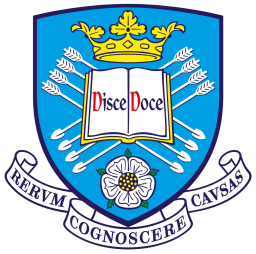


The University Of Sheffield.

Machine Learning Methods for Autonomous Object Recognition and Restoration in Images

By:

Ruilong Chen

A thesis submitted in partial fulfilment

of the requirements for the degree of

Doctor of Philosophy

Department of Automatic Control and Systems Engineering

University of Sheffield

May 2018

Abstract

Image recognition and image restoration are important tasks in the field of image processing. Image recognition are becoming very popular due to the state-of-the-art deep learning methods. However, these models usually require big datasets and high computational costs, which could be challenging. This thesis proposes an online learning framework that deals with both small and big datasets. For small datasets, a Cauchy prior logistic regression classifier is proposed to provide a quick convergence, and the online weight updating scheme is efficient due to the previously trained weights being reused. For big datasets, convolutional neural network could be implemented. For image recognition, non-parametric classifiers are often used for image recognition such as K-nearest neighbours, however, K-nearest neighbours are vulnerable to noise and high dimensional features. This thesis proposes a non-parametric classifier based on Bayesian compressive sensing; the developed classifier is robust and it does not need a training stage. For image restoration, which is usually performed before image recognition as a preprocessing process. This thesis proposes such a joint framework that performs image recognition and restoration simultaneously. In image restoration, image rotation and occlusion are common problems but convolutional neural networks are not suitable to solve these due to the limitation of the convolutional process and pooling process. This thesis develops a joint framework based on capsule networks. The developed joint capsule framework could achieve a good result on recognition, image de-noising, recovering rotation and removing occlusion. The developed algorithms have been evaluated for vehicle logo restora-

tion and recognition, however, they are transferable to other implementations. This thesis also developed an automatic detection and recognition framework for badger monitoring for the first time. Badger plays a key role in the transmission of bovine tuberculosis, which is described by government as the most pressing animal health problem in the UK. An automatic badger monitoring system could help researcher to understand the transmission mechanisms and thereby to develop methods to deal with the transmission between species.

Acknowledgements

First and foremost, I would like to thank my supervisor Prof. Lyudmila Mihaylova for her continuous support. It has been my honour to be one of her PhD students. She encouraged me though my whole PhD study with great enthusiasm and patience. I appreciate her encouragement and support for guiding my PhD. Also, I will never forget that she helped me amend papers sentence by sentence. She helped me to build social connections with other universities and companies, which is very helpful. I could not have imagined having a better supervisor for my PhD study.

I would like to express my gratitude to my second supervisor Dr. Wei Liu, for the insightful discussions. I would also like to thank Dr. Matthew Hawes, for his continuous help and valuable suggestions in writing academic papers. During the PhD study, I have also appreciated the knowledge sharing from collaborations with Dr. Olga Isupova, Dr. Jingjing Xiao, Dr. Hao Zhu and Dr. Ruth Little.

I would like to thank my wife, Sofiana Millati, for her valuable care and love. I am so lucky to have her and she makes my PhD life very happy. I would also like to thank my best friend in Sheffield, Bo Zhang, for his tremendous support and valuable advice. I am grateful to my parents and my sister, for always providing me with unconditional support in my life.

To my family

Contents

List of Symbols and Notations

A list of the symbols and notations used in this thesis is defined below. The definitions will be used throughout unless otherwise stated.

$\mathbf{x}$	Vector
$\mathbf{X}$	Matrix
$\mathbf{x}^*$	A testing data
$\hat{\mathbf{x}}$	The estimation of $\mathbf{x}$
$\mathbf{x}^\mathsf{T}$, $\mathbf{X}^\mathsf{T}$	Transpose of $\mathbf{x}$, $\mathbf{X}$
min	Minimum
max	Maximum
$arg\ min$	Argument that minimises
$arg\ max$	Argument that maximises
$\sum_{i=1}^{N}$	Sum operation from index 1 to N
$\prod_{i=1}^{N}$	Production operation from index 1 to N
$\frac{df(x)}{dx}$	First derivative of the function $f(x)$ with respect to a variable x
$\frac{\partial f(x,y)}{\partial x}$	Partial derivative of the function $f(x, y)$ with respect to a variable x

$\mathcal{N}(\mu, \mathbf{\Sigma})$	Normal distribution with mean μ and covariance $\mathbf{\Sigma}$
$p(\cdot)$	Probability operator
$p(\cdot\|\cdot)$	Conditional probability operator
$\mathbb{R}^M$	Real numbers of dimensions M
$\propto$	Proportional to
$ln(\cdot)$	The natural logarithm
$exp(\cdot)$	Exponential function
$\|\| \cdot \|\|_1$	l_1-norm
$\|\| \cdot \|\|_2$	l_2-norm
$\|\| \cdot \|\|_F$	Frobenius norm
$*$	The convolutional operation
$s.t.$	Subject to
$\| \cdot \|$	The determinant
$\langle \cdot, \cdot \rangle$	Inner product

Nomenclature

BBCS	Back-propagation Bayesian Compressive Sensing
BCS	Bayesian Compressive Sensing
BOW	Bag of Words
bTB	Bovine Tuberculosis
CNNs	Convolutional Neural Networks
CS	Compressive Sensing
DoG	Difference of Gaussian
HOG	Histogram of Oriented Gradient
ITS	Intelligent Transportation Systems
KNN	K Nearest Neighbours
LR	Logistic Regression
PCA	Principle Component Analysis
PSNR	Peak Signal-to-Noise Ratio
ReLU	Rectified Linear Unit
RVM	Relevance Vector Machine

SGD	Stochastic Gradient Descent
SIFT	Scale Invariant Feature Transform
SURF	Speeded Up Robust Features
SVM	Support Vector Machine
VLR	Vehicle Logo Recognition

List of Figures

List of Tables

Chapter 1

Introduction

Image recognition became very popular in many applications following the success of the Convolutional Network Networks (CNNs [1]) in 2012. Before the CNNs, majority image recognition frameworks included a hand-crafted feature detection process and a feature description process [1, 2]. Image features transfer an image from the pixel level to a feature level and then features are used for classification purpose. Unlike the traditional hand-crafted feature methods, CNNs automatically learn the image feature by multiple convolutional operations. A CNN framework has many parameters, therefore, big datasets are required in order to learn these parameters. However, big datasets are not always available and the image data are obtained incrementally in some applications. This leaves the question of what the best solution is for a small dataset or an incremental dataset. In addition, CNNs have recently achieved good results on image restoration such as image de-noising and image super-resolution. However, automatically recovering rotated images and occluded images are challenging tasks but not well studied [3]. Rotation and occlusion are common image degradations and it would be valuable if a degraded image can be recovered to a clear version. Moreover, most existing deep learning frameworks are focusing on dealing with one particular problem such as image recognition and image restoration [4]. This appraises the question of whether we can develop a joint framework that could perform image recognition and restoration simultaneously using a shared

network.

The focus of this thesis is on the development of novel machine learning methods for autonomous image recognition and restoration. Chapter 3 proposes an online recognition framework in order to deal with incremental datasets. Chapter 4 focuses on the development of a spatial image feature method and a non-parametric classification method. Chapter 5 proposes joint frameworks for image recognition and restoration based on CNNs and capsule networks. Chapter 6 develops automatic detection and recognition methods for badger recognition. This thesis starts with the application of Vehicle Logo Recognition (VLR) and some other datasets have been applied. The developed methods could also be extended to other fields.

1.1 Thesis Outline

The structure of the thesis is outlined below:

Chapter 1 introduces the research topic and purpose of this thesis, followed by the outline of this thesis and key contributions in each chapter. Relevant publications are listed in the last section of this chapter.

Chapter 2 reviews some fundamental feature methods and classification methods. The state-of-the-art deep learning methods are introduced for image recognition, followed by an overview of methods for VLR and and image restoration.

Chapter 3 focuses on online learning, considering a small training image dataset at the beginning. In this case, CNNs would not be appropriate due to the high computational costs and a limited size of training data. A dynamic online learning framework is developed for streaming data where models are updated from small to big datasets. The developed framework includes the Histogram of Oriented Gradient (HOG) feature method and the Cauchy prior Logistic Regression (LR) with the conjugate gradient descent. The Cauchy prior assumes most of the weights are near zero-valued; this results in accurate and quick convergence in the weight update scheme. The Cauchy prior LR could be applied online using a weight update

scheme, this further decreases the computational costs. When the training data is big enough, the CNNs could be applied in order to further increase the accuracy and the robustness to image degradations such as noise.

Chapter 4 begins with a recognition framework based on the spatial Scale Invariant Feature Transform (SIFT) features. This feature method takes the advantage of considering spatial information of the SIFT features in an image. The spatial SIFT features could increase the robustness to noise by incorporating the geographical information of the local features. This chapter also investigates non-parametric classifiers and proposes a Similarity-based Bayesian Compressive Sensing (SBCS) classifier. The proposed SBCS classifier takes the advantage of the Bayesian approach when compared with the traditional compressive sensing approaches. In addition, a column-based subspace sampling is introduced in order to select representative samples in the training dataset, aiming at decrease the computational costs while keeping high accuracy. The SBCS proved to be robust to noise, when compared with the state-of-the-art CNNs.

Chapter 5 introduces the learning capsules and develops capsule networks for VLR. Capsule networks contain multiple layers similarly as in CNNs, however, the connections between adjacent layers are changed to vector-vector, rather than scalar-scalar in CNNs and in neural networks. This chapter also proposes joint frameworks targeted at simultaneously performing image recognition and restoration tasks. Combining the recognition and restoration could help the weight updating process by a shared network. Image rotation and occlusion are considered in the restoration process. These image degradations can be recovered while giving high recognition accuracies at the same time.

Chapter 6 develops an automatic recognition framework for badger monitoring. Automatic badger monitoring is beneficial to enhance understanding of the transmission of Bovine Tuberculosis (bTB), which is described by government as the most pressing animal health problem in the UK. This is due to the transmis-

sions of the disease including cattle-badger, badger-badger and badger-cattle. In this chapter, an image dataset has been created. Based on the created dataset, two recognition frameworks have been developed based on CNNs. The trained recognition frameworks are combined with a video frame detection algorithm, which aims at selecting frames of interest in videos.

Chapter 7 gives a summary of the thesis and discusses the future work.

1.2 Key Contributions

Key contributions of this thesis are highlighted below.

- For the first time, an online recognition framework is developed for VLR; this framework considers both small dataset and big dataset.
- The Cauchy prior LR classifier is proposed with the conjugate gradient descent. The developed maximum a posterior model gives a quicker solution when compared with the maximum likelihood model in LR.
- A novel VLR framework has been developed with a spatial SIFT feature method. This framework considers the spatial correlation of different SIFT features and improves the robustness to noise.
- A novel Similarity-based BCS non-parametric classifier is proposed. The Bayesian compressive sensing is applied to estimate the weights vector. The proposed classifier proved to be quicker when compared with the state-of-the-art sparse representation classifier while giving similar recognition results.
- A column-based subspace sampling is developed to pick up representative data from the dataset for VLR. This process significantly decreases the computational costs while keeping the feature space unchanged.
- For the first time, a learning capsule framework is proposed and developed in the field of intelligent transportation systems. The proposed framework

achieves higher accuracy and better robustness to image degradations than the state-of-the-art CNNs.

- Joint frameworks are proposed for image recognition and image restoration based on CNNs and capsule networks. The proposed joint framework based on capsules achieves good results on recognition, image de-noising, rotation correction and occlusion removal.

- For the first time, an image dataset is created for badger recognition.

- For the first time, an automatic recognition framework is developed for badger recognition.

- An automatic detection scheme is developed aiming at identifying frames of interest in videos.

1.3 Publications

The author's publications with relevance to this thesis are listed below:

Peer Reviewed Conference Proceedings

- [$P1$] R. Chen, M. Hawes, L. Mihaylova, J. Xiao, and W. Liu, "Vehicle logo recognition by using spatial-SIFT combined with logistic regression", in *Proc. of International Conf. on Information Fusion*, Heidelberg, Germany, July 2016, pp.1228-1235

- [$P2$] R. Chen, M. Hawes, O. Isupova, L. Mihaylova, and H. Zhu, "Online vehicle logo recognition using Cauchy prior logistic regression", in *Proc. of International Conf. on Information Fusion*, Xi'an, China, July 2017, pp.1-8

- [$P3$] R. Chen, M. A. Jalal, L. Mihaylova, and R. Moore, "Learning capsules for vehicle logo recognition", in *Proc. of International Conf. on Information*

Fusion, Cambridge, UK, July 2018

- [$P4$] M. A. Jalal, R. Chen, R. Moore, and L. Mihaylova, "American sign language posture understanding with deep neural networks", in *Proc. of International Conf. on Information Fusion*, Cambridge, UK, July 2018

Journal Publication Under Review

- [$P5$] R. Chen, M. Hawes, and L. Mihaylova, "Vehicle logo and scene recognition based on back-propagation Bayesian compressive sensing", *Signal Processing*, 2018

Journal Publication in Preparation

- [$P6$] R. Chen, L. Mihaylova, R. Little, R. Cox and R. Delahay "Wildlife monitoring with deep learning methods", *Methods in Ecology and Evolution*, 2018

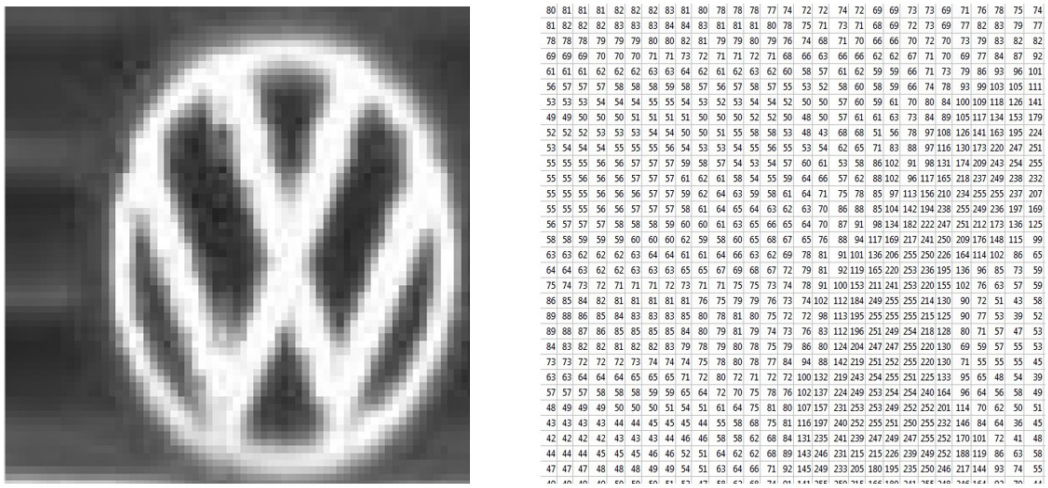
Chapter 2

Literature Review

2.1 Traditional Methods for Image Recognition

Recognising content in an image is easy for humans as we have advanced visual systems which are good at detecting edges and describing the abstract contents. In contrast, computers are good at describing digitised information such as how many pixels are in an image and what their intensity values are. However, it is challenging for computers to describe the abstract content inside an image. For instance, Figure 2.1 shows how humans and computers see an image. We humans automatically detect a big cycle inside the picture, the 'V', and 'W' shape inside the cycle as shown in Figure 2.1 (a). We can even understand that the main content is a vehicle logo representing the Volkswagen company if we have the knowledge in advance. In contrast, Figure 2.1 (b) shows how the image is stored in a computer, only intensity values of pixels are recorded. The digital image can be easily influenced by noise, shift, rotations, and occlusions. For example, if one row in the image matrix is deleted, humans find it hard to detect the difference, whereas the matrix stored in the computer has been totally changed.

Even though recognising images is easy for human beings, we still wish to let computers do such tasks as it can save cost and avoid mistakes by human fatigue. One simple way of letting computers recognise a vehicle logo image is to send pixel



(a) (b)

Figure 2.1: The ways in which humans and computers understand an image, respectively.

intensity values from well-segmented logo images directly into a classifier. However, this is not feasible for images of large size due to the computational costs. More importantly, the accuracy is low. For instance, simply reshaping the raw pixel intensities into a SVM could only achieve an accuracy of 16.40 % on a vehicle logo dataset (methods developed in chapter 5 could achieve up to 100% accuracy on the same dataset by using the capsule networks). While the dataset has ten classes and a random guess has an expectation of 10 % accuracy according to the probability theory. Hence, more advanced techniques are required for image recognition.

2.1.1 Image Features

In order to let computers recognise the content in an image, hand-crafted features that explore the image pattens are often applied instead of using the raw pixel values. Hand-crafted features define an image by engineering rules. For example, colour information including colour histogram and colour moment is used as image features [5]; Yang et al. [6] use shape and contour information as the image features; distinguishable edges and corners information is extracted as features in [2, 7]. Presently, automatic feature extraction methods inspired by neurons in the human visual systems have become the mainstream in the field of image recognition. The

automatic learned features proved to be more accurate when associated with big data [1]. The following introduce different image feature methods, from the traditional hand-crafted features to the state-of-the-art deep learning features.

Global Features

Hand-crafted features can be separated as global features and local features. Global features take all pixels into consideration and represent an image with a single vector. In the following, the well-known global feature method HOG is introduced in detail.

HOG was originally introduced as a representation method for human detection [7, 8]. HOG calculates the horizontal gradient G_x and the vertical gradient G_y on every pixel by use of a 1-D filter [-1, 0, 1]:

$$G_{x(i,j)} = \mathcal{I}(i+1,j) - \mathcal{I}(i-1,j), \tag{2.1}$$

$$G_{y(i,j)} = \mathcal{I}(i,j+1) - \mathcal{I}(i,j-1), \tag{2.2}$$

where $\mathcal{I}(i,j)$ is the intensity value at pixel location index (i,j).

Then the horizontal gradient and vertical gradient can be applied to calculate the orientation of gradient $\theta(i,j)$ and the magnitude of gradient $H(i,j)$ for every pixel in the image:

$$\theta(i,j) = arctan(G_{y(i,j)}/G_{x(i,j)}), \tag{2.3}$$

$$H(i,j) = \sqrt{G^2_{x(i,j)} + G^2_{y(i,j)}}, \tag{2.4}$$

where $\theta(i,j)$ and $H(i,j)$ represent the orientation and the magnitude of the gradient at pixel location index (i,j) respectively.

The next step is quantizing the orientation into bins evenly spaced over $0° - 180°$ in order to build an orientation histogram. The image is divided into cells, with a certain number of cells building up a block. Figure 2.2 shows how an image is

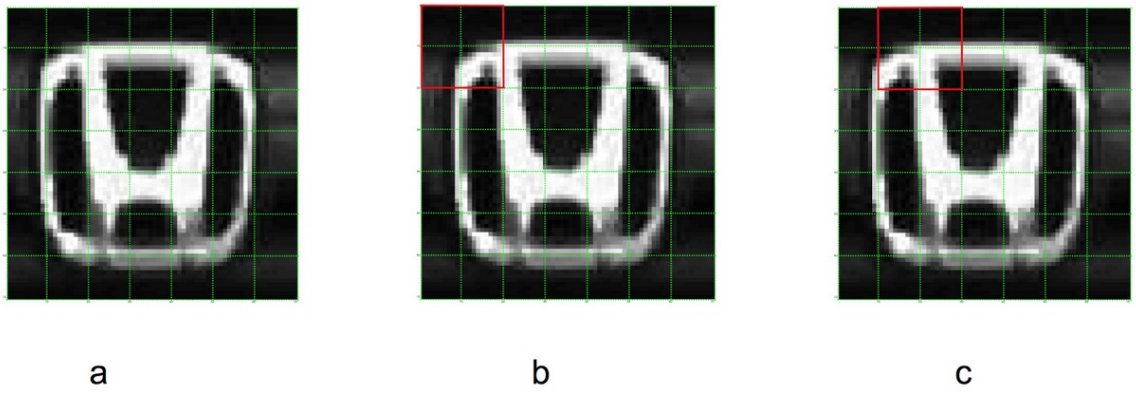


Figure 2.2: The process of dividing an image into cells and blocks in the HOG algorithm. In (a), images are divided into cells, a group of cells form a block denoted as red lines in (b), the block shifts from left to right in (c).

divided into cells and blocks. The original image can be divided into different cells of the same size (Figure 2.2.a). Each cell can be represented as a histogram, with the quantized orientations as histogram bins and the magnitude as weights. Histograms in a block (a block is a 2×2 cell in the example) are concatenated together and get normalised in order to be robust to illuminance variations. The block is then shifted one cell (or one block) forward from either left to right or top to bottom. In the example shown in Figure 2.2, a block is shifted cell by cell; this results in overlaps among adjacent blocks. In the Figure 2.2.b and Figure 2.2.c, the area within red lines indicates the first and second block, respectively. The block shifts from the top left corner to the bottom right corner. Histograms generated from all blocks are then concatenated together in order to generate a vector, which is the HOG feature. The HOG feature method has been applied to many fields, for example, human detection [7, 9], action recognition [10, 11], face and emotion recognition [12–15], handwritten digit recognition [16] and traffic sign recognition [17]. The advantage of the HOG features lies in its efficiency and good performance on certain images without complex content.

Another global feature method is the GIST feature [18], which is inspired by the fact that humans can grasp the "gist" information of an image in a few seconds [19]. The GIST features are designed to describe an image by some perceptual dimensions

using Gabor filters [20]. In the feature generation process, a group of Gabor filters are convolved with the image in order to generate a group of feature maps. These Gabor filters are designed in advance in order to extract different orientations at multiple scales. Each feature map is divided into sub-regions, similar to the HOG algorithm, and the mean values of each sub-region are concatenated together to form a GIST feature vector. The GIST features share similarity with the state-of-the-art CNN features, which will be introduced in later sections. The Gabor filters can be regarded as convolutional kernels in CNNs and the averaging process is a pooling process. However, the CNNs are more advanced by having multi-layers and an automatic weights updating scheme.

The designing process of the global features made their drawbacks straightforward. If the content of interest moves from one area to a distinct area, the feature vector would be changed dramatically. This limits the global features only suitable for well segmented images, or the contents of interest always appear at similar locations. The feature vectors will also be easily influenced by scale variations and orientation changes. In order to deal with these challenges and extract more robust features, local feature methods are proposed and they are more favoured than global features in many applications.

Local Features and Bag of Words

Compared with the global features mentioned above, local descriptors are interested in fractional areas in an image rather than the whole image. In general, global representations are sensitive to challenges such as illuminance variation, noise background, and rotations [21]. In contrast, local features tend to be more robust under these conditions [22]. the SIFT feature method is the most widely used example of the local features.

The SIFT method contains a feature detection process and a description process. In the feature detection process, different 2-D Gaussian filters are convolved with the

original image $\mathcal{I}(x, y)$ to get smoothed images, with each denoted as a $L(x, y, \mathtt{h}\sigma)$, where $\mathtt{h}$ is a constant multiplicative factor. This thesis uses $f(\cdot)$ presenting a function and a 2-D Gaussian filter can be denoted as:

$$f(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}, \tag{2.5}$$

and

$$L(x, y, \sigma) = f(x, y, \sigma) * \mathcal{I}(x, y), \tag{2.6}$$

with x, y as spatial coordinates and σ^2 as the variance of the Gaussian filter; $*$ denotes the convolution operation.

Then the Difference of Gaussians (DoG) is generated by calculating the differences between these Gaussian smoothed images. The DoG map $D(i, j, \sigma)$ is defined as:

$$D(x, y, \sigma) = L(x, y, \mathtt{h}\sigma) - L(x, y, \sigma). \tag{2.7}$$

The DoG is not only applied on the original image but also on the up-sampled and down-sampled images in order to be scale invariant. The potential interest points are extrema among its neighbours in the DoG maps. All the extrema are then revalued in order to reject some unstable extrema, aiming to improve the robustness of the interest points. Finally the location and scale of remaining extrema are chosen as the location and the scale of interest points.

After the location and scale of an interest point have been detected, its neighbourhood area is chosen in order to describe the interest point. All the gradients in the selected area are rotated relative to the main orientations in order to make each interest point invariant to rotation variations. Weights of orientations are controlled by both the magnitude of gradients and a Gaussian kernel which is centred on the interest point [2]. All histograms are then concatenated into a vector of fixed length. Finally, the vector is normalised in order to be invariant to illumination variations. The final normalised vector is a SIFT descriptor.

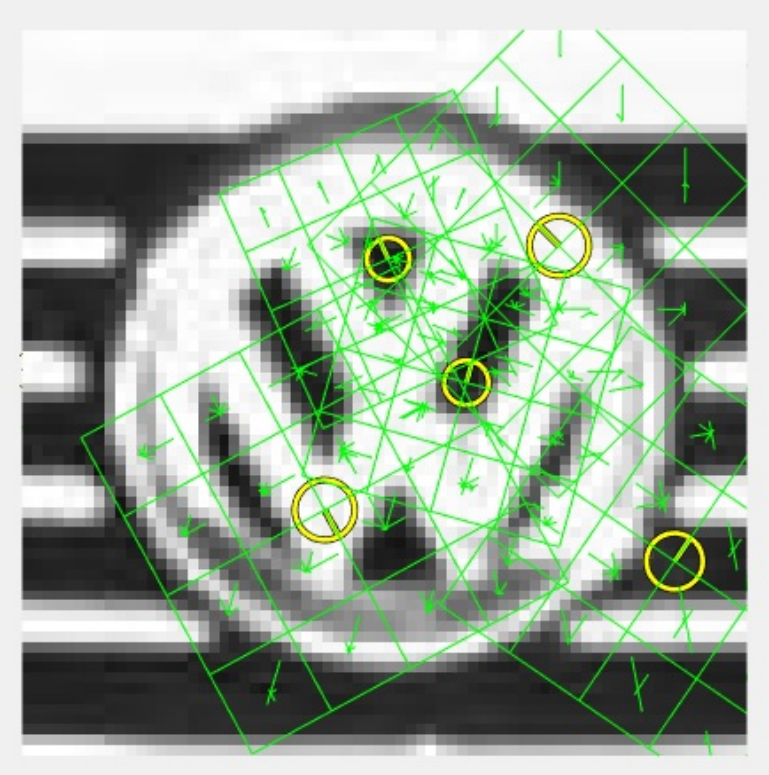


Figure 2.3: An example of SIFT descriptors.

Figure 2.3 gives an illustration of the SIFT descriptor; only five interest points are chosen for the convenience of the illustration. Centres of the yellow circles are locations of interest points, and the arrows inside the circles are the main orientations. Each block of 4×4 cells (green boxes) is chosen based on the location and orientation of the corresponding interest point. Sizes of blocks are different in Figure 2.3 because they are in different scales. Orientations of gradients are quantized into eight bins, with the result that a final SIFT feature has a dimension of $4 \times 4 \times 8 = 128$.

Hand-crafted local features have been well studied in the last decade, including the interest point detection process and the description process. Another popular local feature method is Speeded Up Robust Features (SURF) [23], which use a Hessian matrix of an image as the interest point detector and apply a similar description process similar to that in SIFT, with the gradient histogram replaced by the Haar wavelet response. As local features have both detection and description processes, different combinations can be applied in order to get the local features. For example,

one can use the Harris corner detector [24] to find the interest points and use the SIFT descriptor to generate features. Compared with global features, local features have a broader range of implementations apart from image recognition. For example, image retrieval [25–27] and image alignment [28,29]. This is due to the local features being robust to scale variations, shift and rotations.

Bag of Words

Different images may have a different number of local interest points because the number of interest points is determined by the number of local extrema, which varies from image to image. In other words, images are represented by matrices with different sizes using local features. In order to solve this problem, the Bag of Words (BOW) representation model is required prior to classification.

Csurka et al. [30] proposed the BOW model on top of local features in order to represent an image by a feature histogram. This representation process is efficient in terms of computational costs and practical implementation. The BOW model consists of two main parts: a dictionary generation process by the k-means clustering [31] and a histogram representation process.

The k-means clustering is an unsupervised vector quantisation algorithm. It clusters n observations into k clustering centroids by allocating all the observations into its nearest centroid. The algorithm involves four steps:

Algorithm 1 The process of the k-means clustering

1: Randomly choose k points as the initial group centroids in the training dataset.
2: Assign all the training data points to its nearest centroid.
3: When all data points have been assigned, find the centre of each group and assign it as the new centroid.
4: Repeat steps 2 and 3 until all of the centroids become stable.

Using the k-means clustering method, a dictionary is generated by k 'words' (centroids) and each 'word' has the same dimensions as a feature vector. For an image that consists of a few local interest points, each feature descriptor can find its

closest 'word' from the dictionary, where the closest distance is defined as the minimal l_2-norm distance [32]. If a descriptor has found its nearest 'word' in the dictionary, the number of occurrences of this 'word' will have increased by 1. The BOW model represents an image as a histogram by using each 'word' in the dictionary as a histogram bin and the occurring frequency of each 'word' as its magnitude [30]. The normalised vector is the final representation vector. Using the BOW representation model, all images can be represented as a vector of the same dimensions, no matter how many local features were generated in each image.

2.1.2 Classification

After images are represented as representation vectors (note, a global feature itself is a representation vector because there is no BOW process required), they can then be classified by a classifier. An accurate recognition framework requires both good feature methods and classifiers. In supervised learning, labels for training data are given beforehand. In such a case, denote N training data as $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N)^\mathtt{T}$ with each data $\mathbf{x} \in \mathbb{R}^{M\times 1}$ and their corresponding labels as $\mathbf{y} = (y_1, y_2, \cdots, y_N)^\mathtt{T}$ with $\mathbf{y} \in \mathbb{R}^{N\times 1}$. The goal of a parametric classifier is to learn an accurate model from the training data; this model can then be further applied to predict the label y^* for the testing data point $\mathbf{x}^* \in \mathbb{R}^{M\times 1}$.

Take the VLR problem with a parametric classifier as an example. For a BMW logo image $\mathbf{x}$, its corresponding class y is 'BMW'. For numeric computation purpose, 'BMW' can be denoted as '1', 'Honda' can be denoted as '2' etc. In the testing stage, the model generated in the training stage could then be applied to predict the label information y^* for any incoming image $\mathbf{x}^* \in \mathbb{R}^{M\times 1}$. For instance, if the predicted label y^*=2 is generated for the testing image $\mathbf{x}^*$, the testing image is recognised as a 'Honda' image. The label information for testing data is only used as the ground truth in order to evaluate the performance of a model. In supervised learning where all labels are given, there are parametric approaches and the non-

parametric approaches [33].

Non-parametric Classifier

In parametric approaches such as SVM, the goal in the training stage is to estimate a fixed number of parameters in the model by giving the training data $\mathbf{X}$ and their corresponding labels $\mathbf{y}$. SVM has been one of the most important classification methods in the field of computer vision in recent years [34–36]. Taking the linear SVM as an example, the training data is separated by a maximal margin in a linearly separable space. In the binary classification, denote a set of training data $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$, where $\mathbf{x}_i \in \mathbb{R}^{M\times 1}$ $(i = 1, 2, ..., N)$ is a M-dimensional feature vector and the data belongs to two categories $y_i \in \{-1, 1\}$.

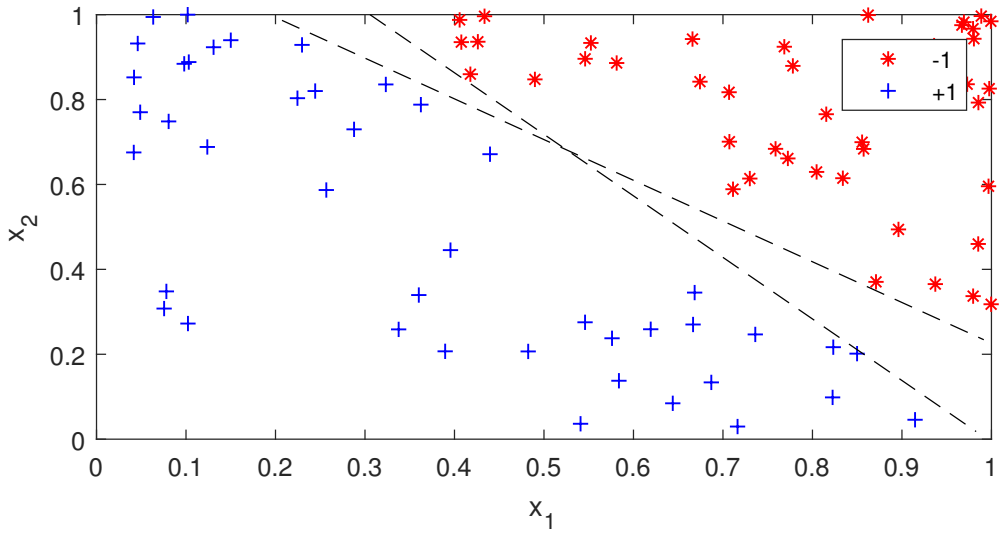


Figure 2.4: An example of data can be separated by infinite lines.

Figure 2.4 gives an example of $\mathbf{x}_i$ in Euclidean space with axis x_1 and x_2. In this situation, there are infinite hyperplanes that could separate all the training data. Hence, the learning algorithm stops when it finds the first line that satisfies the criteria in the training stage. However, the testing data are usually different from the training data. This results in some lines performing better than the others in the testing stage, despite all lines performing perfectly in the training stage.

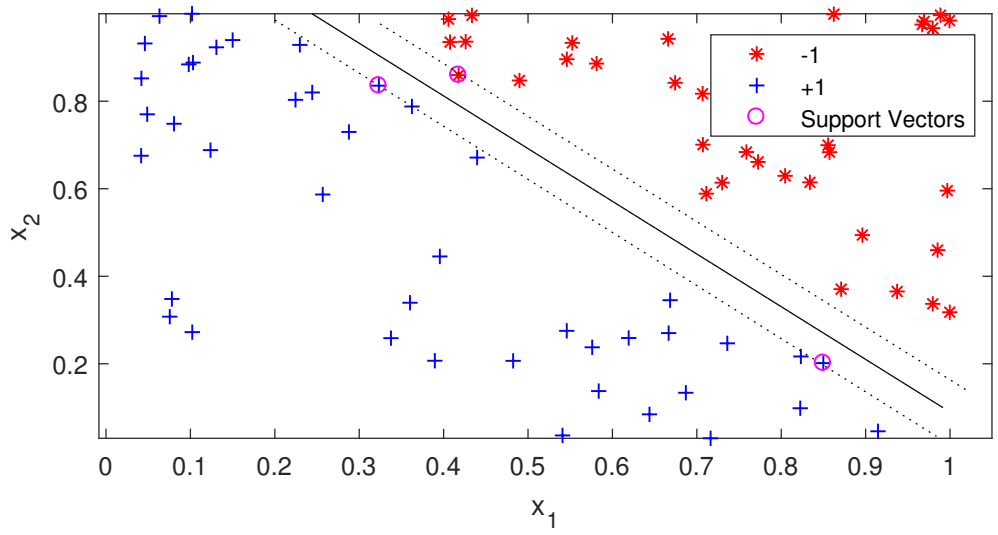


Figure 2.5: An example of the maximum margin in the SVM.

The key process in the SVM classifier is to find the hyperplane that separates the training data with the largest margin, as shown in Figure 2.5. The largest margin minimises the risk of making an error for the testing data, in other words, having a good generalisation ability [37]. If the training data is linearly separable, the two hyperplanes that guarantee the margin (dash lines in Figure 2.5) can be formulated as follows:

$$\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b \geq +1, \quad for\ y_i = +1, \tag{2.8}$$

$$\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b \leq -1, \quad for\ y_i = -1, \tag{2.9}$$

where $\mathbf{w} \in \mathbb{R}^{M\times 1}$ and b are the weights vector and the bias, respectively. These two equations can also be combined into:

$$y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) \geq 1, \quad \forall_i, \tag{2.10}$$

where $\forall_i$ denotes for all i.

Considering the data lie on the hyperplane $\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b = +1$ in equation (2.8), the perpendicular distance between the hyperplane and the origin is $|1-b|/||\mathbf{w}||_2$.

Similarly, for data lying on the hyperplane $\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b = -1$ in equation (2.9), the perpendicular distance between the hyperplane and the origin is $|-1-b|/||\mathbf{w}||_2$, where $||\cdot||_2$ represent the l_2-norm. Therefore, the margin between these two hyperplanes is $2/||\mathbf{w}||_2$. The problem of finding the maximum margin $2/||\mathbf{w}||_2$ equals to finding the minimum value of $||\mathbf{w}||_2^2$ subject to the constraints in equation (2.10), which can be formed as:

$$\underset{\mathbf{w},b}{min}\ \frac{1}{2}||\mathbf{w}||_2^2, \quad s.t. \quad y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) \geq 1, \quad \forall_i. \tag{2.11}$$

The equation (2.11) can be interpreted in the Lagrangian formulation, which finds extrema for the function $f(x, y)$ given its constraints function $g(x, y) = 0$. Similarly, in the SVM, function $f(\mathbf{w})$ needs to be minimised given the inequality constraints function $g(\mathbf{w}, b) = y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) - 1 \geq 0$. Therefore, the problem is solvable by using the positive Lagrange multipliers $\alpha_i, i = 1, \cdots, N$, this gives the primal Lagrangian:

$$\begin{aligned} L_P &= \frac{1}{2}||\mathbf{w}||_2^2 - \sum_i^N \alpha_i y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b - 1) \\ &= \frac{1}{2}||\mathbf{w}||_2^2 - \sum_i^N \alpha_i y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b) + \sum_i^N \alpha_i. \end{aligned} \tag{2.12}$$

Here L_P must be minimised with respect to $\mathbf{w}$ and b without considering α_i first. Then it can be maximised after $\mathbf{w}$ and b are substituted back, subject to the constraints that the $\alpha_i \geq 0$. This is a convex quadratic programming problem [37]. L_P is minimised by taking partial derivatives for $\mathbf{w}$ and b and setting them to 0:

$$\frac{\partial L_P}{\partial \mathbf{w}} = \mathbf{0} \quad \Rightarrow \quad \mathbf{w} = \sum_i^N \alpha_i y_i \mathbf{x}_i, \tag{2.13}$$

$$\frac{\partial L_P}{\partial b} = 0 \quad \Rightarrow \quad b = \sum_i^N \alpha_i y_i = 0. \tag{2.14}$$

After $\mathbf{w}$ and b has been substituted into equation (2.12), the dual Lagrangian is

given:

$$\begin{aligned} L_D &= \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} - \mathbf{w}^{\mathrm{T}}\sum_{i}^{N}\alpha_i y_i \mathbf{x}_i - b\sum_{i}^{N}\alpha_i y_i + \sum_{i}^{N}\alpha_i \\ &= -\frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} + \sum_{i}^{N}\alpha_i \\ &= -\frac{1}{2}\sum_{i}^{N}\sum_{j}^{N}\alpha_i\alpha_j y_i y_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle + \sum_{i}^{N}\alpha_i, \end{aligned} \tag{2.15}$$

where $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ is the inner product of $\mathbf{x}_i$ and $\mathbf{x}_j$. The solution is found by maximising L_D subject to:

$$\sum_{i}^{N}\alpha_i y_i = 0, \quad and \quad \alpha_i \geq 0, \quad \forall_i. \tag{2.16}$$

This can be solved by the convex quadratic programming [37]. Notice that every training data $\mathbf{x}_i$ is associated with a Lagrange multiplier α_i. In the solution, most values of α_i are zeros as they are far away from the margin hyperplane, which satisfy either $y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) > 1$ or $y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) < -1$. Those $\mathbf{x}_i$ with non-zero α_i are the support vectors. If the training data are not linearly separable, there are techniques that can either map the data into a linearly separable space by a kernel function or find a soft margin that is tolerant to some errors. This section gives an example of how a parameter classifier works and further reading about SVM can be found in [38]. In this example, the training process aims to find a fixed number of parameters in $\mathbf{w}$ and b. The learned $\mathbf{w}$ and b can then be applied to classify testing data, with the training data being no longer required in the testing stage.

