uncontrolled environments may cause a significant color variation. Color can be influenced by many factors, such as rain, snow, haze, and illumination change.
2. Certain colors are very difficult to discriminate from other colors under some specific circumstances. For example, white is not easily distinguishable from gray in dusk.
3. Vehicle color recognition is limited by the quality of input images or videos, that are affected by noise and overexposure of surveillance cameras.

In order to tackle these problems, Most previous methods focus on the design of hand-crafted color features (Huang et al., 1997; Gevers et al., 2006; Stokman and Gevers, 2007; Wang et al., 2008; Van De Sande et al., 2010; Hsieh et al., 2015). Since color images are usually represented in RGB space, RGB histogram is a natural representation for color recognition. However, the RGB space is very sensitive to illumination

change since all three channels include a representation of brightness. The normalized RGB histogram (Kender, 1976) is proposed to normalize raw RGB pixels to increase the illumination invariance. The Hue histogram (Van de Weijer et al., 2006) represents color information without illumination, thus it is partially invariant to illumination change. A framework of feature context (Chen et al., 2014) is proposed to encode color features as histograms based on the visual words in a codebook. The encoded features are more robust than original color features. Although these methods obtain reasonable performance gains, they are still far from being satisfactory in real-world applications.

Recently, convolutional neural networks (CNNs) have achieved great success in a variety of vision problems (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015; He et al., 2016). CNNs have shown the powerful ability to learn meaningful representations from input data. A CNN-based method is proposed for vehicle color recognition (Hu et al., 2015). Instead of designing hand-crafted features, this method employs a CNN as the feature extractor to extract features from input images. Then, spatial pyramid pooling is applied to these extracted features to enhance the feature robustness. Finally, a support vector machine (SVM) is trained on these pooled features to predict the color of a given vehicle. This method outperforms other conventional methods by a large margin. Although this method achieves excellent results, its high complexity and large model size can lead to practical difficulties.

In this chapter, we propose a small CNN architecture called ColorNet, which is specifically designed for vehicle color recognition. We develop the ColorNet by optimizing the architecture of AlexNet (Krizhevsky et al., 2012) which is the one of the most dominant CNNs. The major difference between ColorNet and AlexNet is that ColorNet has a structure to learn color transformation which converts input images from the RGB space to an appropriate color space. This color space is learned during training ColorNet. Moreover, since the number of parameters in ColorNet is much less than the original AlexNet, the model size of ColorNet is very small. This means that ColorNet are more feasible to deploy on mobile devices and other hardware with limited memory.

Comprehensive experiments are conducted on the vehicle-color dataset (Chen et al., 2014) which is the most popular dataset for vehicle color recognition. The experimental results show that our ColorNet performs better than all previous methods, including some sophisticated CNN-based methods (Rachmadi and Purnama, 2015; Hu et al., 2015).

6.2 Background

Many existing approaches have been proposed over the last decade to handle the image color recognition task. Most of them can be easily applied to vehicle color recognition. These approaches can be divided into two categories: color histogram-based approaches and CNN-based approaches.

Color histogram-based methods usually design a variety of color histograms to represent colors. Image pixels with similar colors are represented in the same color histogram since similar colors fall into the same bin. A normalized RGB histogram proposed by Kender (Kender, 1976) is used to increase the feature robustness to illumination change. Huang *et al.* (Huang et al., 1997) propose a correlogram for indexing images. The correlogram is a histogram of color pairs and is more robust than the RGB histogram. Qiu *et al.* (Qiu et al., 2004) investigate the redundancy and performance of different color histograms in the context of automatic colour photo categorization. Van de Weijer *et al.* (Van de Weijer et al., 2006) propose to use a Hue histogram to alleviate the effect of overexposure since it only represents information of color without illumination. Wang *et al.* (Wang et al., 2008) propose to extract color features from the HSV space for the color recognition of license plates. Van de Sande *et al.* (Van De Sande et al., 2010) investigate dominant color descriptors and analyse their invariance to illumination change and color shift. They also propose an opponent histogram which is invariant to overexposure and color shift. Hsieh *et al.* (Hsieh et al., 2015) propose a color correction algorithm to reduce lighting effects and color distortions.

CNN-based methods often employ CNNs to extract CNN features and apply a classifier to these CNN features. Rachmadi *et al.* (Rachmadi and Purnama, 2015) propose a bilinear CNN framework for vehicle color recognition. This framework consists of two CNNs which share the same fully-connected layers and the softmax layer. This framework converts input images to two color spaces, HSV and CIE Lab, and then feeds each of them to one CNN of the framework. Hu *et al.* (Hu et al., 2015) propose a CNN-based method which combines the spatial pyramid strategy with the CNN architecture. This method employs the AlexNet to extract CNN features and applies spatial pyramid pooling to these extracted features. The pooled features are than used to train a support vector machine classifier. Since spatial information are embedded into these pooled features, the recognition performance can be further improved.

6.3 Proposed approach

In this section, we propose a light-weight ColorNet to effectively and efficiently recognize the color of vehicles. To improve the accuracy, we employ a mini-CNN to learn the color transformation. To boost the speed of recognition, we optimize the architecture of AlexNet by removing non-essential structures.

6.3.1 Color transformation

Digital color images are usually represented in one color space defined by domain-specific knowledge. Most commonly used color spaces in digital photography are RGB, HSV, YCrCb, and CIE-lab. In image color recognition, visual features are extracted from image pixels represented in one color space. The choice of the color space may influence the quality of extracted features and impact the recognition performance. In the context of deep learning, the commonly used input to CNNs is RGB images. Recently, Mishkin *et al.* (Mishkin et al., 2016) employ different color spaces as the input to CNNs and analyse their performances for generic object classification. Later, Rachmadi *et al.* (Rachmadi and Purnama, 2015) investigate the effectiveness of different color spaces for vehicle color recognition. Their experimental results show that using RGB images as the input to CNNs leads to better performance than other color spaces. Here we ask a question: Can we learn a color space which performs better than the RGB space using the CNN? Our answer to this question is positive.

Inspired by image preprocessing techniques of (Mishkin et al., 2016), we propose a color transformation method that can convert image pixels from the RGB space to a learnt color space. More specifically, the color transformation is implemented by a mini-CNN which consists of two convolutional (Conv) layers and one rectified linear unit. The first Conv layer contains a predefined number of 1×1 Conv filters and the second Conv layer contains three 1×1 Conv filters. 1×1 convolution is a way to increase the non-linearity of the transformation function. Table 6.1 shows several possible architectures of the mini-CNN for color transformation. The input of the mini-CNN is RGB pixels. RGB pixels need to be centred by subtracting the per-pixel mean and scaled by multiplying a constant 0.04. The number of Conv filters in the second Conv layer is fix to 3. This setting restores the original dimensions of input images. The rectified linear unit at the end of the mini-CNN is used to change negative values to zeros and further increases the non-linearity of image representations.

Fig. 6.1 illustrates the visualization of car images in different color spaces. We choose three car images taken from the Vehicle-color dataset (Chen et al., 2014), including a black car, a red car, and a white car. As can be observed from Fig. 6.1,

Mini-CNN	Architecture
RGB	RGB
A	RGB → Conv1x1-3 → Conv1x1-3 → ReLU
B	RGB → Conv1x1-3 → Conv1x1-3 → PReLU
C	RGB → Conv1x1-10 → Conv1x1-3 → PReLU
D	RGB → Conv1x1-16 → Conv1x1-3 → PReLU

Table 6.1: Several architectures of the mini-CNN for color transformation.

(a) RGB space (b) Color space learned by B (c) Color space learned by C

Figure 6.1: Visualization of car images in several color spaces. (a) The RGB space. (b) The space learned from the mini-CNN B. (c) The space learned from the mini-CNN C.

these colors (black, red, white) in RGB space are visually changed in these learnt spaces. Specifically, black is changed to dark red, red is converted to bright pink, and white is changed to bright cyan. The effectiveness of these color spaces will be explored in the Section 6.4.3.

Layer	Architecture
Conv1	Conv11x11-96 $\rightarrow$ ReLU $\rightarrow$ LRN $\rightarrow$ Max-pooling
Conv2	Conv5x5-256 $\rightarrow$ ReLU $\rightarrow$ LRN $\rightarrow$ Max-pooling
Conv3	Conv3x3-384 $\rightarrow$ ReLU
Conv4	Conv3x3-384 $\rightarrow$ ReLU
Conv5	Conv3x3-256 $\rightarrow$ ReLU $\rightarrow$ Max-pooling
FC6	4096 neurons $\rightarrow$ Drop-out
FC7	4096 neurons $\rightarrow$ Drop-out
FC8	1000 neurons $\rightarrow$ Softmax

Table 6.2: Architecture of the AlexNet.