Non-parametric Classifier

For non-parametric approaches such as K Nearest Neighbours (KNN) [39], a testing data is compared with the entire training dataset. KNN is the simplest method for classification where a testing data is classified based on its nearest neighbours in the training dataset. Normally the Euclidean distance is used. When $K = 1$, the input

point is assigned to the same class of its nearest neighbour. When $K > 1$, the input point is assigned to the class that the majority of these K nearest neighbours belong to. Easy implementation and good recognition results make KNN very popular for image classification [40–42].

Algorithm 2 The process of the KNN classification

1: For a testing data, calculate the Euclidean distance with all the training data.
2: Rank the distance from low to high and pick up the first K corresponding labels.
3: Find the majority vote from the K picked up labels and assign the label to the testing data.

The KNN algorithm is an example of a non-parametric method that is often used for classification [43, 44]. However, a straightforward drawback of KNN is that the computational costs increase with the size of the training dataset. This is due to a testing data compared with all training data. Another drawback is that KNN algorithm is not stable in high dimension spaces [45]. This is because the shortest Euclidean distance is not necessarily the best match to the testing data, especially when the number of training data are limited [45, 46]. Besides, the KNN algorithm has proven to be vulnerable to the effects of noise [47]. Meanwhile, the KNN algorithm gives an example that non-parametric classifiers do not need a training stage. While parametric classifiers might learn insufficient weights from the training stage, the non-parametric classifiers do not have this problem.

2.2 Deep Learning Framework for Image Recognition

Traditional image recognition frameworks require image feature methods, followed by classification methods. There are plenty of methods and each method has parameters that need to be defined by users. This raises the question: can we design a "black" box that takes images as the input and their corresponding labels as the output, without choosing image feature methods and classifiers? Neural networks

are designed in this manner but the input reshapes an image to a vector. However, changing an image matrix to a vector loses the spatial information of the content, which is essential for images. Hence, the input of the "black" box should be a matrix in order to keep the spatial information at the beginning. The CNNs are designed to solve this problem. They use convolutional processes to preserve the spatial information of the content; weights in the convolutional processes are automatically learned in a way similar to neural networks.

Lecun et al. proposed the first CNN framework LeNet [48]. Different CNN frameworks have been developed quickly after the AlexNet [1] achieved the best performance on ImageNet in 2012. Unlike neural networks, where neurons in each layer are fully connected to neurons in the next layer, each layer in a CNN shares the weights by using convolutional kernels. This process tremendously decreases the number of weights when compared with neural networks; therefore, it can prevent the over-fitting problem, which is one of the main problems in neural networks [49]. Another advantage is that the spatial information of the content is preserved by the convolutional process, while neural networks simply reshape an image into a vector, without preserving the spatial information. CNN frameworks are mainly composed of the convolution operations and the pooling operations. Figure 2.6 illustrates a typical CNN framework.

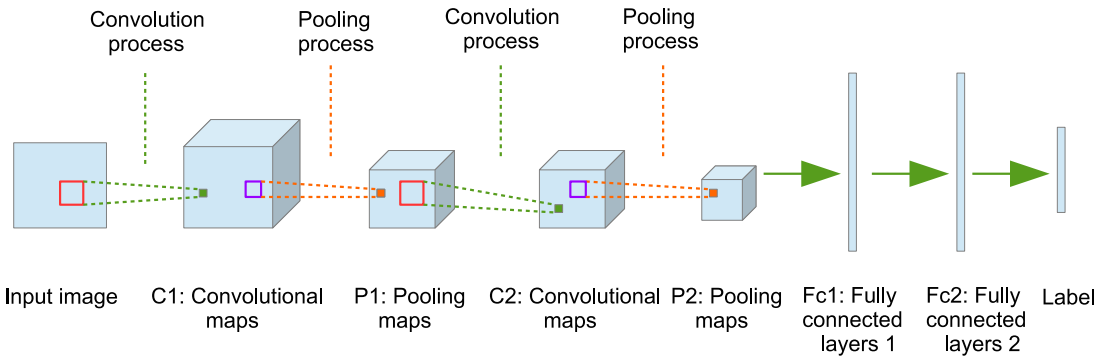


Figure 2.6: A typical CNN architecture example.

In a convolution stage, feature maps are convolved with different convolutional

kernels, which are equivalent to filters in the field of image processing. Kernels can be regarded as the shared weights connecting two layers. Suppose kernels of size $[a \times b \times n]$ ([height $\times$ width $\times$ depth]) are used, the i^{th} $(i = 1, 2, \cdots, n)$ convolutional feature map can be denoted as:

$$\mathbf{C}_i = f\left(\sum_j \mathbf{V}_i * \mathcal{I}_j\right), \tag{2.17}$$

where $\mathbf{V}_i$ is the i^{th} kernel and $\mathcal{I}_j$ $(j = 1, 2, \cdots, J)$ is the j^{th} feature map ($\mathcal{I}_j$ can be a channel of the original image, a pooling map and a convolutional map). Here $f(\cdot)$ denotes a non-linear activation function and $*$ represents the convolutional operation. The Rectified Linear Unit (ReLU), where $g(x) = max(0, x)$, is often applied as the non-linear function [1].

A convolutional process is often followed by a pooling process. In the pooling operation, a pooling process decreases the size of the input feature maps, which can be regarded as a down-sampling operation. Each pooling map $\mathbf{P}_i$ is usually obtained by a pooling operation over the corresponding convolutional map $\mathbf{C}_i$:

$$\mathbf{P}_i = pool(\mathbf{C}_i), \tag{2.18}$$

where $pool(\cdot)$ represents a pooling method [50]. A window shifts on the previous map $\mathbf{C}_i$ and the mean value (or the maximum value) in each window is extracted in order to form a pooling map $\mathbf{P}_i$.

The convolution and pooling operations are the two main techniques in CNNs. As shown in Figure 2.6, these two processes are repeated. Note that convolutional processes are followed by pooling operations in Figure 2.6. However, this is not a requirement; different CNN structures are valid. Different CNN architectures have been developed rapidly subsequent to the AlexNet in 2012. For example, the ZF-Net [51] applied smaller kernel size in order to save more original pixel level information and achieved better results on ImageNet [52]. The VGG-NET [53] also enhanced the

depth of the CNNs up to 19 layers and suggested only using a unique kernel size of $[3 \times 3]$. The Google-Net [54] even increased the number of layers to 22 and applied the inception module, in which different convolutional feature maps (generated by convolutional kernels of different sizes) and the pooling feature maps were combined together. The Res-Net [55] built a 152 layer architecture and introduced the idea of the residual learning, which built short-cut connections between layers and achieved the best result on ImageNet in 2015.

It is found that the CNNs automatically learn simple structures of an image, such as edges in the initial convolutional layers, and more abstract information can be learned in later convolutional layers [51]. This is very similar to Hubel and Wiesel's discovery [56] back in the 1960s, which suggests different orientation activates different groups of neurons in the cat's visual system. The CNNs have achieved great success and they currently dominate various image recognition tasks [57]. For example, face and motion recognition [58–60] and action recognition [61, 62].

2.3 Online Learning for Vehicle Logo Recognition

For the applications, this thesis start with the VLR problem, however, the developed methods can also be adapted to other image recognition tasks. VLR is important in Intelligent Transportation Systems (ITS) as the vehicle logo is one of the most distinguishable marks on a vehicle [63], and can assist in vehicle identification [43]. It has many potential applications in traffic monitoring and vehicle management systems. For instance, VLR can detect fraudulent plates if the combination does not match the data stored on the police security database [64]. As a result, this gives a more robust vehicle identification system. VLR could also provide guidance for autonomous driving systems and intelligent parking systems [65, 66]. In addition, recognising vehicle logos is also useful for commercial investigations [67] and document retrieval [65].

Global features that take all pixels into consideration are often used in VLR. For

example, Rezaei and Farajzadeh [68] used the horizontal and vertical histograms to compare with a pre-defined template; it is, however, not robust to illuminations, noise and rotation variations. Sharpness histogram features [69] are used on well-segmented logo images. However, the accuracy is around 80%, which cannot make an effective recognition system. The Tchebichef moment invariants method is used on VLR but the computation costs are high [65, 70]. Peng [44] proposed a statistical random sparse distribution feature method which performs well in the low-resolution situation. The DenseSIFT is a global feature method that shares the same description method with the SIFT descriptor while using every pixel as an interest point; this saves the computational costs for feature detection. However, it decreases the robustness of interest points. HOG [7] separates an image into small blocks and it calculates the horizontal gradients as well as vertical gradients in each block; it is a very successful global feature method and it is often used for VLR [7, 63, 71, 72].

However, local features are more favoured as they are robust to image noise, shift and rotations. For example, Yang [73] used the Harries corner detector to find the interest point for VLR. Psyllos et al. [43] and Lipikorn et al. [74] applied SIFT [2] features for vehicle logos. Badura and Foltan [75] used Speed Up Robust Features(SURF) [23] and achieved good results. Among local features, the SIFT feature method is the most popular on VLR [43, 74–76].

Currently CNN frameworks have been dominant in VLR, for instance, Gao and Lee [77] built a seven layer CNNs framework and achieved an accuracy of 88.4% on a self-selected dataset. Huang et al. [50] developed a pre-training process on CNNs for VLR, Xia et al. [78] created more images based on the dataset provided by [50] and developed a multi-task CNN framework for VLR. In general, CNN based methods are more accurate when there is a large training dataset. In the literature, transfer learning methods [79] have been developed, which fine tunes the weights based on the trained weight from big datasets in other domains. However, the automatic feature extraction method is particularly suitable for images from the source domain [79] .

In practice, there may only be an initially small training dataset, with additional images becoming available during the implementation. In order to take advantage of these additional images, new models can be built independently when more images become available. However, retraining new models increases the computational costs, especially when new models are updated frequently. CNNs update the weights online; however, it would not be useful in the earlier stage because training a CNN framework needs a huge dataset and involves huge computational costs. This motivates the work in Chapter 3, which develops a Cauchy prior LR framework for small dataset and CNNs are then applied when a big dataset becomes available.

2.4 Back-propagation Bayesian Compressive Sensing Classifier

Parametric classifiers such as SVM and LR assume a functional distribution of the data [46]. Hence, the relationship between the label and the input data can be modelled using a fixed number of parameters. An advantage of parametric classifiers is that an increasing size of training dataset would not increase the number of parameters in the model. Therefore, the computational costs in the testing stage remain constant as classifying a testing data only needs these parameters. However, in practice, parametric classifiers can result in an inadequately trained model because of inappropriate assumptions of prior distributions, leading to inappropriate predictions in the testing phase [33,46]. In contrast, non-parametric classifiers do not assume any particular distribution of the data, neither do they require a model with a fixed number of parameters [46]. In turn this increases the computational costs in the testing stage [80]. However, by avoiding models that can be insufficiently trained, they can be more flexible than the parametric classification methods [80].

A non-parametric classification approach based on sparse representation proposed by Wright et al. [81] has proved to be more accurate than the linear SVM and the

KNN classifier for face recognition. The SRC assumes that the testing data $\mathbf{x}^* \in \mathbb{R}^{M\times 1}$ can be represented as a linear combination of the training samples $\mathbf{X} \in \mathbb{R}^{M\times N}$ where M is the length of the vector representing the data being considered and N gives the number of entries in the training dataset. The linear representation of a testing image can be denoted as:

$$\mathbf{x}^* = \mathbf{X}\mathbf{w} + \mathbf{z}. \tag{2.19}$$

In equation (2.19), $\mathbf{w} \in \mathbb{R}^{N\times 1}$ is the weight vector that controls the contribution of each image in the training dataset to the linear combination representing the testing image, $\mathbf{z} \in \mathbb{R}^{M\times 1}$ is a noise term with $||\mathbf{z}||_2 \leqslant \epsilon$ and ϵ is a threshold constant. Conventionally, the solution of $\mathbf{w}$ is often solved by choosing the minimum l_2-norm distance:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}}(||\mathbf{w}||_2), \quad s.t \;\; ||\mathbf{x}^* - \mathbf{X}\mathbf{w}||_2 \leqslant \epsilon. \tag{2.20}$$

where $\hat{\mathbf{w}} \in \mathbb{R}^{N\times 1}$ is the estimated solution for the weight vector. However, the training dataset is often big enough to make $N > M$. Therefore, equation (2.19) represents an under-determined system and there is no unique solution by using conventional methods [81].

The SRC classification method assumes that a testing image can be sufficiently represented by instances from its corresponding class. Therefore, the solution is naturally sparse as coefficients for unrelated classes are zero valued. For instance, if there are 20 classes, only approximately 5% of the coefficients in $\hat{\mathbf{w}}$ will have non-zero values [81]. In fact, the sparser the recovered $\mathbf{w}$ is, the easier it is to accurately classify the testing image $\mathbf{x}^*$ [81]. This motivates the use of the l_0 to find the sparest solution for $\mathbf{w}$ in equation (2.19), where l_0 represents the number of non-zero entries.

However, l_0 minimisation is an NP hard problem. Instead an l_1-norm minimisa-

tion is typically used as an approximation [82–84], giving:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}}(||\mathbf{w}||_1), \quad s.t \;\; ||\mathbf{x}^* - \mathbf{X}\mathbf{w}||_2 \leqslant \epsilon. \tag{2.21}$$

The solution to the l_1-norm minimisation in equation (2.21) can be solved by kernel reconstruction [85], or estimated by standard linear programming methods such as Basis Pursuit [86], greedy optimisation such as Orthogonal Matching Pursuit [87], and approximate kernel reconstruction method [88]. The solution of equation (2.19) gives the optimal $\mathbf{w}$ for classification purpose in the SRC [81].

Recently the Bayesian Compressive Sensing (BCS) [89] approach has been efficiently applied to synthetic aperture radar target classification [90] and phonetic classification [91]. This Bayesian approach provides an alternative to the l_1-norm minimisation for optimising the linear combination coefficients required for the classification framework. Similar to Zhou et al. [92], by comparing the magnitudes of the coefficients, the testing data can then be classified by assigning it to the class whose coefficients have the highest l_2-norm magnitude. Inspired by the Bayesian approach, this thesis proposes a back-propagation process based on BCS, which will be presented in Chapter 4.

2.5 Image Restoration

The image restoration process aims to recover a clear image from its corrupted version, including noise, rotation and occlusion. It is known as an ill-posed inverse problem [93] and plenty of searches have been conducted in the literature. The majority of works are focused on single image de-noising and single image super-resolution. For example, total variation [94] and BM3D algorithm [95] achieved good performance in single image de-noising. [96–99] achieved a state-of-the-art performance on single image super-resolution [100]. However, these methods are only applicable for a particular image degradation. For example, BM3D is designed only for im-

age de-noising. Deep learning based methods extract image features from groups of images and hence can be used for image de-noising and image super-resolution. In fact, deep neural network based methods take the advantage of the big data and the automatic learning scheme, and outperform the traditional image restoration methods [93, 100]. Recently, the deep neural networks have been developed in order to solve different image restoration problems. For example, Mao et al. [93] proposed a CNN architecture that could perform image de-nosing and image super-resolution. The automatic learned restoration frameworks are more promising as they did not have any assumption about the degradation of the image, hence they are purely data driven [93]. Figure 2.7 shows the general structure of applying CNNs to image restoration similarly to [93].

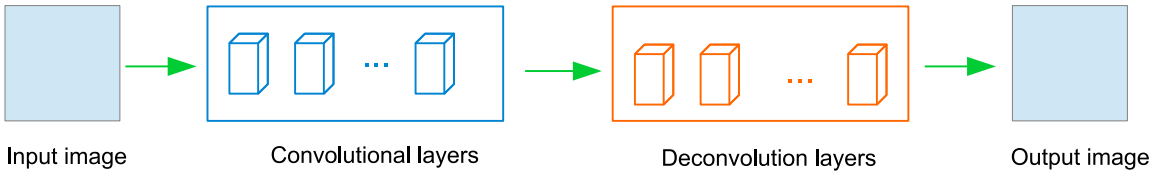


Figure 2.7: A typical restoration architecture based on CNNs.

The restoration framework is based on the same convolutional operations as in CNNs. The main difference lies in there being no pooling operation and fully connected layers for restoration tasks. CNNs for recognition and restoration are different because recognition discards information layer by layer and ends up with a representative feature vector (neurons in the last fully connected layer). In other words, the CNNs for classification discard information step by step in order to extract the most representative information of an image, while CNNs for restoration need to keep detailed information of the input image; hence, the pooling process is not appropriate.

Image degradations also include image occlusion and image rotation. These problems are more challenging than image de-noising and image super-resolution, however, these problems are not well studied using state-of-the-art deep learning methods. This motivates the author to investigate the advanced image restoration methods dealing with image rotation and occlusion. In addition, it will be ben-

eficial if the recognition and restoration could share a joint network, which could perform both tasks at the same time, rather than in a pipeline manner (restoration followed by recognition). The author investigates these problems and develops joint frameworks for image recognition and image restoration. The joint frameworks could simultaneously perform image recognition and restoration with image degradations such as noise, rotation and occlusion. The related work will be presented in Chapter 5.

2.6 Summary

This chapter first introduces traditional recognition frameworks, which comprise image feature methods and classification methods. Among image features, local feature methods such as SIFT features are more robust to image degradations than global feature methods such as HOG features. For the classifier, parametric and non-parametric classifiers are introduced with examples of SVM and KNN. The state-of-the-art recognition methods are shifting from traditional methods to the more advanced deep learning based methods such as CNNs. CNNs take advantage of the automatically learned features on big datasets and perform better than traditional methods in many application fields. However, CNNs contain many parameters that need big data and high computational costs to support. Considering the limitations of the CNNs, an online recognition for VLR is introduced, which will be extended to the proposed online Cauchy prior LR in Chapter 3. Considering the drawbacks for the non-parametric classifier such as KNN, this thesis develops a non-parametric classifier and this will be presented in Chapter 4. CNNs have also been applied for image restoration and classification separately in the literature, with image restoration tasks particularity focused on image de-noising and image super-resolution. Hence, Chapter 5 considers image degradation including rotation and occlusion, and develops deep learning based joint frameworks in order to simultaneously perform image recognition and restoration tasks.

Chapter 3

Online Learning for Vehicle Logo Recognition

3.1 Introduction

Existing VLR frameworks train models on large fixed image training datasets. In practice, there may only be an initially small training dataset, with additional images becoming available during the implementation of the recognition scheme. In order to take advantage of these additional images, new models can be built independently when more images become available. However, retraining new models increases the computational costs, especially when new models are updated frequently. In order to deal with this problem, this chapter proposes a novel online framework for model learning, in which models are rebuilt efficiently using a weight updating scheme when dealing with datasets of an increasing size.

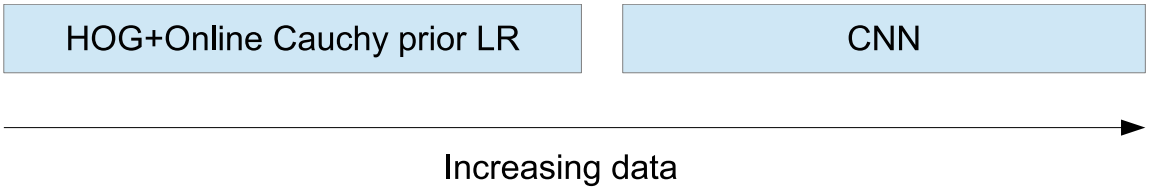


Figure 3.1: The developed framework of VLR with increasing size of dataset.

The general solution is illustrated in Figure 3.1. When the training dataset is

small, the HOG features are applied for feature extraction and the online multinomial Cauchy prior LR classifier is applied for online model updating. The HOG algorithm is applied because of its efficiency. More importantly, it always gives the same feature vector for an image in any training stages. On the contrary, local feature methods involve a dynamic dictionary generation process, which results in different representation vectors for an image in different training stages. For example, using SIFT features with the BOW representation model, an image is represented by two different vectors when the training dataset contains 200 images and 500 images, respectively. In the parametric classification stage, weights are associated with the input vector. Therefore, if an image is represented by irrelevant vectors in different training stages, the corresponding weights will not be relevant. Hence, local features cannot be applied to the online weights updating scheme. Unlike local features requiring a representation model before the classification stage, the HOG algorithm does not need this process and it will always give the same vector despite different training scenarios. When large size datasets are available and high computational costs are acceptable, the CNN classifier can be applied in order to increase the robustness to noise and further improve the accuracy. Unlike hand-crafted features, which use fixed rules, features in CNNs automatically update according to more incoming data; hence, it could be more representative when there are more training images fed in.

For the choice of the classifier, LR can be easily extended for online model updating and it explores the confidence level of the decision that the data has been correctly classified [33, 101]. However, when all training data can be perfectly classified, the LR suffers a common problem called separation, in which the maximum likelihood gives implausible estimates [102]. In order to have a generalised LR classifier without the separation problem, Gelman et al. [103] suggested a default Cauchy prior for LR and the posterior can be computed using Gibbs sampling, which involves high computational costs. This work combines the Conjugate Gradient Descent (CGD) with

LR for both online and offline classification. In section 3.2, the Cauchy prior logistic regression is introduced. Section 3.3 introduces the CGD and Section 3.4 explains the online weight updating scheme. The developed CNNs in Section 3.5 are applied when the large datasets become available. The performance and the summary of this chapter are presented in Section 3.6 and Section 3.7.

3.2 Cauchy Prior Logistic Regression

Given a training data $(\mathbf{x}, y)$, where $\mathbf{x} \in \mathbb{R}^{M\times 1}$, in linear regression, the linear function is applied:

$$y = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b, \tag{3.1}$$

where $\mathbf{w} \in \mathbb{R}^{M\times 1}$ is the weight vector and the scalar b is the bias associated with the linear regression. For the binary classification where y is a label variable that can either be '1' (positive) or '0' (negative). Using a 'logistic' function $f(x) = 1/(1+e^{-x})$, the probability $p(y = 1|\mathbf{x}, \mathbf{w}, b)$ that the training data belongs to class '1' can be expressed by:

$$p(y = 1|\mathbf{x}, \mathbf{w}, b) = s = f(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{w}^{\mathrm{T}}\mathbf{x}+b)}}. \tag{3.2}$$

Therefore, the probability of a negative outcome is $1 - s$:

$$p(y = 0|\mathbf{x}, \mathbf{w}, b) = 1 - s = \frac{e^{-(\mathbf{w}^{\mathrm{T}}\mathbf{x}+b)}}{1 + e^{-(\mathbf{w}^{\mathrm{T}}\mathbf{x}+b)}}. \tag{3.3}$$

Assuming that there are N independent training data $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$, a Bernoulli distribution can be used to form the likelihood function for the i^{th} data by combining equation (3.2) and equation (3.3), which gives:

$$p(y_i|\mathbf{x}_i, \mathbf{w}, b) = s_i^{y_i}(1 - s_i)^{1-y_i}, \tag{3.4}$$

where s_i represents the probability that the i^{th} data belongs to the positive class. The likelihood of all the training data is therefore given by the product:

$$p(\mathbf{y}|\mathbf{w}, \mathbf{X}, b) = \prod_{i=1}^{N} s_i^{y_i}(1 - s_i)^{1-y_i}, \tag{3.5}$$

where $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]$ is the training dataset and $\mathbf{y} \in \mathbb{R}^{N\times 1}$ is a vector representing all the training labels. Maximising the likelihood in equation (3.5) is equivalent to minimising the negative of its natural logarithm likelihood, i.e.

$$\begin{aligned}
\mathbb{L} &= -ln\left(p(\mathbf{y}|\mathbf{w}, \mathbf{X}, b)\right) \\
&= -\sum_{i=1}^{N} y_i ln(s_i) - \sum_{i=1}^{N}(1 - y_i)ln(1 - s_i) \\
&= -\sum_{i=1}^{N} y_i ln\left(f(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b)\right) - \sum_{i=1}^{N}(1 - y_i)ln(1 - f(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b)),
\end{aligned} \tag{3.6}$$

where $ln(\cdot)$ denotes the natural logarithm.

Before going any further, it is worth to mention that the 'logistic' function has a useful property that its derivation is a function of itself:

$$\begin{aligned}
f'(x) &= \frac{d}{dx}\left(\frac{1}{1 + e^{-x}}\right) \\
&= \frac{1}{(1 + e^{-x})^2}(e^{-x}) \\
&= \frac{1}{1 + e^{-x}} \cdot \left(1 - \frac{1}{1 + e^{-x}}\right) \\
&= f(x)(1 - f(x)).
\end{aligned} \tag{3.7}$$

In order to minimise equation (3.6), the partial derivative with respect to $\mathbf{w}$ and b can be used. Combining equation (3.7) gives:

$$\frac{\partial \mathbb{L}}{\partial \mathbf{w}} = -\sum_{i=1}^{N} \frac{y_i}{f(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b)} f'\boldsymbol{x}_i + \sum_{i=1}^{N} \frac{1 - y_i}{1 - f(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b)} f'\boldsymbol{x}_i$$

$$= -\sum_{i=1}^{N} y_i(1 - f(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b))\mathbf{x}_i + \sum_{i=1}^{N}(1 - y_i)f(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b)\mathbf{x}_i$$
$$= \sum_{i=1}^{N}(f(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b) - y_i)\mathbf{x}_i, \tag{3.8}$$

here f' represents the partial derivative of $f(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b)$ with respect to $\mathbf{w}$. In the same way take the partial derivative with respect to b:

$$\frac{\partial \mathbb{L}}{\partial b} = \sum_{i=1}^{N}(f(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b) - y_i). \tag{3.9}$$

Notice that the LR is a maximum likelihood model that does not involve any prior information. However, when the maximum likelihood perfectly separates the training dataset, there are infinite possible solutions caused by the separation problem.

A Cauchy prior on LR can avoid the separation problem. It assumes that the coefficients in LR are sparse, this could provide a quicker convergence in the gradient descent process. A zero mean Cauchy prior is assumed for the weights, this gives:

$$p(\mathbf{w}) = \frac{1}{\pi}\left(\frac{\gamma}{\mathbf{w}^2 + \gamma^2}\right), \tag{3.10}$$

where γ is a scale parameter. According to the Bayes rule:

$$p(\mathbf{w}, b|\mathbf{y}) = \frac{(\mathbf{y}|\mathbf{w}, b)p(\mathbf{w}|b)p(b)}{p(\mathbf{y})}. \tag{3.11}$$

Since $\mathbf{w}$ and b are independent, it gives:

$$p(\mathbf{w}, b|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{w}, b)p(\mathbf{w})p(b). \tag{3.12}$$

The weights are assumed sparse which makes the majority of the weights zero (or close to zero) valued. However, b is the intercept of the decision line which does not have any prior knowledge associated with it. As a result, here assume b is controlled by a non-informative prior. Therefore, maximising the posterior $p(\mathbf{w}, b|\mathbf{y})$

is equivalent to maximising:

$$p(\mathbf{y}|\mathbf{w},b)p(\mathbf{w}) = \frac{1}{\pi}\left(\frac{\gamma}{\mathbf{w}^2+\gamma^2}\right)\prod_{i=1}^{N} s_i^{y_i}(1-s_i)^{1-y_i}. \tag{3.13}$$

Maximising the likelihood in (3.13) is equivalent to minimising the negative of its natural logarithm, which is given by:

$$\begin{aligned}
\mathbb{L} &= -ln\,(\ p(\mathbf{y}|\mathbf{w},b)p(\mathbf{w})) \\
&= -\sum_{i=1}^{N} y_i ln(s_i) - \sum_{i=1}^{N}(1-y_i)ln(1-s_i) - ln(\gamma) + ln((\mathbf{w}^{\mathrm{T}}\mathbf{w}+\gamma^2)\pi) \\
&= -\sum_{i=1}^{N} y_i ln\left(f(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)\right) + ln\left((\mathbf{w}^{\mathrm{T}}\mathbf{w}+\gamma^2)\pi\right) \\
&\qquad\qquad - \sum_{i=1}^{N}(1-y_i)ln\left(1-f(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)\right) - ln(\gamma).
\end{aligned} \tag{3.14}$$

In order to minimise (3.14), taking the partial derivative with respect to $\mathbf{w}$ gives:

$$\begin{aligned}
\frac{\partial \mathbb{L}}{\partial \mathbf{w}} &= -\sum_{i=1}^{N}\frac{y_i}{f(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)}f'(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)\mathbf{x}_i + \frac{2\mathbf{w}}{\mathbf{w}^{\mathrm{T}}\mathbf{w}+\gamma^2} \\
&\qquad\qquad + \sum_{i=1}^{N}\frac{(1-y_i)}{1-f(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)}f'(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)\mathbf{x}_i \\
&= -\sum_{i=1}^{N} y_i(1-f(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b))\mathbf{x}_i + \frac{2\mathbf{w}}{\mathbf{w}^{\mathrm{T}}\mathbf{w}+\gamma^2} + \sum_{i=1}^{N}(1-y_i)(f(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b))\mathbf{x}_i \\
&= \frac{2\mathbf{w}}{\mathbf{w}^{\mathrm{T}}\mathbf{w}+\gamma^2} + \sum_{i=1}^{N}(f(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)-y_i)\mathbf{x}_i.
\end{aligned} \tag{3.15}$$

In the same way taking the partial derivative with respect to b gives:

$$\begin{aligned}
\frac{\partial \mathbb{L}}{\partial b} &= -\sum_{i=1}^{N}\frac{y_i}{f(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)}f'(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b) + \sum_{i=1}^{N}\frac{(1-y_i)}{1-f(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)}f'(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b) \\
&= -\sum_{i=1}^{N} y_i(1-f(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)) + \sum_{i=1}^{N}(1-y_i)(f(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)) \\
&= \sum_{i=1}^{N}(f(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)-y_i).
\end{aligned} \tag{3.16}$$

For a new testing image $\mathbf{x}^*$, the probability that it belongs to the positive class is:

$$p(y^* = 1|\mathbf{w}, b) = \frac{1}{1 + e^{-(\mathbf{w}^\mathrm{T}\mathbf{x}^* + b)}} \tag{3.17}$$

and the probability that it belongs to the negative class is therefore:

$$p(y^* = 0|\mathbf{w}, b) = 1 - p(y^* = 1|\mathbf{w}, b). \tag{3.18}$$

Here, y^* represents the predicted label for a testing image. Hence, the testing image can be allocated into the class that has the higher probability.

The Cauchy prior LR in binary classification can be easily extended to multinomial classification. Given the training data from C categories $y_i \in \{1, 2, \cdots, C\}$, the probability of $p(y_i = c|\mathbf{W}, \mathbf{b})$ for each $c = (1, 2, \cdots, C)$ can be denoted as:

$$\begin{bmatrix} p(y_i=1|\mathbf{W},\mathbf{b}) \\ p(y_i=2|\mathbf{W},\mathbf{b}) \\ \vdots \\ p(y_i=C|\mathbf{W},\mathbf{b}) \end{bmatrix} = \frac{1}{\sum_{c=1}^{C} e^{(\mathbf{w}_c^\mathrm{T}\mathbf{x}_i + b_c)}} \begin{bmatrix} e^{(\mathbf{w}_1^\mathrm{T}\mathbf{x}_i + b_1)} \\ e^{(\mathbf{w}_2^\mathrm{T}\mathbf{x}_i + b_2)} \\ \vdots \\ e^{(\mathbf{w}_C^\mathrm{T}\mathbf{x}_i + b_C)} \end{bmatrix}, \tag{3.19}$$

where $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_C]$ is a matrix consisting of the weights and $\mathbf{b} = [b_1, b_2, \cdots, b_C]$ is the bias of the multi-class LR models. The term $\sum_{c=1}^{C} e^{(\mathbf{w}_c^\mathrm{T}\mathbf{x}_i + b_c)}$ normalises the distribution so that all of the probabilities sum up to one. Hence, for a testing image $\mathbf{x}^*$, the probability that its label y^* equals c is :

$$p(y^* = c|\mathbf{W}, \mathbf{b}) = \frac{e^{(\mathbf{w}_c^\mathrm{T}\mathbf{x}^* + b_c)}}{\sum_{c=1}^{C} e^{(\mathbf{w}_c^\mathrm{T}\mathbf{x}^* + b_c)}}. \tag{3.20}$$

The incoming testing image is then assigned to the class that has the highest probability.

3.3 Conjugate Gradient Descent

Carpenter [104] proposed using the Stochastic Gradient Descent (SGD) to solve this problem and the Cauchy gradient is derived without considering a bias term in LR. However, the key disadvantage of SGD is that it requires manual tuning of parameters such as learning rates and stopping criteria [105]. Meanwhile, the CGD [106] automatically chooses a learning rate that could avoid this problem [105].

Equation (3.15) and equation (3.16) are optimisation problems that can be solved using gradient descent which updates the weights iteratively:

$$\mathbf{w}^{(j+1)} = \mathbf{w}^{(j)} - \eta \frac{\partial \mathbb{L}}{\partial \mathbf{w}^{(j)}}, \tag{3.21}$$

$$b^{(j+1)} = b^{(j)} - \eta \frac{\partial \mathbb{L}}{\partial b^{(j)}}, \tag{3.22}$$

where η is a fixed learning rate that controls the speed of convergence and $j \in \{1, \cdots, J\}$ is the iteration index.

One key disadvantage of gradient descent methods, such as batch gradient descent and SGD is that a good learning rate is difficult to find [105]. In order to avoid this problem, this chapter proposes using the Cauchy prior LR with CGD, which automatically chooses a learning rate in each iteration.

Denote all the variables as $\mathbf{v} = (b, \mathbf{w}^{\mathrm{T}})$. Therefore $\mathbb{L}$ in equation (3.14) can be written as a function of $\mathbf{v}$, giving $\mathbb{L} = g(\mathbf{v})$. For the first iteration, the gradient update for all the variables is:

$$\mathbf{v}_2 = \mathbf{v}_1 - \eta_1 \frac{\partial \mathbb{L}}{\partial \mathbf{v}_1}, \tag{3.23}$$

where $\mathbf{v}_1$ represents the initial bias and weights (initialised as zero values) when $j = 1$. A line search is applied to find the initial learning rate [106]:

$$\eta_1 = \underset{\eta}{\operatorname{argmin}}\, g\left(\mathbf{v}_1 - \eta \frac{\partial \mathbb{L}}{\partial \mathbf{v}_1}\right). \tag{3.24}$$

For the following iterations where $j > 1$, gradients are along the conjugate directions. In order to avoid a zig-zagging path, the new gradient direction combines the gradient $-\frac{\partial \mathbb{L}}{\partial \mathbf{v}_j}$ and the previous direction:

$$\mathbf{d}_{j+1} = -\eta_j \frac{\partial \mathbb{L}}{\partial \mathbf{v}_j} + \beta_j \mathbf{d}_j, \tag{3.25}$$

with $\mathbf{d}_1 = -\frac{\partial \mathbb{L}}{\partial \mathbf{v}_1}$. According to the Polak-Ribiere rule [107], the value of β_j is given by:

$$\beta_j = \frac{\left(\frac{\partial \mathbb{L}}{\partial \mathbf{v}_j}\right)^{\mathsf{T}} \left(\frac{\partial \mathbb{L}}{\partial \mathbf{v}_j} - \frac{\partial \mathbb{L}}{\partial \mathbf{v}_{j-1}}\right)}{\left(\frac{\partial \mathbb{L}}{\partial \mathbf{v}_j}\right)^{\mathsf{T}} \frac{\partial \mathbb{L}}{\partial \mathbf{v}_j}}. \tag{3.26}$$

The gradient update process is:

$$\mathbf{v}_{j+1} = \mathbf{v}_j + \eta_j \mathbf{d}_j, \tag{3.27}$$

and a line search process is applied to find the optimal learning rate:

$$\eta_j = \underset{\eta}{\operatorname{argmin}}\, g\left(\mathbf{v}_j + \eta \mathbf{d}_j\right). \tag{3.28}$$

3.4 Online Weight Updating

In order to deal with training datasets that increase in size, the classifier needs to be retrained as more training images become available. However, rather than retraining different classifiers independently, the weights trained for the previous stages can be useful. Figure 3.2 shows the general process of retraining models when the size of training images is increasing.

Using the HOG features, each image is represented by a vector $\mathbf{x}$ and its label y. Algorithm 3 shows the offline method, which retrains a new model independently

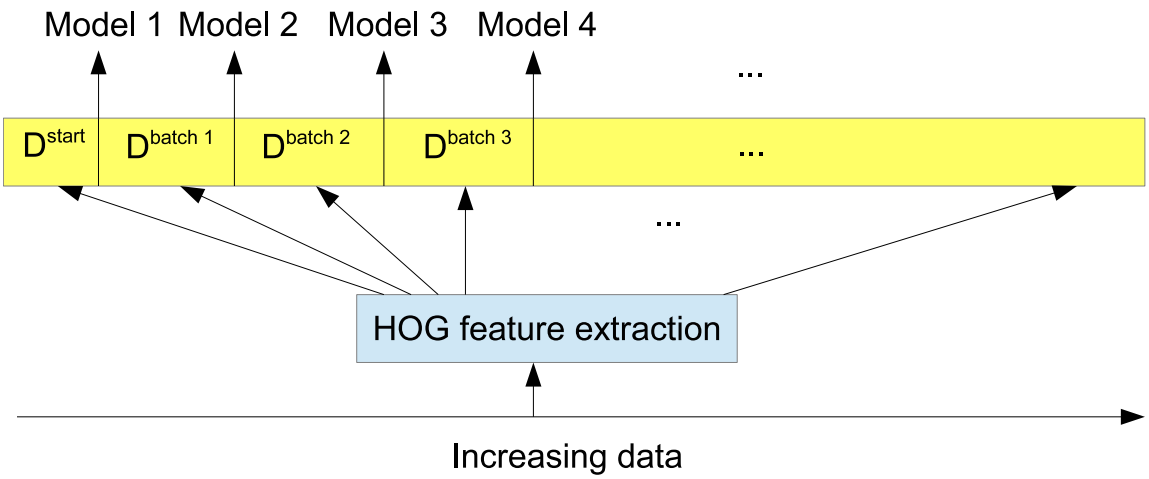


Figure 3.2: The developed online recognition framework for VLR.

when additional training data become available. More specifically, the initial model was trained using a small amount of training data $\mathbf{D}^{start}$. When there are extra training data available, $\mathbf{D}^{batch}$, the model is retrained using all the available data $\mathbf{D}^{ava}$. This now includes both the additional data and the previously available data. This process is repeated each time when additional data become available. The batch size is a parameter that controls how often the model is updated, i.e. the number of additional data required before retraining occurs.

Using the offline methods, models are retrained independently as $\mathbf{W}$ and $\mathbf{b}$ are initialised to either random values or zeros. However, $\mathbf{W}$ and $\mathbf{b}$ from the previous models might be good initial values which could help the current model converge faster. Therefore, the current model can be updated based on a previously trained model rather than a model retrained independently. Algorithm 4 shows the general process of online model updating.

Algorithm 3 Framework of offline Cauchy prior logistic regression

Input:

The initial training data $\mathbf{D}^{start}$

The sequential training data, $\mathbf{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots (\mathbf{x}_i, y_i), \cdots\}$, with i is the index of the i^{th} image and label

Output:

The model parameters in Cauchy prior LR and accuracies on the testing dataset

1: Apply the Cauchy prior LR on the initial training data $\mathbf{D}^{start}$ and save the initial model (Model 1 in Figure 3.2)
2: **for** each $i = 1, i++$ **do**
3: **if** i/(batch size)==int **then**
4: Retrain a new model using the all available training data $\mathbf{D}^{ava} = \{(\mathbf{x}_1, y), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_i, y_i)\}$ with Cauchy prior LR
5: **end if**
6: Use the retrained model to classify the testing images
7: **end for**
8: **return** The model parameters and accuracies

3.5 Convolutional Neural Networks for Online Learning

The HOG features have an advantage and limitation. The advantage is that the HOG is easy and fast to implement while the disadvantage lies in the accuracy bottleneck. The HOG features could not make full use of a big dataset; the accuracy saturates at some point. In contrast, CNNs need a big dataset to support. In order to take advantage of both, HOG features and CNNs can be combined to deal with different scenarios.

Figure 3.3 shows the developed CNN architecture for VLR. It contains three convolutional layers, three pooling layers, and two fully connected layer. In the first convolutional stage, six kernels with a size of $[5 \times 5]$ ([width $\times$ height]) are convolved with an input image in order to generate six convolutional maps in $C1$. A Max-pooling process with $[2 \times 2]$ is applied to 6 convolutional maps; therefore six pooling

Algorithm 4 Framework of online Cauchy prior logistic regression

Input:

The initial training data $\mathbf{D}^{start}$

The sequential training data, $\mathbf{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_i, y_i), \cdots\}$, with i is the index of the i^{th} image and label

Output:

The model parameters in Cauchy prior LR and accuracies on the testing dataset

1: Apply the Cauchy prior LR on the initial training data $\mathbf{D}^{start}$ and save the initial model (Model 1 in Figure 3.2)
2: **for** each $i = 1, i++$ **do**
3: **if** i/(batch size)==int **then**
4: Update the model using all available training data $\mathbf{D}^{ava} = \{(\mathbf{x}_1, y), (\mathbf{x}_2, y_2), \cdots (\mathbf{x}_i, y_i)\}$ with the previous $\mathbf{W}$ and b are used as the initial start values of the model parameters.
5: **end if**
6: Use the updated model to classify the testing dataset
7: **end for**
8: **return** The model parameters and accuracies

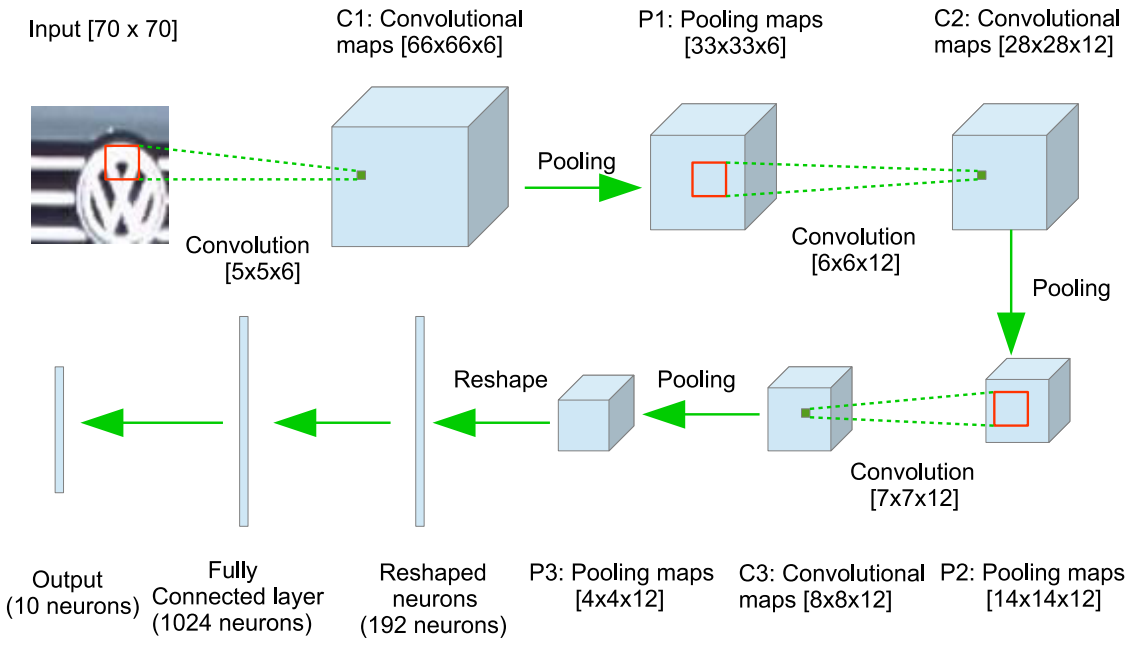


Figure 3.3: The designed CNN architecture.

maps are generated in $P1$. The same process is repeated in $C2$ with 12 kernels of size $[6 \times 6]$ and in $C3$ with 12 kernels of size $[7 \times 7]$, respectively. The same Max-pooling is

applied in $P2$ and $P3$ with a non-overlapping window of size $[2 \times 2]$. The 12 pooling maps with the size of $[4 \times 4]$ in $P3$ are then reshaped into a vector form, which is further fully connected with the 1024 neurons. These neurons are then connected to a softmax output layer. A dropout process [49] is applied to the fully connected layer with a dropout rate of 0.5.

3.6 Performance Evaluation

In this section, the proposed framework is evaluated on the open dataset provided by Huang et al. [50] in order to evaluate the performance. This dataset is currently the biggest available vehicle logo dataset; it has ten categories and each category contains 1000 training images and 150 testing images. All images have a size of $[70 \times 70]$ pixels. Figure 3.4 shows an example of these ten vehicle categories by randomly choosing one image from each category in the training dataset and Figure 3.5 shows some challenging testing images, which can be easily mis-classified.

Figure 3.4: Image examples of the vehicle logo dataset.

The performance evaluation of the online Cauchy LR with HOG features is conducted in MATLAB 2015 on a computer with the following specification: Intel CPU I5-4590 (3.4Ghz) and 24GB of RAM. The proposed Cauchy prior LR is compared with LR and the Cauchy prior LR is evaluated for training datasets that increase in size. The performance of each method is measured in terms of accuracy (percentage

Figure 3.5: Examples of some challenge images in the testing dataset.

of correctly classified images among the entire testing dataset), total number of misclassified images and the computation time (to indicate the relative computational complexities). Accuracies and computational times are given as average values taken from 30 simulation runs.

In the implementation, the HOG features are different from the original HOG method as implemented in [7]. Here, a histogram vector is built for each block rather than each cell and 12 bins with uniform spacings are applied on the angular range from 0° to 180°. The block window shifts on the whole image taking the size of a cell as the sliding size. A block window is made up by 2 × 2 cells and each cell is made from [5×5] pixels. These techniques give an improvement of accuracy more than 3% when the model is trained on the whole training dataset (from 93.53% to 97.13% when LR with CGD is applied). Each HOG feature vector is normalised with zero mean and the standard deviation is set to 1. This process is able to increase accuracy about 2% (from 97.13% to 98.80% when LR with CGD is applied).

Finding the learning rate is a difficult issue in SGD. Using the whole training dataset with the testing dataset as the validation data, the best accuracy SCD (95.35%) achieved is about 3% lower when compared with CGD (98.80%). However, when applied in practice, the testing dataset is not known in advance. As a result, it is not possible to find the optimal learning rate for use in classification. This means a further degradation in performance would be expected for methods based on SGD. In the following, the optimised HOG features with normalisation and CGD are applied in order to compare LR and Cauchy prior LR.

3.6.1 Comparisons of the Logistic Regression with the Cauchy Prior Logistic Regression

Table 3.1: Performance comparisons between LR and Cauchy prior LR when dataset size is increasing (average values from 30 simulation runs).

Training size	100	500	1000	2000	3000	4000
LR (%)	67.05	91.02	96.07	98.11	98.56	98.67
Misclassified images	494.24	135.70	58.95	28.35	21.60	19.95
Time (s)	4	23	49	94	135	179
Cauchy LR (%)	66.24	90.35	95.33	97.59	98.06	98.35
Misclassified images	506.40	144.75	70.05	36.15	29.10	24.75
Time (s)	2	8	17	36	54	74
Training size	5000	6000	7000	8000	9000	10000
LR (%)	98.72	98.84	98.83	98.76	98.72	98.80
Misclassified	19.20	17.40	17.55	18.60	19.20	18
Time (s)	216	261	302	339	386	**437**
Cauchy LR (%)	98.38	98.42	98.42	98.51	98.48	98.80
Misclassified images	24.30	23.70	23.70	22.35	22.80	18
Time (s)	92	119	149	175	180	**182**

Different training dataset sizes are tested and accuracies are evaluated on the complete testing dataset. Results are given in Table 3.1 and Figure 3.6. The accuracies of both classifiers are close while the Cauchy prior LR has a significant reduction on computational costs. Taking the training size that equals 10000 as an example, the Cauchy prior is able to decrease the computational time from seven minutes (437 seconds) to approximately three minutes (182 seconds) when the whole dataset is applied, i.e. a 58% reduction in computational time. This can be explained by the prior information resulting in a quicker convergence. As a result, only LR with the Cauchy prior will be considered in the remaining comparisons of online and offline training.

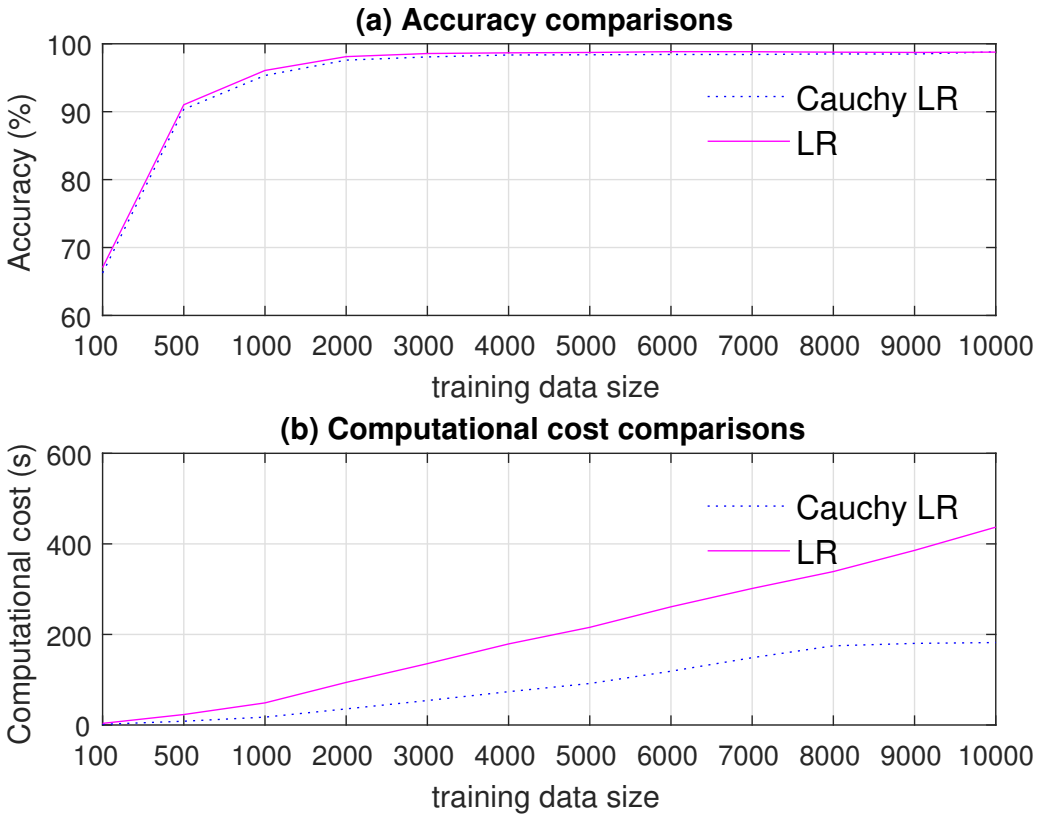


Figure 3.6: Accuracy (a) and computational costs (b) comparisons between LR and Cauchy prior LR when the dataset size is increasing (average values of 30 simulation runs).

3.6.2 Comparisons of Online and Offline Cauchy Prior Logistic Regression

Figure 3.7 shows the performances of the online method and the offline method using the Cauchy prior LR. A random set of training data for each class is picked and used for the initial classifier training (the same initial set for each method). When the training size is increasing, training models are updated when the available number of training data meets the requirement described in Algorithm 3. Here the batch size $\mathbf{D}^{batch}$=100 is used. The offline method means the weights are retrained on the available images, with all weights initialised to 0, while the online method involves a weight initialisation from the previous models. For Cauchy prior LR, different γ have been tested. It is found that when γ is larger than 15, the accuracy becomes

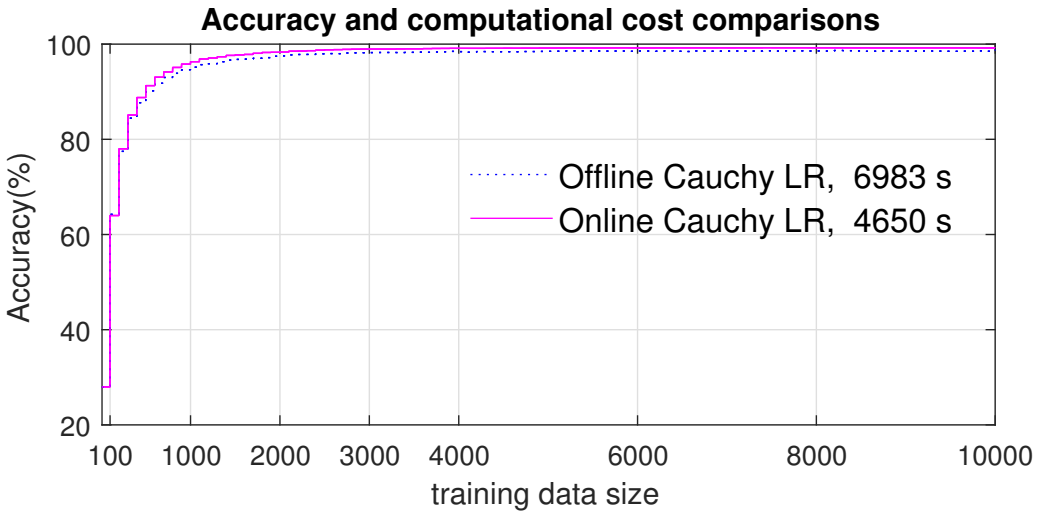


Figure 3.7: Accuracy and computational costs comparisons between offline and online Cauchy prior LR up to 10000 training images ($\mathbf{D}^{batch} = 100$).

saturate. In the following experiment, γ=30 is used for Cauchy prior LR.

It is shown in Figure 3.7 that the HOG features can achieve a good accuracy when there is a small dataset (90% accuracy is achieved when the training size is around 500). Once the training size is above 2000, both accuracies become high and stable. The time in seconds shows the computation time for the whole process, which includes the testing scenario and the model updating process. The online scheme reduced the computational time by 33%, which indicates that the weights initialisation can help with the convergence in CGD.

Figure 3.7 indicates the accuracies become stable when the training size is above 2000. Therefore a more detailed comparison can be made by using a smaller dataset while updating the model more frequently because a smaller batch size gives more comparison results. Figure 3.8 shows more detailed results by setting $\mathbf{D}^{batch}$ =20 and the training size varies from 100 to 3000. It indicates that the online method provides a quicker convergence speed.

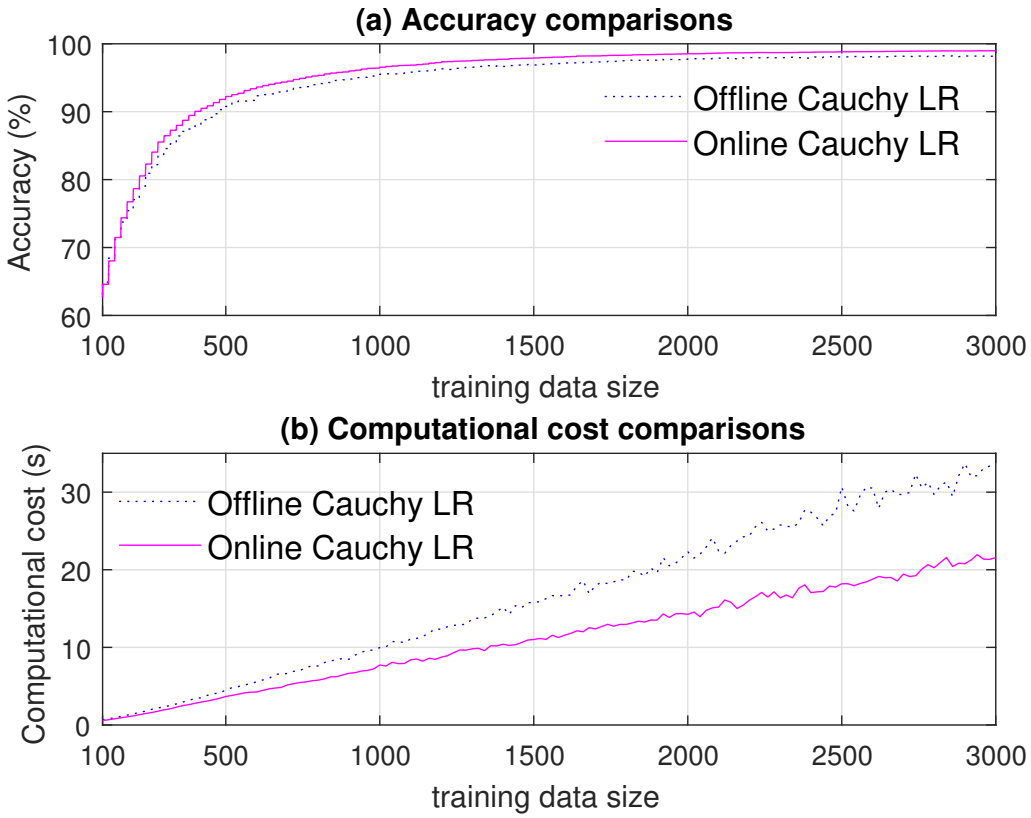


Figure 3.8: Accuracy (a) and computational costs (b) comparisons between offline and online Cauchy prior LR up to 3000 random training images with ($\mathbf{D}^{batch}$ =20).

3.6.3 Convolutional Neural Networks for Large Datasets with the Noise Robustness Evaluation

Unlike the HOG features, which can achieve a good result using a small dataset, CNNs need a large dataset to automatically learn good features, otherwise the over-fitting problems can be serious. Over-fitting is a common problem; it means the model fits too well on the training dataset but has a bad generalisation ability on the testing dataset. The experiment in this subsection considers different dataset sizes, from small to big, with an equal number of training images in each class. The experiment for CNNs was conducted on a laptop with the following specification: Intel CPU I5 and Nvidia GTX 1070 (extended GPU); the Tensorflow toolbox is used and the accuracy is evaluated once the weights are learned, after 100 training epochs.

Table 3.2 shows that when the training dataset is below 2000, HOG features combined with the Cauchy prior LR achieve a higher accuracy than CNNs. With an increasing size of the training data, the CNNs outperform the HOG features in terms of accuracy. The highest accuracy achieved by CNNs is 99.87%; this means only two testing images were wrongly classified among 1500 testing images. The CNNs need high computational costs; taking the whole training dataset as an example, the CNNs take about five hours running on an Nvidia GTX 1070 GPU using Tensorflow, while HOG features combined with Cauchy LR only need about 13 minutes. The improvement in terms of accuracy seems trivial (from 98.80% to 99.87%) when compared with the computational costs invested in CNNs. This is due to the images in the dataset being clear and the HOG features therefore can be representative.

Table 3.2: Accuracy comparisons between Cauchy prior LR and CNNs when dataset size is increasing.

Training size	100	500	1000	2000	3000	4000
Cauchy LR (%)	**70.33**	**90.47**	**95.07**	**96.87**	97.73	97.73
CNNs (%)	29.6	58.2	72.40	89.00	**98.13**	**98.47**
Training size	5000	6000	7000	8000	9000	10000
Cauchy LR (%)	98.20	98.47	98.73	98.80	98.33	98.80
CNNs (%)	**99.53**	**99.67**	**99.60**	**99.67**	**99.60**	**99.87**

In practice, it is unlikely that the logos being classified will be clearly visible. Figure 3.9 shows an example of three clear testing images and the effects of adding noise with increasing variances. The intensity values of all pixels are normalised, giving values between 0 and 1. Zero mean white Gaussian noise is then added to each pixel, with the effects of noise variances of 0.1, 0.2 and 0.3. As shown in Figure 3.9, an image is highly contaminated if the variance of the Gaussian noise is above 0.2, and even difficult to distinguish by human vision when the noise variance has been increased to 0.3.

In this simulation, image from both training and testing images are added with zero mean Gaussian white noise, with the noise variance σ^2 being a random value

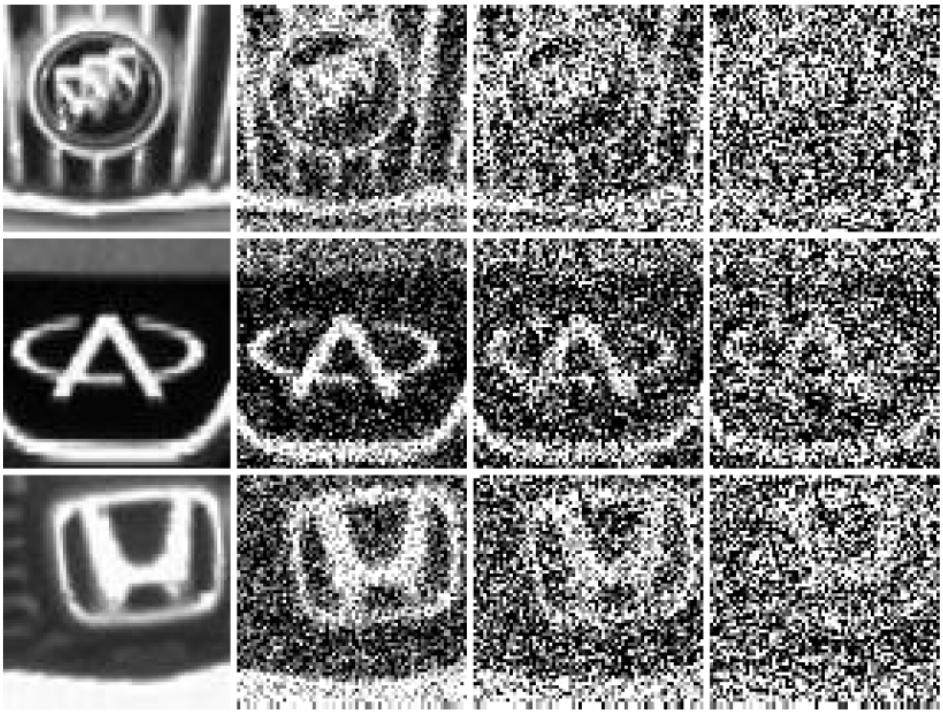


Figure 3.9: An example of three testing images (the first column) and the effects by adding Gaussian white noise to image intensities with zero mean and variances of 0.1, 0.2, 0.3 from left to right respectively.

between 0 and σ^2_{max}, where $\sigma^2_{max} = \{0.1, 0.2, 0.3\}$.

Table 3.3: Comparisons between Cauchy prior LR with CNNs when training and testing images are noisy.

σ^2_{max}	0.1	0.2	0.3
HOG	79.47	68.20	62.20
CNNs	**98.80**	**97**	**93.40**

Table 3.3 indicates that HOG features are sensitive to noise, while CNNs are robust to noise. For example, when random zero-mean Gaussian noises are generated with the variance within 0 to 0.2, the HOG features could only achieve an accuracy of 68.20% while CNNs achieve an accuracy of 97%. As aforementioned the 1% of accuracy improvement on clear images seems trivial; the robustness to noise suggests CNNs can be applied when HOG features fail.

3.7 Summary

State-of-the-art VLR approaches typically consider training models on large datasets. However, there might only be a small training dataset to start with and more data can be obtained during the real-time applications. This chapter proposes an online image recognition framework which provides solutions for both small and large datasets. Using this recognition framework, models are built efficiently using a weight updating scheme. The Cauchy prior logistic regression with CGD is proposed to deal with the multinomial classification tasks. The motivation of using a Cauchy prior is due to the sparsity on the weights. The Cauchy prior results in a quick convergence speed for the weight updating process which could decrease the computational costs. By testing with a publicly available dataset, the Cauchy prior LR decreases the classification time by 58%. The Cauchy prior LR has been tested in both offline and online situations, and the online weight updating scheme further decreases the computational costs. The HOG features and Cauchy prior LR framework give an efficient solution to VLR when there are fewer than 300 images in each category. However, when giving more training datasets for training, the HOG features could not improve the accuracy any further. This chapter further develops a CNN framework when the dataset is very large. The CNNs take advantage of the automatic feature extraction scheme based on large training dataset. Hence, it could be more accurate and robust to image challenges such as noise. Notice that CNN frameworks and HOG features with Cauchy prior LR are two independent systems. However, future work could be conducted in order to make smooth transition between the two systems.

Chapter 4

Spatial Invariant Feature Transform and Back-propagation Bayesian Compressive Sensing Classifier

4.1 Introduction

An efficient traditional recognition framework requires both good feature representation and effective classification methods. The reason for investigating the traditional framework lies in the deep learning based methods requiring high computational costs and big datasets, which might not be available in some applications; in contrast traditional frameworks could give good results in some applications. For the image features, this chapter develops such a framework based on spatial SIFT features. The performance of the developed framework is compared to methods based on the HOG features and SIFT features. The spatial information of SIFT features gives more representative vectors than in HOG features and SIFT features.

For the classification, this chapter proposes a novel non-parametric classifier.

State-of-the-art parametric classifiers such as LR and SVM can result in inadequately trained models, leading to inappropriate predictions in the testing phase. Unlike parametric classifiers, the proposed SBCS classifier does not train a model and thus avoids this problem. In addition, the proposed classifier is combined with a column-based subspace sampling process in order to deal with high computational costs situations. By testing with a publicly available vehicle logo dataset, it is shown that the proposed classifier gives accurate results when compared with the state-of-the-art non-parametric classifiers. The SBCS is also tested on other applications such as scene recognition and the CIFAR-10 dataset. The proposed approach outperforms the CNNs in noisy conditions by using only a small fraction of the original dataset.

The rest of this chapter is organised as follows. Section 4.2 introduces how the pyramid idea based on the SIFT feature is applied and how the LR is employed in order to solve the multinomial classification problem. Section 4.3 introduces the proposed SBCS classifier. Evaluations of the spatial SIFT features and the SBCS classifier are presented in Section 4.4 and Section 4.4. Section 4.6 summarises this chapter.

4.2 Spatial Scale Invariant Feature Transform

In the BOW model, the magnitude of each 'word' in the histogram is only decided by its occurring frequency in an image. The information of where the feature originates from does not influence the representation vector. In fact, the geographical information of the interest points is deliberately avoided in order to ensure that interest points at different locations can be matched, making the process invariant to shifts. However, applications such as vehicle logos often occupy the entirety of the training and testing images after a segmentation process. The geographical information might be useful in such a case. For example, the 'V' is always above the 'W' in a detected Volkswagen logo image and using such information can potentially give a more accurate classification.

Lazebnik et al. [108] proposed the idea of partitioning the image into sub-regions and then using the BOW model over each sub-region for natural images. Specifically, the original image is first partitioned into four sub-regions, then into 16 sub-regions in the next level and so on. The BOW model is applied over each sub-region and the final feature is formed by concatenating the histograms from the original image and all the sub-regions. The result is a pyramid-like structure, where each level, as you move down the pyramid, is focused on a smaller region of the image. Each level is often called a pyramid scale.

The pyramid idea has been applied in VLR tasks by using the Dense-SIFT global descriptor [67, 109]. However, the Dense-SIFT feature method regards all pixels in the original map as interest points, which makes the interest points not robust as there is a lack of feature detection process [110]. Instead, this chapter proposes using the pyramid idea with the SIFT descriptor for VLR.

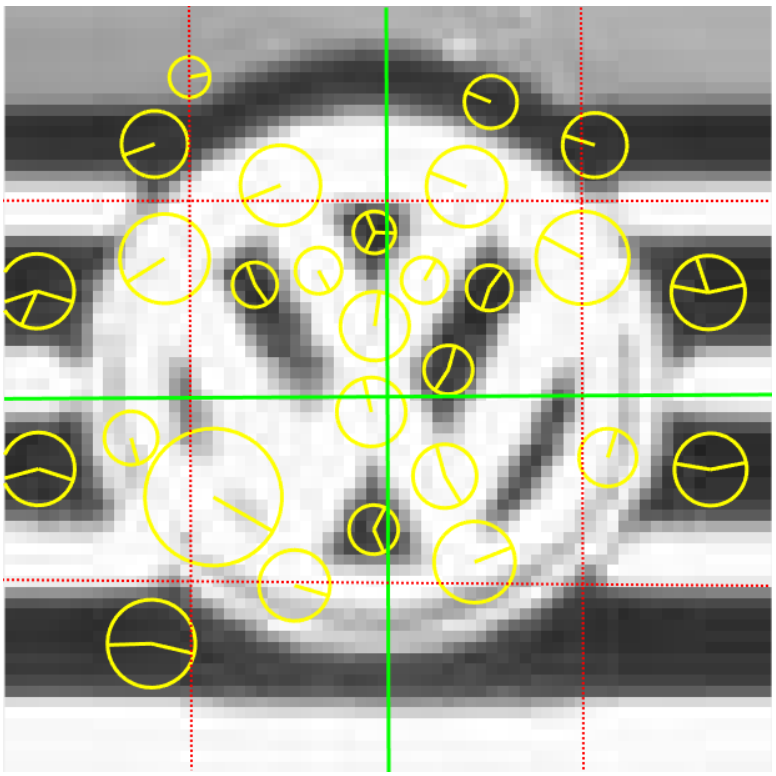


Figure 4.1: An example of spatial pyramid interest points. The centre of the yellow circles are locations of SIFT features in their corresponding maps and the yellow bars represent the main orientations. Since the interest points are from different DOG maps, the size of the yellow circles varies.

Figure 4.1 illustrates the pyramid partition of an image. By using the BOW representation model, the original image can be represented by a histogram of k dimensions (k is defined by k-means clustering) in the first pyramid scale; then, the image is divided into four sub-regions and 16 sub-regions in the second and third pyramid scales, respectively. The BOW model is applied over each region to obtain histogram vectors and all these histogram vectors generated from both original scale and sub-scales are concatenated into a histogram vector to represent the image.

Figure 4.2 shows an example of how the BOW model represents the image in Figure 4.1, using the SIFT features and the spatial SIFT features. For illustration purpose, k=50 is applied in the k-means clustering and two pyramid levels are applied for the spatial SIFT features. By using the SIFT features, the BOW is only applied to the original image therefore the image is represented by a histogram vector of length 50 (Figure 4.2 (a)). However, using the spatial SIFT features, BOW is applied to both the original image and the sub-regions. Hence, the image is represented by a vector of length 250 (Figure 4.2 (b)). As both SIFT and spatial SIFT features are sharing the same dictionary, the SIFT vector forms the first portion of the spatial SIFT vector.

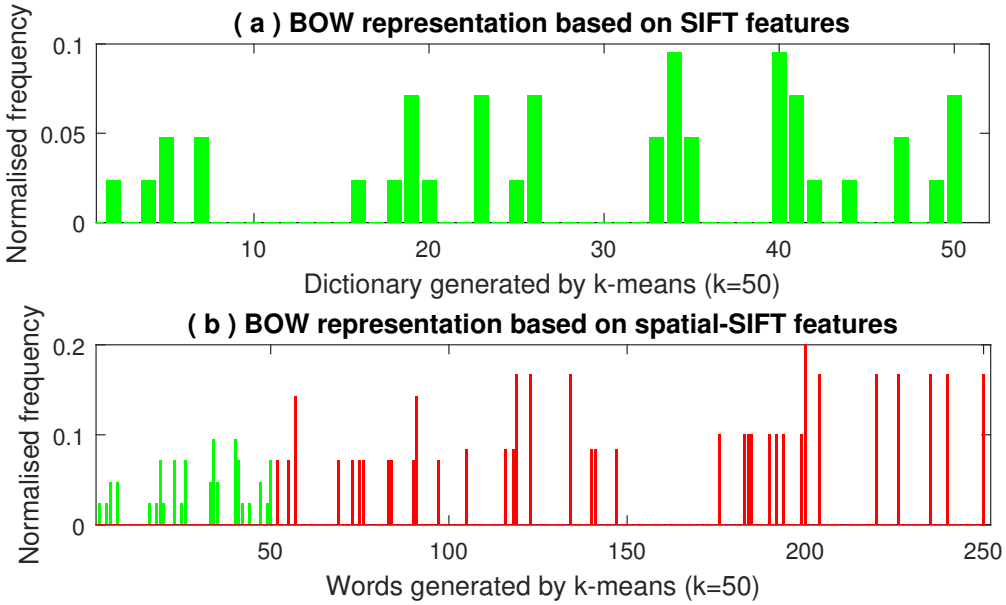


Figure 4.2: The BOW representation for the image in Figure 4.1 based on the SIFT features (a) and the spatial SIFT features (b).

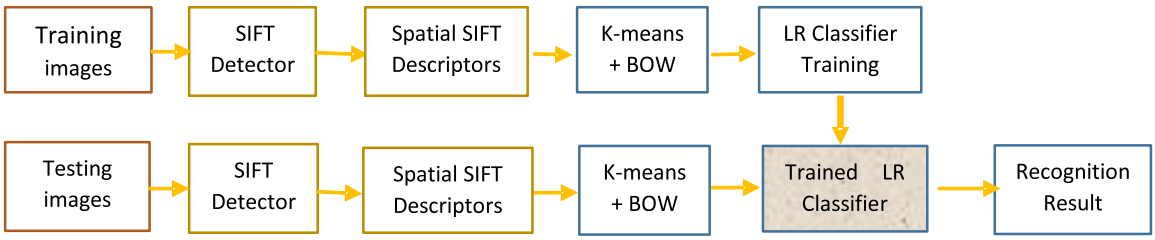


Figure 4.3: The developed recognition framework by using Spatial SIFT features.

In the performance evaluations, the spatial SIFT features are compared with the HOG and SIFT features. Three common classifiers are applied as the choice of classifiers (LR, SVM and KNN). The developed VLR framework, which combines the spatial SIFT features with the LR classifier, is shown in Figure 4.3. According to the experimental results, it is found that the non-parametric classifier KNN is vulnerable to noise when compared with parametric classifiers such as LR and SVM. KNN does not require a training stage, therefore, it would not suffer from insufficient training models. This motivates the author to find such a non-parametric classifier that could get rid of the training stage while achieving good robustness to noise.

4.3 Back-propagation Bayesian Compressive Sensing

4.3.1 Bayesian Compressive Sensing

The Sparse Representation Classifier (SRC) proposed by Wright et al. [81] is a non-parametric classifier based on sparse representation. It has proved to be more accurate than the linear SVM and the KNN classifier for face recognition. However, the high computational costs associated with the SRC can be a problem. In addition, the SRC works only when the system is under-determined [82]. In practice, this criterion cannot be met when there is a lack of training data. The Bayesian Compressive Sensing (BCS) method [89] uses the same assumption that a testing image can be represented as a linear combination of few training images, where the relative

importance of each training image is controlled by the weights vector $\mathbf{w} \in \mathbb{R}^{N\times 1}$. It provides an alternative to the l_1-norm minimisation as in SRC by incorporating prior knowledge with a Bayesian approach. Here, $\mathbf{w}$ can be separated into $\mathbf{w}_v \in \mathbb{R}^{N\times 1}$ and $\mathbf{w}_e \in \mathbb{R}^{N\times 1}$, where $\mathbf{w}_v$ contains the significant weights and $\mathbf{w}_e$ the remaining negligible weights. Hence, $\mathbf{w} = \mathbf{w}_v + \mathbf{w}_e$ and equation (2.19) can be written as:

$$\mathbf{x}^* = \mathbf{X}\mathbf{w}_v + \mathbf{X}\mathbf{w}_e + \mathbf{z}. \tag{4.1}$$

Both $\mathbf{X}\mathbf{w}_e$ and $\mathbf{z}$ can be approximated as a zero-mean Gaussian noise [89], allowing equation (4.1) to be written as:

$$\mathbf{x}^* = \mathbf{X}\mathbf{w}_v + \mathbf{n}, \tag{4.2}$$

where $\mathbf{n} = \mathbf{X}\mathbf{w}_e + \mathbf{z}$. The variance of $\mathbf{n}$ is then given by $\mathbf{\Sigma}_n = \sigma^2\mathbf{I}_M$, where $\mathbf{I}_M$ is an identity matrix of size $[M \times M]$. Note that each entry in $\mathbf{n}$ has the same variance σ^2 and hence the likelihood function can be given by:

$$p(\mathbf{x}^*|\mathbf{w}, \sigma^2) = (2\pi\sigma^2)^{-\frac{M}{2}} exp\left\{-\frac{||\mathbf{x}^* - \mathbf{X}\mathbf{w}||_2^2}{2\sigma^2}\right\}, \tag{4.3}$$

rather than in the standard multivariate form which includes the covariance matrix $\mathbf{\Sigma}_n$. Note, in equation (4.3) and henceforth the subscript v is dropped on $\mathbf{w}$.

A zero mean Gaussian prior is used to impose a belief that $\mathbf{w}$ is sparse. This is given by:

$$\begin{aligned} p(\mathbf{w}|\boldsymbol{\alpha}) &= \prod_{i=1}^{N} \mathcal{N}(w_i|0, \alpha_i^{-1}) \\ &= \prod_{i=1}^{N} (2\pi\alpha_i^{-1})^{-\frac{1}{2}} exp\left\{-\frac{1}{2}\alpha_i w_i^2\right\} \\ &= (2\pi)^{-\frac{N}{2}} |\mathbf{A}|^{\frac{1}{2}} exp\left\{-\frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{A}\mathbf{w}\right\}, \end{aligned} \tag{4.4}$$

where $\mathbf{A} = diag(\alpha_1, \alpha_2, \cdots, \alpha_N)$ and $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_N]^\intercal$, α_i is a precision value and $|\cdot|$ denotes the determinant. Furthermore, Gamma hierarchical priors are considered over α_i and σ^2:

$$p(\boldsymbol{\alpha}) = \prod_{i=1}^{N} Gamma(\alpha_i|a, b), \tag{4.5}$$

$$p(\sigma^2) = Gamma(\sigma^2|c, d), \tag{4.6}$$

where a, b, c and d are shape and scale parameters.