6.3.2 Model optimization

The mini-CNN is only used for color transformation, we still need a main CNN to implement the vehicle color recognition task. In this work, we develop our portable CNN model by optimizing the architecture of AlexNet (Krizhevsky et al., 2012). Table 6.2 shows the general layout of AlexNet. In general, AlexNet has five Conv layers and three fully-connected (FC) layers. The default input of the AlexNet is a fixed-size 227×227 RGB image. The input image is passed through a sequence of five Conv layers. These Conv layers are followed by three FC layers: the first two have 4096 neurons each, the last one contains 1000 neurons and performs 1000-way classification. The final layer is the soft-max layer. To increase the non-linearity of the model, all Conv layers and FC layers are equipped with the rectification (ReLU) layers. We use ConvX to denote the Xth Conv layer in the AlexNet. FC6, FC7, and FC8 are used to denote three FC layers, respectively.

Since AlexNet was originally pre-trained on the ImageNet dataset (Deng et al., 2009) for generic object classification, the original AlexNet is not suitable for real-time vehicle color classification. Therefore, we optimize the architecture of AlexNet by compressing its model size and simplifying its redundant structure. Optimized CNN architectures provide at least two advantages: (1) Smaller CNNs are more feasible to deploy on mobile devices and other hardware with limited memory. (2) Simpler CNNs require less computational resources in execution.

The model size of CNNs is determined by the number of parameters. In the original AlexNet, FC layers contain the majority of parameters. Hence, one way to compress the model size is to reduce the number of neurons on FC layers. However, setting the number of neurons to be too small can hurt the performance as the CNN can not converge in the training phase (under-fitting). A recently proposed global average pooling (GAP) (Lin et al., 2013) can be used to remedy this problem.

Instead of adding FC6 and FC7 layers on top of the feature maps of Conv5, a GAP layer computes the average of each feature map, and the pooled feature vector is fed directly into the last softmax layer. Compared to FC layers, the GAP layer is more native to the convolution structure by enforcing correspondences between convolutional feature maps and object categories. The GAP layer acts as a regularizer which prevents the over-fitting during training, so drop-out layers can be eliminated. The model size of AlexNet can be remarkably reduced by replacing FC6 and FC7 layers with the GAP layer.

Compared to the large-scale generic object classification (ImageNet), the vehicle color classification is relatively simple. The architecture of AlexNet may be too complex to implement this task. We can simplify AlexNet by the following three aspects: narrowing the size of input images, reducing the number of Conv filters, and decreasing the number of Conv layers. Since vehicles usually occur a large area of the car image, we conjecture that even a small image can keep enough discriminative information for color classification. We can change the resolution of input images to provide a better trade-off between the accuracy and the speed. Superfluous Conv filters or Conv layers may generate redundant features which are useless to improve the performance, but bring additionally computational burdens. We can prune some Conv filters and Conv layers to accelerate the speed. The detailed discussion on this point is provided in Section 6.4.3.

6.3.3 Model integration

The mini-CNN takes RGB pixels as input, converts them to a learnt color space, then feeds them to a main CNN which is used for vehicle color classification. Since the architecture of the mini-CNN is very simple, it can be easily embedded into many dominant CNNs (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015; He et al., 2016). For example, a mini-CNN can be embedded into AlexNet by inserting it between the input layer and the first Conv layer of AlexNet. Embedding a mini-CNN into a main CNN simplifies the training process. The mini-CNN can be trained jointly with the main CNN, parameters of the mini-CNN can be learned in the end-to-end training. The integrated CNN architecture is referred as the ColorNet.

Once we embedded a mini-CNN into a main CNN, we can investigate the visual features learned from the integrated model. Fig 6.2 shows the visualization of 96 filters of the Conv1 layer of AlexNet embedded with the mini-CNN B. We observe that most filters are visualized as various color patches. This is very different from those filters of the pre-trained AlexNet which have large response to borders and edges. This observation shows that filters of the integrated model can obtain large

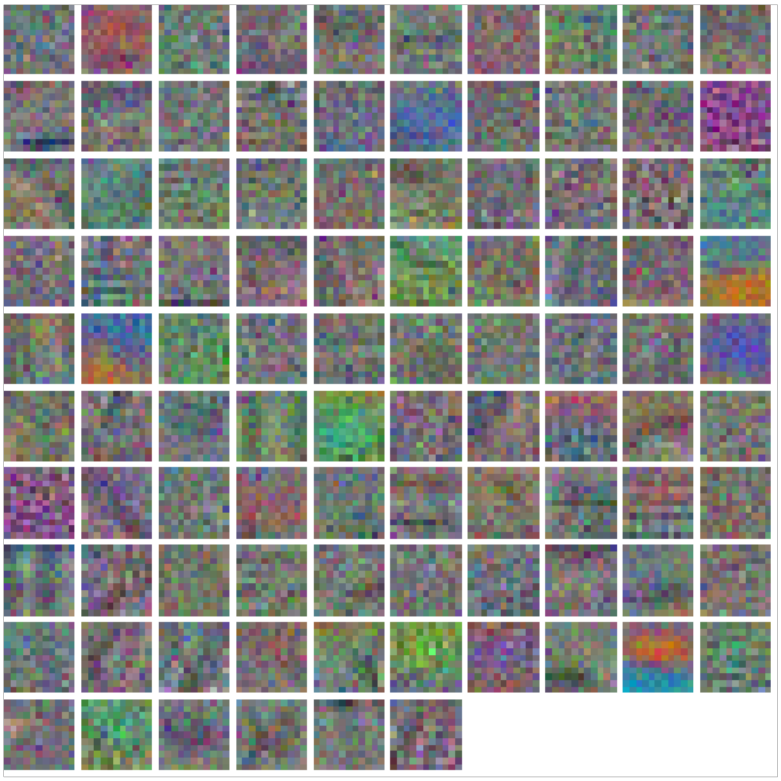

Figure 6.2: Visualization of 96 filters of the first convolutional layer of the AlexNet with mini-CNN B. These filters of size 11×3 are learned from the $227 \times 227 \times 3$ input images.

response in these colors.

6.4 Experiments

In this section, we comprehensively investigate the design space of our method and evaluate the proposed ColorNet on the Vehicle-color dataset (Chen et al., 2014).

6.4.1 Dataset and evaluation metric

The Vehicle-color dataset (Chen et al., 2014) is a vehicle image dataset which contains 15,601 vehicle images of 8 color categories. Vehicle images are captured from the frontal viewpoint under different illuminations and weather conditions. The dataset is randomly split into the training set of 7,802 images and the test set of 7,799 images. Each category is roughly split about 50-50. We evaluate the performance of various methods using the average accuracy which is computed by averaging the classification accuracy of each category. Table 6.3 shows the data distribution for each color

Category	# Images
Black	3442
Blue	1086
Cyan	281
Gray	3046
Green	482
Red	1941
White	4742
Yellow	581
Total	15601

Table 6.3: Data distribution for each color category on the Vehicle-color dataset.

category on the Vehicle-color dataset.

6.4.2 Implementation details

To adapt AlexNet to the vehicle color classification task, we trained the model on the Vehicle-color dataset. We replace the 1000-way classification layer trained on ImageNet dataset with a randomly initialized C-way soft-max layer, where C is the number of categories in the dataset. We resize the image dimension to 256×256. Then a 227×227 crop is randomly sampled from the image or its horizontal flip, with the per-pixel mean subtracted. Since the number of colors is much less than the number of categories in ImageNet dataset, we modify the original AlexNet by reducing the number of neurons on FC6 and FC7 layers from 4096 to 512. We use the stochastic gradient descent (SGD) with a mini-batch size of 128. The initial learning rate is set to 0.001 for Conv layers. The learning rate is divided by 10 after each 40 training epochs. These models are trained for up to 100 epochs. We use a weight decay of 0.0005 and a momentum of 0.9. We use the Caffe (Jia et al., 2014) framework to implement our experiments.

6.4.3 Experimental design

We further investigate the design space of the proposed method. For each experiment, we evaluate the performance of different settings on the Vehicle-color dataset. All experiments are carried out on a computer with an octa-core Intel Xeon 2.50GHz processor and a Nvidia GTX780 video card.

Architecture	Ave. Accuracy	Model Size
AlexNet ($N = 4096$)†	93.70%	227.6MB
AlexNet ($N = 512$)	94.15%	29.3MB
AlexNet-gap	94.15%	**9.6**MB

Table 6.4: Performance of the AlexNet with traditional FC layers or with GAP layer on the Vehicle-color dataset. N is the number of neurons on FC6 and FC7 layers. † is the default setting of AlexNet.