The overall prior over $\mathbf{w}$ can be evaluated by marginalising over the hyper-parameters $\boldsymbol{\alpha}$:

$$p(\mathbf{w}|a, b) = \prod_{i=1}^{N} \int_0^\infty \mathcal{N}(w_i|0, \alpha_i^{-1}) Gamma(\alpha_i|a, b) d\alpha_i. \tag{4.7}$$

Since the prior of $\mathbf{w}$ is assumed to be a zero-mean Gaussian distribution which conjugates to a Gamma prior, the probability density $p(\mathbf{w}|a, b)$ corresponds to a Student's t-distribution [111]. This achieves sparsity as the Student's t-distribution can be strongly peaked at $w_i = 0$ with appropriate choices of a and b [89, 111].

Combining the likelihood and the prior given in equations (4.3) and (4.4), the posterior distribution of the weights can be found from:

$$p(\mathbf{w}|\mathbf{x}^*, \boldsymbol{\alpha}, \sigma^2) = \frac{p(\mathbf{x}^*|\mathbf{w}, \sigma^2) p(\mathbf{w}|\boldsymbol{\alpha})}{p(\mathbf{x}^*|\boldsymbol{\alpha}, \sigma^2)}. \tag{4.8}$$

As the likelihood and prior are both Gaussian, the posterior distribution over $\mathbf{w}$ is also a Gaussian distribution:

$$\begin{aligned} p(\mathbf{w}|\mathbf{x}^*, \boldsymbol{\alpha}, \sigma^2) &= \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma}), \\ &= (2\pi)^{-N/2} |\boldsymbol{\Sigma}|^{-1/2} exp\left\{ -\frac{1}{2} (\mathbf{w} - \boldsymbol{\mu})^\intercal \boldsymbol{\Sigma}^{-1} (\mathbf{w} - \boldsymbol{\mu}) \right\}, \end{aligned} \tag{4.9}$$

where the mean and covariance are given by:

$$\boldsymbol{\mu} = \sigma^{-2}\boldsymbol{\Sigma}\mathbf{X}^{\mathrm{T}}\mathbf{x}^{*} \tag{4.10}$$

and

$$\boldsymbol{\Sigma} = (\mathbf{A} + \sigma^{-2}\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}, \tag{4.11}$$

respectively.

Notice that $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are dependent on σ^2 and $\boldsymbol{\alpha}$. Therefore, the goal is to find the posterior probability density function over all the unknown parameters given the training data and the testing image. Hence, the aim is to find the values for $\mathbf{w}$, $\boldsymbol{\alpha}$ and σ^2 which maximise the following posterior probability density function:

$$p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2|\mathbf{x}^{*}) = p(\mathbf{w}|\mathbf{x}^{*}, \boldsymbol{\alpha}, \sigma^2)p(\boldsymbol{\alpha}, \sigma^2|\mathbf{x}^{*}). \tag{4.12}$$

Finding the optimal $\mathbf{w}$, $\boldsymbol{\alpha}$ and σ^2 involves two steps. Firstly, for the current values of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ the values of $\boldsymbol{\alpha}$ and σ^2 are calculated to maximise $p(\boldsymbol{\alpha}, \sigma^2|\mathbf{x}^{*})$. Then these values are substituted to re-evaluate $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. This process is then repeated until a convergence criterion is met. In the first step $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are fixed then maximising equation (4.12) is equivalent to maximising:

$$p(\boldsymbol{\alpha}, \sigma^2|\mathbf{x}^{*}) = \frac{p(\mathbf{x}^{*}|\boldsymbol{\alpha}, \sigma^2)p(\boldsymbol{\alpha})p(\sigma^2)}{p(\mathbf{x}^{*})}, \tag{4.13}$$

where the denominator is independent of $\boldsymbol{\alpha}$ and σ^2. Therefore, only $p(\mathbf{x}^{*}|\boldsymbol{\alpha}, \sigma^2)p(\boldsymbol{\alpha})p(\sigma^2)$ has to be maximised. Furthermore, by selecting a, b, c and d to be small positive values, there are flat, uninformative priors over α and σ^2 [111]. Thus, maximising equation (4.13) is approximately equal to maximising the marginal likelihood:

$$p(\mathbf{x}^{*}|\boldsymbol{\alpha}, \sigma^2) = \int p(\mathbf{x}^{*}|\mathbf{w}, \sigma^2)p(\mathbf{w}|\boldsymbol{\alpha})d\mathbf{w}, \tag{4.14}$$

with $p(\mathbf{x}^*|\mathbf{w}, \sigma^2)$ and $p(\mathbf{w}|\boldsymbol{\alpha})$ being given in equations (4.3) and (4.4), respectively. Details for the derivation of the marginal likelihood are given in Appendix A.

Equation (4.14) is a convolution of two zero-mean Gaussians and the natural logarithm of the result gives:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\alpha}, \sigma^2) &= ln\left(p(\mathbf{x}^*|\boldsymbol{\alpha}, \sigma^2)\right) \\ &= ln\left(\mathcal{N}(\mathbf{x}^*|0, \mathbf{C})\right) \\ &= -\frac{1}{2}\left(Mln(2\pi) + ln|\mathbf{C}| + \mathbf{x}^{*T}\mathbf{C}^{-1}\mathbf{x}^*\right), \end{aligned} \tag{4.15}$$

where the $M \times M$ matrix $\mathbf{C}$ is given by:

$$\mathbf{C} = \sigma^2\mathbf{I}_M + \mathbf{X}\mathbf{A}^{-1}\mathbf{X}^{\intercal}. \tag{4.16}$$

A type-II maximum likelihood approximation can be used to estimate $\boldsymbol{\alpha}$ and σ^2 [111], which gives:

$$\alpha_i^{new} = \frac{1 - \alpha_i\Sigma_{ii}}{\mu_i^2}, \tag{4.17}$$

$$(\sigma^{new})^2 = \frac{||\mathbf{x}^* - \mathbf{X}\boldsymbol{\mu}||_2^2}{M - \sum_i^N(1 - \alpha_i\Sigma_{ii})}, \tag{4.18}$$

where Σ_{ii} is the i-th diagonal element of $\boldsymbol{\Sigma}$ in equation (4.11). Notice that $\boldsymbol{\alpha}$ and σ^2 are functions of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, while $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are functions of $\boldsymbol{\alpha}$ and σ^2. As previously discussed, this suggests an iterative algorithm to update each variable until a convergence criterion has been met. A derivation for the update equations in (4.17) and (4.18) is provided in Appendix A.

4.3.2 Back-propagation Bayesian Compressive Sensing with the Column-based Subspace Sampling

Back-propagation Bayesian Compressive Sensing

Given that the training images in $\mathbf{X}$ belong to C classes, where the class label $i \in \{1, 2, \cdots, C\}$, the training images can be separated according to their labels. This gives $\mathbf{X} = [\mathbf{X}^1, \mathbf{X}^2, \cdots, \mathbf{X}^i, \cdots, \mathbf{X}^C]$, where $\mathbf{X}^i$ contains all of the training images belonging to the i^{th} class. Suppose that there are n_i samples in the i^{th} class, then all of the training images in the i^{th} class are given by $\mathbf{X}^i = [\mathbf{x}_1^i, \mathbf{x}_2^i, \cdots, \mathbf{x}_{n_i}^i]$. Notice, this process only separates the training images by their labels; the total number of training images does not change. Hence, $\sum_i^C n_i = N$.

Therefore the original testing image can be reconstructed by $\hat{\mathbf{w}}$:

$$\tilde{\mathbf{x}}^* = [\mathbf{X}^1, \mathbf{X}^2, \cdots, \mathbf{X}^C] \begin{bmatrix} \hat{\mathbf{w}}^1 \\ \hat{\mathbf{w}}^2 \\ \vdots \\ \hat{\mathbf{w}}^C \end{bmatrix}, \tag{4.19}$$

where $\tilde{\mathbf{x}}^*$ is an estimation of the original image $\mathbf{x}^*$ and $\hat{\mathbf{w}} = \left[[\hat{\mathbf{w}}^1]^\intercal, [\hat{\mathbf{w}}^2]^\intercal, \cdots, [\hat{\mathbf{w}}^C]^\intercal\right]^\intercal$. Based on the assumption that a testing image is a linear combination of a few images from its corresponding class, non-zero valued elements in $\hat{\mathbf{w}}$ should be only in $\hat{\mathbf{w}}^i$ if the testing image belongs to the class i. The BCS approach [90, 91] assigns the testing image to the class i if it has the highest norm-2 magnitude of $\hat{\mathbf{w}}^i$.

However, when there are training images with no or a very small number of points of interest, most of the resulting feature vectors will be zero valued. This would allow large weight values in $\hat{\mathbf{w}}$ without detrimentally affecting the likelihood value when evaluating equation (4.3). These inappropriately large weight values can lead to a logo being misclassified when using the l_2-norm of the weights as a classification mechanism. To overcome this problem this chapter proposes a classification approach based on a back-propagation process.

The proposed method reconstructs the testing images by a BCS process in which the images are represented by equation (4.19). The weights vector $\hat{\mathbf{w}}$ is separated into C vectors with each vector keeping the value in its corresponding weights locations and setting the remaining values to zero:

$$\begin{bmatrix} \hat{\mathbf{w}}^1 \\ \hat{\mathbf{w}}^2 \\ \vdots \\ \hat{\mathbf{w}}^C \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{w}}^1 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \hat{\mathbf{w}}^2 \\ \vdots \\ \mathbf{0} \end{bmatrix} + \cdots + \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \hat{\mathbf{w}}^C \end{bmatrix}$$

$$\hat{\mathbf{w}} \quad = \quad \tilde{\mathbf{w}}^1 \; + \quad \tilde{\mathbf{w}}^2 \; + \cdots + \quad \tilde{\mathbf{w}}^C, \tag{4.20}$$

where $\tilde{\mathbf{w}}^i \in \mathbb{R}^{N\times 1}$ and $i \in \{1, 2, \cdots, C\}$. Each $\tilde{\mathbf{w}}^i$ is used to reconstruct the testing image $\mathbf{x}^i_{cons}$ as follows:

$$\mathbf{x}^i_{cons} = \mathbf{X}\tilde{\mathbf{w}}^i. \tag{4.21}$$

The testing image $\mathbf{x}^*$ will be assigned to the class with the corresponding reconstructed image to which it is most similar. More specifically, if the testing image recovered by $\tilde{\mathbf{w}}^i$ has the highest similarity with the original testing image $\mathbf{x}^*$, then this testing image can be classified into the i^{th} class. In order to compute the similarity between the image recovered by $\tilde{\mathbf{w}}^i$ and the original image $\mathbf{x}^*$, an error term is defined for each class:

$$Err(i) = ||\mathbf{x}^* - \mathbf{x}^i_{cons}||_2. \tag{4.22}$$

Then the testing image can be classified into the class which gives the minimum error.

Column-based Subspace Sampling

Since all the training data are used to evaluate the weights, it is time consuming to implement the algorithm when the dataset is very large. Methods in [112, 113] map the data into a reduced dimension space, as for Principle Component Analysis (PCA). However, these new latent spaces make the original data difficult to interpret. To combat this issue the column-based subspace sampling can instead be used [114–116].

In this case it is still possible to work in the original space, just with fewer data points.

Estimating the coefficients in equation (4.2) for SBCS can be time consuming when $\mathbf{X}$ is in a high dimension. PCA can solve this problem by mapping the data into a lower dimensional data space. However, as the space has been altered, each entry can be difficult to interpret. In addition, all the testing data are transferred into the new space. This restricts the system to only work in offline applications, as adding more data would change the previous feature space entirely. The column-based subspace sampling method can avoid these problems [114]. It selects the "best" subset of d columns from $\mathbf{X}$, where $d < N$.

Let $\mathbf{X}_n$ represent the "best" rank-n approximation to $\mathbf{X}$ by singular value decomposition. The output matrix $\mathbf{D} \in \mathbb{R}^{M \times d}$ consists of d columns from $\mathbf{X}$ such that the inequality in equation (4.23) is valid for a probability at least $1 - \delta$.

$$||\mathbf{X} - \mathbf{D}\mathbf{D}^{+}\mathbf{X}||_F \leqslant (1 + \rho)||\mathbf{X} - \mathbf{X}_n||_F, \tag{4.23}$$

where $||\cdot||_F$ is the Frobenius norm, $\mathbf{D}^{+}$ is a Moore-Penrose generalized inverse of $\mathbf{D}$, ρ is an error parameter and δ is the failure probability.

Define a score for each column in "$\mathbf{X}$" in the following form:

$$\pi_j = \frac{1}{n}\sum_{\xi=1}^{n}(\mathbf{v}_j^{\xi})^2, \tag{4.24}$$

where $\mathbf{v}_j^{\xi}$ $(j = 1, 2, \cdots, N)$ is the j^{th} coordinate of $\mathbf{v}^{\xi}$ and $\mathbf{v}^{\xi} \in \mathbb{R}^{N \times 1}$ $(\xi = 1, 2, \cdots, n)$ is the top right n singular vectors of $\mathbf{X}$. A random sampling process is applied on $\mathbf{X}$ and the j^{th} column of $\mathbf{X}$ is adopted with probability $min\{1, d\pi_j\}$, where $d = O(n\ln(n/\rho^2))$. All the adopted columns then generate the target matrix $\mathbf{D}$, with d examples to represent the original dataset. The detailed proof is given in [114, 116].

4.4 Performance Evaluations on Spatial Scale Invariant Feature Transform

In this section, the proposed framework is applied on the open dataset provided by Huang et al. [50] in order to evaluate the performance. The dataset is the same as the dataset used in Chapter 3. The performance evaluation is conducted in MATLAB on a computer with the following specification: I5, 3.4G Quad-core, and 8G memory. The open source library VLFeat [117] is used for SIFT features extraction and LIBSVM toolbox [118] is used for SVM classification. The following result shows the performance of the HOG, SIFT and spatial SIFT features when they are combined with different classifiers such as the SVM, LR and KNN. Different levels of noise are added in order to examine the robustness of the proposed framework.

4.4.1 Feature Comparisons

Three classifiers are used in this section and the following for feature classification, which are the KNN, SVM and LR. The SVM uses the default radial basis function kernel in LIBSVM and $\lambda = 0.1$ is set in the LR classifier.

Table 4.1: Performance of HOG by using different classifiers.

HOG features			
Classifier	SVM	LR	KNN
Accuracy (%)	88.40	**97.53**	95.67
Misclassified images from 1500 testing images	174	**37.05**	64.95

From Table 4.1 it can be seen that LR outperforms SVM and KNN in terms of classification accuracy. This validates the use of LR in VLR. For local features such as SIFT and spatial SIFT, the dictionary size will influence the performance in terms of accuracy. Hence, the SVM, KNN and LR with ten different dictionary sizes

are tested. Centroids of k-means clustering are randomly initialised, which results in different outcomes with each run; therefore, the experiments are conducted 30 times and the mean results with variances are presented.

Table 4.2: Classification accuracies ($\mu \pm \sigma$) on 1500 testing images using SIFT features, according to different dictionary sizes in the k-means clustering (30 runs).

SIFT features					
K	50	100	200	300	400
SVM (%)	88.09 ± 1.01	92.87 ± 0.55	95.89 ± 0.61	97.09 ± 0.46	97.64 ± 0.27
Misclassified images	178.65	106.95	61.65	43.65	35.40
LR (%)	88.76 ± 1.13	95.18 ± 0.62	98.56 ± 0.32	$\mathbf{99.11} \pm 0.22$	99.37 ± 0.16
Err(/1500)	170.40	72.30	21.60	13.35	9.45
KNN (%)	96.55 ± 2.33	97.93 ± 0.37	98.55 ± 0.33	98.60 ± 0.26	98.46 ± 0.28
Misclassified images	51.75	31.05	21.75	21	23.10

K	500	600	700	800	900
SVM (%)	97.77 ± 0.25	97.96 ± 0.26	98 ± 0.19	98 ± 0.25	98.05 ± 0.19
Misclassified images	33.45	30.60	30	30	29.25
LR (%)	99.54 ± 0.12	99.63 ± 0.11	99.70 ± 0.10	99.70 ± 0.13	$\mathbf{99.76} \pm 0.14$
Misclassified images	6.90	5.55	4.50	4.50	3.60
KNN (%)	98.66 ± 0.25	98.68 ± 0.27	98.73 ± 0.29	98.69 ± 0.25	98.81 ± 0.24
Misclassified images	20.10	20.10	19.05	19.65	17.85

Table 4.2 shows that increasing dictionary size improves the classification accuracy. This is because the feature naturally contains more information in a higher dimensional space. Among these three classifiers, the LR classifier always achieves a slightly higher accuracy than the SVM on this dataset, while the KNN achieves high accuracy than the rest when the dictionary size is smaller than 300. However, when the dimension increases, the improvement of KNN is not as obvious as for both the SVM and LR. Furthermore, the variance of accuracy achieved when using the LR is smaller than for both SVM and KNN on this dataset. This indicates that the performance of the LR classifier is more stable. By combining SIFT features with the LR classifier, a recognition accuracy of 99.76 ± 0.14 is achieved on this dataset, which is slightly higher than the previous highest accuracy achieved by PCA-CNNs (99.13 ± 0.24) [50].

Table 4.3: Classification accuracies ($\mu \pm \sigma$) on 1500 testing images by using spatial SIFT features, according to different dictionary sizes in the the k-means clustering (30 runs).

Spatial SIFT features					
K	50	100	200	300	400
SVM (%)	89.52 ± 0.88	94.26 ± 0.59	96.36 ± 0.37	96.84 ± 0.32	97.04 ± 0.38
Misclassified images	157.20	86.10	54.60	47.40	44.40
LR (%)	95.34 ± 0.63	98.55 ± 0.25	99.55 ± 0.16	$\mathbf{99.71} \pm 0.12$	99.81 ± 0.13
Misclassified images	69.90	21.75	6.75	4.35	2.85
KNN (%)	92.48 ± 0.45	93.11 ± 0.60	93.56 ± 0.69	93.85 ± 0.59	94.12 ± 0.53
Misclassified images	112.80	103.35	96.60	92.25	88.20
K	500	600	700	800	900
SVM (%)	97.10 ± 0.26	97.12 ± 0.25	97.08 ± 0.31	97.08 ± 0.26	97.20 ± 0.24
Misclassified images	43.50	43.20	43.80	43.80	42
LR (%)	99.87 ± 0.09	99.86 ± 0.08	99.90 ± 0.09	99.92 ± 0.08	$\mathbf{99.93} \pm 0.07$
Misclassified images	1.95	2.10	1.50	1.20	1.05
KNN (%)	97.14 ± 0.49	96.82 ± 0.68	97.02 ± 0.60	96.98 ± 0.35	96.77 ± 0.38
Misclassified images	42.90	47.70	44.70	45.30	48.45

The pyramid level is set to 2 as the number of SIFT features is very limited in such low resolution images. The recognition accuracies using the spatial SIFT features and different classifiers are shown in Table 4.3. For both SVM and LR, the result indicates that the spatial SIFT features achieves a slightly higher accuracy on this dataset than the SIFT features no matter how large the dictionary is. However, for KNN, using spatial SIFT features have the opposite effect and accuracy is reduced. This lies on the features having been extended into a high dimensional space by segmenting the image into pyramid sub-regions and studies show that KNN is not sufficient when it is applied in a high dimensional space [45].

The efficiency of these features using the LR classifier has been compared. In this subsection and the one that follows only the LR is adopted as the classifier. This is because the results up to this point indicate that it is the most accurate of the classifiers. The experiment sets k=300 as a compromise between computational costs and accuracy. It is a dilemma here as increasing the dictionary size also increases the computational costs. For example, by the proposed framework, recognising all

Table 4.4: Computational costs by using different features with the LR classifier on 10000 training images and 1500 testing images.

Features	**HOG**	**SIFT**	**spatial SIFT**
Accuracy (%)	97.53	99.11	**99.71**
Misclassified images	37.05	13.35	4.35
Time-whole-process (s)	190	571	967
Time-per-test (s)	0.06	0.08	0.23

the testing images needs 847 and 1291 seconds when $k = 200$ and $k = 400$, respectively with its accuracy almost unchanged. The result in Table 4.4 indicates that spatial SIFT obtained the an average accuracy of 99.71%, with the SIFT achieving an average accuracy of 99.11%, and the HOG of 97.53%, respectively. The difference in accuracies seems negligible, however, the coming subsection will show that the spatial SIFT features are more robust to noise.

4.4.2 Feature Robustness to Noise

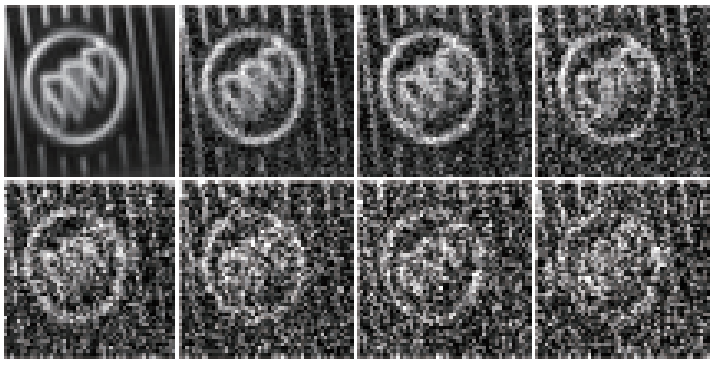


Figure 4.4: An example of a training image and effects by adding Gaussian noise with zero means and variance values 0.02, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3 from left to right, respectively.

In practice, we would not expect to always have clear logos in the images. As a result, here noises are added to the images in order to test the robustness of these features with the LR classifier. The noise is Gaussian with zero mean and differing levels of variance. Since the Gaussian noise is random, all experiments have been

conducted for ten times and their mean values are applied. Figure 4.4 shows an original training example and the effects by adding noise with increasing variances. Normally an image is highly contaminated if the Gaussian noise variance is above 0.2. The noise is added to the training and testing images separately with variances given by σ_{train} and σ_{test}, respectively.

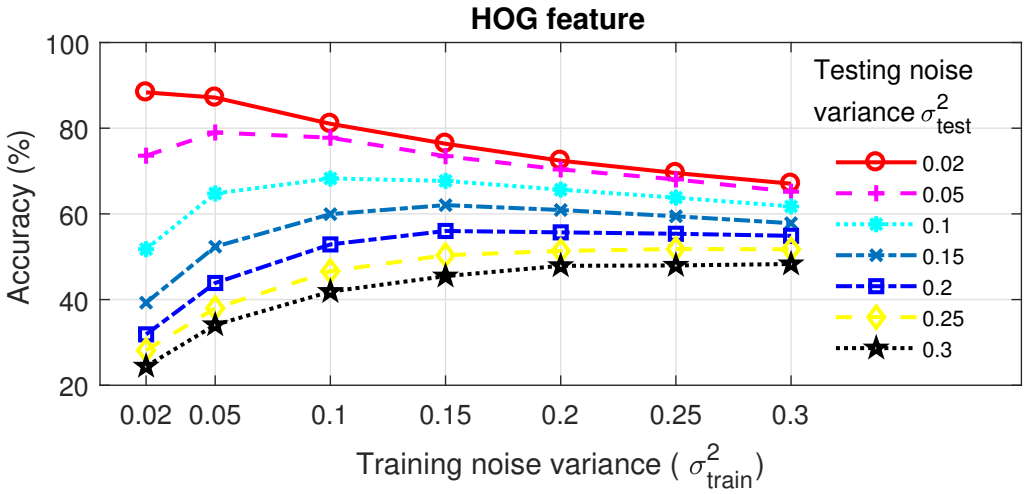


Figure 4.5: Accuracy of the recognition framework by using the HOG features.

Figure 4.5 illustrates how different noise variance levels for both training and testing images influence the accuracy when using the HOG features. Without surprise, adding noise in the image decreases the accuracy compared to the noise free case. Generally speaking, when the noise variance in the training set is fixed, the higher the noise variance added to the test images, the lower the accuracy will be. For example, σ^2_{test}=0.02 always outperforms σ^2_{test}=0.3 in terms of accuracy. However, when the noise in the testing images is fixed, the highest accuracy tends to be found when the training images have similar noise variance levels. For instance, the highest accuracy for σ^2_{test}=0.05 is found when σ^2_{train}=0.05; in contrast, the model trained by clearer training images (when σ^2_{train}=0.02) gives a less accurate recognition result. As a result, a higher accuracy can be achieved by matching σ^2_{train} to σ^2_{test}.

Compared with the HOG features, the SIFT features are more robust to noise as shown in Figure 4.6. The main reason why some misclassifications occur in extreme noise scenarios is that no SIFT features are detected. However, this does not always

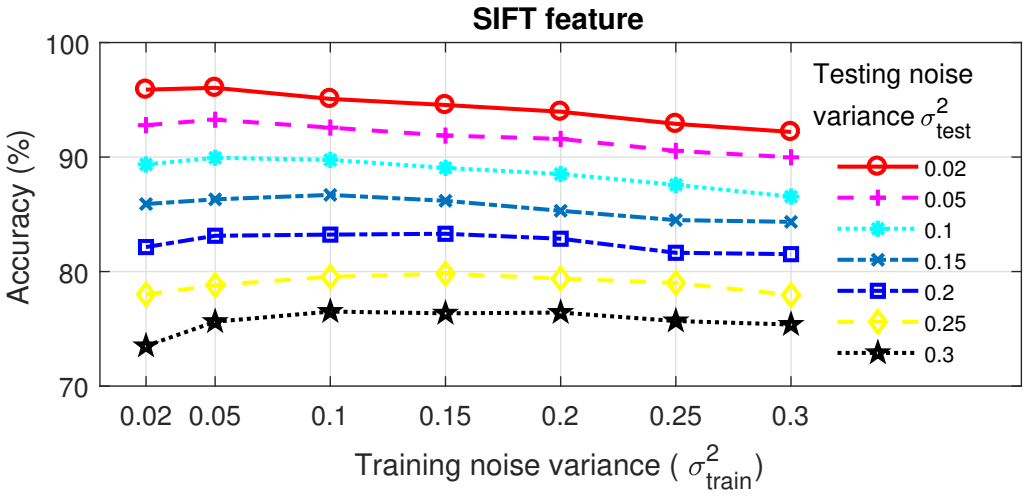


Figure 4.6: Accuracy of the recognition framework by using the SIFT features.

happen, meaning a good overall recognition accuracy is achieved.

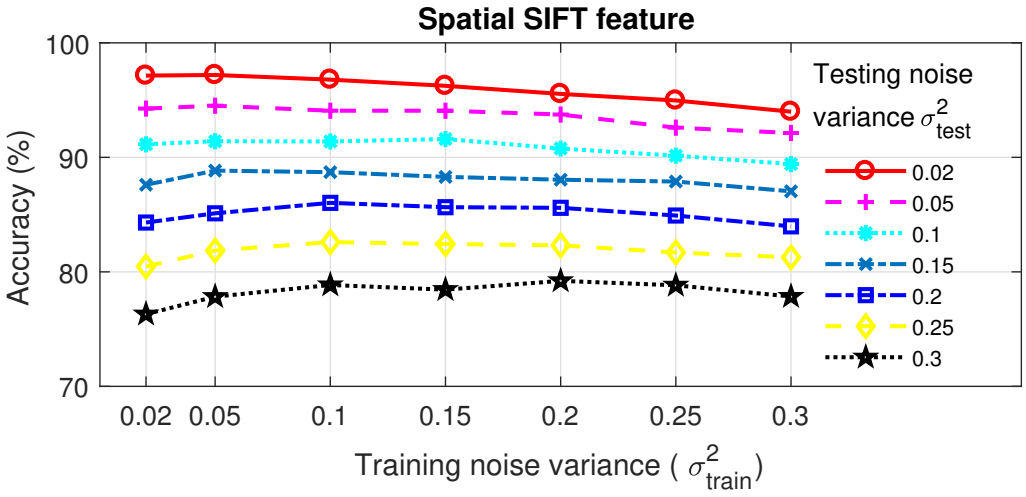


Figure 4.7: Accuracy of the recognition framework using the spatial SIFT features.

As shown in Figure 4.7, the spatial SIFT features give the highest accuracy when compared with the SIFT features and the HOG features. All accuracy results in Figure 4.7 are above the corresponding ones in Figure 4.6. This improvement shows that the geographic information included in spatial SIFT features have resulted in a more robust performance than for SIFT features.

4.5 Performance Evaluations on Non-parametric Classification Methods

This section focuses on the evaluations of the proposed SBCS classifier. Three different datasets are evaluated: vehicle logo dataset [50], traffic scene dataset [119] and CIFAR-10 dataset [120].

4.5.1 Evaluations on the Vehicle Logo Recognition Dataset

As the focus is the non-parametric classifier comparisons, the SIFT features are applied in this section due to its accuracy and efficiency. The open source library VLFeat [117] is applied for extracting the SIFT features. A comparison is made with the SRC (implemented using CVX [121, 122]), BCS classifier and KNN classifier. In the experiment, K=1 achieves the best result for clear images. Different K values influence the result when images are noisy, while the prior knowledge of images is unknown. Therefore, as is commonly done in the literature, [81, 115], here, a value of $K = 1$ is selected for all considered examples. The performance of each method is evaluated in terms of accuracy (percentage of correctly classified images), the total number of misclassified images and the computation time (to indicate the relative computational complexities).

This subsection compares the performances of the classification methods when applied to the images that are provided in the dataset [50], the same dataset as in Chapter 3. The simulation is repeated 30 times and the average accuracy is found and given with the corresponding standard deviation. The computation time and the number of misclassified images are also given as the mean results for all the simulation runs.

Table 4.5 shows that the SBCS classifier achieves the highest accuracy of 98.91%. Table 4.5 also indicates that the BCS classifier is less accurate than the SRC and SBCS classifier. For example, when k=300, the BCS classifier incorrectly classifies

Table 4.5: Non-parametric classifiers comparisons using SIFT descriptors with k=100, 200, 300, 400, 500, where k is the number of centroid in k-means clustering.

Classifiers	K**NN**	**SRC**	**BCS**	**SBCS**
M=100 (Accuracy%)	98.29 ± 0.36	**98.30** ± 0.44	92.17 ± 0.77	98.24 ± 0.32
Misclassified images	25.65	**25.50**	117.45	26.40
Time (s)	**0.97**	6357	868	868
M=200 (Accuracy%)	98.72 ± 0.24	**98.73** ± 0.25	91.36 ± 0.54	98.60 ± 0.28
Misclassified images	19.20	**19.05**	129.60	21
Time (s)	**1.84**	7804	2358	2358
M=300 (Accuracy%)	98.63 ± 0.27	98.78 ± 0.24	90.77 ± 0.75	**98.86** ± 0.22
Misclassified images	20.55	18.30	138.45	**17.10**
Time (s)	**2.70**	8360	3120	3120
M=400 (Accuracy%)	98.67 ± 0.30	98.83 ± 0.23	90.37 ± 0.77	**98.91** ± 0.24
Misclassified images	19.95	17.55	144.45	**16.35**
Time (s)	**3.54**	9116	3360	3360
M=500 (Accuracy%)	98.74 ± 0.23	**98.86** ± 0.19	90.25 ± 0.95	98.84 ± 0.25
Misclassified images	18.90	**17.10**	146.25	17.40
Time (s)	**4.17**	9582	3497	3497

138 images, while this is reduced to 17 images for the SBCS classifier. In this case, the number of misclassifications is reduced by 88% without increasing the computational costs. For all the values of k considered, there was a mean reduction in the number of misclassified logos of 87% for SBCS classifier as compared to the BCS classifier. The computation times in Table 4.5 show that this improvement in classification accuracy comes without an increase in computational time.

The SRC and SBCS classifier give very similar classification accuracies. However, the SBCS classifier has a significant advantage in terms of computational time. For the example when k=300, the proposed SBCS classifier reduces the computational time by 63% when compared with the SRC whilst giving a slightly improved accuracy compared with the SRC algorithm. When comparing the computation times of the proposed SBCS classifier to the SRC, for all values of k considered, there is a mean reduction in the computation time of 68%. It only takes about two seconds to recognise an image using the SBCS classifier (note, that the times in Table 4.5

are for classifying all images in the testing dataset). The computation times show that the KNN classifier is quicker than the proposed SBCS classification approach. However, later results will show that it is more vulnerable to the effects of noise.

Note that, according to these results, the computation times for the BCS and SBCS based classifiers are the same. However, the accuracy is consistently lower for the BCS classifier as compared to the SBCS classifier. The accuracy of the other two classifiers considered in the comparison also outperforms the BCS based method. As a result, the BCS based classifier will not be considered further in this performance evaluation.

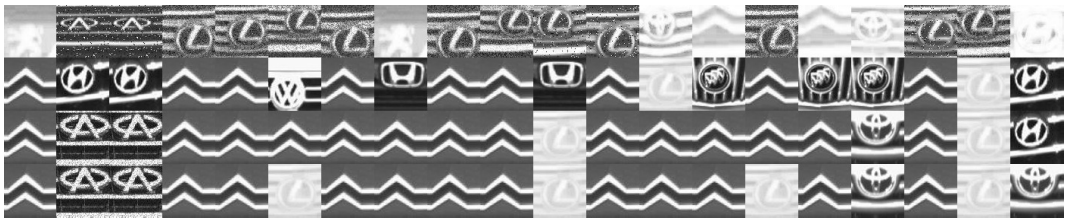


Figure 4.8: The first row illustrates some challenging images, the second, third and fourth rows are the corresponding results classified by KNN, SRC and SBCS, respectively.

Figure 4.8 shows 20 images (from the original testing dataset) that the KNN algorithm fails to satisfactorily classify. The first row gives the images that are under consideration and the second row gives the classification results from the KNN classifier. For comparison the SRC and SBCS classification results are shown in rows 3 and 4, respectively. The relative performances of the three methods are also further summarised in Table 4.6. Here it can be seen that both methods outperform the KNN algorithm in terms of classification accuracy. Note that the 30 independent simulation runs are conducted with the final selected class being the most frequent overall.

Table 4.6: Accuracies obtained using challenging data.

Classifier	K**NN**	**SRC**	**SBCS**
Accuracy	19.17%	43.83%	**47**%

Classification Comparisons with Noise

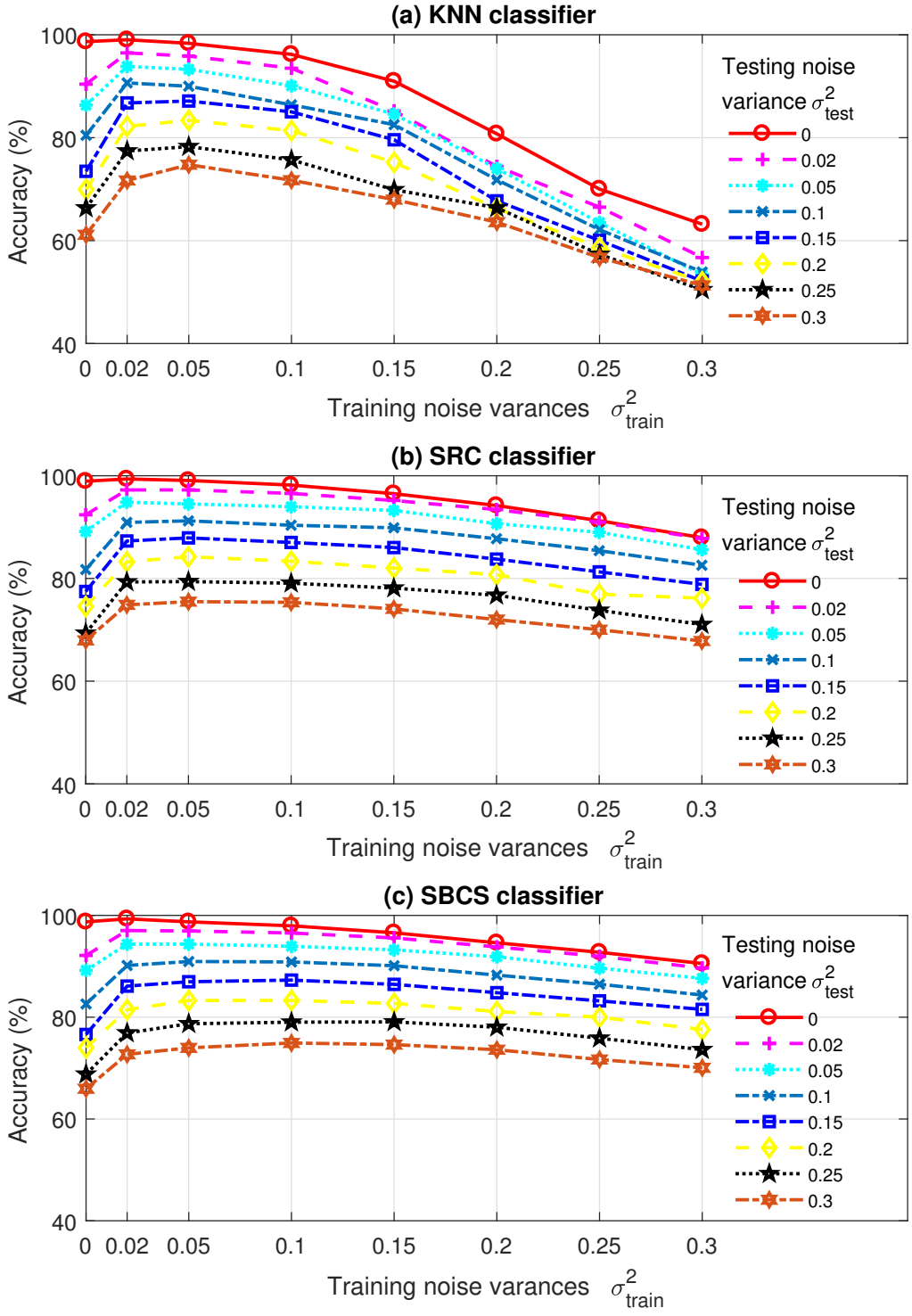


Figure 4.9: Noise robustness comparisons of KNN, SRC and SBCS using the full training dataset.

The same noise scheme as shown in Figure 4.4 is applied to compare to the

performances of KNN, SRC and SBCS. Figure 4.9 shows that the KNN classifier is the most vulnerable to the effects of noise. This can be explained by the fact that the KNN classifier only calculates the Euclidean distance between two feature vectors, while the other two allow for some errors when modelling a testing image as a linear combination of the training images. The performances of the SBCS classifier and the SRC are similar, while the SBCS classifier tends to be more accurate compared with the SRC when the training images are heavily contaminated by noise. For instance, when the noise variances are 0.25 in the training and testing images, the SBCS classifier, the SRC and KNN achieve 75.87 %, 73.79% and 57.31%, respectively. Furthermore, when the noise variances increase to 0.3, the SBCS classifier, the SRC and KNN can achieve 70.05 %, 67.82% and 51.30%, respectively.

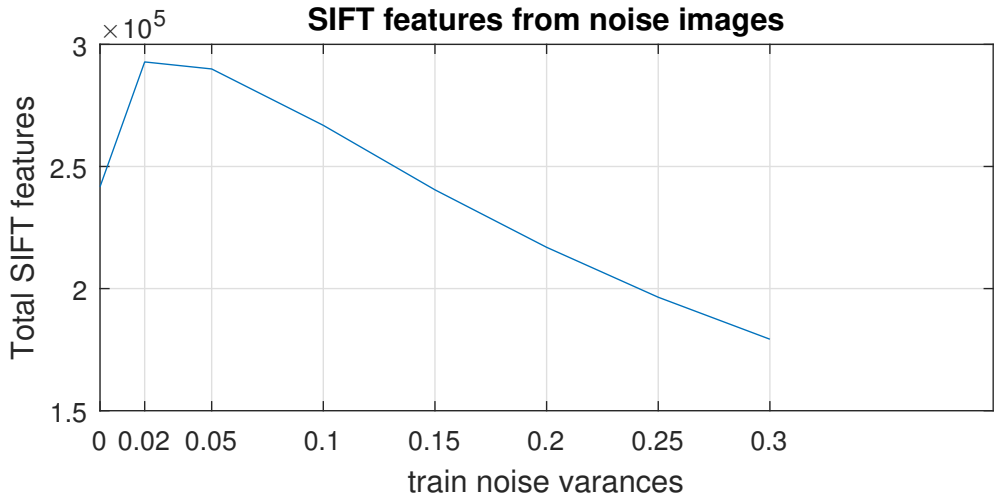


Figure 4.10: The total number of SIFT features detected from images with different noise variances.

Figure 4.10 shows that different numbers of SIFT features could be detected from the training dataset by adding noise with different variances. The result shows that in slightly noisy images there are more SIFT features could be detected; this could result in the image representation vector being more representative. However, when increasing the noise variance, less SIFT features could be detected as the images are seriously damaged by noise. Apart from the number of SIFT features, the prior knowledge of the noise can also influence the result. When the testing and training

images have the same noise variance, this could potentially perform better than other scenarios.

Column-based Subspace Sampling

In this subsection, a reduced number of training images are used to evaluate the situation where the size of the dictionary is large. Table 4.7 shows the time and computational costs comparisons for different classifiers. Using the column-based subspace sampling method, the partial dictionary size is decreased to 20% and 10% (denoted as $p1$ and $p2$, respectively) when compared to the original dataset (denoted as *all*). The computational costs decrease about six times ($p1$) and 11 times ($p2$), while the accuracy drops slightly. The proposed SBCS approach requires an overall time 500 and 277 seconds, respectively, which is 0.3s and 0.18s per image. The experiments are performed over 1500 images. This could still be applied to real-time applications. Even though the computational costs of the proposed algorithm are still higher than the computational costs of the KNN algorithm, it is more robust than the KNN when applied to noisy images. Since 10% data reduction does not decrease the accuracy significantly, the next experiments are performed with 10% data reduction as a trade-off between the computational costs and accuracy.

Table 4.7: Comparisons between using the full and partial dictionaries.

Classifiers	K**NN**(f)	**SRC**(f)	**SBCS**(f)
Accuracy(%)	98.63 ± 0.27	98.78 ± 0.24	$\mathbf{98.86} \pm 0.22$
Misclassified images	26.33	18.30	**17.10**
Time(s)	**2.70**	8360	3120
Classifiers	K**NN**$(p1)$	**SRC**$(p1)$	**SBCS**$(p1)$
Accuracy(%)	97.32 ± 0.47	$\mathbf{98.54} \pm 0.31$	98.24 ± 0.35
Misclassified images	40.20	**21.83**	26.83
Time(s)	**0.25**	1436	500
Classifiers	K**NN**$(p2)$	**SRC**$(p2)$	**SBCS**$(p2)$
Accuracy(%)	96.75 ± 0.86	$\mathbf{97.49} \pm 0.61$	96.94 ± 0.52
Misclassified images	40.20	**21.83**	26.83
Time(s)	**0.13**	1170	277

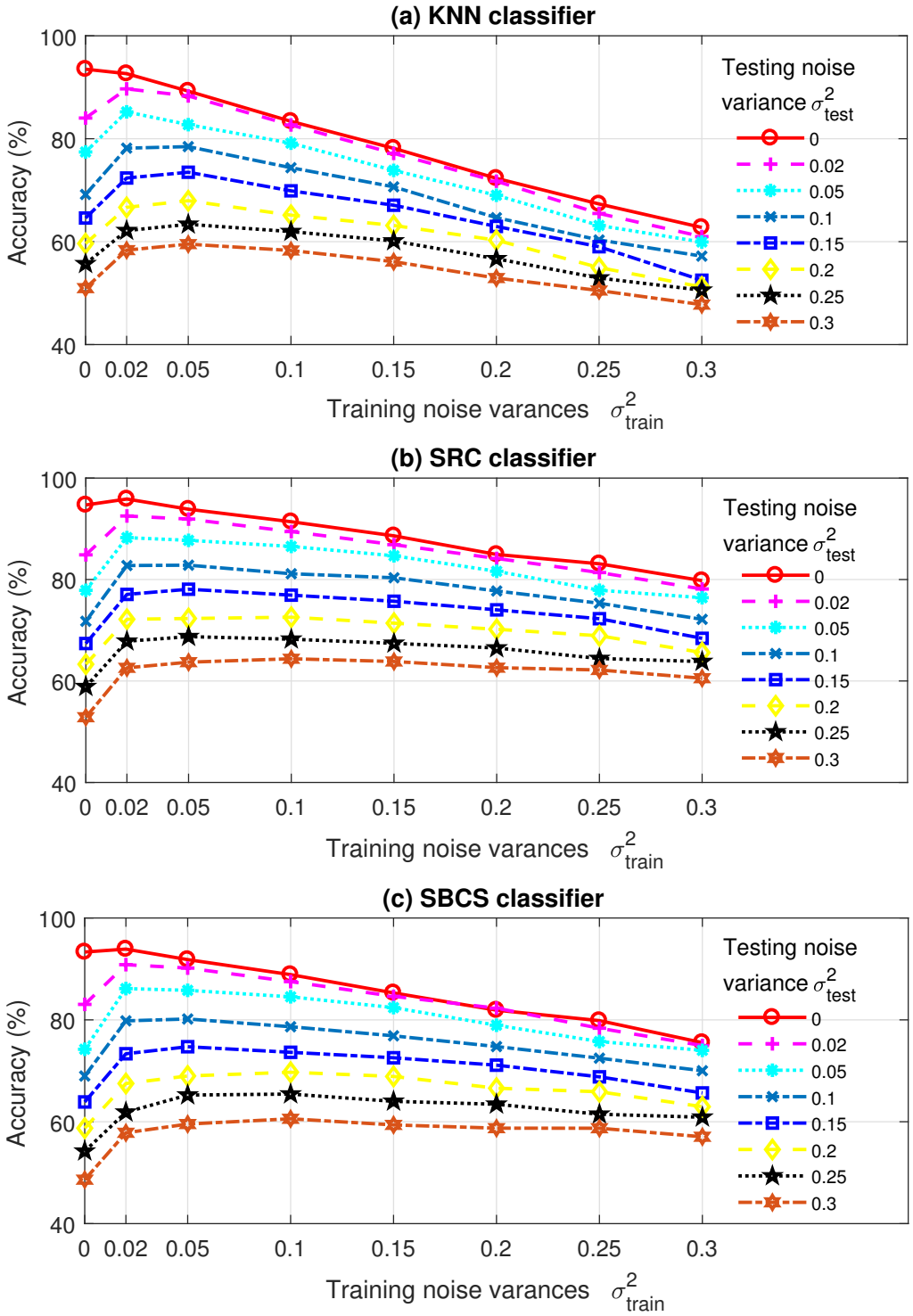


Figure 4.11: Noise robustness comparisons of KNN, SRC and SBCS when there are 10% training examples in each class using the column-based subspace sampling.

Figure 4.11 shows the result of different classifiers when the dictionary size is decreased to 10% by the column-based subspace sampling method. When comparing the accuracies to those shown in Figure 4.9, the accuracy of each classification method

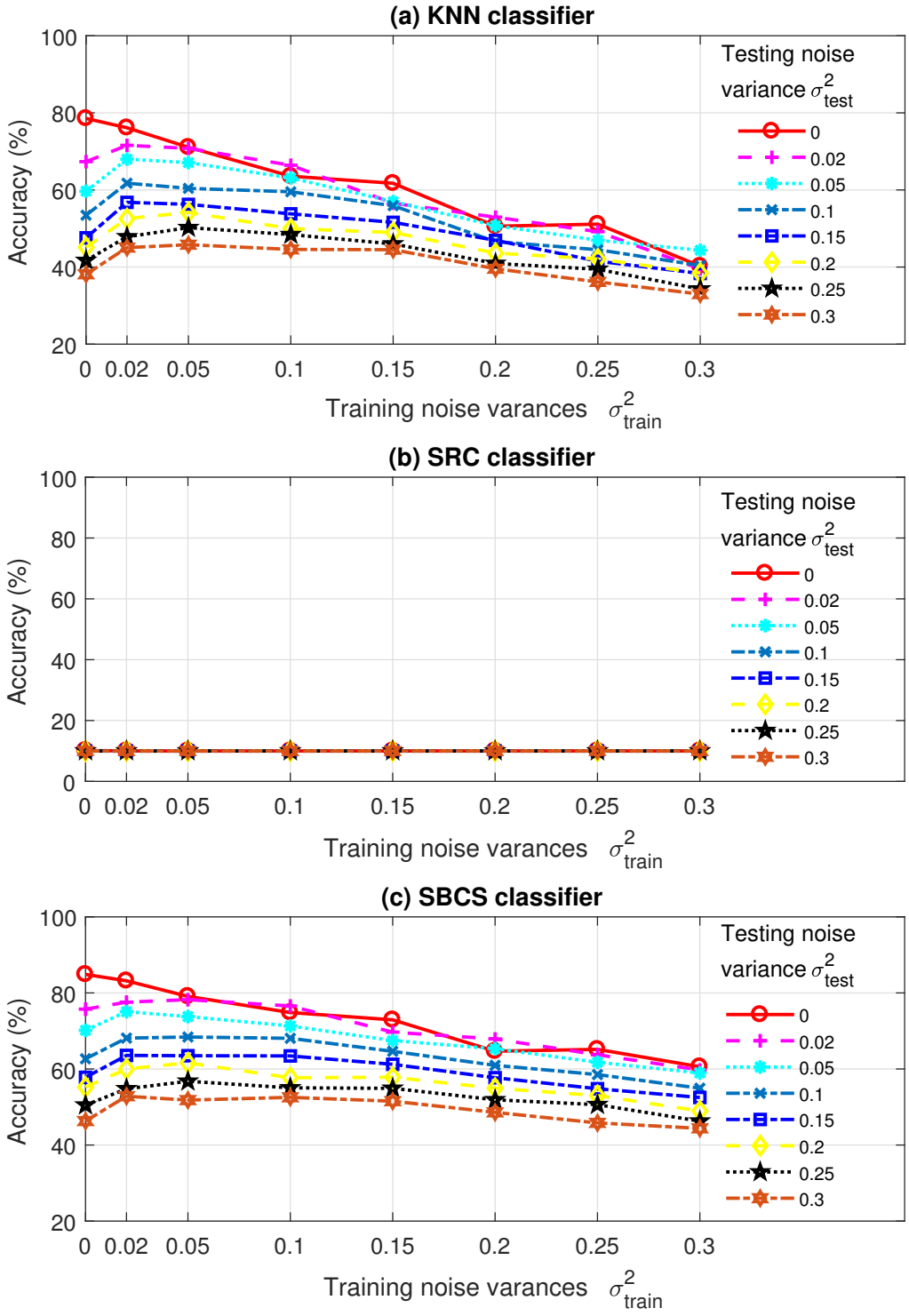


Figure 4.12: Noise robustness comparisons of KNN, SRC and SBCS when there are 1% training examples in each class using the column-based subspace sampling.

has been reduced when compared to Figure 4.11. Moreover, Figure 4.11 shows that the KNN classifier is significantly affected by noise and the SRC is only marginally more accurate than the SBCS classifier, despite having previously been shown to be

less computationally efficient. However, the computational costs are dropped as the dictionary size has decreased by 10 times.

The size of the training dataset is further decreased to only 1% selected images for each class in each of the ten independent simulations, with the resulting classification accuracies being shown in Figure 4.12. In this case, the accuracies of the KNN classifier are not as high as the SBCS algorithm, especially when the noise levels increase. The SRC no longer works since $M > N$ and the system is no longer under-determined. Note, that the conventional compressive sensing framework (as used in the SRC) is specifically designed for systems which are under-determined [82]. This leads to a random guess which can only achieve 10% accuracy as there are ten classes with equal number of logos in each class.

4.5.2 Performance Evaluation for Scene Recognition

So far, the last section has considered the application of SBCS for VLR. Traffic scene recognition is a very similar topic in Smart Cities. Here, the FM2 dataset [119] is considered. This dataset contains 6237 images from eight classes: highway, road, tunnel, tunnel exit, settlement, overpass, toll booth and dense traffic. Seventy percent of the images are randomly chosen for the training stage and the other 30% of images are for testing purposes. Figure 4.13 illustrates some examples of the FM2 dataset. A pre-trained framework (AlexNet [1]) is used for feature extraction. Instead of using the original weights from the network, which was trained on other images, this work replaces the last fully connected layer to 200 neurons and fine tunes the weights based on traffic scene images. Hence, each image is represented by a vector of length 200. Note that the focus is on the classification method rather than on the image feature extraction.

The column-based subspace sampling is applied to each training group. Since each class has an imbalanced training data, the experiment set a maximum number of 200 to each class. When a class has more than 200 training images, the column-based

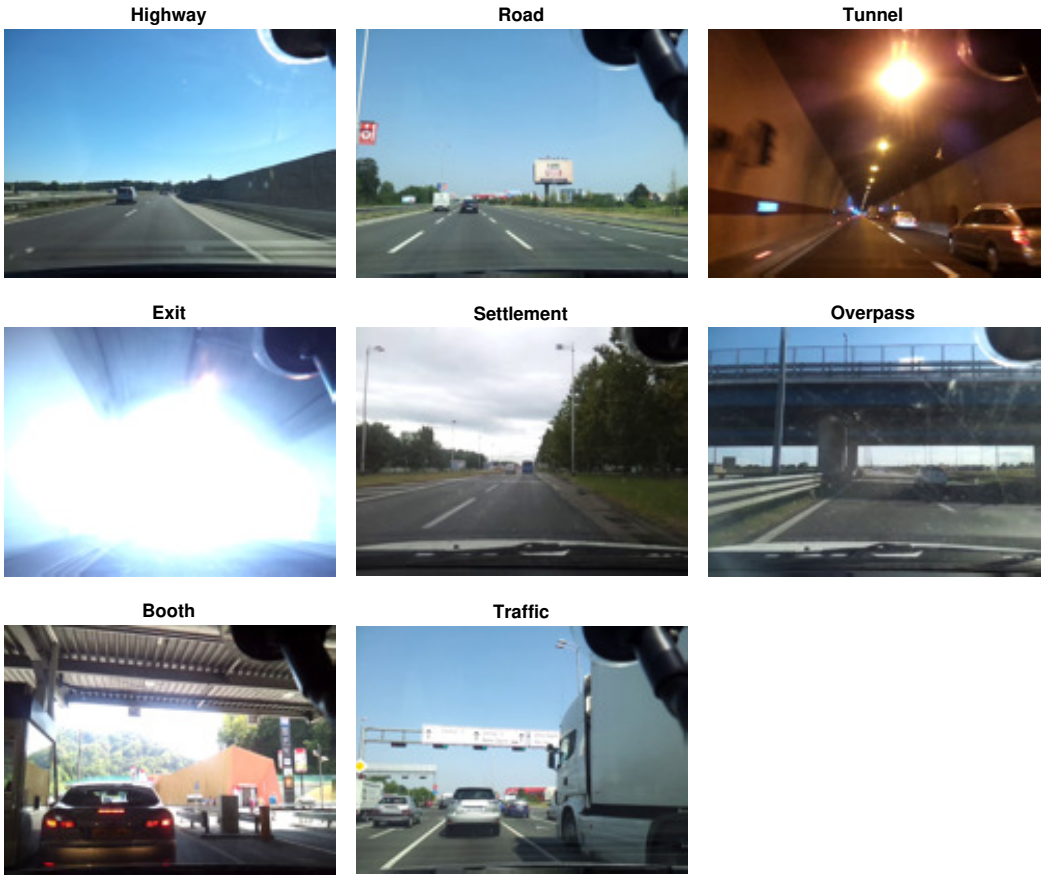


Figure 4.13: Image examples in the traffic scene dataset.

subspace sampling method is applied to this class. A comparison with a recently developed deep learning approach, the CNNs from [1] is performed, where the weights are trained for classification. Note that in CNNs the classification is applied directly without using column-based subspace sampling. Since the parameters are fixed based on the whole training dataset, there is no need of retraining a network using a much smaller dataset. However, the results for KNN, SBCS and SRC are achieved on the new dataset after the column based sub-sampling.

Table 4.8 shows the result from each classifier. Zero-mean Gaussian noises with different noise variances are applied on these training images and testing images. Without adding any noise, the CNNs achieve the highest accuracy. However, when increasing the noise, the CNNs become fragile. Similar research shows that when changing a few pixel values the classification result changes [123, 124]. However,

Table 4.8: CNNs, KNN, SBCS and SRC comparisons on traffic scene dataset using features extracted by AlexNet.

Noise variance	**CNN**(%)	K**NN**(%)	**SRC**(%)	**SBCS**(%)
0	**87.70**	84.41	87.00	86.31
0.01	57.01	73.21	79.73	**79.89**
0.1	10.59	56.04	57.59	**64.39**
0.2	7.43	52.03	42.51	**54.33**

using the extracted features from CNNs and applying them to other classifiers leads to better results. Increasing the noise level, the proposed SBCS achieves the best results. This is important as the real images are not always clear. Figure 4.14 illustrates how different noise levels influence an image.

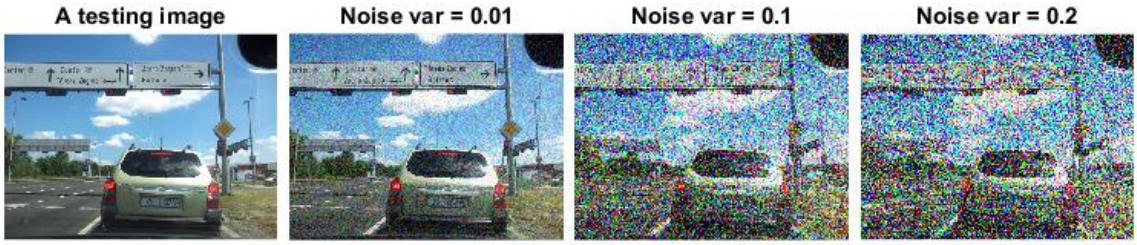


Figure 4.14: An example of a traffic scene image with Gaussian noises.

4.5.3 Evaluations on External Dataset

The proposed SBCS approach has the potential to be applied to other areas, not only restricted to VLR and traffic scene recognition. In the performance validation the CIFAR-10 dataset [120] is used. This dataset consists of 50000 training images and 10000 testing images. Here, a CNN framework similar to [1] is trained on the new dataset, with the last fully connected layer extracted to give the feature vector. Hence, each image is represented by a vector of length 200. Parameters of the network can be found in Appendix A.

The column-based subspace sampling is applied to each training group. This process picks 200 image feature vectors from 5000 image feature vectors in each group (4% of the original size). Hence, in order to avoid using all image feature vectors, the

dictionary $\mathbf{X}$ is formed by only 2000 representative image feature vectors. Both the CNNs and SBCS approaches train the weights for classification. Similarly, in CNNs the classification is applied directly without using column-based subspace sampling.

Table 4.9: CNNs, KNN, SBCS and SRC comparisons on CIFAR-10 dataset using features extracted by AlexNet.

Noise variance	**CNN**(%)	K**NN**(%)	**SRC**(%)	**SBCS**(%)
0	**81.87**	68.79	78.53	73.40
0.01	47.60	52.77	52.51	**58.36**
0.02	36.37	42.39	43.80	**46.98**

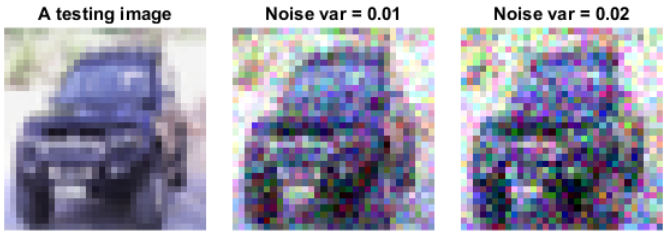


Figure 4.15: An example of an image from CIFAR-10 with Gaussian noises.

Table 4.9 gives the performance of each classifier. Zero-mean Gaussian noises with different noise variances are added on these training images and testing images. Note that here the noise level is lower than the noise added in the VLR dataset. The reason for this is the images in CIFAR-10 are tiny colour images. A small colour image can be easily contained by adding up the noise effects from each channel. Figure 4.15 illustrates the effect of the noise contamination. Similar to the traffic scene dataset, the result shows that the CNN classifier is not robust to noise. However, using the features extracted by the CNNs and applying it to other classifiers could achieve better accuracy. This is important as clear images are not always guaranteed in real applications. Table 4.9 also shows that SRC should achieve good accuracy when the images are noise free, even if only 4% training images are applied. However, when the images are noisy, the SBCS algorithm achieves the best accuracy.

4.6 Summary

In this chapter, a framework based on spatial SIFT features combined with LR has been proposed for VLR. The spatial SIFT features, which include the geographical knowledge of SIFT features, are more robust than both SIFT and HOG features in noisy images. Three classifiers (SVM, LR, and KNN) are tested and the LR shows an overall higher accuracy than both the SVM and KNN on this dataset. The proposed framework achieved an average recognition accuracy of 99.93%, which exceeded the previous record.

This chapter also proposes a novel non-parametric classification approach, namely the BBCS classifier. The novelty of the work has two main components: *i*) the proposed back propagation process, *ii*) the proposed column-based subspace sampling to reduce the size of the dataset and associated computation costs. The developed approach relies on the construction of the testing image using partial information from the weights estimated by BCS. Note, that for each class there is a corresponding reconstructed image. By comparing the reconstructed images with the testing image, the objects of interest are reconstructed and classified. The proposed back-propagation process gives a significant reduction of the misclassification error. For VLR, the number of misclassified testing images reduces by 87% when compared with the BCS classifier. Compared with the SRC, the SBCS algorithm gives a similar recognition accuracy while decreasing the mean computational time by 68%. However, the SRC does not work when the training dataset is small while the SBCS algorithm gives a good performance in the same situation. Moreover, the KNN classier is the most often applied non-parametric classifier for VLR and many other applications. However, the proposed classifier and column-based subspace sampling have been shown to be robust to the effects of heavy noise, unlike the KNN classifier. The proposed approach is also valid on traffic scene recognition dataset and the CIFAR-10 image dataset by using CNNs as feature method.

Chapter 5

Learning Capsules and Joint Frameworks

5.1 Introduction

In real applications, images can be damaged by different causes; for instance, an image can be corrupted by occlusion, blurring, noise, etc. Recognising these images usually requires an image restoration process in advance. The aim of image restoration is to recover an image from its corrupted version. Restoration methods are usually developed only for a particular image degradation, for example, BM3D [95] is only designed for image de-noising and the work in [99] is only designed for image super-resolution. Recently, the deep learning based methods, such as CNNs, achieved state-of-the-art results in image de-noising and super-resolution. However, image occlusion and orientation are common and challenging problems but less explored by the state-of-the-art deep learning methods.

Conventional methods use the restoration and classification pipeline, in which a restoration stage is followed by a classification process [4]. However, a joint framework that simultaneously performs recognition and restoration would be valuable. This chapter aims to develop such a joint framework, in which recognition and restoration share weights and both tasks are performed simultaneously. Figure 5.1

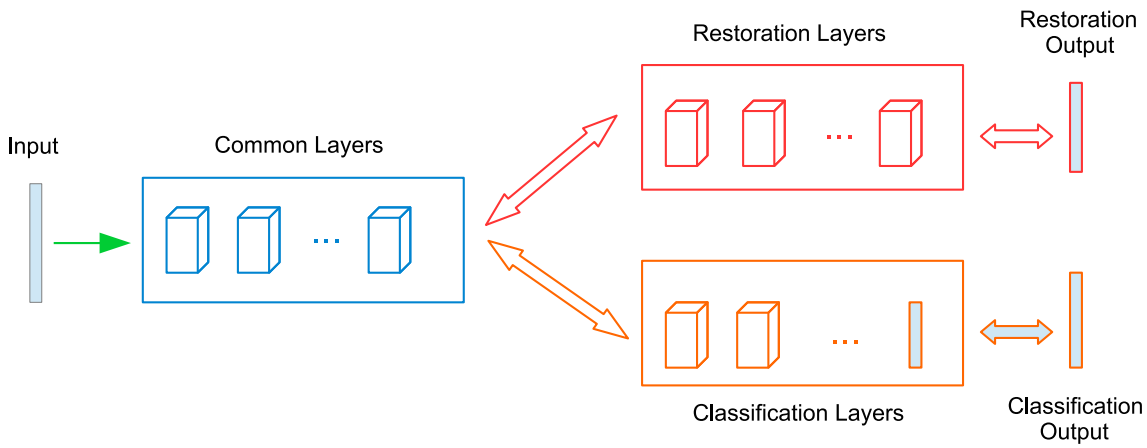


Figure 5.1: A general joint framework for image restoration and recognition.

illustrates the general architecture of a joint framework. In the restoration part, image rotation and occlusion are considered. The proposed joint framework could perform classification, remove the noise, correct a rotated image and recover an occluded image simultaneously. In Section 5.2, the joint framework based on CNNs is introduced. Section 5.3 explains the mechanism of the capsule networks and how to extend the capsule networks for the restoration task. Section 5.4 evaluates the performance and Section 5.5 summarises this chapter.

5.2 Joint Framework for Recognition and Restoration by Convolutional Neural Networks

The proposed joint framework composes both restoration and recognition processes by incorporating common layers, restoration layers and classification layers. All common layers and restoration layers are generated by convolutional processes, while classification layers consist of both pooling layers and convolutional layers. For image restoration, which requires the output matrix to have the same size as the input matrix, zero padding convolutions are applied to the common layers and restoration layers. For the same reason, no pooling process shall be applied for these layers. On the contrary, classification needs the pooling process and the valid padding convolu-

tion process in order to decrease the size of the feature maps.

Common layers are layers shared by both restoration and classification tasks. In the forward propagation, the corrupted images (input) are convolved with kernels in the common layers and restoration layers. For each input image, the restoration error function is given:

$$\mathbb{L}_{res} = \frac{1}{nm} \sum_{i=1}^{i=n} \sum_{j=1}^{j=n} \left(R(i,j) - \mathcal{I}(i,j) \right)^2 , \tag{5.1}$$

where $R(i,j)$ is the predicted value at the location index (i,j) in the last restoration layer and the $\mathcal{I}(i,j)$ is the ground truth training image intensity at the location index (i,j).

The loss function for the classification is the cross-entropy loss is defined as:

$$\mathbb{L}_{cla} = \sum_{i} \left(-y_i ln(\hat{y}_i) - (1 - y_i) ln(1 - \hat{y}_i) \right) , \tag{5.2}$$

here $\mathbf{y}$ and $\hat{\mathbf{y}}$ are the ground truth label and the predicted label for an image in the one-hot coded matter (a vector only contain a value of 1, the rests are zero valued), with the i^{th} entry denoted as y_i and $\hat{y}_i$. The gradient descent method is then applied to minimise $\mathbb{L}_{res}$ and $\mathbb{L}_{cla}$ for the three-pathway framework. Hence, the total loss function is given as:

$$\mathbb{L}_{total} = \mathbb{L}_{cla} + \lambda \mathbb{L}_{rec}, \tag{5.3}$$

where λ is the controlling factor for the restoration and classification. Then the weights in both section could be updated according to the gradient. Algorithm 5 summarises the process of a joint framework.

Figure 5.2 illustrates a proposed joint framework based on CNNs. The joint framework is composed of two common layers, two restoration layers and five classification layers (three pooling layers and two convolutional layers). In order to keep the image size, kernels of the size $[5 \times 5]$ are applied to all the common and restora-

Algorithm 5 Training process of a joint framework

Input:

The original training images and labels.

The designed joint framework. The image degradation parameters.

Output:

Random initialise the convolutional kernels.

Separate the training data into small batches.

1: **for** *iteration index*=1, *iteration index* ++ **do**
2: **for** *batch index*=1, *batch index* ++ **do**
3: Contaminate the source image with image degradations such as rotation and occlusion.
4: Run the network forward using the degraded images.
5: Calculate the total loss, combined by the classification loss and the restoration loss.
6: Calculate the gradient of all the weights and update the weights.
7: **end for**
8: Decrease the learning rate.
9: **end for**
10: **return** The weights in the common layers, restoration layers and classification layers.

tion layers with a padding size of [2 2]. For recognition, two convolutional processes and three pooling processes are applied. A fully connected layer is then connected with the output label. The softmax process is applied in the last stage of the classification; hence, the output can present the probability of the input data belonging to the corresponding class.

Notice that the loss function in the restoration is calculated from pixel to pixel; this is naturally fragile to image rotation, as the rotation does not change the content while involving a huge variation in the loss function. In fact, the rotation can seriously influence the classification accuracy in CNNs as its inherent properties of the spatial pooling process [125,126]. A recent idea of capsules has been proposed by Sabour et al. [127] in order to deal with the limitations of CNNs. A capsule is a group of neurons, whose length represents the probability of the object's (or part of the object) existence, and the orientation represents the instantiation parameters [127]. Compared with a convolutional process which transfers scalar inputs to scalar out-

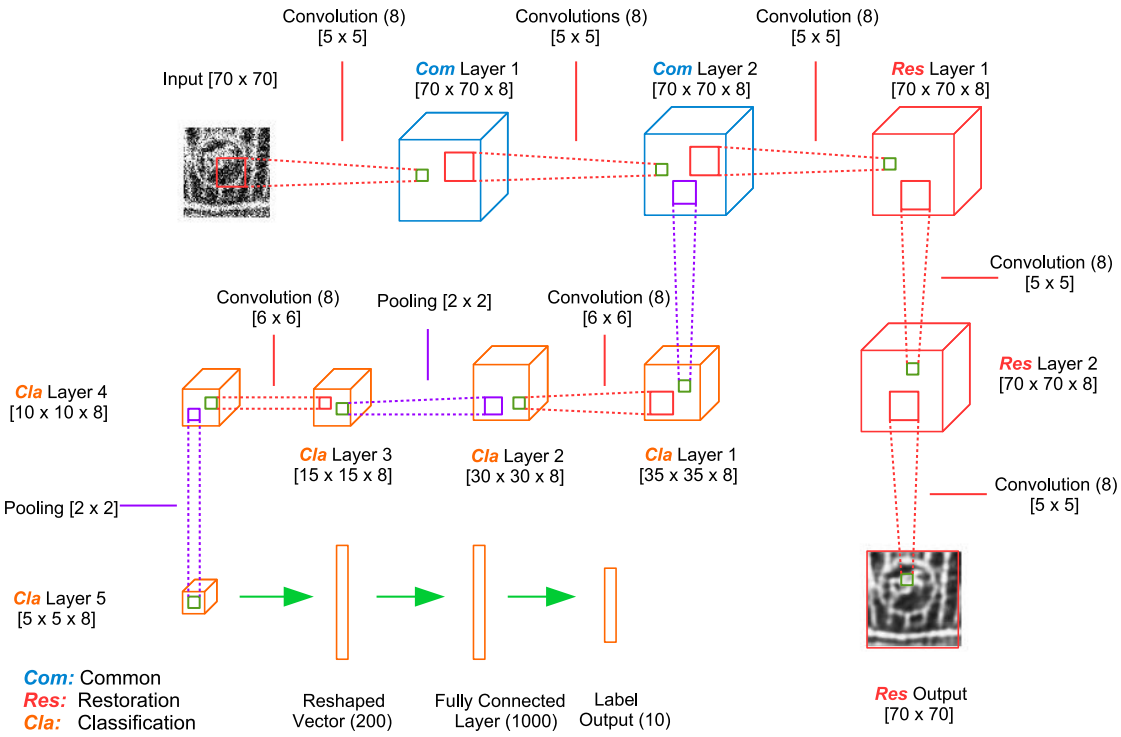


Figure 5.2: The joint CNN framework for image recognition and restoration.

puts, a capsule transfers data from a group of neurons to a group of neurons between adjacent capsule layers. Instead of using the Max-pooling process which only finds the local response from an individual layer, a routing process is applied in capsule networks in order to detect active capsules cross layers. Using a routing process, each capsule predicts the output of higher level capsules. A lower level capsule becomes active if its prediction agrees with the true output of higher level capsules using a dot product measurement. In the last fully connected capsule layer, weights are optimised by a margin loss function. In the following, the capsule networks [127] are introduced, with a proposed joint framework based on the learning capsules.

5.3 Learning Capsules for Recognition and Restoration

In CNNs, connections between layers and layers are scalar-scalar. However, in a capsule network, a group of neurons are combined in order to present an object or part

of an object. Therefore, a neuron is replaced with a group of neurons and the connections between capsule layers become vector-vector. For each capsule (represented as a vector), a non-linear squash function $f(\cdot)$ is defined:

$$f(\mathbf{x}) = \frac{||\mathbf{x}||_2^2}{1 + ||\mathbf{x}||_2^2} \frac{\mathbf{x}}{||\mathbf{x}||_2}, \tag{5.4}$$

with $\mathbf{x}$ is the input vector of the squash function. This function makes the length of short vectors shrink close to 0 and long vectors shrink close to 1, notice that the length of the vectors is decided by the values in each input dimension. Hence, the output length can be used to represent the probability that an object (or part of an object) exists. The output of the capsule j ($\mathbf{v}_j$) is given by:

$$\mathbf{v}_j = f(\mathbf{h}_j), \tag{5.5}$$

where $\mathbf{h}_j$ is the input of the capsule j. Parameters in each capsule represent various properties such as position, scale and orientation of a particular object (or part of an object) [127].

Except the capsules in the first capsule layer, the total input of the capsule $\mathbf{h}_j$ is a weighted sum of all "predictions" $\mathbf{o}_{j|i}$ (the predicted output of capsule j in the current layer by the input capsule i from the previous layer) is given by:

$$\mathbf{h}_j = \sum_i c_{ij} \mathbf{o}_{j|i}, \tag{5.6}$$

where c_{ij} are coefficients determined by a routing process. Let q_{ij} denote the log prior probabilities that the capsule i (in the previous layer) is coupled with the capsule j (in the current layer); the coefficients c_{ij} can then be denoted as:

$$c_{ij} = \frac{exp(q_{ij})}{\sum_d exp(q_{id})}, \tag{5.7}$$

where the index d refers to capsules in the current layer. q_{ij} are initialised with zeros

and updated by a routing algorithm. In the routing algorithm, q_{ij} is updated by the following process:

$$q_{ij}^{(r+1)} = q_{ij}^{(r)} + \left\langle \mathbf{v}_j, \mathbf{o}_{j|i} \right\rangle, \tag{5.8}$$

where r is an iteration index. Note that the term $\left\langle \mathbf{v}_j, \mathbf{o}_{j|i} \right\rangle$ is the inner product of the predicted output and its actual output (of the capsule j in the current layer). The assumption is intuitive as for the capsule j in the current layer, all capsules from the previous layer will predict its value. If the prediction made by the capsule i from the previous layer is similar to the actual output $\mathbf{v}_j$, the capsule i should have a high probability of the contribution. Hence, the coupling coefficient c_{ij} increases.

In equations (5.6) and (5.8), the predictions $\mathbf{o}_{j|i}$ can be calculated by the output capsules $\mathbf{u}_i$ from the previous layer:

$$\mathbf{o}_{j|i} = \mathbf{W}_{ij}\mathbf{u}_i, \tag{5.9}$$

where $\mathbf{W}_{ij}$ are transformation matrices connecting capsules between two adjacent layers.

Suppose there are C classes, then the final capsule layer has C capsules, with the length of each capsule representing the existence probability of the corresponding object. To allow multiple classes exist in the same image, a margin loss is used, with the loss $\mathbb{L}_i$ for the class i ($i = 1, 2, \cdots, C$) is given by:

$$\mathbb{L}_i = y_i \; max(0, m^+ - ||\mathbf{v}_i||_2)^2 + \lambda(1 - y_i) \; max(0, ||\mathbf{v}_i||_2 - m^-)^2, \tag{5.10}$$

where $y_i = 1$ if and only if the object of the class i exists and $||\mathbf{v}_i||_2$ is the length of the vector $\mathbf{v}_i$ in the final capsule layer. This encourages the length of the capsule $\mathbf{v}_i$ to be above m^+ if an object of the class i is present, and encourages the length of the capsule $\mathbf{v}_i$ to be below m^- when an object of the class i is absent. Here λ is a controlling parameter and the total classification loss is calculated by $\mathbb{L}_{cla} = \sum_i \mathbb{L}_i$, which simply sums the losses from all the final layer capsules.

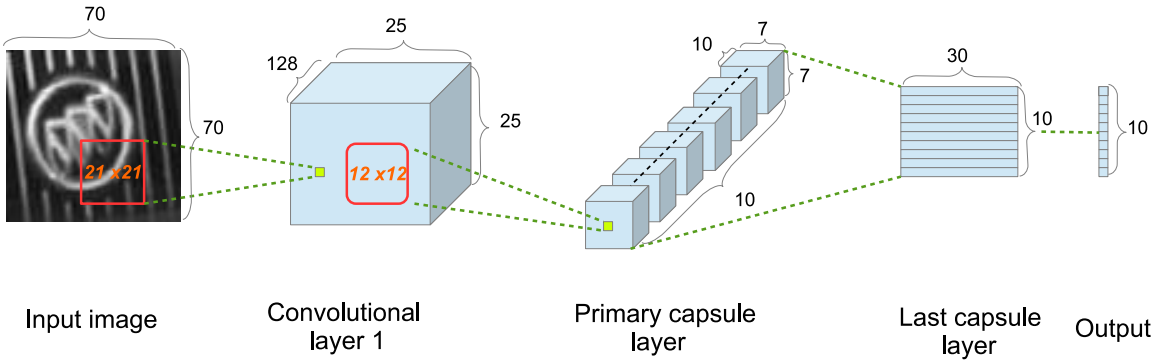


Figure 5.3: The architecture of capsule networks (similar to [127]).

In capsule networks, the back-prorogation is applied to update the convolutional kernels and the transformation matrices. A routing process is applied to update the weights for the coupling coefficients c and the log prior probabilities q. In capsule networks, the vector-vector transformation could potentially extract more robust features than scalar-scalar transformation in CNNs. In order to illustrate the general architecture, a developed capsule network's architecture for VLR is shown in Figure 5.3. It contains two convolutional layers and one fully connected layer. The size of the first convolutional kernels is $[21 \times 21 \times 128]$ ([height $\times$ width $\times$ depth]), and a convolution operation is applied with a stride of two, followed by a ReLU non-linear activation function. Hence, the output size of the first convolutional layer is $[25 \times 25 \times 128]$.