6.4.3.1 Global average pooling (GAP)

We propose to use GAP to replace traditional FC layers in AlexNet. GAP can reduce the number of parameters and alleviate the computational burdens on FC layers. Since there is no parameter in the GAP layer, over-fitting can be avoided and drop-out layers can be eliminated. The GAP sums out the spatial information, hence it is more robust to spatial translations of input images. We apply the GAP layer to AlexNet by replacing the max-pooling in the Conv5 with GAP and removing FC6 and FC7 layers. We refer to this model as the AlexNet-gap.

We compare the performance of AlexNet with traditional FC layers or with the GAP layer. The average accuracies are shown in Table 6.4. We observe that applying the GAP layer can remarkably reduce the model size of the original AlexNet from 227.6MB to 9.6MB with a reasonable performance gain. We conjecture that the GAP layer alleviates the negative effect of the over-fitting and improves the accuracy. Decreasing the number of neurons on FC6 and FC7 layers from 4096 to 512 can achieve the similar result with the AlexNet-gap. For the rest of our experiments, we employ the AlexNet-gap as it gives the best average accuracy with the smallest model size.

6.4.3.2 Size of input image

The size of input image directly affects the accuracy and speed on many visual classification tasks. Large images often bring additional information and fine-grained details. For each Conv filter, more training samples are provided by large images. On the other hand, it takes more time to process large images. We investigate the relationship between the image size and the average accuracy.

We compare three different sizes of the input image from {80x80, 128x128, 227x227}. The results are presented in Table 6.5. We observe that the average accuracy improves as we enlarge the input image. The highest accuracy is achieved by using the largest image size 227x227. However, larger images bring additionally computational burdens. The average runtime (runtime per image) also increases when we select a large image size. For the rest of our experiments, we set the size of input images to be

Input image size	Ave. Accuracy	Ave. Runtime
80x80	93.74%	**0.008**s
128x128	94.00%	0.009s
227x227	**94.15**%	0.015s

Table 6.5: Performance of the AlexNet-gap with different input image sizes on the Vehicle-color dataset.

BN Placement	Ave. Accuracy
without BN	94.00%
BN before ReLU	**94.22**%
BN after ReLU	94.13%

Table 6.6: Average accuracy of the AlexNet-gap with different BN placements on the Vehicle-color dataset.

128x128 as it gives a better trade-off between the average accuracy and the average runtime.

6.4.3.3 Batch normalization (BN)

Batch normalization addresses the gradient exploding or vanishing problem by reducing the internal covariate shift of each mini-batch. By normalizing activations throughout the network, the training process of a CNN can be accelerated by using a high learning rate. BN also acts as a regularizer to avoid the over-fitting, it can improve the generalization performance of the CNN.

To apply BN to the AlexNet-gap, we need to decide the placement of BN - before or after the non-linearity ReLU. Table 6.6 shows the results of different BN placements. We observe that it is beneficial to apply BN as it avoids the over-fitting. In both two options, BN slightly improves the average accuracy. The better accuracy is achieved by the Conv-BN-ReLU setting. For the next experiment, we apply BN to the AlexNet-gap and place BN before the ReLU layer.

6.4.3.4 Number of convolutional filters

AlexNet employs a plenty of Conv filters on each Conv layer. These Conv filters learn specific features or patterns that are presented in input images. We investigate the relationship between the number of Conv filters and the performance.

We set the number of Conv filters of the original AlexNet as the baseline (100%),

# Conv filters	Ave. Accuracy	Ave. Runtime	Model size
100%	94.22%	0.011s	9.4MB
75%	94.16%	0.010s	5.3MB
50%	93.80%	0.010s	2.4MB
25%	93.41%	0.009s	0.62MB
12.5%	92.67%	0.008s	0.17MB

Table 6.7: Performance of the AlexNet-gap with different number of convolutional filters on the Vehicle-color dataset.

# Conv layers	Ave. Accuracy	Ave. Runtime	Model size
1	85.02%	0.008s	0.15MB
2	93.02%	0.009s	1.4MB
3	93.49%	0.009s	4.9MB
4	93.93%	0.010s	7.6MB
5	94.22%	0.011s	9.4MB

Table 6.8: Performance of the AlexNet-gap with different number of convolutional layers on the Vehicle-color dataset.

we decrease the number of Conv filters on each Conv layer by multiplying the baseline with a ratio. We trained 5 different models with the number of Conv filters from {100%, 75%, 50%, 25%, 12.5%}. The results are shown in Table 6.7. We observe that the average accuracy slightly decreases as we reduce the number of Conv filters. However, using less Conv filters remarkably reduces the model size and slightly accelerates the speed of recognition. For the rest of our experiments, we set the number of Conv filters as the same to the original AlexNet since it provides the best performance.

6.4.3.5 Number of convolutional layers

AlexNet contains five Conv layers. Shallow Conv layers have high responses to edges and borders. On the contrary, deep Conv layers contain high-level feature representations which provide more discriminative information.

In Table 6.8, we compare the performance of AlexNet-gap with different number of Conv layers. We observe that the average accuracy continuously increases as we add Conv layers one by one. Adding the Conv2 significantly improve the accuracy. We conjecture that the Conv2 extracts texture information which are crucial for color classification. Adding more deeper Conv layers can further improve the accuracy, but improvements tend to be saturated. Using more Conv layers brings more parameters and computational burdens. The best result is achieved by the model with five Conv

Architecture	Ave. Accuracy
AlexNet-gap	94.22%
AlexNet-gap + Mini-CNN A	94.25%
AlexNet-gap + Mini-CNN B	94.28%
AlexNet-gap + Mini-CNN C	**94.36**%
AlexNet-gap + Mini-CNN D	94.31%

Table 6.9: Performance of the ColorNet with the different mini-CNNs for color transformation on the Vehicle-color dataset.

layers. For the next experiment, we set the number of Conv layers to five as it achieves the best result.

6.4.3.6 Color transformation

As we described in Section 6.3.1, we employ a mini-CNN to implement color transformation. The mini-CNN converts input images from the RGB space to the learnt color space and feeds them to the AlexNet-gap. The mini-CNN can be embedded into the AlexNet-gap to form the ColorNet.

We evaluate the performance of the ColorNet with different mini-CNNs from {A, B, C, D}. The performance of different models is shown in Table 6.9. By comparing mini-CNN A and mini-CNN B, we observe that the parametric rectified linear unit (PReLU) (He et al., 2015b) performs better than ReLU, since PReLU generalizes the traditional ReLU and improves accuracy at negligible extra computational cost. It can be observed that using more Conv filters in the mini-CNN can bring additional performance gains. However, setting the number of Conv filters in the mini-CNN to be too large without enlarging the size of images can hurt the performance. The best accuracy of 94.36% is achieved by using the mini-CNN C. We embed the mini-CNN C into the AlexNet-gap and refer to the integrated model as the ColorNet.

6.4.4 Comparison with state-of-the-art approaches

Table 6.10 shows the comparison of classification results on the Vehicle-color dataset. We observe that all CNN-based methods perform better than the color histogram-based methods. This shows that the effectiveness of CNN features is clearly superior than conventional hand-crafted features. The original AlexNet achieves the average accuracy of 93.29%. BilinearNet (Rachmadi and Purnama, 2015) improves the accuracy to 93.77% by employing a two stream architecture. The recently proposed squeezeNet (Iandola et al., 2017) performs on a par with the BilinearNet with a much