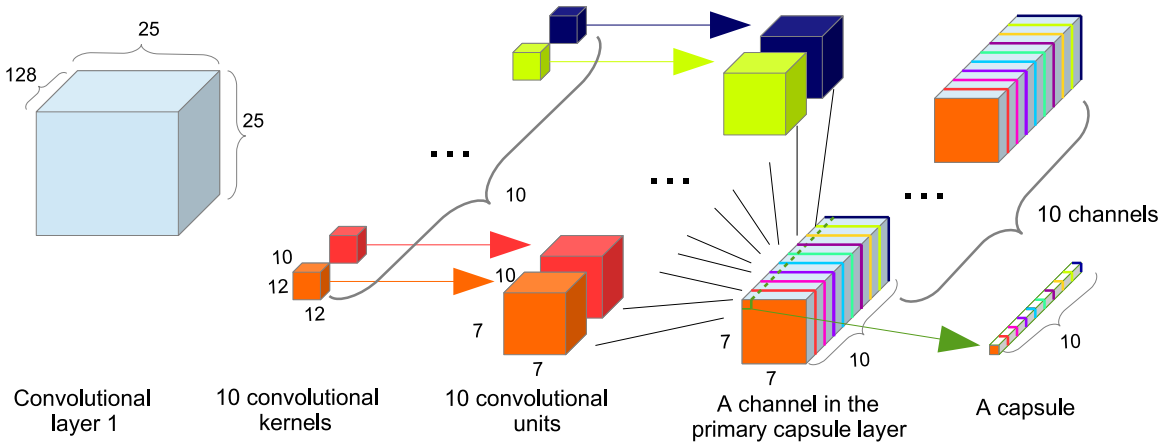


Figure 5.4: The capsule generation process in the proposed primary capsule layer.

The second convolutional process generates the primary capsule layer. Figure 5.4

illustrates the capsule generation process from a convolutional layer. There are ten groups of convolutional kernels, each is of the size $[12 \times 12 \times 10]$ and is applied with a stride of two. This process generates ten convolutional units, each of them is of the size $[7 \times 7 \times 10]$. These units are re-grouped into ten channels, each channel containing one layer from all convolutional units. Each channel is made up from $7 \times 7 = 49$ capsules with each capsule being a vector with ten entries. The primary capsule layer connects with the final capsule layer by transformation matrices. This process is the same with a fully connected layer in neural networks, except the scalar-scalar transform is changed to a vector-vector transform.

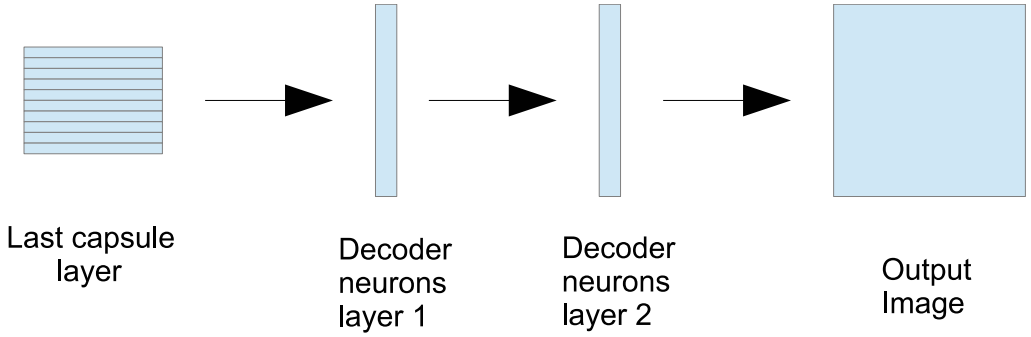


Figure 5.5: The reconstruction process of the capsule networks.

The reconstruction process is a decoder using neural networks, with each layer having 2048 neurons. By combining the recognition and reconstruction, the joint framework could perform both recognition and reconstruction. However, if the input is a degraded image, the capsule network could potentially be developed for the restoration purpose. The joint framework based on capsule networks is named as Joint-Cap-Net in this thesis. The training process of the Joint-Cap-Net is summarised in Algorithm 6. The routing process is applied only between the primary capsule layer and the final capsule layer. There are $7 \times 7 \times 10 = 490$ capsules in the primary capsule layer and ten (C=10) capsules in the final capsule layer. This requires 4900 transformation matrices with the size of $[10 \times 30]$. The length of each capsule in the final capsule layer represents the existence probability of the corresponding object.

Algorithm 6 The training process of the proposed Joint-Cap-Net

Input:

Input images.

Number of iterations r for the routing algorithm.

Output:

A convolutional operation with ReLU is applied to the input images.
A convolutional operation is applied to the convolutional layer 1 using convolution kernel groups.
Reshape the primary capsule layer to capsules $\mathbf{u}_i$, each $\mathbf{u}_i$ is squashed by the function in (5.4).
Define a final capsule layer.
Define a corresponding restoration network.
For all capsule i in the primary capsule layer and capsule j in the final capsule layer: initialise q_{ij} to zeros.
for r iterations **do**
    for all capsule i in the primal capsule layer, apply equations (5.7) and (5.9) in order to get c_{ij} and $\mathbf{o}_{j|i}$.
    for all capsule j in the final capsule layer, apply equation (5.6) in order to get $\mathbf{h}_j$.
    for all capsule j in the final capsule layer, apply equation (5.5) in order to get $\mathbf{v}_j$.
    update all q_{ij}, apply equation (5.8).
    return $\mathbf{v}_j$
end for
Calculate the loss and update the weights.

5.4 Performance Evaluation

In order to compare the CNN classifier with the joint framework based on CNNs, it would be fair to use the same CNN architecture. Hence, the architecture in Figure 5.2 is applied as the designed CNN architecture in both scenarios, with the restoration parts deleted for the recognition task. Another reason of not using big CNN frameworks such as AlexNet [1] and VGG [53] is that there are too many parameters in big CNN frameworks, which cause over-fitting problems. Meanwhile, the proposed CNN framework has proved to be a good model as it has achieved more than 99% accuracy on the testing dataset while containing many fewer parameters than in AlexNet and VGG. In the proposed capsule network, there are three iterations

in the routing process and the restoration loss is considered for the weights updating. The largest open VLR dataset provided by Huang et al. [50] is used to evaluate the proposed recognition approach. The performance evaluations of both the CNNs and the capsule networks are conducted in Python with the Pytorch toolbox on a laptop with the following specification: Intel CPU I5 and Nvidia GTX 1070 (extended GPU). The performance of each method is measured in terms of accuracy (percentage of correctly classified images) on the entire testing dataset (1500 images).

5.4.1 Capsule Networks for Classification

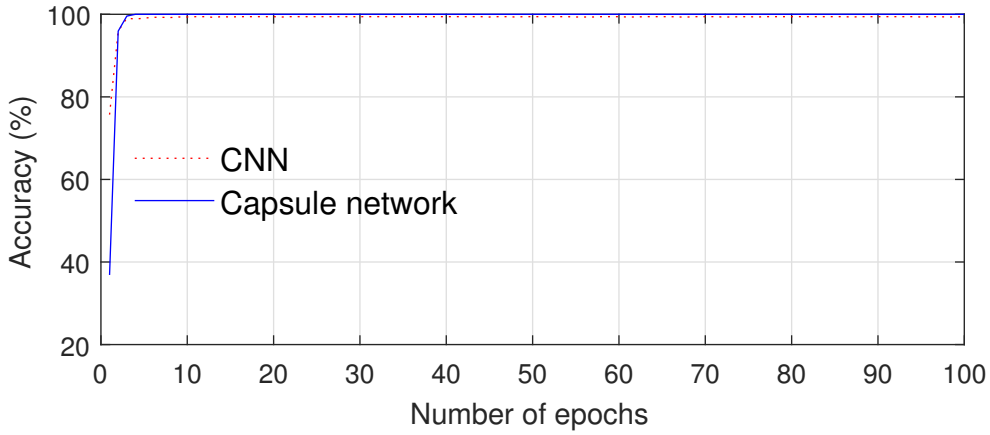


Figure 5.6: The accuracy of the CNNs and the developed capsule networks on the original testing data.

The model is trained on the original training dataset with 100 epochs. In each training epoch, the corresponding testing accuracy is recorded in order to give more detailed results. Figure 5.6 illustrates the testing accuracies in each training epoch, up to 100 epochs. Both the CNNs (accuracy of 99.42% at the 100^{th} epoch) and the capsule networks (accuracies keep at 100% after the 4^{th} epoch) can achieve good results after a limited number of epochs. However, the advantage could not be significant as there is a limit space for the improvement. Hence, image degradations are applied to evaluate the trained CNNs and the capsule networks.

Figure 5.7 illustrates the effects of adding noise and rotation to 20 random testing images. The first two rows are original testing images and the third and fourth rows

are the effects of adding zero-mean Gaussian noises with the variance of 0.1. The fifth and sixth rows are the effects of images that are rotated randomly within the angle within -50^o to 50^o. The last two rows are the combined effects of noise and random rotation. These scenarios are tested in order to compare performances of the developed CNNs and the developed capsule networks.

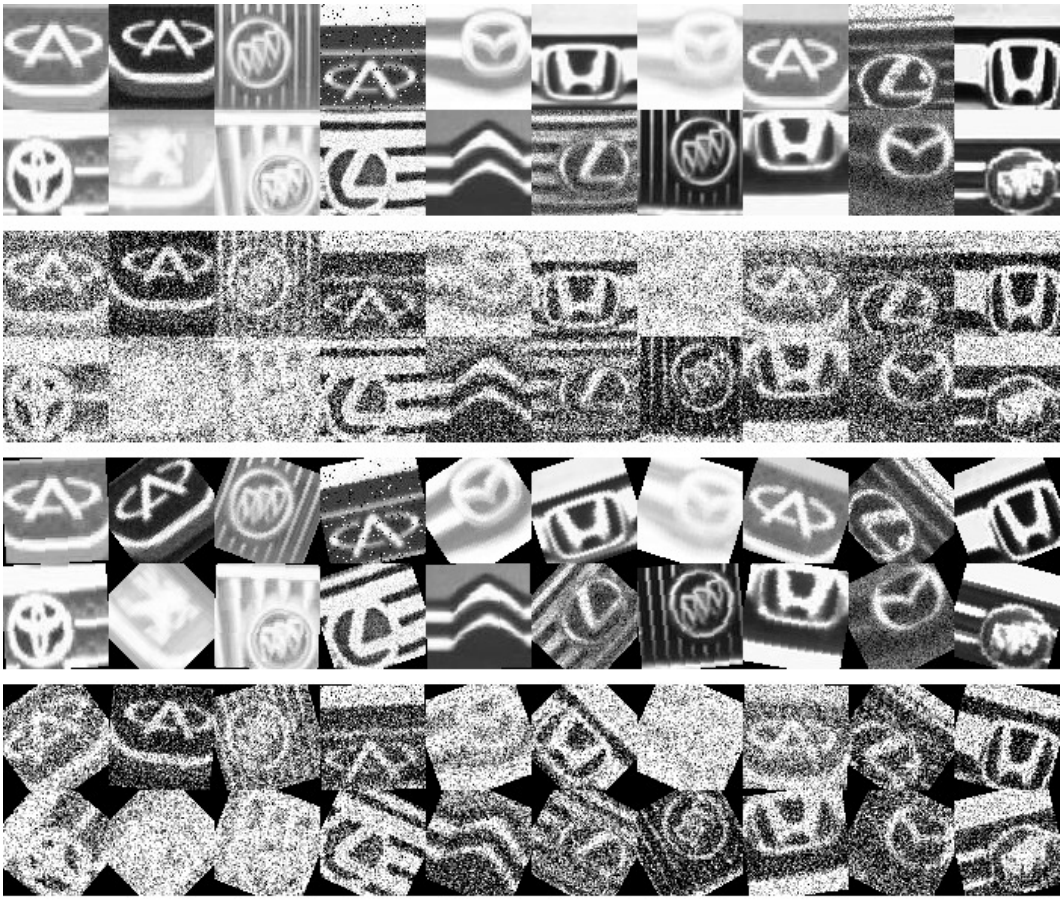


Figure 5.7: Illustration of rotation and noise effects on 20 random testing images. The first and second rows are clear image, the third and fourth rows are the noise effects on clear image, the fifth and sixth rows are the effects by adding rotation, and the last two rows are the effects with both noise and rotation.

Figure 5.8 shows the accuracies of the CNNs and the developed capsule networks on the challenge testing dataset. Both accuracies have dropped because of the testing images become more challenging. However, the capsule networks are more robust to noises and rotation variations. Compared with the CNNs which achieve an accuracy of 81.02% after 100 training epochs, the capsule networks achieve an accuracy of 98.49% in the same scenario. In terms of robustness to rotation, the capsule networks

achieve 63.55% while the CNNs achieve 51.40% after 100 training epochs. When the noise and rotation are combined, the testing images become even more challenging. In this case, the capsule networks achieve an accuracy 59.09% and the CNNs achieve an accuracy of 39.47% after 100 training epochs.

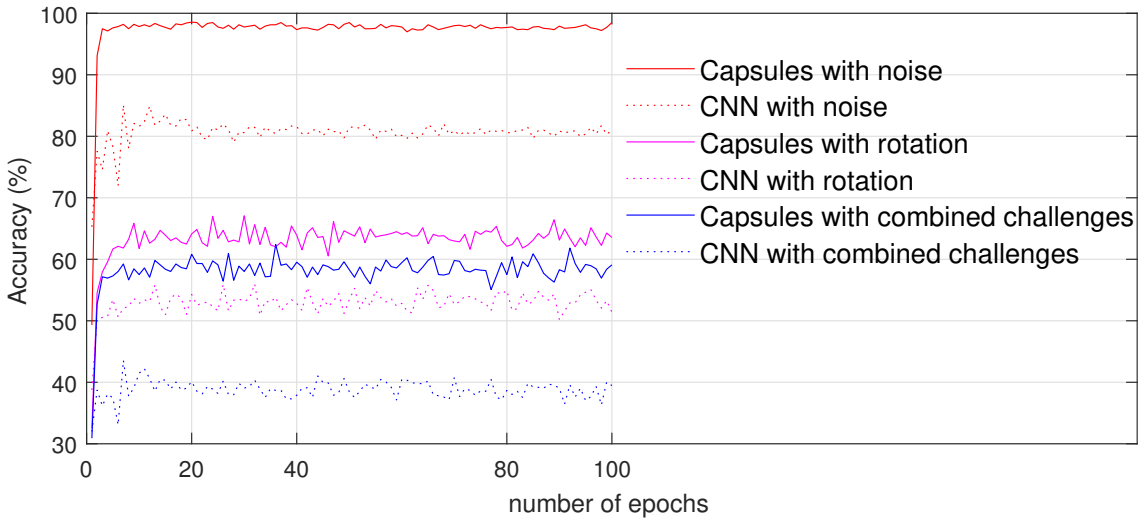


Figure 5.8: The accuracy of the CNNs and the developed capsule networks on the challenge testing dataset with noise, rotation and the combined challenges.

The reason for decreasing performances in both frameworks is the images become more challenging; this results in the features extracted in the testing images not being representative. Another important reason is the weights are learned from clear training images. This results in the automatic learned features not considering changes such as noise and rotation. However, the capsule networks perform much better than the CNNs in the same scenarios. This indicates the capsule networks automatically learn more robust image features than the CNNs. One possible solution in order to improve the robustness of features is to add challenging data in the training dataset. However, when there are degraded images like the images in the last two rows in Figure 5.8, it would be also good to recover them to their clear versions. In the following section, joint frameworks for both recognition and restoration are developed.

5.4.2 Joint Framework for Image Recognition and Restoration

In this subsection, the Joint-CNN-Net and the Joint-Cap-Net are evaluated with noise, rotation and occlusion. In the joint frameworks, all the degraded training images are used as the input and the ground truth image and labels are used to update the weights. All the accuracies are evaluated using the model generated at the 100^{th} epoch in the training stage. The Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM) index [128] are applied with each original testing image being compared with its degraded version (O-D) and its recovered version (O-R). PSNR (ranges from 0 to 1, the higher the better) and SSIM (ranges from -1 to 1, the higher the better) are measurements for comparing the differences between two images. The PSNR focuses on the difference of the pixel-pixel intensity values and the SSIM considers the structure image within an image [129]. Figure 5.9 illustrates 20 testing images, which will be used for demonstration purpose.

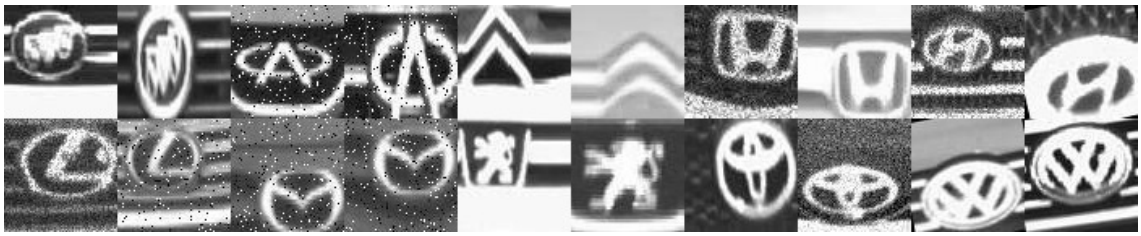


Figure 5.9: Twenty testing images for illustration purpose.

Noise Robustness Evaluation

The first two rows in Figure 5.10 show the noisy effects when noises are added to 20 testing images. All the images are noisy by adding zero-mean Gaussian noises with the variance of 0.1. In the training stage, the degraded noisy images are the input of both frameworks, the original training images and their corresponding labels are the ground truth for updating the weights. In the testing stage, noisy images are used to evaluate the trained model. The middle two rows illustrate the covered image by

the Joint-CNN-Net, and clearly the noisy effects have been removed. The last two rows illustrate the restoration effects by Joint-Cap-Net. Clearly, the noise effects have also been removed.

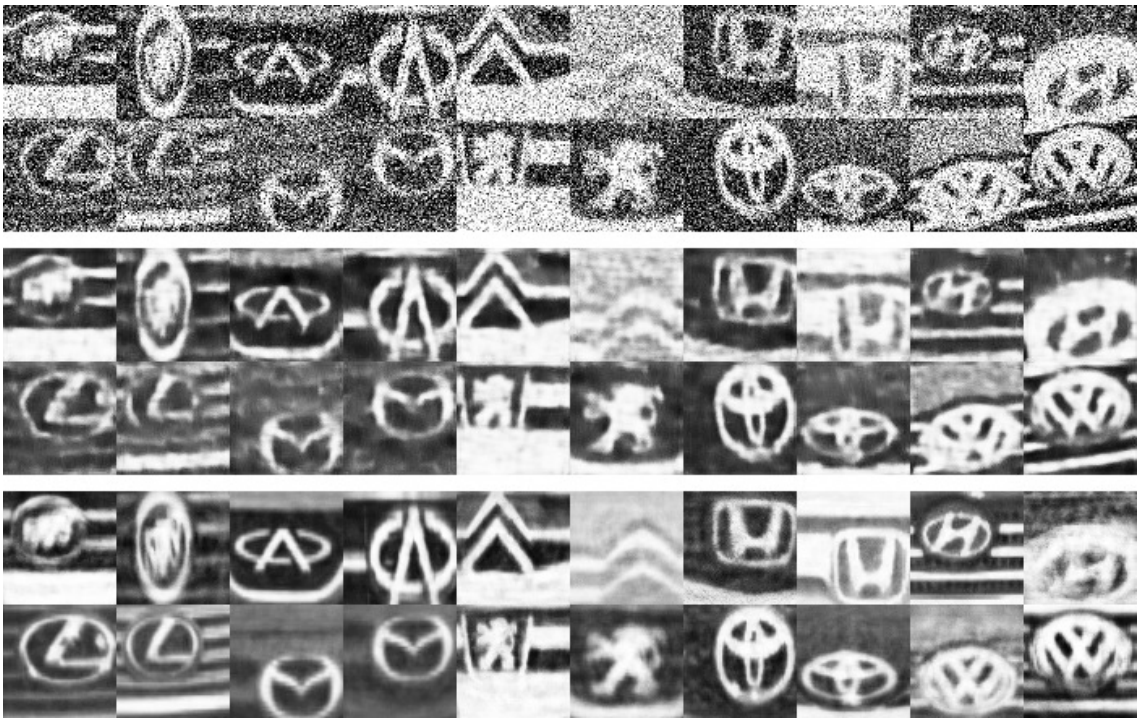


Figure 5.10: De-noising results by the Joint-CNN-Net and the Joint-Cap-Net.

By comparing these restored images with their corresponding ground truth as shown in Figure 5.9, some of the recovered images by Joint-Cap-Net have even better visual quality than the ground truth. For example, the two ground truth "Lexus" images have slight noise inside, while the Joint-Cap-Net removes these noise in the recovered images. For image de-noising, both frameworks achieve good results. The Joint-Cap-Net is not always better than Joint-CNN-Net, for instance, the second "Hyundai" image recovered by the Joint-Cap-Net has a much lower visual quality than the recovered image by Joint-CNN-Net. Notice that the last "VW" image has been rotated automatically. These are due to the 2D convolutional kernels preserving the spatial information in CNNs while the capsules are not restricted on pixel-to-pixel recovery.

Table 5.1 summarises the performance of the Joint-CNN-Net and the Joint-Cap-

Table 5.1: Performance of the Joint-CNN-Net and the Joint-Cap-Net on noisy images.

	Accuracy%	PSNR (O-D)	PSNR (O-R)	SSIM (O-D)	SSIM (O-R)
Joint-CNN-Net	96.83%	12	21.18	0.32	0.61
Joint-Cap-Net	**99.57**%	12	**22.16**	0.32	**0.65**

Net. The Joint-CNN-Net achieves an accuracy of 96.83% with noisy images. Meanwhile, the Joint-Cap-Net is slightly more robust to noise and 99.57% accuracy is achieved under the same noise condition. The PSNR and SSIM are both improved by the restoration process due to the noisy effects having been removed. The improvement in PSNR and SSIM indicates the recovered images are more similar to the original images than the noisy images. Compared with the PSNR and SSIM index, the Joint-Cap-Net has better performance than the Joint-CNN-Net. The PSNR and SSIM only indicate how "similar" two images are. However, both indices are based on pixel-pixel comparison. This would be unfair if the Joint-Cap-Net changes the rotation of an image. Even in this case, the PSNR and SSIM of Joint-Cap-Net are higher than in Joint-CNN-Net.

Rotation Robustness Evaluation

Since the Joint-Cap-Net shows the ability of automatically rotating an image, different rotation angles are tested for both the Joint-CNN-Net and the Joint-Cap-Net. In both training and testing stages, rotated images are the input of the joint frameworks. Each image is randomly rotated within the maximum bounds of 20°, 40°, 60° and 80°.

Figure 5.11 shows the rotation restoration results of the Joint-CNN-Net by setting the rotation angle bound to 40°. The first two rows are the input images and the last two rows are the corresponding restoration outputs. The restoration results show the recovered images are becoming blurred with the main objects keeping the rotation unchanged. Figure 5.11 only shows the effects when a maximum rotation of 40° is applied; higher bounds would result in more blurred effects on the recovered images.

Figure 5.11: The rotation restoration result of the Joint-CNN-Net when the rotation angle bound is 40°.

Since the loss function of the restoration is based on pixel-pixel, every pixel in the restored image is forced to be close to the ground truth value. However, the correct mapping between the input pixels and the restoration pixels has been changed when the input image is rotated. This mapping distortion requires that an input pixel be close to both the corresponding ground truth pixels and its neighbourhood pixels. Hence, the restoration image becomes blurred.

Table 5.2: Performance of the Joint-CNN-Net on rotated images.

Angles	0	20°	40°	60°	80°
Accuracy	98.92 %	97%	95.83 %	91.88%	91.23%
PSNR (O-D)	100	16.36	11.39	10.48	9.87
PSNR (O-R)	**30.79**	15.21	13.16	12.52	12.23
SSIM (O-D)	1	0.35	0.19	0.14	0.11
SSIM (O-R)	**0.89**	0.30	0.12	0.07	0.05

The corresponding accuracy, PSNR and SSIM are given in Table 5.2. The PSNR and SSIM in Table 5.2 indicates the performance of Joint-CNN-Net is bad. The SSIM (O-R) is even lower than SSID (O-D) in the Joint-CNN-Net when images are rotated. This means the recovering process even decreases the image similarity when compared with the original image degradation. According to the result, the CNNs are not suitable for image rotation recovering.

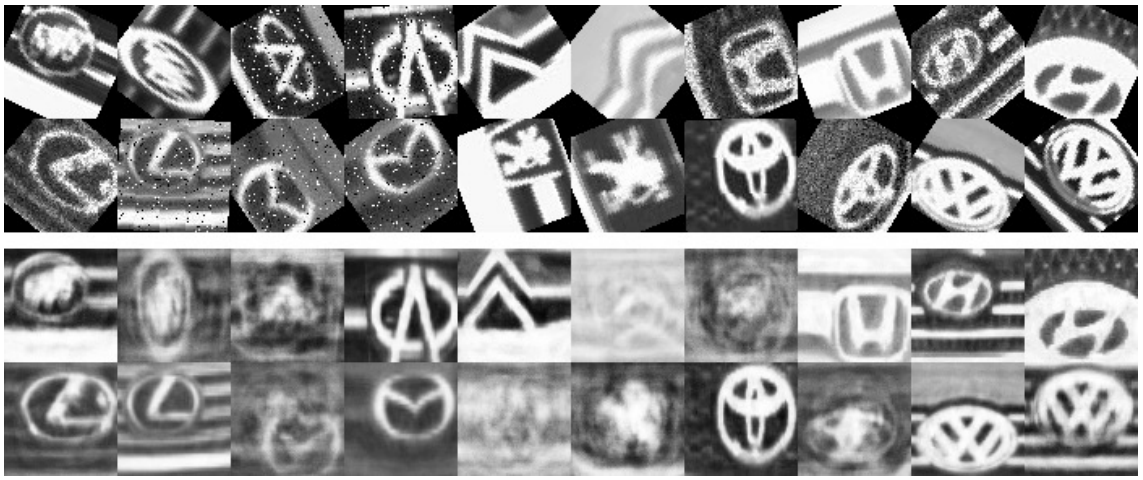


Figure 5.12: The rotation restoration result of the Joint-Cap-Net when the rotation angle bound is 80°.

Figure 5.12 illustrates the rotation restoration effects by the capsule networks with the rotation bound is set to 80°. Again the first two rows are the input images and the last two rows are the corresponding recovered images. Due to the good performance achieved by the Joint-Cap-Net, a maximum rotation up to 80° is illustrated for demonstration. As shown in Figure 5.12, the Joint-Cap-Net is able to recover the rotated images automatically when large rotations are applied.

Table 5.3: Performance of the Joint-Cap-Net on rotated images.

Angles	0°	20°	40°	60°	80°
Accuracy	**100**%	**100**%	**100** %	**99.78**%	**99.57**%
PSNR (O-D)	100	15.89	11.45	10.84	9.78
PSNR (O-R)	23.61	**22.26**	**21.65**	**20.55**	**20.84**
SSIM (O-D)	1	0.37	0.19	0.14	0.11
SSIM (O-R)	0.68	**0.65**	**0.63**	**0.59**	**0.60**

Different angles have been tested and the result is summarised in Table 5.3. When comparing the performance with Joint-CNN-Net shown in 5.2, the Joint-Cap-Net has much better performance in terms of accuracy and robustness to image rotations. For instance, when all training and testing images are randomly rotated with the angle range from $-60°$ to 60°, the Joint-Cap-Net achieves an accuracy of 99.78%, while Joint-CNN-Net could only achieve an accuracy of 91.88%. This is due to the

capsules extracting more robust features than the Max-pooling in CNNs. The PSNR and SSIM have been greatly improved. Notice that the PSNR and SSIM are pixel-pixel based calculations, therefore, improvement in Joint-Cap-Net does not mean better visual quality.

Occlusion Robustness Evaluation

In addition, the developed Joint-Cap-Net could recover images which are partly occluded. In both training and testing stages, occluded images are used as the input of the joint framework. A white square box of a random size is applied to cover image contents in order to simulate the occlusion effects. The occlusion boxes are randomly located in an image and the length of the box is a random integer varying from 0 to 30 pixels.

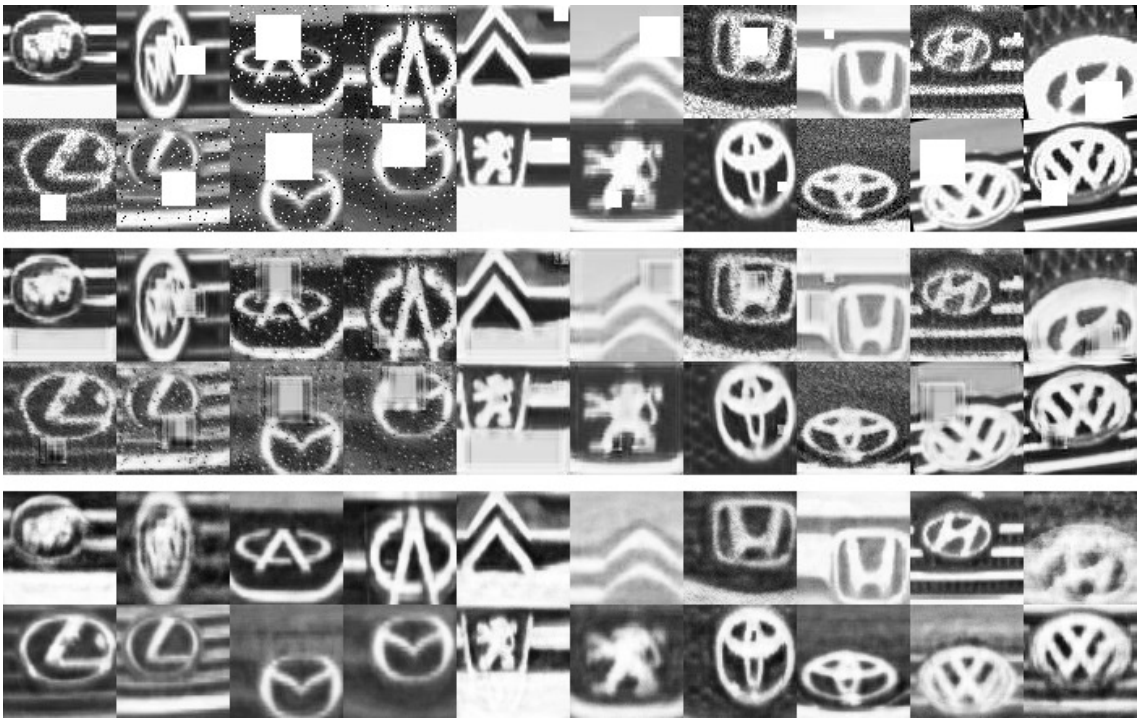


Figure 5.13: Occlusion restoration results by the Joint-CNN-Net and the Joint-Cap-Net with the maximum occlusion box of size $[30 \times 30]$.

The first two rows in Figure 5.13 show the effects when the occlusion boxes are added to 20 example images. The intermediate two rows show the recovered version of the corresponding testing images by the Joint-CNN-Net. As shown in Table 5.4,

the Joint-CNN-Net achieves an accuracy of 95.76%. The last two rows illustrate the corresponding recovered images by the Joint-Cap-Net and an accuracy of 99.78% is achieved. This indicates the Joint-Cap-Net achieves a slightly higher recognition accuracy than the Joint-CNN-Net.

Table 5.4: Performance of the Joint-CNN-Net and the Joint-Cap-Net on occluded images.

	Accuracy%	PSNR (O-D)	PSNR (O-R)	SSIM (O-D)	SSIM (O-R)
Joint CNNs	95.76%	26.36	22.08	0.92	**0.76**
Joint-Cap-Net	**100**%	26.92	**22.42**	0.92	0.67

In terms of the restoration result, the Joint-CNN-Net has limited recovery effects, and the white boxes become slightly transparent. However, the occlusion effects remain. In contrast, the Joint-Cap-Net has removed the blocking effects totally. The PSNR and SSIM of recovered images have decreased in both situations when compared with degradations. The reasons lie in the occlusion only changing a limited small area of the image, while the recovered images by the Joint-CNN-Net and Joint-Cap-Net change value on every pixel. However, the occlusion effects have been removed and the visual qualities have been improved by both frameworks, especially by the Joint-Cap-Net .

5.4.2.1 Mixed-degradation Evaluation

The first two rows in Figure 5.14 show the combined degradation including zero-mean Gaussian noises (variance =0.05), rotation (random angle from $-60°$ to $60°$) and the occlusion (a square white box with a random size from 0 to 30 pixels). The intermediate two rows show the recovered images by Joint-CNN-Net: clearly there is no more noise effect; however, the rotation and occlusion are not recovered. This results in the recovered images being difficult to distinguish by human vision. The last two rows illustrate the restoration result by the Joint-Cap-Net. The recovered images become blurred like the Joint-CNN-Net; however, it successfully recovers the

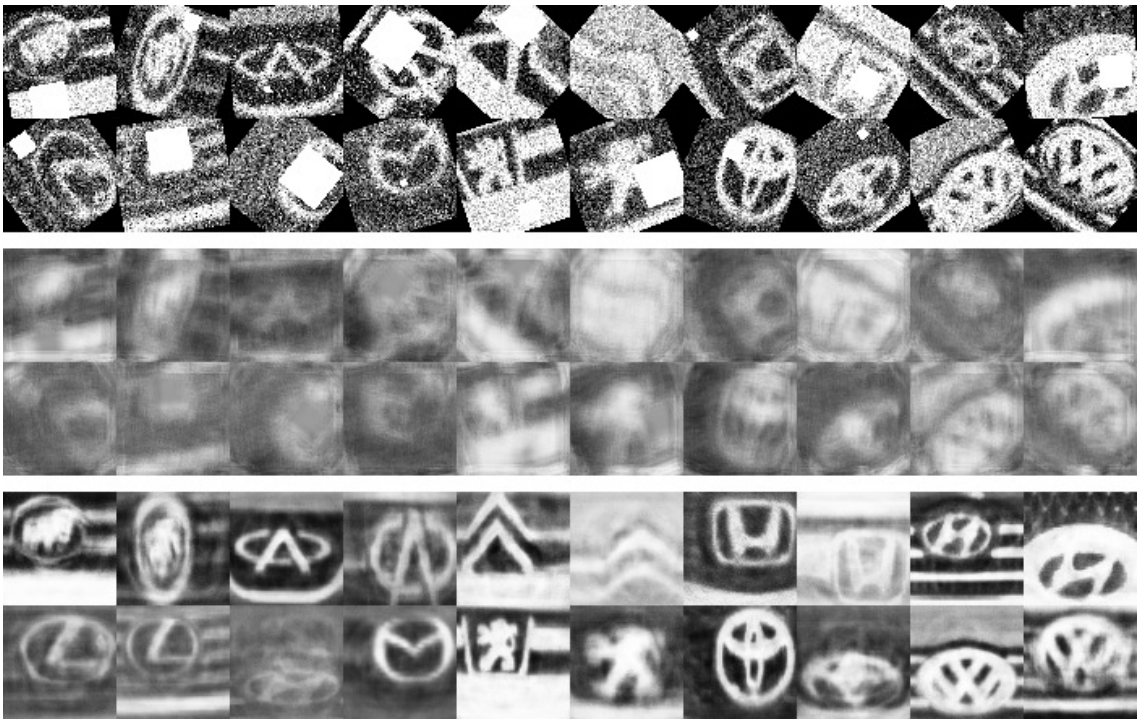


Figure 5.14: The restoration result of the Joint-CNN-Net and the Joint-Cap-Net with Gaussian noise, rotation and concussion.

rotation, occlusion and noise.

Table 5.5: Performance of the Joint-CNN-Net and the Joint-Cap-Net on combined degradations.

	Accuracy%	PSNR (O-D)	PSNR (O-R)	SSIM (O-D)	SSIM (O-R)
Joint-CNN-Net	66.57%	8.03	12.19	0.06	0.06
Joint-Cap-Net	**94.75**%	8.03	**18.37**	0.06	**0.49**

For the recognition, the Joint-CNN-Net achieves an accuracy of 66.57% and the Joint-Cap-Net achieves an accuracy of 94.75%. In terms of the restoration, the recovered images by the Joint-CNN-Net are poor as it is not suitable for image occlusion and image rotation. In contrast, the Joint-Cap-Net gives a good restoration result. Table 5.5 shows the corresponding improvement in term of PSNR and SSIM.

5.5 Summary

Image restoration and recognition are traditionally implemented in a pipeline manner. For example, noise images are first de-noised, followed by a recognition process. This chapter develops a joint framework for recognition and restoration. A joint framework has three parts: common layers, classification layers and restoration layers. Two joint frameworks are developed, one is based on CNNs and the other is based on the learning capsules. An image can be damaged by many degradations, such as noise, rotation and occlusion, this chapter also investigates the robustness of the two proposed joint frameworks. The capsule networks achieve higher recognition accuracy and better visual quality under different image degradations than the joint CNN framework on the VLR dataset. The key to the success of learning capsules is due to a more effective routing process rather than the pooling process in CNNs. In addition, the method is not only restricted on the VLR application but also has potential in other image recognition and restoration applications.

Chapter 6

Wildlife Monitoring Based on Deep Learning Methods

In previous chapters different methods have been developed for image recognition and restoration. This chapter extends image recognition to videos by incorporating a detection process. The application of this chapter focuses on badger recognition, which is helpful for studying the transmissions route of Bovine tuberculosis (bTB). Transmissions of the disease are mainly cattle-cattle, but also can be cattle-badger, badger-badger and badger-cattle. Therefore, an automatic badger monitoring system would be beneficial in order to enhance the understanding of transmissions of the disease. This chapter, for the first time, develops a deep learning based automatic recognition framework capable of identifying and isolating badger activity in still image and video footages. Section 6.1 gives an introduction on the background of badger recognition. Section 6.2 develops two CNN frameworks for badger recognition. Section 6.3 introduces the developed frame detection algorithm in videos. Experimental results are presented in Section 6.4. Section 6.5 summarises this chapter.

6.1 Introduction

Bovine tuberculosis (bTB) is described by government as the most pressing animal health problem in the UK. Taxpayers paid over 100 million pounds in 2015 alone in order to deal with bTB in England. The Strategy to Achieve Officially TB Free Status (OTF) for England was published in 2014 and established a framework for controlling all routes of transmission of the disease. The transmission of bTB is principally from cattle-cattle, but also includes cattle-badger, badger-badger and badger-cattle [130]. There remains a lack of clear understanding of how the disease is transmitted between cattle and badgers and vice versa [131]. Hence, monitoring the badger activity could help us to understand the transmission mechanisms and thereby to develop methods to deal with the transmission between species.

Similar research has been conducted for elephant monitoring [132]. However, this uses the colour information to segment the elephant and the background, which is not valid in badger videos. This is due to badgers being mainly active at night and the lighting conditions meaning that the image does not contain much colour information. Automatic video processing has also been trailed in the dairy sector to provide information on the mobility of dairy cows without requiring human intervention; the research team succeeded in showing how cows can be accurately located and tracked in video [133]. It uses the hand-crafted SIFT features while the-state-of-art deep learning features would give better results than SIFT features on big datasets.