Method	Black	Blue	Cyan	Gray	Green	Red	White	Yellow	Ave. Accuracy
Color Correlogram (Huang et al., 1997)	80.83%	74.22%	90.78%	55.68%	65.98%	94.75%	81.70%	86.94%	78.86%
Normalized RG Hist (Gevers et al., 2006)	77.31%	82.29%	87.66%	58.96%	68.03%	96.10%	82.70%	83.86%	79.62%
Hue Hist (Van de Weijer et al., 2006)	86.54%	84.37%	90.14%	60.32%	76.30%	97.29%	85.50%	88.69%	83.64%
Transformed Color Hist (Van De Sande et al., 2010)	92.06%	84.79%	86.76%	75.70%	64.61%	95.28%	91.92%	96.34%	85.18%
Opponent Hist (Van De Sande et al., 2010)	93.12%	81.72%	95.18%	78.56%	63.50%	97.53%	92.50%	87.96%	86.26%
Combined Color Hist (Chen et al., 2014)	97.14%	93.63%	96.60%	82.18%	78.59%	98.85%	94.15%	93.95%	91.89%
AlexNet (Krizhevsky et al., 2012)	96.46%	96.69%	96.43%	86.08%	80.50%	98.97%	94.69%	96.55%	93.29%
BilinearNet (Rachmadi and Purnama, 2015)	97.38%	94.10%	96.45%	86.08%	82.57%	98.89%	**96.66**%	**97.94**%	93.77%
SqueezeNet (Iandola et al., 2017)	97.91%	96.87%	95.71%	88.90%	81.33%	98.97%	94.77%	95.86%	93.79%
SP-CNN+FC (Hu et al., 2015)	96.74%	95.94%	**99.28**%	84.04%	**84.23**%	99.07%	94.81%	97.58%	93.96%
SP-CNN+SVM (Hu et al., 2015)	97.32%	96.50%	98.57%	86.80%	83.40%	98.76%	95.90%	97.24%	94.31%
Ours(AlexNet-gap)	97.68%	**98.34**%	97.14%	88.77%	80.50%	98.97%	95.15%	97.24%	94.22%
Ours(ColorNet)	**98.02**%	97.79%	97.86%	**90.41**%	80.08%	**99.18**%	94.60%	96.90%	**94.36**%

Table 6.10: Comparison of classification results on the Vehicle-color dataset.

smaller model size. SP-CNN (Hu et al., 2015) method applies spatial pyramid pooling to CNN features to boost the performance. SP-CNN achieves the average accuracy of 93.96%. This accuracy can be further improved to 94.31% by replacing the softmax classifier with a SVM classifier.

Compared to the original AlexNet, our optimized AlexNet-gap considerably improves the accuracy from 93.29% to 94.22% with much fewer parameters. Moreover, we can get a reasonable performance gain by embedding the mini-CNN C into the AlexNet-gap. Our ColorNet achieves the best accuracy of 94.36% outperforming all previously-reported results. Table 6.11 presents a comparison between ColorNet and

Method	Ave. Accuracy	Model Size
SqueezeNet	93.77%	2.9MB
ColorNet	**94.36**%	9.4MB
ColorNet (50% #Conv filters)	93.87%	**2.4**MB

Table 6.11: Comparison between ColorNet and SqueezeNet on the Vehicle-color dataset. 50% #Conv filters indicate that only half number of Conv filters are used in each Conv layer of the ColorNet.

the existing compact model SqueezeNet on the Vehicle-color dataset. SqueezeNet is a small CNN architecture which achieves AlexNet-level accuracy on ImageNet with much fewer parameters. We observe that SqueezeNet has smaller model size than the proposed ColorNet. However, the accuracy of SqueezeNet is lower than ColorNet. Furthermore, if we halve the number of Conv filters in each Conv layer of the ColorNet, the model size can be further compressed from 9.4 MB to 2.4 MB with a slight accuracy loss.

6.5 Conclusion

In this chapter, we have proposed a mini-CNN to implement color transformation for vehicle color recognition. The mini-CNN takes RGB pixels as input, converts them to a learnt color space, then feeds them to the main CNN. We also proposed a portable ColorNet, which optimizes the architecture of original AlexNet and embeds the mini-CNN of color transformation. The ColorNet requires less computational resources, but achieves the best reported results on the Vehicle-color dataset. Since the model size of ColorNet is very small, we plan to deploy the ColorNet on mobile devices and develop an application to recognize vehicle color in real-world scenarios.

Chapter 7

Conclusion and Future Directions

7.1 Conclusion

In this thesis, we concentrated on several tasks of traffic scene understanding, such as traffic sign detection, car detection, pedestrian detection, fine-grained car recognition, etc.. The main contributions presented in this thesis are:

- In chapter 3, we propose a common detection framework to detect three important classes of objects (traffic signs, cars, cyclists) in traffic scenes. The proposed framework introduces spatially pooled features as a part of aggregated channel features to enhance the feature robustness and employs detectors of these three classes to detect multiple objects. To tackle objects with a large intra-class variation, we propose an object subcategorization method to improve the generalization performance by capturing the intra-class variation of objects.

- In chapter 4, we propose a simple-yet-powerful pedestrian detector, which reuses inner layers of convolutional features extracted by a properly fine-tuned VGG16 model. We show that by re-using the convolutional feature maps of the fine-tuned VGG16 model as visual features to train an ensemble of boosted decision forests, the resulting ensemble model outperforms all competing CNN-based methods. We also show that a pixel labelling model can be used to improve performance by simply incorporating the labelling scores with the detection scores of a standard pedestrian detector.

- In chapter 5, we propose a spatially weighted pooling (swp) strategy which considerably enhances the discriminative power of CNN representations for fine-grained car classification. The swp pools the extracted features of deep CNNs with the guidance of its learnt masks, which measures the importance of the spatial units in terms of discriminative power. We also collect two challenging fine-grained car datasets: CarFlag-563 and CarFlag-1532. These two

datasets are more challenging than existing datasets due to the large number of car objects and the rich diversity of car models.

- In chapter 6, we propose a color transformation method that converts image pixels from the RGB space to a learnt space for improving the recognition performance. We also propose a small CNN architecture called ColorNet, which optimizes the architecture of AlexNet and embeds a mini-CNN of color transformation for vehicle color recognition.

7.2 Future work

Even though the work in this thesis has made considerable progress in several tasks of traffic scene understanding, several important issues remain unresolved. We point out future directions for these tasks.

- In chapter 3 and chapter 4 we addressed several detection tasks with a common detection framework and a CNN-based pedestrian detector. Since these tasks are still implemented by two independent methods, one interesting research direction is to integrate the pedestrian detector into the common detection framework. This idea can be achieved by generalizing feature representations of these two methods. Many powerful CNN-based detection frameworks have been proposed (Ren et al., 2015; Redmon et al., 2016; Liu et al., 2016; Li et al., 2016a) in recent years for generic object detection. These frameworks achieve the excellent performance in detection accuracy and speed. We plan to optimize one of these state-of-the-art frameworks to implement detection tasks of traffic scene understanding. It would be interesting to use a single CNN-based framework to detect every objects of interest in traffic scenes.

- In chapter 5 and 6 we addressed the recognition task for vehicle model and vehicle color. Since vehicles contain a variety of properties, including brand, model, year of manufacture, color, license plate, etc., it is not very efficient to design specific methods to recognize each of these properties. We are interested in designing a portable recognizer which can simultaneously recognizes all kinds of information of a given vehicle. Moreover, we plan to deploy this recognizer on mobile devices or FPGAs for practical applications.

Bibliography

Agarwal, S.; Awan, A.; and Roth, D., 2004. Learning to detect objects in images via a sparse, part-based representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26, 11 (2004), 1475–1490. (cited on page 39)

Ahonen, T.; Hadid, A.; and Pietikäinen, M., 2004. Face recognition with local binary patterns. In *Proc. Eur. Conf. Comp. Vis.*, 469–481. Springer. (cited on pages 1, 9, 21, 23, 25, 31, and 32)

Azizpour, H.; Razavian, A. S.; Sullivan, J.; Maki, A.; and Carlsson, S., 2016. Factors of transferability for a generic convnet representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38, 9 (2016), 1790–1802. (cited on page 2)

Behley, J.; Steinhage, V.; and Cremers, A. B., 2013. Laser-based segment classification using a mixture of bag-of-words. In *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 4195–4200. IEEE. (cited on pages 44 and 46)

Branson, S.; Van Horn, G.; Belongie, S.; and Perona, P., 2014. Bird species categorization using pose normalized deep convolutional nets. In *Proc. British Mach. Vis. Conf.* (cited on page 49)

Broggi, A.; Cappalunga, A.; Cattani, S.; and Zani, P., 2008. Lateral vehicles detection using monocular high resolution cameras on terramax. In *Proc. IEEE Intell. Vehicles Symp.*, 1143–1148. IEEE. (cited on page 26)

Cai, Z.; Saberian, M.; and Vasconcelos, N., 2015. Learning complexity-aware cascades for deep pedestrian detection. In *Proc. IEEE Int. Conf. Comp. Vis.*, 3361–3369. (cited on pages xx, 3, 49, 50, 59, 61, 68, 69, and 72)

Chai, Y.; Lempitsky, V.; and Zisserman, A., 2013. Symbiotic segmentation and part localization for fine-grained categorization. In *Proc. IEEE Int. Conf. Comp. Vis.*, 321–328. (cited on page 89)

Chen, L. C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; and Yuille, A. L., 2015a. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *Proc. Int. Conf. Learning Representations*. (cited on pages 59 and 60)

Chen, P.; Bai, X.; and Liu, W., 2014. Vehicle color recognition on urban road by feature context. *IEEE Trans. Intell. Transportation Syst.*, 15, 5 (2014), 2340–2346. (cited on pages 100, 102, 106, and 112)

Chen, X.; Kundu, K.; Zhu, Y.; Berneshawi, A. G.; Ma, H.; Fidler, S.; and Urtasun, R., 2015b. 3d object proposals for accurate object class detection. In *Proc. Adv. Neural Inf. Process. Syst.*, 424–432. (cited on pages 71 and 72)

Cimpoi, M.; Maji, S.; and Vedaldi, A., 2015. Deep filter banks for texture recognition and segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 3828–3836. (cited on page 76)

Coates, A. and Ng, A. Y., 2011. The importance of encoding versus training with sparse coding and vector quantization. In *Proc. Int. Conf. Mach. Learn.*, 921–928. (cited on page 32)

Cordts, M.; Omran, M.; Ramos, S.; Scharwächter, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; and Schiele, B., 2015. The cityscapes dataset. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn. Workshops*. (cited on page 59)

Cui, J.; Liu, F.; Li, Z.; and Jia, Z., 2010. Vehicle localisation using a single camera. In *Proc. IEEE Intell. Vehicles Symp.*, 871–876. IEEE. (cited on page 26)

Dalal, N. and Triggs, B., 2005. Histograms of oriented gradients for human detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 886–893. IEEE. (cited on pages 1, 9, 10, 21, 22, 23, 25, 28, 37, 49, and 53)

Dalal, N.; Triggs, B.; and Schmid, C., 2006. Human detection using oriented histograms of flow and appearance. In *Proc. Eur. Conf. Comp. Vis.*, 428–441. Springer. (cited on page 9)

de la Escalera, A.; Armingol, J. M.; and Mata, M., 2003. Traffic sign recognition and analysis for intelligent vehicles. *Image Vis. Comput.*, 21, 3 (2003), 247–258. (cited on pages 21 and 24)

De La Escalera, A.; Moreno, L. E.; Salichs, M. A.; and Armingol, J. M., 1997. Road traffic sign detection and classification. *IEEE Trans. Industrial Electronics*, 44, 6 (1997), 848–859. (cited on pages 21 and 24)

Demiriz, A.; Bennett, K. P.; and Shawe-Taylor, J., 2002. Linear programming boosting via column generation. *Mach. Learn.*, 46, 1-3 (2002), 225–254. (cited on page 53)

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 248–255. (cited on pages 2, 19, 49, 76, and 104)

Divvala, S. K.; Efros, A. A.; and Hebert, M., 2012. How important are "deformable parts" in the deformable parts model? In *Proc. Eur. Conf. Comp. Vis. Workshop*, 31–40. Springer. (cited on pages 22, 23, and 25)

Dollár, P.; Appel, R.; Belongie, S.; and Perona, P., 2014. Fast feature pyramids for object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36, 8 (2014), 1532–1545. (cited on pages xix, 49, 56, 61, and 67)

Dollár, P.; Appel, R.; Belongie, S.; and Perona, P., 2014. Fast feature pyramids for object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36, 8 (2014), 1532–1545. (cited on pages 9, 11, 22, 24, 25, 28, and 30)

Dollár, P.; Belongie, S.; and Perona, P., 2010. The fastest pedestrian detector in the west. In *Proc. British Mach. Vis. Conf.*, 1–11. (cited on pages 1, 3, and 37)

Dollár, P.; Tu, Z.; Perona, P.; and Belongie, S., 2009. Integral channel features. In *Proc. British Mach. Vis. Conf.*, 1–11. (cited on pages 23, 28, 30, and 31)

Dollar, P.; Wojek, C.; Schiele, B.; and Perona, P., 2012. Pedestrian detection: An evaluation of the state of the art. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34, 4 (2012), 743–761. (cited on page 52)

Donahue, J.; Jia, Y.; Vinyals, O.; Hoffman, J.; Zhang, N.; Tzeng, E.; and Darrell, T., 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *Proc. Int. Conf. Mach. Learn.*, vol. 32, 647–655. (cited on page 2)

Enzweiler, M.; Eigenstetter, A.; Schiele, B.; and Gavrila, D. M., 2010. Multi-cue pedestrian classification with partial occlusion handling. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 990–997. (cited on page 50)

Everingham, M.; Gool, L. J. V.; Williams, C. K. I.; Winn, J. M.; and Zisserman, A., 2010. The pascal visual object classes (VOC) challenge. *Int. J. Comp. Vis.*, 88, 2 (2010), 303–338. (cited on pages 35, 37, and 41)

Fang, C.; Chen, S.; and Fuh, C., 2003. Road-sign detection and tracking. *IEEE Trans. Vehicular Technol.*, 52, 5 (2003), 1329–1341. (cited on page 24)

Felzenszwalb, P. F.; Girshick, R. B.; McAllester, D. A.; and Ramanan, D., 2010. Object detection with discriminatively trained part-based models. *IEEE Trans. Pat-*

tern Anal. Mach. Intell., 32, 9 (2010), 1627–1645. (cited on pages 22, 23, 25, 43, 44, 46, 50, and 75)

Fidler, S.; Mottaghi, R.; Urtasun, R.; et al., 2013. Bottom-up segmentation for top-down detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 3294–3301. (cited on page 51)

Freund, Y. and Schapire, R. E., 1999. A short introduction to boosting. (1999), 1401–1406. (cited on pages 11 and 53)

Friedman, J.; Hastie, T.; and Tibshirani, R., 2000. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *J. Annals Stat.*, 28, 2 (2000), 337–407. (cited on page 14)

Gao, X. W.; Podladchikova, L.; Shaposhnikov, D.; Hong, K.; and Shevtsova, N., 2006. Recognition of traffic signs based on their colour and shape features extracted using human vision models. *J. Visual Comm. Image Repr.*, 17, 4 (2006), 675–685. (cited on page 25)

Geiger, A.; Lenz, P.; Stiller, C.; and Urtasun, R., 2013. Vision meets robotics: The KITTI dataset. *Int. J. Robotic Res.*, 32, 11 (2013), 1231–1237. (cited on pages 26 and 40)

Geiger, A.; Lenz, P.; and Urtasun, R., 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 3354–3361. (cited on page 53)

Geiger, A.; Wojek, C.; and Urtasun, R., 2011. Joint 3d estimation of objects and scene layout. In *Proc. Adv. Neural Inf. Process. Syst.*, 1467–1475. (cited on pages 22, 23, and 25)

Getreuer, P., 2012. Automatic color enhancement (ace) and its fast implementation. *Image Process. Line*, 2 (2012), 266–277. (cited on page 37)

Gevers, T.; Van De Weijer, J.; and Stokman, H., 2006. Color feature detection. (cited on pages 99 and 112)

Girshick, R., 2015. Fast R-CNN. In *Proc. IEEE Int. Conf. Comp. Vis.*, 1440–1448. (cited on pages 3, 19, and 49)

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J., 2014a. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 580–587. (cited on pages 3, 19, 49, and 78)

Girshick, R.; Iandola, F.; Darrell, T.; and Malik, J., 2015. Deformable part models are convolutional neural networks. *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, (2015), 437–446. (cited on page 50)

Girshick, R. B.; Donahue, J.; Darrell, T.; and Malik, J., 2014b. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 580–587. (cited on pages 2 and 23)

Gómez-Moreno, H.; Maldonado-Bascón, S.; Gil-Jiménez, P.; and Lafuente-Arroyo, S., 2010. Goal evaluation of segmentation algorithms for traffic sign recognition. *IEEE Trans. Intell. Transportation Syst.*, 11, 4 (2010), 917–930. (cited on page 24)

González, A.; Villalonga, G.; Xu, J.; Vázquez, D.; Amores, J.; and López, A. M., 2015. Multiview random forest of local experts combining rgb and lidar data for pedestrian detection. In *Proc. IEEE Intell. Vehicles Symp.*, 356–361. IEEE. (cited on page 46)

Gosselin, P.-H.; Murray, N.; Jégou, H.; and Perronnin, F., 2014. Revisiting the fisher vector for fine-grained classification. *Patt. Recogn. Lett.*, 49 (2014), 92–98. (cited on page 89)

Gu, H.-Z. and Lee, S.-Y., 2013. Car model recognition by utilizing symmetric property to overcome severe pose variation. *Mach. vis. app.*, 24, 2 (2013), 255–274. (cited on page 78)

Hariharan, B.; Arbeláez, P.; Girshick, R.; and Malik, J., 2014. Simultaneous detection and segmentation. In *Proc. Eur. Conf. Comp. Vis.*, 297–312. (cited on pages 49 and 51)