While previous studies have proved that automatic recognition is feasible and adaptable to other animal species, they have limitations due to the insufficient image detection and image feature methods. This work aims at developing a robust and accurate automatic wildlife monitoring framework. The framework is tested on badger images but it could be also implemented in other wildlife species monitoring. Badger activity detection and monitoring is a challenging task for a number of reasons - the video data can be collected under different illumination conditions and

from cameras situated at different positions with respect to the monitored area, lack of colour information as badger images are usually captured at night, changeable weather and other factors. However, the ability to identify and isolate badger activity would have multiple benefits. In the short term, it would enable more efficient and cost effective analysis of existing footage. In the longer term, automated detection, using cheaper and more effective badger recognition on live-feed cameras, could increase the number of farms and locations that can be monitored. Further applications could include the development of a hand-held application, alerting farmers to badger activity on their farm. This would benefit disease control strategies by providing more efficient badger monitoring on farms to help farmers identify areas for intervention (e.g. to prevent badger entry or badger-cattle contact), thus reducing opportunities for disease transmission. In order to monitor the badger activities, this chapter develops deep learning based frameworks for image recognition; the developed recognition frameworks are combined with a developed detection process in order to only classify frames of interest in videos.

6.2 Deep Learning for Badger Recognition

In this section, two CNN frameworks have been proposed to recognise the badger images. The first framework is a self trained CNN framework based on the created badger dataset, which is named Badger-CNN-1. The second framework is a transferred framework based on AlexNet [1], followed by a fine-tuning process on the badger dataset, which is named Badger-CNN-2. Developing two different frameworks give options for users. For example, the size of the Badger-CNN-1 is about 20Mb but gives a less accurate result. In contrast, the size of the Badger-CNN-2 is about 200Mb.

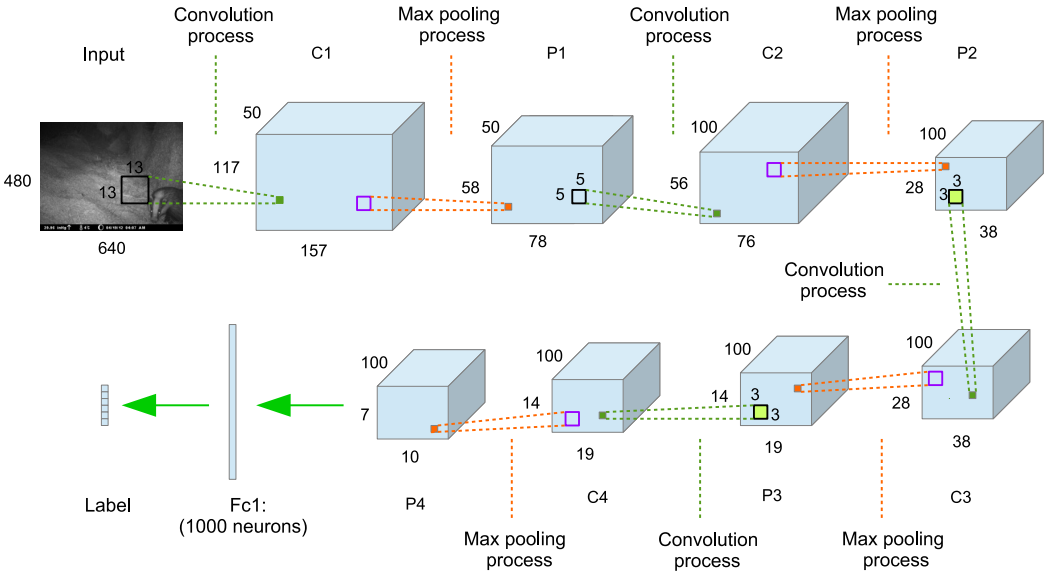


Figure 6.1: The architecture of the Badger-CNN-1.

6.2.1 Badger-CNN-1 Diagram

Figure 6.1 shows the architecture of the proposed deep learning framework Badger-CNN-1. There are four convolutional layers and four Max-pooling layers and a fully connected layer. The input image of size $[480 \times 640 \times 3]$ is transferred to 50 convolutional maps of size $[117 \times 157]$ in the first convolutional layer (C1). In the first pooling layer (P1), 50 pooling maps are generated based on C1. This transformation is achieved by using 50 convolutional kernels of the size $[13 \times 13]$ with a stride of [4 4].

The second convolutional process is applied on P1 by using 100 convolutional kernels; hence, 100 convolutional maps are generated in C2. The same process is repeated in P2, C3, P3, C4 and P4. In P4, there are 100 pooling maps with the size of $[7 \times 10]$. Elements in P4 are reshaped to a vector form of 7000 neurons, and these neurons are fully connected to 1000 neurons in the first fully connected layer (Fc1). Fc1 is then fully connected with the output neurons, which represent the corresponding label information. Algorithm 8 in the Appendices section gives the detailed steps of the Badger-CNN-1.

6.2.2 Badger-CNN-2 Diagram

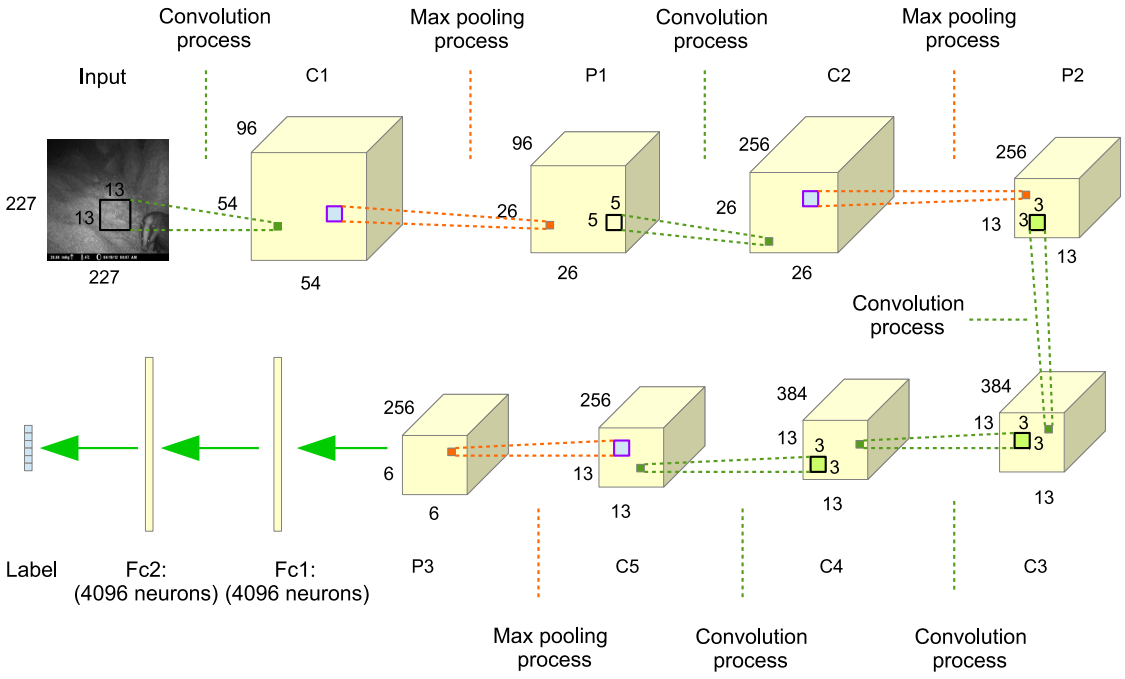


Figure 6.2: The architecture of the Badger-CNN-2.

In Badger-CNN-1, all weights are randomly initialised and updated based on a badger training dataset. However, there are ways to "borrow" weights from other trained models. Researches show that the CNNs learned from a large-scale dataset in the source domain can be effectively transferred to a new target domain [134,135]. The transfer learning uses the already trained weights as the initial weights, this gives a start point and the weights are then fine-tuned based on the task dataset. Here, the weights are transferred from the AlexNet, which were trained on a very big image dataset consisting of 1.2 million labelled images with 1000 categories (ImageNet [52]). A badger training dataset is only applied in the fine-tuning process. Figure 6.2 illustrates the architecture of Badger-CNN-2. The Badger-CNN-2 keeps all the weights except for the last three layers from the AlexNet. The output layer is self-defined, and weights are fine-tuned based on a badger dataset. The developed Badger-CNN-2 framework has five convolutional layers, three Max-pooling layers and

two fully connected layers. Algorithm 9 in the Appendices section gives the detailed parameters for the Badger-CNN-2.

6.3 Detection Algorithm in Videos

The trained CNNs can directly be applied to videos as videos are sequential image frames. Most of the time, cameras only capture the background because animal activities rarely happen in front of a camera being installed in the wild. Therefore, detecting images of interest in the first place and then classifying them would be beneficial. For example, an alarm system could be built for badger monitoring. In the system, the user can get immediate feedback when there is a badger moving in front of a camera. In addition, this could also save storage space by only saving frames of interest. For this purpose, this section develops a detection algorithm combined with the aforementioned recognition framework for video recognition and live streaming data recognition. In order to speed up the detection process, all images are converted to grey scale. If an image is detected as an interest frame, the colour framework is used for recognition.

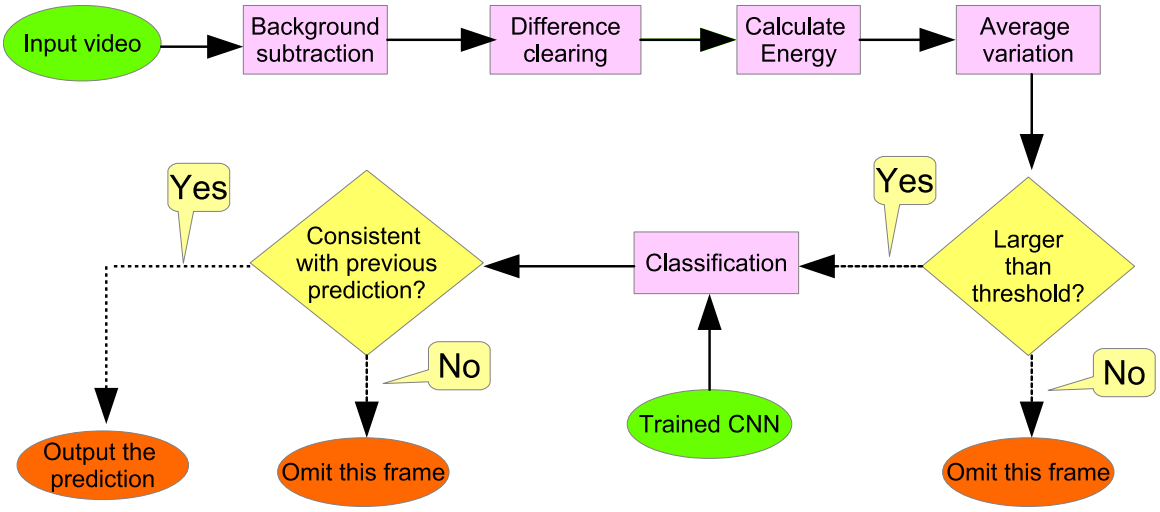


Figure 6.3: The diagram of applying the trained CNNs to videos.

Images of interest should contain objects of interest (animals). An object of interest, such as a badger or a fox, has motions and these motions result in pixel

value variations in adjacent frames. Intuitively, frame difference among adjacent frames could be calculated. Here, instead of directly applying frame difference, a dynamic background ($\mathcal{B}$) is involved with the following updating process:

$$\mathcal{B}^t(i,j) = (1-\alpha)\mathcal{I}^t(i,j) + \alpha\mathcal{B}^{t-1}(i,j), \tag{6.1}$$

where (i,j) represents the vertical and horizontal pixel location index, and $\mathcal{I}^t(i,j)$ represents the current pixel value at the location index (i,j). The initial $\mathcal{B}$ is set as the first input frame and it dynamically updates. Hence, the difference between the current frame and the background is given:

$$E^t(i,j) = |\mathcal{I}^t(i,j) - \mathcal{B}^t(i,j)|, \tag{6.2}$$

when $|\cdot|$ calculates the absolute value.

Frame difference does not necessarily indicate that there is an animal. The difference has many causes, for instance, grass motion caused by wind or noise caused by camera sensors. Here, the following assumptions are made in order to decrease the false positive detections: 1) if an animal moves, the frame difference should be considerably large; 2) the movement of the animal is the main cause of the pixel changes and the camera is not occluded by the animal's body.

In order to remove tiny variations, a dynamic thresholding process is applied based on the maximum value among all $E^t(i,j)$:

$$E^t(i,j) = \begin{cases} 0 & if \quad E^t(i,j) < \beta \cdot max(\mathbf{E}^t), \\ E^t(i,j) & Otherwise, \end{cases} \tag{6.3}$$

where $max(\mathbf{E}^t)$ is the maximum value in $\mathbf{E}^t$. This process is followed by a median filter aimed at removing noise. As animal movements usually happen in a small area, if a big area is moving, this means the camera is either moving or it has been blocked

by the animal's body. Hence, a frame is omitted when its **E** has either too little or too many non-zero values. Here, an animal size is restricted between 200 pixels and half of the total number of pixels in an image. Frames that have large pixel variations are removed in order to decrease the false positive detections caused by other factors such as the camera movement, extremely windy weather and suddenly changing the scene.

For the considered frames, an energy term E^t_{total} is accumulated by summing all the non-zero values in $\mathbf{E}^t$:

$$E^t_{total} = \sum_{i,j} E^t(i,j). \tag{6.4}$$

The average variation of each pixel is given by:

$$E^t_{avg} = E^t_{total}/n^t, \tag{6.5}$$

where n is the number of non-zero valued pixels in **E**. For frames that have animal motions, the image should have big total energy E^t_{total}. In addition, the pixel variations caused by animals should be larger than other factors, which results in the variations caused by animals contributing the main portion of the total energy. Hence, its E^t_{avg} should be large. Hence, by comparing the E^t_{avg} with a threshold, the frame t would further be sent to the classification stage if its value was beyond the threshold.

If an animal is detected, the classification result should be consistent in a short period (for example, 0.1 second). Therefore, a confirmation process as shown in Figure 6.3 is applied in order to further decrease false alarms. When the prediction agrees with the previous prediction, the classification result will be confirmed as the output.

6.4 Performance Evaluation on the Badger Dataset

6.4.1 Dataset Generation

Table 6.1: Categories and number of images in the badger dataset.

Categories	Total images	Training images	Testing images
Badger	1556	1089	467
Bird	1528	1070	458
Cat	1083	758	325
Fox	2693	1885	808
Rat	570	399	171
Rabbit	938	657	281
	8368	5858	2510

The images were captured in the UK and all images were unlabelled. In order to make a new dataset, all images were manually checked and assigned to their corresponding classes. The created dataset contains six categories: badger, bird, cat, fox, rat and rabbit. Seventy percent of the images are randomly chosen as the training images and the rest are assigned for the testing. Detailed information of the created dataset is given in Table 6.1. Figure 6.4 shows some random images in the testing dataset. From the first row to the last row are badger, bird, cat, fox, rat and rabbit, respectively.

In this section, two different scenarios are considered. The first scenario considers the binary classification problem, which distinguishes an image either belonging to the badger or the non-badger category. The second scenario considers the multinomial classification problem, which further distinguishes what animal is in a non-badger image, e.g. a fox or a rabbit.

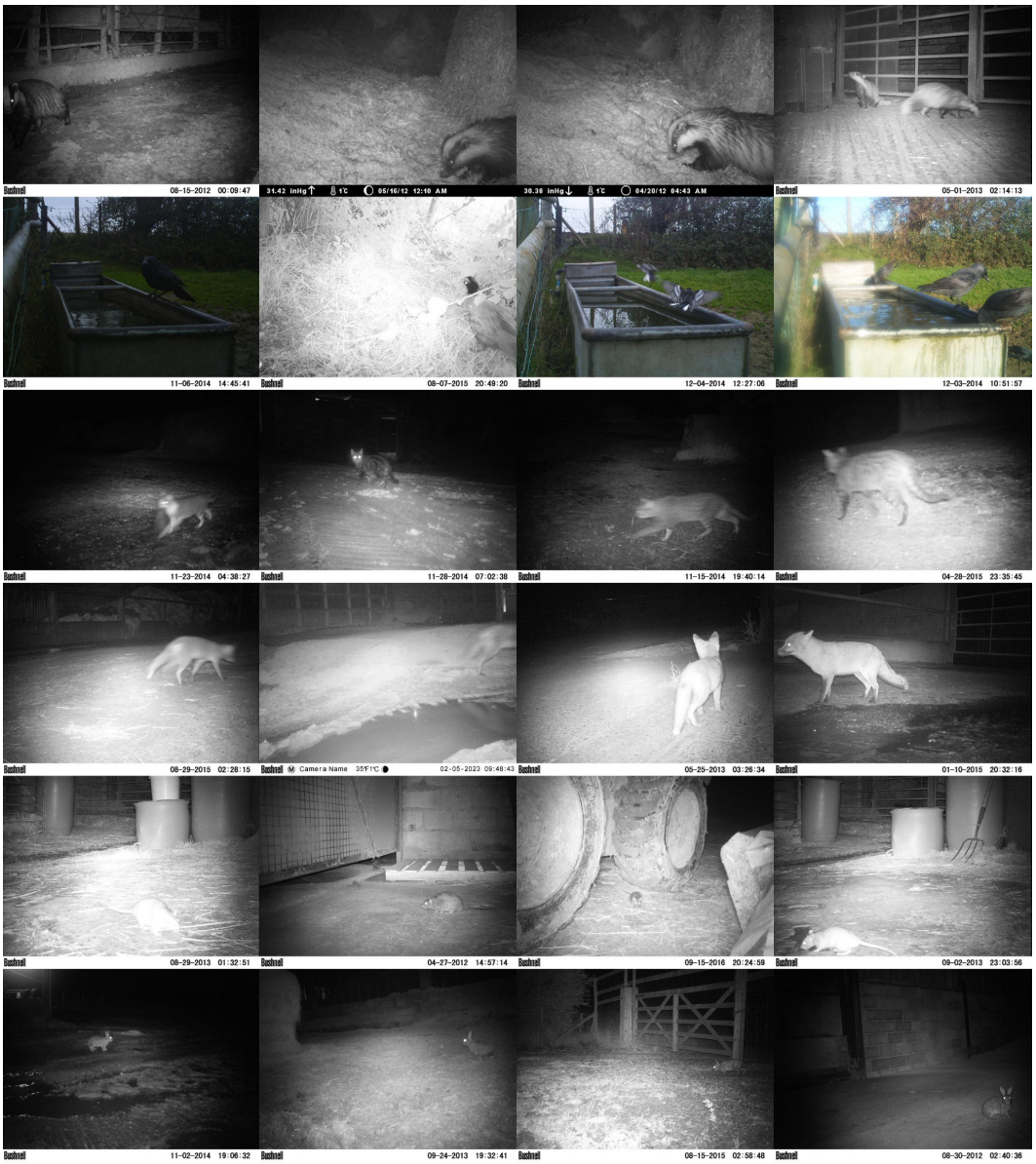


Figure 6.4: Some random images of the testing dataset.

6.4.2 Badger and Non-Badger Classification

In the badger dataset, images are grouped into only two categories: the badger group and the non-badger group. Details are given in Table 6.2. In this subsection, both frameworks are evaluated based on the binary classification dataset.

Table 6.2: Badgers and non-badgers in badger dataset.

Categories	Total images	Training images	Testing images
Badger	1556	1089	467
Non-Badger	6812	4769	2043

Performance of Badger-CNN-1

Table 6.3 gives the performance of the Badger-CNN-1 framework. The True Positive (TP) refers to the number of the badger testing images that are correctly classified to the badger category. The False Positive (FP) refers to the number of the non-badger testing images that are wrongly classified to the badger category. False Negative (FN) represents the number of the badger testing images that are wrongly classified to the non-badger category, and True Negative (TN) gives the number of the non-badger testing images that are correctly classified to their corresponding category.

Table 6.3: The performance of the Badger-CNN-1.

	Badger(test)	Non-Badger(test)	Accuracy	F1 score
Predicted as badger	384 (TP)	28 (FP)	95.58%	0.87
Predicted as non-badger	83 (FN)	2015 (TN)		

Here two evaluation methods, accuracy and F1 score, are applied to evaluate the binary classification result. Accuracy represents the ratio between correctly classified testing images and the entire testing dataset. F1 score is the harmonic average of the precision ($\frac{TP}{TP+FP}$) and recall ($\frac{TP}{TP+FN}$). Its value is between 0 and 1, the higher the better.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}. \tag{6.6}$$

$$F1\ score = \frac{2TP}{2TP + FP + FN} \tag{6.7}$$

Table 6.3 indicates that the false negative rate ($\frac{FN}{FN+TP} = 17.77\%$) is much higher

than the false positive rate ($\frac{FP}{FP+TN}$=1.37%). This is due to there being unbalanced data in each category, which results in a testing image having a higher probability to be classified to the majority group in the training dataset. In order to decrease this effect, a re-sampling process is applied to the minority group. In the training dataset, images under the badger folder are re-sampled four more times in order to provide an equivalent number of images in both categories. This re-sampling process drops the false negative rate from 17.77% to 10.71% and improves the F1 score from 0.87 to 0.89 as shown in Table 6.4.

Table 6.4: The performance of the Badger-CNN-1 with a re-sampling process.

	Badger (test)	Non-Badger (test)	Accuracy	F1 score
Predicted as badger	416 (TP)	53 (FP)	95.86%	0.89
Predicted as non-badger	51 (FN)	1990 (TN)		

Performance of Badger-CNN-2

As the imbalanced training dataset causes biased results as shown in the previous subsection, here considers both the training dataset with a re-sampling process and without.

Table 6.5: The performance of the Badger-CNN-2.

	Without re-sampling		With re-sampling	
	Badger (test)	Non-Badger (test)	Badger (test)	Non-Badger (test)
Predicted as badger	429 (TP)	22 (FP)	442 (TP)	24 (FP)
Predicted as non-badger	38 (FN)	2021 (TN)	25 (FN)	2019 (TN)
	Accuracy and F1 score			
	Accuracy	F1 Score	Accuracy	F1 Score
	97.61%	0.93	**98.05%**	**0.95**

As shown in Table 6.5, the performance of Badger-CNN-2 performs better than the Badger-CNN-1. The highest accuracy of 98.05% is achieved by Badger-CNN-2. Compared with Badger-CNN-1, Badger-CNN-2 takes the advantage of weights that have learned from millions of images from other image sources.

6.4.3 Multinomial Classification Based on the Badger Dataset

The previous subsection evaluates the frameworks with the purpose of distinguishing badger and non-badger images. However, it would also be helpful to give more specific answers when the testing images belong to the non-badger category, e.g, a badger or a fox. In this subsection, the binary classification problem is extended to the multinomial-classification problem. The performances of both Badger-CNN-1 and Badger-CNN-2 are evaluated for the multinomial classification task.

Performance of the Badger-CNN-1

The accuracies of six categories can be illustrated in a Confusion Matrix. In the multinomial classification case, the F1 score is not valid. Accuracy and mean accuracy are instead applied to evaluate the performance. Mean accuracy is obtained by averaging all the accuracy from individual classes. Table 6.6 and Table 6.7 show the results of the multinomial classification without and with a re-sampling process being applied, respectively. Both accuracies are lower than the binary classification. The re-sampling process slightly improves both accuracies and the mean accuracy in the Badger-CNN-1 framework. This can be explained by the imbalanced training data causing bias. The re-sampling process improves the accuracy of categories that have less training dataset, such as cat, rat and rabbit. The general accuracies are above 83% for multinomial classification.

Table 6.6: The performance of the Badger-CNN-1 without a re-sampling process.

Testing data / Predictions	Badger	Bird	Cat	Fox	Rat	Rabbit	Accuracy	Mean Accuracy
Badger	395	2	6	29	9	6	83.07%	79.98%
Bird	1	441	3	7	2	4		
Cat	4	3	207	34	10	6		
Fox	54	11	90	704	24	46		
Rat	7	0	5	3	122	3		
Rabbit	6	1	14	31	4	214		
Individual Accuracy	84.58%	96.29%	63.69%	87.13%	71.35%	76.87%		

Table 6.7: The performance of the Badger-CNN-1 with a re-sampling process.

Testing data / Predictions	Badger	Bird	Cat	Fox	Rat	Rabbit	Accuracy	Mean Accuracy
Badger	402	2	2	34	8	5	83.51%	82.71%
Bird	0	438	4	12	0	3		
Cat	4	3	235	41	8	4		
Fox	11	0	9	652	12	29		
Rat	11	0	9	7	132	3		
Rabbit	5	5	14	62	11	237		
Individual Accuracy	86.08%	95.63%	72.31%	80.69%	77.19%	84.34%		

Performance of Badger-CNN-2

Table 6.8: Result of the Badger-CNN-2 without a re-sampling process

Testing data / Predictions	Badger	Bird	Cat	Fox	Rat	Rabbit	Accuracy	Mean Accuracy
Badger	431	1	6	10	3	11	**90.32%**	**87.57%**
Bird	2	447	3	5	3	5		
Cat	3	0	251	8	6	5		
Fox	16	5	47	763	10	15		
Rat	5	2	3	6	133	3		
Rabbit	10	3	15	16	16	242		
Individual Accuracy	92.29%	97.60%	77.23%	94.43%	77.78%	86.12%		

Table 6.8 and Table 6.9 show the results of the Badger-CNN-2 without and with a re-sampling process being applied, respectively. Even though there are six categories in the dataset, Badger-CNN-2 could achieve an accuracy of 90.32%. Table 6.8 indicates that the accuracies are much higher and more balanced than results achieved by Badger-CNN-1. The lowest accuracy (77.23%) is in the cat category; this can be explained by their similar appearances, especially when looking from the back. Comparing Table 6.8 with Table 6.9, the re-sampling process does not have any improvement on Badger-CNN-2. This is due to the well trained weights from AlexNet being less likely to have over-fitting problems. Over-fitting means the model perfectly performs on the training dataset only, while giving bad performance on the testing dataset.

Table 6.9: The performance of the Badger-CNN-2 with a re-sampling process.

Testing data / Predictions	Badger	Bird	Cat	Fox	Rat	Rabbit	Accuracy	Mean Accuracy
Badger	434	3	6	24	2	11	86.85%	87.04%
Bird	2	439	1	2	3	4		
Cat	13	2	281	90	7	7		
Fox	7	1	26	644	7	8		
Rat	6	5	2	13	137	6		
Rabbit	5	8	9	35	15	245		
Individual Accuracy	92.93%	95.85%	86.46%	79.70%	80.12%	87.19%		

6.4.4 Detection and Classification to Videos

This section gives the result of applying the trained Badger-CNN-2 on videos. Figure 6.5 (a) shows an input frame. Figure 6.5 (b) shows the average variation of the activated pixels in the current frame. A threshold is set to 20. A frame whose average variation above 20 is sent for classification. Figure 6.5 (c) illustrates the cleared version of the activated pixels, with the optical flow indicating the motion estimations. Figure 6.5 (d) shows the classification result of the interest frame. In videos, the recognition results for adjacent frames shall be agreed; this process would further decrease the mis-recognition probability.

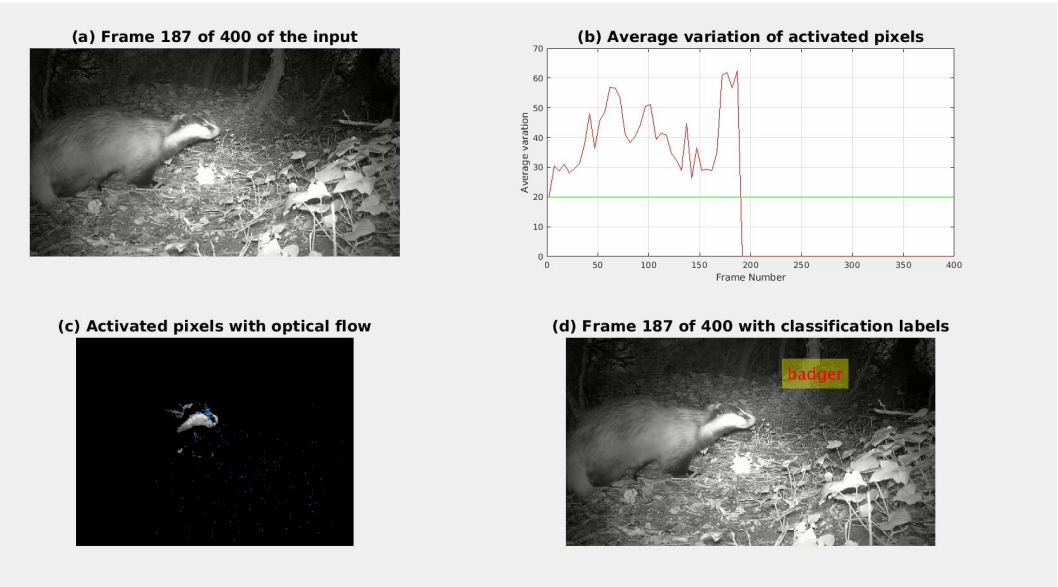


Figure 6.5: Illustration of a detected frame activated by a badger.

6.5 Summary

Monitoring badger activities is important for understanding the transmission of bTB, which is the most pressing animal health problem in the UK as described by government. This work, for the first time, develops an automatic recognition framework capable of identifying and isolating badger activity in still images and video footages. This work has shown that machine learning algorithms can be beneficial to wildlife monitoring applications. Instead of involving human labour for video inspection, a machine learning based framework could detect an animal from a video sequence and identify the species of animal. This would be helpful for automatically monitoring wildlife such as badgers.

In this work, a new image dataset has been created that contains 8368 images belonging to six categories: badger, bird, cat, fox, rat and rabbit. For each category, 70% of images are randomly chosen for the training purpose. The remaining 30% of images are used in order to test the accuracies of the trained models. Two CNN frameworks are developed for automatically recognising animal images. Both frameworks give good performances on the testing dataset. For badger and non-badger recognition, they achieve accuracies of 95.86% and 98.05%, respectively. For the multinomial classification, they achieve accuracies of 83.07% and 90.32%, respectively. The proposed recognition frameworks give options according to different equipment that may be available to organisations. The more accurate framework requires more computational memories. In addition, a detection algorithm has been developed for videos. Therefore, activated frames can be detected and further be classified using the trained recognition models.

Chapter 7

Conclusions and Future Works

7.1 Conclusions

This thesis aims to develop machine learning methods for autonomous object recognition and restoration in images. The main development includes: development of an efficient online recognition framework valid for both small datasets and big datasets; development of a non-parametric classifier based on Bayesian compressive sensing; development of automatic learned joint networks for image recognition and restoration. The proposed methods are mainly evaluated on the application of vehicle logo recognition, an important research area in intelligent transportation systems. However, the developed methods can also be extended to other application domains.

The developed online recognition framework providing solutions for both small datasets and big datasets has been presented in Chapter 3. When the dataset is small, the HOG feature method and the proposed Cauchy prior LR provides a quick and accurate solution. When the dataset size is increasing, the weights in the Cauchy prior LR can be updated based on the previously trained models. The Cauchy prior assumes the useful weights are sparse, this results in a quick convergence speed for the weight updating process. When the dataset becomes big, a developed CNN framework is involved in order to further increase the accuracy and robustness.

This thesis also developed a framework based on spatial SIFT features, in Chap-

ter 4. As the feature representation model only counts the occurring frequency of detected local features, the spatial SIFT features incorporate the spatial information of SIFT features and achieve good results. A non-parametric classifier, SBCS, is also proposed in Chapter 4. The proposed SBCS relies on the construction of the testing image using partial information from the weights estimated by BCS. Note that for each class there is a corresponding reconstructed image. By comparing the reconstructed images with the testing image, the testing image can be classified to the class that has the least Euclidean distance. In addition, a column based subspace sampling process is combined with SBCS which gives good results while only using a small fraction of the training data.

Image restoration and recognition are often addressed in a pipeline manner, where the recognition process is applied after the image restoration. Chapter 5 proposed deep neural network based joint frameworks for simultaneously classifying and recovering images. The joint frameworks consist of common layers in which weights are shared for both tasks, classification layers which are trained for recognition, and restoration layers that aim to recover a corrupted input image to a clear version. For the restoration, the image rotation and occlusion are considered and the joint framework based on learning capsules gives a good performance.

Chapter 6 developed a recognition framework for badger detection and badger recognition. In order to build a robust badger recognition framework, a dataset is created. Two CNN frameworks have been developed. A detection process is proposed in order to detect frames of interest, which will be further classified using the trained recognition models. The proposed frameworks give effective and accurate badger recognition solutions.

7.2 Direction for Future Works

Image recognition and restoration remain challenging tasks with a large scope for future researches. Based on the findings in this thesis, there are several extensions,

potential new frontiers and applications to be explored in future:

- Chapter 3 developed a Cauchy prior logistic regression classifier with the conjugate gradient descent. The Cauchy prior assumes the weights are sparse, which is similar to dropout [49] and Adam optimiser [136] in CNNs. Since the softmax layer in CNNs is based on LR, the proposed Cauchy prior LR has the potential to be extended in a deep learning framework.

- Object detection and segmentation methods could be developed in future. The state-of-the-art methods are based on R-CNN [137], which is a regional based CNN. In R-CNN, small regions are extracted in order to compute the image features. The feature extraction mechanism is the same as in CNNs. However, this thesis shows that the learning capsules are more robust to image degradation such as noise, occlusion and rotation. Therefore, the learning capsules and the joint capsule framework could potentially be extended to R-CNN. In addition, the recognition and restoration layers are separate, future work could be considered to link both layers similarly as in Generative Adversarial Networks (GANs) [138].

- For the wild monitoring, algorithms can be further developed to a customised camera system which will give real-time alerts and will only record images of interest, without requiring huge amounts of storage capacity. In the current stage, the work in Chapter 6 considers the situation of an image containing an animal; it shall also work for animals in the same category. However, this framework is not designed for multi-label classification, e.g. it will only give one answer if an image contains two categories. In the future, this problem can also be investigated. In addition, the work in Chapter 6 only recognises which animal appears in an image, not how many animals. However, an automatic animal population estimation algorithm may be helpful and could be developed in future.