Hariharan, B.; Arbeláez, P.; Girshick, R.; and Malik, J., 2015. Hypercolumns for object segmentation and fine-grained localization. *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, (2015), 447–456. (cited on pages 51 and 59)

Hastie, T.; Tibshirani, R.; Friedman, J.; and Franklin, J., 2005. The elements of statistical learning: data mining, inference and prediction. *J. Math. Intelligencer*, 27, 2 (2005), 83–85. (cited on pages 13 and 54)

He, H.; Shao, Z.; and Tan, J., 2015a. Recognition of car makes and models from a single traffic-camera image. *IEEE Trans. Intell. Transportation Syst.*, 16, 6 (2015), 3182–3192. (cited on page 77)

He, K.; Zhang, X.; Ren, S.; and Sun, J., 2015b. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. IEEE Int. Conf. Comp. Vis.*, 1026–1034. (cited on pages 17 and 111)

He, K.; Zhang, X.; Ren, S.; and Sun, J., 2016. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 770–778. (cited on pages 2, 17, 19, 76, 77, 83, 85, 87, 88, 100, and 105)

Hejrati, M. and Ramanan, D., 2012. Analyzing 3d objects in cluttered images. In *Proc. Adv. Neural Inf. Process. Syst.*, 602–610. (cited on pages 22 and 25)

Hosang, J.; Omran, M.; Benenson, R.; and Schiele, B., 2015. Taking a deeper look at pedestrians. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 4073–4082. (cited on pages 49, 52, 69, and 72)

Houben, S., 2011. A single target voting scheme for traffic sign detection. In *Proc. IEEE Intell. Vehicles Symp.*, 124–129. IEEE. (cited on page 24)

Houben, S.; Stallkamp, J.; Salmen, J.; Schlipsing, M.; and Igel, C., 2013. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *Proc. Int. Joint Conf. Neural Net.*, 1–8. IEEE. (cited on pages 25, 36, and 39)

Hsiao, E.; Sinha, S. N.; Ramnath, K.; Baker, S.; Zitnick, L.; and Szeliski, R., 2014. Car make and model recognition using 3d curve alignment. In *Proc. IEEE Win. Conf. App. Comp. Vis.*, 1–1. IEEE. (cited on page 78)

Hsieh, J.-W.; Chen, L.-C.; Chen, S.-Y.; Chen, D.-Y.; Alghyaline, S.; and Chiang, H.-F., 2015. Vehicle color classification under different lighting conditions through color correction. *IEEE Sensors Journal*, 15, 2 (2015), 971–983. (cited on pages 99 and 101)

Hu, C.; Bai, X.; Qi, L.; Chen, P.; Xue, G.; and Mei, L., 2015. Vehicle color recognition with spatial pyramid deep learning. *IEEE Trans. Intell. Transportation Syst.*, 16, 5 (2015), 2925–2934. (cited on pages 100, 101, and 112)

Hu, Q.; Paisitkriangkrai, S.; Shen, C.; van den Hengel, A.; and Porikli, F., 2016. Fast detection of multiple objects in traffic scenes with a common detection framework. *IEEE Transactions on Intelligent Transportation Systems*, 17, 4 (2016), 1002–1014. (cited on page 75)

Huang, J.; Kumar, S. R.; Mitra, M.; Zhu, W.-J.; and Zabih, R., 1997. Image indexing using color correlograms. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 762–768. IEEE. (cited on pages 99, 101, and 112)

Iandola, F. N.; Han, S.; Moskewicz, M. W.; Ashraf, K.; Dally, W. J.; and Keutzer, K., 2017. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, (2017). (cited on pages 17, 111, and 112)

Ioffe, S. and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, (2015). (cited on pages 17 and 18)

Jain, A. K.; Murty, M. N.; and Flynn, P. J., 1999. Data clustering: a review. *ACM computing surveys (CSUR)*, 31, 3 (1999), 264–323. (cited on page 30)

Janssen, R.; Ritter, W.; Stein, F.; and Ott, S., 1993. Hybrid approach for traffic sign recognition. In *Proc. IEEE Intell. Vehicles Symp.*, 390–395. (cited on page 24)

Jégou, H.; Douze, M.; Schmid, C.; and Pérez, P., 2010. Aggregating local descriptors into a compact image representation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 3304–3311. IEEE. (cited on pages 4 and 76)

Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; and Darrell, T., 2014. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, (2014). (cited on page 107)

Ke, Y. and Sukthankar, R., 2004. Pca-sift: A more distinctive representation for local image descriptors. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, vol. 2, 506–513. IEEE. (cited on page 9)

Kender, J. R., 1976. Saturation, heu, and normalized color: Calculation, digitization effects, and use. Technical report, DTIC Document. (cited on pages 5, 100, and 101)

Krähenbühl, P. and Koltun, V., 2011. Efficient inference in fully connected CRFs with Gaussian edge potentials. In *Proc. Adv. Neural Inf. Process. Syst.* (cited on page 59)

Krause, J.; Jin, H.; Yang, J.; and Fei-Fei, L., 2015. Fine-grained recognition without part annotations. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 5546–5555. (cited on pages 75, 78, and 89)

Krause, J.; Stark, M.; Deng, J.; and Fei-Fei, L., 2013. 3d object representations for fine-grained categorization. In *Proc. IEEE Int. Conf. Comp. Vis. Workshops*, 554–561. (cited on pages 75, 77, 78, 79, 80, 85, and 87)

Krizhevsky, A. and Hinton, G., 2009. Learning multiple layers of features from tiny images. Technical report, University of Toronto. (cited on page 49)

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E., 2012. Imagenet classification with deep convolutional neural networks. In *Proc. Adv. Neural Inf. Process. Syst.*, 1097–1105. (cited on pages 2, 5, 17, 19, 49, 76, 77, 83, 85, 87, 100, 104, 105, and 112)

Kuo, C. and Nevatia, R., 2009. Robust multi-view car detection using unsupervised sub-categorization. In *Proc. App. Comp. Vis. Workshop*, 1–8. IEEE. (cited on pages 22 and 26)

Kuo, W. and Lin, C., 2007. Two-stage road sign detection and recognition. In *Proc. IEEE Int. Conf. Multimedia Expo.*, 1427–1430. IEEE. (cited on pages 21 and 24)

Kyo, S.; Koga, T.; Sakurai, K.; and Okazaki, S., 1999. A robust vehicle detecting and tracking system for wet weather conditions using the imap-vision image processing board. In *Proc. IEEE Int. Conf. Intell. Transportation Syst.*, 423–428. IEEE. (cited on page 26)

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P., 1998. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86, 11 (1998), 2278–2324. (cited on page 15)

Li, B.; Wu, T.; and Zhu, S., 2014a. Integrating context and occlusion for car detection by hierarchical and-or model. In *Proc. Eur. Conf. Comp. Vis.*, 652–667. Springer. (cited on page 44)

Li, L.-J.; Su, H.; Lim, Y.; and Fei-Fei, L., 2014b. Object bank: An object-level image representation for high-level visual recognition. *Int. J. Comput. Vision*, 107, 1 (2014), 20–39. (cited on page 61)

Li, Y.; He, K.; Sun, J.; et al., 2016a. R-fcn: Object detection via region-based fully convolutional networks. In *Proc. Adv. Neural Inf. Process. Syst.*, 379–387. (cited on page 116)

Li, Y.; Liu, L.; Shen, C.; and Hengel, A. v. d., 2016b. Mining mid-level visual patterns with deep cnn activations. *Int. J. Comp. Vis.*, (2016), 1–21. (cited on page 81)

Li, Y.; Liu, L.; Shen, C.; and van den Hengel, A., 2015. Mid-level deep pattern mining. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 971–980. IEEE. (cited on page 81)

Liang, M.; Yuan, M.; Hu, X.; Li, J.; and Liu, H., 2013. Traffic sign detection by ROI extraction and histogram features-basedrecognition. In *Proc. Int. Joint Conf. Neural Net.*, 1–8. IEEE. (cited on pages 25 and 39)

Liao, L.; Hu, R.; Xiao, J.; Wang, Q.; Xiao, J.; and Chen, J., 2015. Exploiting effects of parts in fine-grained categorization of vehicles. In *Proc. IEEE Int. Conf. Image Proc.*, 745–749. IEEE. (cited on page 77)

Lin, G.; Shen, C.; Reid, I.; et al., 2015a. Efficient piecewise training of deep structured models for semantic segmentation. *arXiv:1504.01013*, (2015). (cited on pages 59 and 60)

Lin, H.-T.; Lin, C.-J.; and Weng, R. C., 2007. A note on platt's probabilistic outputs for support vector machines. *Mach. Learn.*, 68, 3 (2007), 267–276. (cited on page 34)

Lin, L.; Wang, X.; Yang, W.; and Lai, J.-H., 2015b. Discriminatively trained and-or graph models for object shape detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37, 5 (2015), 959–972. (cited on page 50)

Lin, M.; Chen, Q.; and Yan, S., 2013. Network in network. *arXiv preprint arXiv:1312.4400*, (2013). (cited on page 104)

Lin, T.-Y.; RoyChowdhury, A.; and Maji, S., 2015c. Bilinear cnn models for fine-grained visual recognition. In *Proc. IEEE Int. Conf. Comp. Vis.*, 1449–1457. (cited on pages 2, 19, 75, 76, 78, and 89)

Lin, Y.-L.; Morariu, V. I.; Hsu, W.; and Davis, L. S., 2014. Jointly optimizing 3d model fitting and fine-grained classification. In *Proc. Eur. Conf. Comp. Vis.*, 466–480. Springer. (cited on page 78)

Liu, L.; Shen, C.; and van den Hengel, A., 2015. The treasure beneath convolutional layers: Cross-convolutional-layer pooling for image classification. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 4749–4757. (cited on pages 2, 4, 19, 76, 78, and 81)

Liu, L.; Shen, C.; Wang, L.; van den Hengel, A.; and Wang, C., 2014. Encoding high dimensional local features by sparse coding based fisher vectors. In *Proc. Adv. Neural Inf. Process. Syst.*, 1143–1151. (cited on page 81)

Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; and Berg, A. C., 2016. Ssd: Single shot multibox detector. In *Proc. Eur. Conf. Comp. Vis.*, 21–37. Springer. (cited on pages 19 and 116)

Long, C.; Wang, X.; Hua, G.; Yang, M.; and Lin, Y., 2014. Accurate object detection with location relaxation and regionlets re-localization. In *Proc. Asian Conf. Comp. Vis.*, 3000–3016. IEEE. (cited on pages 43 and 46)

Long, J.; Shelhamer, E.; and Darrell, T., 2015. Fully convolutional networks for semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 3431–3440. (cited on pages 51 and 59)

Lowe, D. G., 1999. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2, 1150–1157. Ieee. (cited on pages 1, 4, and 76)

Lowe, D. G., 2004. Distinctive image features from scale-invariant keypoints. *Int. J. Comp. Vis.*, 60, 2 (2004), 91–110. (cited on page 9)

Loy, G. B. and Barnes, N. M., 2004. Fast shape-based road sign detection for a driver assistance system. In *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 70–75. IEEE. (cited on page 24)

Luo, P.; Tian, Y.; Wang, X.; and Tang, X., 2014. Switchable deep network for pedestrian detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 899–906. (cited on page 50)

Maldonado-Bascón, S.; Lafuente-Arroyo, S.; Gil-Jiménez, P.; Gómez-Moreno, H.; and López-Ferreras, F., 2007. Road-sign detection and recognition based on support vector machines. *IEEE Trans. Intell. Transportation Syst.*, 8, 2 (2007), 264–278. (cited on page 24)

Martinez, E.; Diaz, M.; Melenchon, J.; Montero, J.; Iriondo, I.; and Socoro, J., 2008. Driving assistance system based on the detection of head-on collisions. In *Proc. IEEE Intell. Vehicles Symp.*, 913–918. IEEE. (cited on page 26)

Masnadi-Shirazi, H. and Vasconcelos, N., 2011. Cost-sensitive boosting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33, 2 (2011), 294–309. (cited on page 40)

Mathias, M.; Benenson, R.; Timofte, R.; and Van Gool, L., 2013a. Handling occlusions with franken-classifiers. In *Proc. IEEE Int. Conf. Comp. Vis.*, 1505–1512. (cited on page 50)

Mathias, M.; Timofte, R.; Benenson, R.; and Gool, L. J. V., 2013b. Traffic sign recognition - how far are we from the solution? In *Proc. Int. Joint Conf. Neural Net.*, 1–8. IEEE. (cited on pages 21, 24, 25, and 39)

Mishkin, D.; Sergievskiy, N.; and Matas, J., 2016. Systematic evaluation of cnn advances on the imagenet. *arXiv preprint arXiv:1606.02228*, (2016). (cited on page 102)

Mogelmose, A.; Trivedi, M. M.; and Moeslund, T. B., 2012. Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey. *IEEE Trans. Intell. Transportation Syst.*, 13, 4 (2012), 1484–1497. (cited on page 24)

Nam, W.; Dollár, P.; and Han, J. H., 2014. Local decorrelation for improved pedestrian detection. In *Proc. Adv. Neural Inf. Process. Syst.*, 424–432. (cited on pages 9 and 69)

Ng, A. Y.; Jordan, M. I.; Weiss, Y.; et al., 2002. On spectral clustering: Analysis and an algorithm. *Proc. Adv. Neural Inf. Process. Syst.*, 2 (2002), 849–856. (cited on page 30)

Ohn-Bar, E. and Trivedi, M. M., 2014. Fast and robust object detection using visual subcategories. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn. Workshop*, 179–184. (cited on pages 22, 24, 26, 43, and 44)

Ohn-Bar, E. and Trivedi, M. M., 2015. Learning to detect vehicles by clustering appearance patterns. *IEEE Trans. Intell. Transportation Syst.*, 16, 5 (2015), 2511–2521. (cited on page 29)

Ojala, T.; Pietikäinen, M.; and Harwood, D., 1996. A comparative study of texture measures with classification based on featured distributions. *Patt. Recogn.*, 29, 1 (1996), 51–59. (cited on page 10)

Ojala, T.; Pietikainen, M.; and Maenpaa, T., 2002. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24, 7 (2002), 971–987. (cited on page 10)

Ouyang, W. and Wang, X., 2012. A discriminative deep model for pedestrian detection with occlusion handling. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 3258–3265. (cited on page 50)

Ouyang, W. and Wang, X., 2013a. Joint deep learning for pedestrian detection. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2056–2063. (cited on page 50)

Ouyang, W. and Wang, X., 2013b. Single-pedestrian detection aided by multi-pedestrian detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 3198–3205. (cited on page 50)

Paisitkriangkrai, S.; Shen, C.; and Hengel, A. v. d., 2016. Pedestrian detection with spatially pooled features and structured ensemble learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38, 6 (2016), 1243–1257. (cited on pages 49, 53, 63, 69, and 72)

Paisitkriangkrai, S.; Shen, C.; and van den Hengel, A., 2014a. Asymmetric pruning for learning cascade detectors. *IEEE Trans. Multimedia*, 16, 5 (2014), 1254–1267. (cited on page 40)

Paisitkriangkrai, S.; Shen, C.; and van den Hengel, A., 2014b. Strengthening the effectiveness of pedestrian detection with spatially pooled features. In *Proc. Eur. Conf. Comp. Vis.*, 546–561. Springer. (cited on pages 2, 9, 23, 24, 28, 30, 31, 32, 33, 56, and 69)

Pepik, B.; Stark, M.; Gehler, P.; and Schiele, B., 2015. Multi-view and 3d deformable part models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37, 11 (2015), 2232–2245. (cited on page 46)

Pepik, B.; Stark, M.; Gehler, P. V.; and Schiele, B., 2013. Occlusion patterns for object class detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 3286–3293. IEEE. (cited on pages 22, 25, 43, and 44)

Perronnin, F.; Sánchez, J.; and Mensink, T., 2010. Improving the fisher kernel for large-scale image classification. In *Proc. Eur. Conf. Comp. Vis.*, 143–156. Springer. (cited on pages 4 and 76)

Pettersson, N.; Petersson, L.; and Andersson, L., 2008. The histogram feature-a resource-efficient weak classifier. In *Proc. IEEE Intell. Vehicles Symp.*, 678–683. IEEE. (cited on page 25)

Platt, J. C., 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Proc. Adv. Large Margin Classifiers*, 61–74. Citeseer. (cited on page 34)

Prisacariu, V. A.; Timofte, R.; Zimmermann, K.; Reid, I.; and Gool, L. J. V., 2010. Integrating object detection with 3d tracking towards a better driver assistance system. In *Proc. Int. conf. Patt. Recogn.*, 3344–3347. IEEE. (cited on page 25)

Qiu, G.; Feng, X.; and Fang, J., 2004. Compressing histogram representations for automatic colour photo categorization. *Patt. Recogn.*, 37, 11 (2004), 2177–2193. (cited on page 101)

Qui, Z.; Yao, D.; Zhang, Y.; Ma, D.; and Liu, X., 2003. The study of the detection of pedestrian and bicycle using image processing. *IEEE Trans. Intell. Transportation Syst.*, 1 (2003), 340–345. (cited on page 26)

Rachmadi, R. F. and Purnama, I., 2015. Vehicle color recognition using convolutional neural network. *arXiv preprint arXiv:1510.07391*, (2015). (cited on pages 100, 101, 102, 111, and 112)

Redmon, J.; Divvala, S.; Girshick, R.; and Farhadi, A., 2016. You only look once: Unified, real-time object detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 779–788. (cited on pages 19 and 116)

Ren, S.; He, K.; Girshick, R.; and Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proc. Adv. Neural Inf. Process. Syst.*, 91–99. (cited on pages 19, 93, and 116)

Ren, S.; He, K.; Girshick, R.; Zhang, X.; and Sun, J., 2016. Object detection networks on convolutional feature maps. *IEEE Trans. Pattern Anal. Mach. Intell.*, (2016). (cited on page 51)

Rogers, S. and Papanikolopoulos, N. P., 2000. Counting bicycles using computer vision. In *Proc. IEEE Conf. Intell. Transportation Syst.*, 33–38. IEEE. (cited on page 26)

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L., 2015. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comp. Vis.*, 115, 3 (2015), 211–252. doi:10.1007/s11263-015-0816-y. (cited on page 19)

Sermanet, P. and LeCun, Y., 2011. Traffic sign recognition with multi-scale convolutional networks. In *Proc. Int. Joint Conf. Neural Net.*, 2809–2813. IEEE. (cited on page 25)

Shechtman, E. and Irani, M., 2007. Matching local self-similarities across images and videos. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 1–8. IEEE. (cited on page 49)

Simonyan, K. and Zisserman, A., 2014. Two-stream convolutional networks for action recognition in videos. In *Proc. Adv. Neural Inf. Process. Syst.*, 568–576. (cited on page 49)

Simonyan, K. and Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. In *Proc. Int. Conf. Learning Representations*. (cited on pages 2, 3, 17, 19, 49, 50, 51, 54, 76, 77, 83, 85, 100, and 105)

Sivaraman, S. and Trivedi, M. M., 2013. Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis. *IEEE Trans. Intell. Transportation Syst.*, 14, 4 (2013), 1773–1795. (cited on pages 21 and 25)

Sochor, J.; Herout, A.; and Havel, J., 2016. Boxcars: 3d boxes as cnn input for improved fine-grained vehicle recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 3006–3015. (cited on pages xxi, 78, and 90)

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15, 1 (2014), 1929–1958. (cited on page 18)

Stokman, H. and Gevers, T., 2007. Selection and fusion of color models for image feature detection. *IEEE transactions on pattern analysis and machine intelligence*, 29, 3 (2007). (cited on page 99)

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A., 2015. Going deeper with convolutions. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 1–9. (cited on pages 3, 17, and 49)

Tang, S.; Andriluka, M.; and Schiele, B., 2014. Detection and tracking of occluded people. *Int. J. Comp. Vis.*, 110, 1 (2014), 58–69. (cited on page 50)

Tian, Y.; Luo, P.; Wang, X.; and Tang, X., 2015. Deep learning strong parts for pedestrian detection. In *Proc. IEEE Int. Conf. Comp. Vis.*, 1904–1912. (cited on pages 3, 49, 50, 59, 68, 69, and 72)

Timofte, R.; Zimmermann, K.; and Gool, L. J. V., 2009. Multi-view traffic sign detection, recognition, and 3d localisation. In *Proc. App. Comp. Vis. Workshop*, 1–8. IEEE. (cited on pages 24 and 39)

Tompson, J. J.; Jain, A.; LeCun, Y.; and Bregler, C., 2014. Joint training of a convolutional network and a graphical model for human pose estimation. In *Proc. Adv. Neural Inf. Process. Syst.*, 1799–1807. (cited on page 49)

Torres, J. J. Y.; Bergasa, L. M.; Arroyo, R.; and Lazaro, A., 2014. Supervised learning and evaluation of kitti's cars detector with DPM. In *Proc. IEEE Intell. Vehicles Symp.*, 768–773. (cited on pages 23, 25, 43, 44, and 46)

Tuzel, O.; Porikli, F.; and Meer, P., 2006. Region covariance: A fast descriptor for detection and classification. In *Proc. Eur. Conf. Comp. Vis.*, 589–600. Springer. (cited on pages 1, 9, 10, and 31)

Van De Sande, K.; Gevers, T.; and Snoek, C., 2010. Evaluating color descriptors for object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32, 9 (2010), 1582–1596. (cited on pages 99, 101, and 112)

Van de Weijer, J.; Gevers, T.; and Bagdanov, A. D., 2006. Boosting color saliency in image feature detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28, 1 (2006), 150–156. (cited on pages 5, 100, 101, and 112)

Vedaldi, A. and Fulkerson, B., 2010. Vlfeat: an open and portable library of computer vision algorithms. In *Proc. IEEE Int. Conf. Multimedia*, 1469–1472. ACM. (cited on page 33)

Viola, P. and Jones, M. J., 2004. Robust real-time face detection. *Int. J. Comp. Vis.*, 57, 2 (2004), 137–154. (cited on pages 22, 23, 27, 39, and 40)

Wang, F.; Man, L.; Wang, B.; Xiao, Y.; Pan, W.; and Lu, X., 2008. Fuzzy-based algorithm for color recognition of license plates. *Patt. Recogn. Lett.*, 29, 7 (2008), 1007–1020. (cited on pages 99 and 101)

Wang, G.; Ren, G.; Wu, Z.; Zhao, Y.; and Jiang, L., 2013a. A robust, coarse-to-fine traffic sign detection method. In *Proc. Int. Joint Conf. Neural Net.*, 1–5. IEEE. (cited on pages 21, 25, and 39)

Wang, H.; Chen, Q.; and Cai, W., 2006. Shape-based pedestrian/bicyclist detection via onboard stereo vision. In *Proc. Multiconf. Computational Eng. Syst. App.*, 1776–1780. IEEE. (cited on page 26)

Wang, J.; Bebis, G.; and Miller, R., 2005. Overtaking vehicle detection using dynamic and quasi-static background modeling. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn. workshop*, 64–64. IEEE. (cited on page 26)

Wang, X.; Han, T. X.; and Yan, S., 2009. An HOG-LBP human detector with partial occlusion handling. In *Proc. IEEE Int. Conf. Comp. Vis.*, 32–39. IEEE. (cited on pages 23 and 33)

Wang, X.; Yang, M.; Zhu, S.; and Lin, Y., 2013b. Regionlets for generic object detection. In *Proc. IEEE Int. Conf. Comp. Vis.*, 17–24. IEEE. (cited on pages 23, 43, 44, 46, 72, and 75)

Wu, J.; Brubaker, S. C.; Mullin, M. D.; and Rehg, J. M., 2008. Fast asymmetric learning for cascade face detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30, 3 (2008), 369–382. (cited on page 40)

Yan, J.; Yu, Y.; Zhu, X.; Lei, Z.; and Li, S. Z., 2015. Object detection by labeling superpixels. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 5107–5116. (cited on page 51)

Yan, J.; Zhang, X.; Lei, Z.; Liao, S.; and Li, S. Z., 2013. Robust multi-resolution pedestrian detection in traffic scenes. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 3033–3040. IEEE. (cited on page 23)

Yang, B.; Yan, J.; Lei, Z.; and Li, S. Z., 2015a. Convolutional channel features. In *Proc. IEEE Int. Conf. Comp. Vis.*, 82–90. (cited on pages 51, 52, 54, and 55)

Yang, L.; Luo, P.; Change Loy, C.; and Tang, X., 2015b. A large-scale car dataset for fine-grained categorization and verification. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 3973–3981. (cited on pages 75, 78, 79, 80, 85, 89, and 90)

Zhang, N.; Farrell, R.; and Darrell, T., 2012. Pose pooling kernels for sub-category recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 3665–3672. IEEE. (cited on pages 4 and 75)

Zhang, S.; Benenson, R.; and Schiele, B., 2015. Filtered channel features for pedestrian detection. *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, (2015), 1751–1760. (cited on pages 9, 49, 50, 51, 52, 53, 57, 61, 67, 69, and 72)

Zheng, S.; Jayasumana, S.; Romera-Paredes, B.; Vineet, V.; Su, Z.; Du, D.; Huang, C.; and Torr, P., 2015. Conditional random fields as recurrent neural networks. *arXiv:1502.03240*, (2015). (cited on pages 59 and 60)