Appendix A

SBCS Extended and the Deep Learning Architecture in CIFAR-10 Dataset

A.1 Marginal Likelihood Maximisation

The following gives a detailed derivation for the marginal likelihood in equation (4.14). By combining equations (4.3) and (4.4), the marginal likelihood can be expanded to:

$$\begin{aligned}
&p(\mathbf{x}^*|\boldsymbol{\alpha}, \sigma^2) \\
&= \int p(\mathbf{x}^*|\mathbf{w}, \sigma^2)\ p(\mathbf{w}|\boldsymbol{\alpha}) d\mathbf{w} \\
&= \int (2\pi\sigma^2)^{-\frac{M}{2}} exp\left\{-\frac{1}{2\sigma^2}||\mathbf{x}^* - \mathbf{X}\mathbf{w}||_2^2\right\} (2\pi)^{-\frac{N}{2}} |\mathbf{A}|^{\frac{1}{2}} exp\left\{-\frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{A}\mathbf{w}\right\} d\mathbf{w} \\
&= (2\pi\sigma^2)^{-\frac{M}{2}} (2\pi)^{-\frac{N}{2}} |\mathbf{A}|^{\frac{1}{2}} \int exp\left\{-\frac{1}{2\sigma^2}||\mathbf{x}^* - \mathbf{X}\mathbf{w}||_2^2 + \mathbf{w}^\mathsf{T}\mathbf{A}\mathbf{w}\right\} d\mathbf{w}. \qquad \text{(A.1)}
\end{aligned}$$

In order to simply equation (A.1), define:

$$\mathbb{Q} = \frac{1}{2}\left\{\frac{1}{\sigma^2}||\mathbf{x}^* - \mathbf{X}\mathbf{w}||_2^2 + \mathbf{w}^\mathsf{T}\mathbf{A}\mathbf{w}\right\}. \qquad \text{(A.2)}$$

Combining with equations (4.10) and (4.11), (A.2) can be given as:

$$\begin{aligned}
\mathbb{Q} &= \frac{1}{2}\left(\frac{\mathbf{x}^{*\mathsf{T}}\mathbf{x}^*}{\sigma^2} - \frac{2\mathbf{x}^{*\mathsf{T}}\mathbf{X}\mathbf{w}}{\sigma^2} + \frac{\mathbf{w}^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}\mathbf{X}\mathbf{w}}{\sigma^2} + \mathbf{w}^{\mathsf{T}}\mathbf{A}\mathbf{w}\right) \\
&= \frac{1}{2}\left(\frac{\mathbf{x}^{*\mathsf{T}}\mathbf{x}^*}{\sigma^2} - \frac{2\mathbf{x}^{*\mathsf{T}}\mathbf{X}\mathbf{w}}{\sigma^2} + \mathbf{w}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\mathbf{w}\right) \\
&= \frac{1}{2}\left(\frac{\mathbf{x}^{*\mathsf{T}}\mathbf{x}^*}{\sigma^2} - \frac{2(\boldsymbol{\Sigma}\mathbf{X}^{\mathsf{T}}\mathbf{x}^*)^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\mathbf{w}}{\sigma^2} + \mathbf{w}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\mathbf{w}\right) \\
&= \frac{1}{2}\left(\frac{\mathbf{x}^{*\mathsf{T}}\mathbf{x}^*}{\sigma^2} - 2\boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\mathbf{w} + \mathbf{w}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\mathbf{w}\right) \\
&= \frac{1}{2}\left(\frac{\mathbf{x}^{*\mathsf{T}}\mathbf{x}^*}{\sigma^2} - \boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right) + \frac{1}{2}(\mathbf{w}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{w}-\boldsymbol{\mu}).
\end{aligned} \tag{A.3}$$

In order to simply equation (A.3), set

$$\mathbb{T} = \frac{1}{2}\left(\frac{\mathbf{x}^{*\mathsf{T}}\mathbf{x}^*}{\sigma^2} - \boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right), \tag{A.4}$$

Therefore the integral part in the right hand side of equation (A.1) is given by:

$$\begin{aligned}
&\int exp\{-\mathbb{Q}\}d\mathbf{w} \\
&= \int exp\left\{-\mathbb{T} - \frac{1}{2}(\mathbf{w}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{w}-\boldsymbol{\mu})\right\}d\mathbf{w} \\
&= exp\left\{-\mathbb{T}\right\}\int exp\left\{-\frac{1}{2}(\mathbf{w}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{w}-\boldsymbol{\mu})\right\}d\mathbf{w} \\
&= exp\left\{-\mathbb{T}\right\}(2\pi)^{\frac{N}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}\int \frac{1}{(2\pi)^{\frac{N}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}} exp\left\{-\frac{1}{2}(\mathbf{w}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{w}-\boldsymbol{\mu})\right\}d\mathbf{w} \\
&= (2\pi)^{\frac{N}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}} exp\left\{-\mathbb{T}\right\}.
\end{aligned} \tag{A.5}$$

Substituting this back in equation (A.1) have:

$$\begin{aligned}
&p(\mathbf{x}^*|\boldsymbol{\alpha},\sigma^2) \\
&= (2\pi\sigma^2)^{-\frac{M}{2}}(2\pi)^{-\frac{N}{2}}|\mathbf{A}|^{\frac{1}{2}}(2\pi)^{\frac{N}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}} exp\{-\mathbb{T}\} \\
&= (2\pi\sigma^2)^{-\frac{M}{2}}|\mathbf{A}|^{\frac{1}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}} exp\{-\mathbb{T}\}.
\end{aligned} \tag{A.6}$$

This can be further simplified by:

$$\begin{aligned} p(\mathbf{x}^*|\boldsymbol{\alpha},\sigma^2) &= (2\pi\sigma^2)^{-\frac{M}{2}}|\mathbf{A}|^{\frac{1}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}exp\{-\mathbb{T}\} \\ &= (2\pi\sigma^2)^{-\frac{M}{2}}\frac{|\boldsymbol{\Sigma}|^{\frac{1}{2}}}{|\mathbf{A}^{-1}|^{\frac{1}{2}}}exp\{-\mathbb{T}\} \\ &= (2\pi)^{-\frac{M}{2}}\frac{1}{\sigma^M}\frac{1}{|\mathbf{A}^{-1}\boldsymbol{\Sigma}^{-1}|^{\frac{1}{2}}}exp\{-\mathbb{T}\} \\ &= (2\pi)^{-\frac{M}{2}}\frac{1}{\sigma^M}\frac{1}{|\mathbf{I}_N+\sigma^{-2}\mathbf{A}^{-1}\mathbf{X}^\mathsf{T}\mathbf{X}|^{\frac{1}{2}}}exp\{-\mathbb{T}\}, \end{aligned} \tag{A.7}$$

where $\mathbf{I}_N = \mathbf{A}^{-1}\mathbf{A}$. Using the matrix determinants properties [139] that $|\mathbf{I}_N + \mathbf{X}^\mathsf{T}\mathbf{Y}| = |\mathbf{I}_M + \mathbf{X}\mathbf{Y}^\mathsf{T}|$ with $\mathbf{X} \in \mathbb{R}^{M\times N}$ and $\mathbf{Y} \in \mathbb{R}^{M\times N}$, the above equation can be updated to:

$$\begin{aligned} p(\mathbf{x}^*|\boldsymbol{\alpha},\sigma^2) &= (2\pi)^{-\frac{M}{2}}\frac{1}{\sigma^M}\frac{1}{|\mathbf{I}_M+\sigma^{-2}\mathbf{A}^{-1}\mathbf{X}\mathbf{X}^\mathsf{T}|^{\frac{1}{2}}}exp\{-\mathbb{T}\} \\ &= (2\pi)^{-\frac{M}{2}}\frac{1}{|\sigma^2\mathbf{I}_M|^{\frac{1}{2}}}\frac{1}{|\mathbf{I}_M+\sigma^{-2}\mathbf{A}^{-1}\mathbf{X}\mathbf{X}^\mathsf{T}|^{\frac{1}{2}}}exp\{-\mathbb{T}\} \\ &= (2\pi)^{-\frac{M}{2}}\frac{1}{|\sigma^2\mathbf{I}_M+\mathbf{X}\mathbf{A}^{-1}\mathbf{X}^\mathsf{T}|^{\frac{1}{2}}}exp\{-\mathbb{T}\}. \end{aligned} \tag{A.8}$$

Recall the $\mathbb{T}$ is given in equation (A.4) and it can be expressed as follows:

$$\begin{aligned} \mathbb{T} &= \frac{1}{2}\left(\frac{\mathbf{x}^{*\mathsf{T}}\mathbf{x}^*}{\sigma^2} - (\sigma^{-2}\boldsymbol{\Sigma}\mathbf{X}^\mathsf{T}\mathbf{x}^*)^\mathsf{T}\boldsymbol{\Sigma}^{-1}\sigma^{-2}\boldsymbol{\Sigma}\mathbf{X}^\mathsf{T}\mathbf{x}^*\right) \\ &= \frac{1}{2}\left(\frac{\mathbf{x}^{*\mathsf{T}}\mathbf{x}^*}{\sigma^2} - \sigma^{-2}\mathbf{x}^{*\mathsf{T}}\mathbf{X}\boldsymbol{\Sigma}\mathbf{X}^\mathsf{T}\mathbf{x}^*\sigma^{-2}\right) \\ &= \frac{1}{2}\left(\mathbf{x}^{*\mathsf{T}}\left[\sigma^{-2}\mathbf{I}_M - \sigma^{-2}\mathbf{X}(\mathbf{A}+\sigma^{-2}\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\sigma^{-2}\right]\mathbf{x}^*\right). \end{aligned} \tag{A.9}$$

According to the Woodbury inversion identity [111]:

$$[\sigma^{-2}\mathbf{I}_M - \sigma^{-2}\mathbf{X}(\mathbf{A}+\sigma^{-2}\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\sigma^{-2}] = (\sigma^2\mathbf{I}_M + \mathbf{X}\mathbf{A}^{-1}\mathbf{X}^\mathsf{T})^{-1}, \tag{A.10}$$

equation (A.9) can be expressed as:

$$\mathbb{T} = \frac{1}{2}\left(\mathbf{x}^{*\mathrm{T}}(\sigma^2\mathbf{I}_M + \mathbf{X}\mathbf{A}^{-1}\mathbf{X}^{\mathrm{T}})^{-1}\mathbf{x}^*\right). \tag{A.11}$$

Therefore, equation (A.8) can be given as:

$$\begin{aligned} &p(\mathbf{x}^*|\boldsymbol{\alpha}, \sigma^2) \\ &= (2\pi)^{-\frac{M}{2}}|\sigma^2\mathbf{I}_M + \mathbf{X}\mathbf{A}^{-1}\mathbf{X}^{\mathrm{T}}|^{-\frac{1}{2}} exp\left\{-\frac{1}{2}\left(\mathbf{x}^{*\mathrm{T}}(\sigma^2\mathbf{I}_M + \mathbf{X}\mathbf{A}^{-1}\mathbf{X}^{\mathrm{T}})^{-1}\mathbf{x}^*\right)\right\} \\ &= \frac{1}{\sqrt{(2\pi)^M|\mathbf{C}|}} exp\left\{-\frac{1}{2}\mathbf{x}^{*T}\mathbf{C}^{-1}\mathbf{x}^*\right\}, \end{aligned} \tag{A.12}$$

which links back to equation (4.15), with the $M \times M$ matrix $\mathbf{C}$ is given by:

$$\mathbf{C} = \sigma^2\mathbf{I}_M + \mathbf{X}\mathbf{A}^{-1}\mathbf{X}^{\mathrm{T}}. \tag{A.13}$$

A.2 Evidence Approximation

This subsection presents the derivation of the marginal log-likelihood function and its maximisation with respect to α_i and σ^2. Notice that $\mathbb{T}$ in equation (A.4) can be expressed as follows:

$$\begin{aligned} \mathbb{T} &= \frac{1}{2}\left(\frac{1}{\sigma^2}\mathbf{x}^{*\mathrm{T}}\mathbf{x}^* - \boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right) \\ &= \frac{1}{2\sigma^2}\left(\mathbf{x}^{*\mathrm{T}}\mathbf{x}^* - \sigma^{-2}\mathbf{x}^{*\mathrm{T}}\mathbf{X}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right) \\ &= \frac{1}{2\sigma^2}\mathbf{x}^{*\mathrm{T}}(\mathbf{x}^* - \mathbf{X}\boldsymbol{\mu}) \\ &= \frac{1}{2\sigma^2}\left(||\mathbf{x}^* - \mathbf{X}\boldsymbol{\mu})||_2^2 + \mathbf{x}^{*\mathrm{T}}\mathbf{X}\boldsymbol{\mu} - \boldsymbol{\mu}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\mu}\right) \\ &= \frac{1}{2\sigma^2}\left(||\mathbf{x}^* - \mathbf{X}\boldsymbol{\mu})||_2^2 + \mathbf{x}^{*\mathrm{T}}\mathbf{X}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - \boldsymbol{\mu}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\mu}\right) \\ &= \frac{1}{2\sigma^2}\left(||\mathbf{x}^* - \mathbf{X}\boldsymbol{\mu})||_2^2 + \sigma^2\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - \boldsymbol{\mu}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\mu}\right) \\ &= \frac{1}{2\sigma^2}||\mathbf{x}^* - \mathbf{X}\boldsymbol{\mu})||_2^2 + \frac{1}{2}\boldsymbol{\mu}^{\mathrm{T}}\mathbf{A}\boldsymbol{\mu}. \end{aligned} \tag{A.14}$$

Hence, taking the logarithm of the marginal likelihood given in equation (A.6), the logarithm of the marginal likelihood can be obtained in the following form:

$$\mathcal{L}(\boldsymbol{\alpha}, \sigma^2) = -\frac{M}{2} ln(\sigma^2) - \frac{M}{2} ln(2\pi) + \frac{1}{2}\sum_{i=1}^{N} ln(\alpha_i) + \frac{1}{2} ln|\boldsymbol{\Sigma}| - \frac{1}{2\sigma^2}||\mathbf{x}^* - \mathbf{X}\boldsymbol{\mu})||_2^2 - \frac{1}{2}\boldsymbol{\mu}^{\mathrm{T}}\mathbf{A}\boldsymbol{\mu}. \tag{A.15}$$

The procedure of maximising equation (A.15) with respect to α_i and σ^2 is known as the evidence approximation procedure.

Following the approach in [140], the derivative of $ln|\boldsymbol{\Sigma}|$ with respect to α_i is:

$$\frac{\partial}{\partial \alpha_i} ln|\boldsymbol{\Sigma}| = \frac{\partial}{\partial \alpha_i}(-ln|\boldsymbol{\Sigma}|^{-1}) = -Trace\boldsymbol{\Sigma} = -\Sigma_{ii}, \tag{A.16}$$

where Σ_{ii} is the i^{th} diagonal component of the posterior covariance matrix $\boldsymbol{\Sigma}$ and $Trace$ is the trace of a matrix. Therefore, the derivative of equation (A.15) with respect to α_i gives:

$$\frac{\partial \mathcal{L}(\boldsymbol{\alpha}, \sigma^2)}{\partial \alpha_i} = \frac{1}{2\alpha_i} - \frac{1}{2}\Sigma_{ii} - \frac{1}{2}\mu_i^2. \tag{A.17}$$

Setting the derivative to zero gives equation (4.17).

In order to simplify the $\frac{\partial \mathcal{L}(\boldsymbol{\alpha}, \sigma^2)}{\partial \sigma^2}$, set $\beta = 1/\sigma^2$. Following the approach in [141], the derivative of $ln|\boldsymbol{\Sigma}|$ with respect to β is:

$$\begin{aligned}
\frac{\partial}{\partial \beta} ln|\boldsymbol{\Sigma}| &= \frac{\partial}{\partial \beta}(-ln|\boldsymbol{\Sigma}|^{-1}) \\
&= -Trace(\boldsymbol{\Sigma}\mathbf{X}^{\mathbf{T}}\mathbf{X}) \\
&= -Trace(\boldsymbol{\Sigma}\mathbf{X}^T\mathbf{X} + \beta^{-1}\boldsymbol{\Sigma}\mathbf{A} - \beta^{-1}\boldsymbol{\Sigma}\mathbf{A}) \\
&= -Trace(\boldsymbol{\Sigma}(\beta\mathbf{X}^T\mathbf{X} + \mathbf{A})\beta^{-1} - \beta^{-1}\boldsymbol{\Sigma}\mathbf{A}) \\
&= -Trace(\mathbf{I}_N - \boldsymbol{\Sigma}\mathbf{A})\beta^{-1}
\end{aligned} \tag{A.18}$$

Therefore, the derivative of equation (A.15) with respect to β is:

$$\frac{\partial \mathcal{L}(\boldsymbol{\alpha}, \sigma^2)}{\partial \beta} = \frac{M}{2\beta} - \frac{1}{2}||\mathbf{x}^* - \mathbf{X}\boldsymbol{\mu})||_2^2 - \frac{1}{2}Trace(\mathbf{I}_N - \boldsymbol{\Sigma}\mathbf{A})\beta^{-1}. \tag{A.19}$$

Setting the derivative to zero gives equation (4.18).

A.3 CIFAR-10 Feature Extraction Parameters

Algorithm 7 gives the detailed steps of image feature extraction in the CIFAR-10 dataset. The last fully connected layer is use as the feature. Hence, each image is represented by a vector of length 200.

Algorithm 7 CNN feature extraction parameters

Output:

1: Image input layer [32×32×3] (image size [32×32] and image channel [3]).
2: Convolution layer [3×3×48] (kernel size [3×3] and number of kernels [48], zero padding is applied).
3: Batch normalisation.
4: Relu non-linear function.
5: Max pooling with size [2 2] with stride 2.
6: Convolution layer [3×3×96].
7: Batch normalisation.
8: Relu non-linear function.
9: Max pooling.
10: Convolution layer [3×3×192].
11: Batch normalisation.
12: Relu non-linear function.
13: Max pooling.
14: Dropout with probability is set to 0.25.
15: Fully connected layer with 512 neurons.
16: Relu non-linear function.
17: Dropout with probability is set to 0.5.
18: Fully connected layer with 200 neurons.
19: Dropout with probability is set to 0.5.
20: Softmax and classification layer.

Appendix B

Deep Learning Framework Parameters for Badger Recognition

This section gives the detailed parameters in the developed Badger-CNN-1 and Badger-CNN-2 in Section .

Algorithm 8 Badger-CNN-1 procedures

Input:

Load images from the source, weights are randomly initialised.

Output:

1: Image input layer [480×640×3] with 'zero centre' normalisation.
2: Convolution layer 1. 50 [13×13×3] convolution kernels with stride of [4 4], no padding is applied.
3: ReLU non-linear function 1.
4: Max pooling 1, with size [3 3] with stride 2, no padding is applied.
5: Batch normalisation 1.
6: Convolution layer 2. 80 [5×5×50], with padding [1 1 1 1] applied ([top bottom left right]).
7: ReLU non-linear function 2.
8: Max pooling 2, with size [2 2] with stride [2 2], no padding is applied.
9: Batch normalisation 2.
10: Convolution layer 3. 100 [3×3×80] with stride of [1 1], with padding [1 1 1 1] is applied.
11: ReLU non-linear function 3.
12: Max pooling 3, with size [2 2] with stride [2 2], no padding is applied.
13: Batch normalisation 3.
14: Convolution layer 4. 100 [3×3×100] with stride of [1 1], with padding [1 1 1 1] is applied.
15: ReLU non-linear function 4.
16: Max pooling 4, with size [2 2] with stride [2 2], with padding [0 0 0 1] is applied.
17: Batch normalisation 4.
18: Fully connected layer 1, with 1000 neurons.
19: ReLU non-linear function.
20: Dropout layer, with probability is set to 0.5.
21: Fully connected layer 2, with 6 neurons. (If the task is only to distinguish badger and non-badger, change the number of neurons to 2.)
22: Softmax layer.
23: classification layer.

Algorithm 9 Badger-CNN-2 procedures

Input:

Resize the input images to the size of [227×227×3] (image size [227×227] and image channel [3]).

Output:

1: Image input layer [227×227×3] with 'zero centre' normalisation.
2: Convolution layer 1. 96 [11×11×3] convolution kernels with stride of [4 4], no padding is applied.
3: ReLU non-linear function 1.
4: Max pooling 1, with size [3 3] with stride 2, no padding is applied.
5: Batch normalisation 1.
6: Convolution layer 2. 256 [5×5×48] with stride of [1 1], padding size of [2 2 2 2].
7: ReLU non-linear function 2.
8: Batch normalisation 2.
9: Max pooling 2.
10: Convolution layer 3. 384 [3×3×256] with stride of [1 1], padding size of [1 1 1 1].
11: ReLU non-linear function 3.
12: Convolution layer 4. 384 [3×3×192] with stride of [1 1], padding size of [1 1 1 1].
13: ReLU non-linear function 4.
14: Convolution layer 5. 256 [3×3×192] with stride of [1 1], padding size of [1 1 1 1].
15: ReLU non-linear function 5.
16: Max pooling 5.
17: Fully connected layer 1, with 4096 neurons.
18: ReLU non-linear function.
19: Dropout layer 1, with probability is set to 0.5.
20: Fully connected layer 2, with 4096 neurons.
21: Relu non-linear function.
22: Dropout layer 2, with probability is set to 0.5.
23: Fully connected layer 3, with 6 neurons. (If the task is only to distinguish badger and non-badger, change the number of neurons to 2.)
24: Softmax layer.
25: classification layer.

Bibliography

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. of 25th International Conf. on Neural Information Processing Systems*, Lake Tahoe, Nevada, USA, Dec. 2012, pp. 1097–1105.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[3] P. Fischer, A. Dosovitskiy, and T. Brox, "Image orientation estimation with convolutional networks," in *Proc. of German Conf. on Pattern Recognition*, Aachen, Germany, Oct. 2015, pp. 368–378.

[4] G. Chen, Y. Li, and S. N. Srihari, "Joint visual denoising and classification using deep learning," in *Proc. of IEEE International Conf. on Image Processing*, Phoenix, AZ, USA, Sept. 2016, pp. 3673–3677.

[5] D. ping Tian *et al.*, "A review on image feature extraction and representation techniques," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 8, no. 4, pp. 385–396, 2013.

[6] M. Yang, K. Kpalma, and J. Ronsin, "A survey of shape feature extraction techniques," in *Pattern Recognition*. IN-TECH, 2008, pp. 43–90.

[7] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. of IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, San Diego, CA, USA, July 2005, pp. 886–893.

[8] O. L. Junior, D. Delgado, V. Gonçalves, and U. Nunes, "Trainable classifier-fusion schemes: an application to pedestrian detection," in *Proc. of 12th IEEE International Conf. on Intelligent Transportation Systems*, St.Louis, MO, USA, Oct. 2009, pp. 1–6.

[9] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 743–761, 2012.

[10] C. Thurau and V. Hlavác, "Pose primitive based human action recognition in videos or still images," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, Anchorage, AK, USA, June 2008, pp. 1–8.

[11] E. Ohn-Bar and M. M. Trivedi, "Joint angles similarities and HOG2 for action recognition," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, Portland, OR, USA, June 2013, pp. 465–470.

[12] O. Déniz, G. Bueno, J. Salido, and F. De la Torre, "Face recognition using histograms of oriented gradients," *Pattern Recognition Letters*, vol. 32, no. 12, pp. 1598–1603, 2011.

[13] A. Albiol, D. Monzo, A. Martin, J. Sastre, and A. Albiol, "Face recognition using HOG–EBGM," *Pattern Recognition Letters*, vol. 29, no. 10, pp. 1537–1543, 2008.

[14] C. Shu, X. Ding, and C. Fang, "Histogram of the oriented gradient for face recognition," *Tsinghua Science and Technology*, vol. 16, no. 2, pp. 216–224, 2011.

[15] M. Dahmane and J. Meunier, "Emotion recognition using dynamic grid-based HOG features," in *Proc. of IEEE International Conf. on Automatic Face and Gesture Recognition and Workshops*, Santa Barbara, CA, USA, Mar. 2011, pp. 884–888.

[16] R. Ebrahimzadeh and M. Jampour, "Efficient handwritten digit recognition based on histogram of oriented gradients and SVM," *International Journal of Computer Applications*, vol. 104, no. 9, pp. 10–13, 2014.

[17] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The german traffic sign recognition benchmark: a multi-class classification competition," in *Proc. of IEEE International Joint Conf. on Neural Networks*, San Jose, CA, USA, Aug. 2011, pp. 1453–1460.

[18] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.

[19] J. Li and N. M. Allinson, "Subspace learning-based dimensionality reduction in building recognition," *Neurocomputing*, vol. 73, no. 1-3, pp. 324–330, 2009.

[20] A. Vinay, B. Gagana, V. S. Shekhar, B. Anil, K. B. Murthy, and S. Natarajan, "A double filtered GIST descriptor for face recognition," *Procedia Computer Science*, vol. 79, pp. 533–542, 2016.

[21] D. A. Lisin, M. A. Mattar, M. B. Blaschko, E. G. Learned-Miller, and M. C. Benfield, "Combining local and global image features for object class recognition," in *Proc. of IEEE Computer Society Conf. on Computer Vision and Pattern Recognition workshops*, San Diego, CA, USA, Sept. 2005, pp. 47–47.

[22] C. Wallraven, B. Caputo, and A. Graf, "Recognition with local features: the kernel recipe," in *Proc. of 9th IEEE International Conf. on Computer Vision*, Nice, France, Oct. 2003, pp. 257–264.

[23] H. Bay, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," vol. 110, pp. 346–359, 2008.

[24] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. of 4th Alvey Vision Conf.*, 1988, pp. 147–151.

[25] L. Ledwich and S. Williams, "Reduced SIFT features for image retrieval and indoor localisation," in *Proc. of Australian Conf. on Robotics and Automation*, Canberra, Australia, Dec. 2004, pp. 3–11.

[26] X. Yuan, J. Yu, Z. Qin, and T. Wan, "A SIFT-LBP image retrieval model based on bag of features," in *Proc. of 15th IEEE International Conf. on Image Processing*, Okayama, Japan, June 2011, pp. 1061–1064.

[27] Y. Yang and S. Newsam, "Geographic image retrieval using local invariant features," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 51, no. 2, pp. 818–832, 2013.

[28] X. Xiong and F. De la Torre, "Supervised descent method and its applications to face alignment," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, Portland, OR, USA, June 2013, pp. 532–539.

[29] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman, "SIFT flow: Dense correspondence across different scenes," in *Proc. of European Conf. on Computer Vision*, Marseille, France, Oct. 2008, pp. 28–42.

[30] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Proc. of Workshop on Statistical Learning in Computer Vision*, Prague, Czech Republic, May 2004, pp. 1–22.

[31] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proc. of the 5th Berkeley symposium on mathematical statistics and probability*, Oakland, CA, USA, 1967, pp. 281–297.

[32] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Proc. of 9th IEEE International Conf. on Computer Vision*, Nice, France, Oct. 2003, pp. 1470–1477.

[33] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, USA: Springer, 2006.

[34] S. Tong and E. Chang, "Support vector machine active learning for image retrieval," in *Proc. of ACM International Conf. on Multimedia*, New York, NY, USA, Oct. 2001, pp. 107–118.

[35] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang, "Large-scale image classification: fast feature extraction and SVM training," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, Providence, RI, USA, June 2011, pp. 1689–1696.

[36] X. Li, L. Wang, and E. Sung, "Multilabel SVM active learning for image classification," in *Proc. of IEEE International Conf. on Image Processing*, Singapore, Singapore, Oct. 2004, pp. 2207–2210.

[37] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[38] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge, UK: Cambridge University Press, 2000.

[39] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Trans. on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[40] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Proc. of Advances in Neural Information Processing Systems*, Vancouver, BC, Canada, Dec. 2005, pp. 1473–1480.

[41] L. Ma, M. M. Crawford, and J. Tian, "Local manifold learning-based k-nearest-neighbor for hyperspectral image classification," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 48, no. 11, pp. 4099–4109, 2010.

[42] O. Boiman, E. Shechtman, and M. Irani, "In defense of nearest-neighbor based image classification," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, Anchorage, AK, USA, June 2008, pp. 1–8.

[43] A. P. Psyllos, C.-N. E. Anagnostopoulos, and E. Kayafas, "Vehicle logo recognition using a SIFT-based enhanced matching scheme," *IEEE Trans. on Intelligent Transportation Systems*, vol. 11, no. 2, pp. 322–328, 2010.

[44] H. Peng, X. Wang, H. Wang, and W. Yang, "Recognition of low-resolution logos in vehicle images based on statistical random sparse distribution," *IEEE Trans. on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 681–691, 2015.

[45] A. Hinneburg, C. C. Aggarwal, and D. A. Keim, "What is the nearest neighbor in high dimensional spaces?" in *Proc. of the International Conf. on Very Large Data Bases*, San Francisco, CA, USA, Sept. 2000, pp. 506–515.

[46] J. S. Sánchez, F. Pla, and F. J. Ferri, "On the use of neighbourhood-based non-parametric classifiers," *Pattern Recognition Letters*, vol. 18, no. 11, pp. 1179–1186, 1997.

[47] R. Chen, M. Hawes, L. Mihaylova, J. Xiao, and W. Liu, "Vehicle logo recognition by spatial-SIFT combined with logistic regression," in *Proc. IEEE International Conf. on Information Fusion*, Heidelberg, Germany, July 2016, pp. 1228–1235.

[48] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[49] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[50] Y. Huang, R. Wu, Y. Sun, W. Wang, and X. Ding, "Vehicle logo recognition system based on convolutional neural networks with a pretraining strategy," *IEEE Trans. on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1951–1960, 2015.

[51] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. European Conf. on Computer Vision*, Zurich, Switzerland, Sept. 2014, pp. 818–833.

[52] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, Miami, FL, USA, June 2009, pp. 248–255.

[53] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. of International Conf. on Learning Representations*, San Diego, CA, USA, Dec. 2015, pp. 1–14.

[54] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich *et al.*, "Going deeper with convolutions," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Boston, MA, USA, June 2015, pp. 1–9.

[55] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, June 2016, pp. 770–778.

[56] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of Physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[57] N. Kriegeskorte, "Deep neural networks: a new framework for modeling biological vision and brain information processing," *Annual Review of Vision Science*, vol. 1, pp. 417–446, 2015.

[58] O. M. Parkhi, A. Vedaldi, A. Zisserman *et al.*, "Deep face recognition," in *Proc. of British Machine Vision Conf.*, Swansea, UK, Sept. 2015, pp. 6–18.

[59] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Boston, MA, USA, June 2015, pp. 815–823.

[60] G. Hu, Y. Yang, D. Yi, J. Kittler, W. Christmas, S. Z. Li, and T. Hospedales, "When face recognition meets with deep learning: an evaluation of convolutional neural networks for face recognition," in *Proc. of the IEEE International Conf. on Computer Vision Workshops*, Santiago, Chile, Dec. 2015, pp. 142–150.

[61] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, 2013.

[62] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi, "Human action recognition using factorized spatio-temporal convolutional networks," in *Proc. of the IEEE International Conf. on Computer Vision*, Santiago, Chile, Dec. 2015, pp. 4597–4605.

[63] D. Llorca, R. Arroyo, and M. Sotelo, "Vehicle logo recognition in traffic images using HOG features and SVM," in *Proc. of 16th IEEE International Conf. on Intelligent Transportation Systems*, The Hague, Netherlands, Oct. 2013, pp. 2229–2234.

[64] L. Figueiredo, I. Jesus, J. T. Machado, J. Ferreira, and J. M. De Carvalho, "Towards the development of intelligent transportation systems," in *Proc. of IEEE Intelligent Transportation Systems*, Oakland, CA, USA, Aug. 2001, pp. 1206–1211.

[65] Z. Zhang, X. Wang, W. Anwar, and Z. L. Jiang, "A comparison of moments-based logo recognition methods," in *Proc. of Abstract and Applied Analysis*, vol. 2014, 2014, pp. 1–8.

[66] C. Huang, B. Liang, W. Li, and S. Han, "A convolutional neural network architecture for vehicle logo recognition," in *Proc. of IEEE International Conf. on Unmanned Systems*, Beijing, China, Oct. 2017, pp. 282–287.

[67] Y. Ou, H. Zheng, S. Chen, and J. Chen, "Vehicle logo recognition based on a weighted spatial pyramid framework," in *Proc. of 17th IEEE International Conf. on Intelligent Transportation Systems*, Qingdao, China, Oct. 2014, pp. 1238–1244.

[68] N. Farajzadeh and N. S. Rezaei, "Vehicle logo recognition using image matching and textural features," in *Proc. of Scientific Cooperations International Workshops on Electrical and Computer Engineering Subfields*, Istanbul,Turkey, Aug. 2014, pp. 82–87.

[69] J. Xiao, W. Xiang, and Y. Liu, "Vehicle logo recognition by weighted multi-class support vector machine ensembles based on sharpness histogram features," *IET Image Processing*, vol. 9, no. 7, pp. 527–534, 2015.

[70] S. Dai, H. Huang, Z. Gao, K. Li, and S. Xiao, "Vehicle logo recognition method based on tchebichef moment invariants and SVM," in *Proc. on IEEE WRI World Congress on Software Engineering*, Xiamen, China, May 2009, pp. 18–21.

[71] H. Pan and B. Zhang, "An integrative approach to accurate vehicle logo detection," *Journal of Electrical and Computer Engineering*, vol. 2013, p. 18, 2013.

[72] Q. Sun, X. Lu, L. Chen, and H. Hu, "An improved vehicle logo recognition method for road surveillance images," in *Proc. of 7th International Symposium*

on Computational Intelligence and Design, Hangzhou, China, Dec 2014, pp. 373–376.

[73] H. Yang, L. Zhai, L. Li, Z. Liu, Y. Luo, Y. Wang, H. Lai, and M. Guan, "An efficient vehicle model recognition method," *Journal of Software*, vol. 8, no. 8, pp. 1952–1959, 2013.

[74] R. Lipikorn, N. Cooharojananone, S. Kijsupapaisan, and T. Inchayanunth, "Vehicle logo recognition based on interior structure using SIFT descriptor and neural network," in *Proc. of International Conf. on Information Science, Electronics and Electrical Engineering*, Sapporo, Japan, Apr. 2014, pp. 1595–1599.

[75] S. Badura and S. Foltan, "Advanced scale-space, invariant, low detailed feature recognition from images - car brand recognition," in *Proc. of the International Multiconference on Computer Science and Information Technology*, Wisla, Poland, Oct. 2010, pp. 19–23.

[76] S. Sotheeswaran and A. Ramanan, "A classifier-free codebook-based image classification of vehicle logos," in *Proc. of IEEE International Conf. on Industrial and Information Systems*, Gwalior, India, Dec. 2014, pp. 1–6.

[77] Y. Gao and H. J. Lee, "Vehicle make recognition based on convolutional neural network," in *Proc. of International Conf. on Information Science and Security*, Seoul, South Korea, Dec. 2015, pp. 1–4.

[78] Y. Xia, J. Feng, and B. Zhang, "Vehicle logo recognition and attributes prediction by multi-task learning with CNN," in *Proc. of IEEE International Conf. on Natural Computation, Fuzzy Systems and Knowledge Discovery*, Changsha, China, Aug. 2016, pp. 668–672.

[79] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[80] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: The MIT Press, 2012.

[81] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, 2009.

[82] D. L. Donoho, "For most large underdetermined systems of linear equations the minimal l_1-norm solution is also the sparsest solution," *Communications on Pure and Applied Mathematics*, vol. 59, no. 6, pp. 797–829, 2006.

[83] E. Amaldi and V. Kann, "On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems," *Theoretical Computer Science*, vol. 209, no. 1, pp. 237–260, 1998.

[84] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. on Information Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[85] B. Bayar, N. Bouaynaya, and R. Shterenberg, "Kernel reconstruction: An exact greedy algorithm for compressive sensing," in *Proc. of 2014 IEEE Global Conference on Signal and Information Processing*, Atlanta, GA, USA, Dec. 2014, pp. 1390–1393.

[86] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1998.

[87] D. L. Donoho, Y. Tsaig, I. Drori, and J. L. Starck, "Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit," *IEEE Trans. on Information Theory*, vol. 58, no. 2, pp. 1094–1121, 2012.

[88] G. Ditzler, N. C. Bouaynaya, and R. Shterenberg, "AKRON: An algorithm for approximating sparse kernel reconstruction," *Signal Processing*, vol. 144, pp. 265–270, 2018.

[89] S. Ji, Y. Xue, and L. Carin, "Bayesian compressive sensing," *IEEE Trans. on Signal Processing*, vol. 56, no. 6, pp. 2346–2356, 2008.

[90] X. Zhang, J. Qin, and G. Li, "SAR target classification using Bayesian compressive sensing with scattering centers features," *Progress In Electromagnetics Research*, vol. 136, pp. 385–407, 2013.

[91] T. N. Sainath, A. Carmi, D. Kanevsky, and B. Ramabhadran, "Bayesian compressive sensing for phonetic classification," in *Proc. of IEEE International Conf. on Acoustics Speech and Signal Processing*, Dallas, TX, USA, Mar 2010, pp. 4370–4373.

[92] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Proc. of Advances in Neural Information Processing Systems*, Whistler, British Columbia, Canada, Dec. 2003, pp. 321–328.

[93] X. Mao, C. Shen, and Y.-B. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *Proc. of Advances in Neural Information Processing Systems*, Barcelona, Spain, Dec. 2016, pp. 2802–2810.

[94] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1-4, pp. 259–268, 1992.

[95] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Trans. on Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007.

[96] K. I. Kim and Y. Kwon, "Single-image super-resolution using sparse regression and natural image prior," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 32, no. 6, pp. 1127–1133, 2010.

[97] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image," in *Proc. of IEEE International Conf. on Computer Vision*, Kyoto, Japan, Sept. 2009, pp. 349–356.

[98] R. Timofte, V. De, and L. Van Gool, "Anchored neighborhood regression for fast example-based super-resolution," in *Proc. of IEEE International Conf. on Computer Vision*, Sydney, NSW, Australia, Dec. 2013, pp. 1920–1927.

[99] J. Yang, Z. Lin, and S. Cohen, "Fast image super-resolution based on in-place example regression," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 2013, pp. 1059–1066.

[100] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2016.

[101] S. Dreiseitl and L. Ohno-Machado, "Logistic regression and artificial neural network classification models: a methodology review," *Journal of Biomedical Informatics*, vol. 35, no. 5, pp. 352–359, 2002.

[102] C. Rainey, "Dealing with separation in logistic regression models," *Political Analysis*, vol. 24, no. 3, pp. 339–355, 2016.

[103] A. Gelman, A. Jakulin, M. G. Pittau, and Y.-S. Su, "A weakly informative default prior distribution for logistic and other regression models," *The Annals of Applied Statistics*, pp. 1360–1383, 2008.

[104] B. Carpenter, "Lazy sparse stochastic gradient descent for regularized multinomial logistic regression," Tech. Rep., 2008.

[105] J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, Q. V. Le, and A. Y. Ng, "On optimization methods for deep learning," in *Proc. of the International Conf. on Machine Learning*, Bellevue, Washington, USA, June 2011, pp. 265–272.

[106] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," Pittsburgh, PA, USA, Tech. Rep., 1994.

[107] L. Grippo and S. Lucidi, "A globally convergent version of the polak-ribiere conjugate gradient method," *Mathematical Programming*, vol. 78, no. 3, pp. 375–391, 1997.

[108] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. of IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, New York, NY, USA, Jun 2006, pp. 2169–2178.

[109] S. Yu, S. Zheng, H. Yang, and L. Liang, "Vehicle logo recognition based on bag-of-words," in *Proc. of 10th IEEE International Conf. on Advanced Video and Signal Based Surveillance*, Krakow, Poland, Aug. 2013, pp. 353–358.

[110] T. Hassner, V. Mayzels, and L. Zelnik-Manor, "On SIFTs and their scales," in *Proc. on IEEE Conf. on Computer Vision and Pattern Recognition*, Providence, RI, USA, June 2012, pp. 1522–1528.

[111] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *The Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.

[112] M. Shi, T. Furon, and H. Jégou, "A group testing framework for similarity search in high-dimensional spaces," in *Proc. of ACM International Conf. on Multimedia*, New York, NY, USA, Nov. 2014, pp. 407–416.

[113] A. Iscen, M. Rabbat, and T. Furon, "Efficient large-scale similarity search using matrix factorization," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, June 2016, pp. 2073–2081.

[114] M. W. Mahoney and P. Drineas, "CUR matrix decompositions for improved data analysis," *Proceedings of the National Academy of Sciences*, vol. 106, no. 3, pp. 697–702, 2009.

[115] E. Elhamifar, G. Sapiro, and R. Vidal, "See all by looking at a few: Sparse modeling for finding representative objects," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, Providence, RI, USA, June 2012, pp. 1600–1607.

[116] C. Boutsidis, M. W. Mahoney, and P. Drineas, "An improved approximation algorithm for the column subset selection problem," in *Proc. of Annual ACM-SIAM Symposium on Discrete Algorithms*, New York, New York, Jan. 2009, pp. 968–977.

[117] A. Vedaldi and B. Fulkerson, "VLFeat - an open and portable library of computer vision algorithms," in *Proc. of 18th ACM International Conf. on Multimedia*, New York, NY, USA, Oct. 2010, pp. 1469–1472.

[118] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.

[119] I. Sikiri, K. Brki, J. Krapac, and S. egvi, "Image representations on a budget: Traffic scene classification in a restricted bandwidth scenario," in *Proc. IEEE Intelligent Vehicles Symposium Proceedings*, Dearborn, MI, USA, June 2014, pp. 845–852.

[120] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Master's Thesis, Department of Computer Science, University of Toronto*, 2009.

[121] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," http://cvxr.com/cvx, Mar 2014.

[122] M. C. Grant and S. P. Boyd, "Graph implementations for nonsmooth convex programs," in *Recent Advances in Learning and Control*. Springer, 2008, pp. 95–110.

[123] S. M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, June 2016, pp. 2574–2582.

[124] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *CoRR*, vol. abs/1710.08864, 2017.

[125] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," in *Proc. of Advances in Neural Information Processing Systems*, Montral, Quebec, Canada, Dec. 2015, pp. 2017–2025.

[126] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *Proc. of International Conf. on Artificial Neural Networks*, Thessaloniki, Greece, Sept. 2010, pp. 92–101.

[127] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proc. of Advances in Neural Information Processing Systems*, Long Beach, CA, USA, Dec. 2017, pp. 3859–3869.

[128] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[129] H. R. Sheikh, M. F. Sabir, and A. C. Bovik, "A statistical evaluation of recent full reference image quality assessment algorithms," *IEEE Trans. on Image Processing*, vol. 15, no. 11, pp. 3440–3451, 2006.

[130] M. Böhm, M. R. Hutchings, and P. C. White, "Contact networks in a wildlife-livestock host community: Identifying high-risk individuals in the transmission of bovine TB among badgers and cattle," *PLoS One*, vol. 4, no. 4, p. e5016, 2009.

[131] H. C. J. Godfray, C. A. Donnelly, R. R. Kao, D. W. Macdonald, R. A. McDonald, G. Petrokofsky, J. L. Wood, R. Woodroffe, D. B. Young, and A. R. McLean, "A restatement of the natural science evidence base relevant to the control of bovine tuberculosis in Great Britain," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 280, no. 1768, pp. 1–9, 2013.

[132] M. Zeppelzauer, "Automated detection of elephants in wildlife video," *European Association for Signal Processing on Image and Video Processing*, vol. 2013, no. 1, pp. 46–69, 2013.

[133] C. A. Martinez-Ortiz, R. M. Everson, and T. Mottram, "Video tracking of dairy cows for assessing mobility scores," in *Proc. of Joint European Conf. on Precision Livestock Farming*, Leuven, Belgium, Sept. 2013, pp. 1–8.

[134] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *Proc. of International Conf. on Machine Learning*, Beijing, China, June 2014, pp. 647–655.

[135] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proc. of Advances in Neural Information Processing Systems*, Montral, Quebec, Canada, Dec. 2014, pp. 3320–3328.

[136] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of International Conf. on Learning Representations*, San Diego, CA, USA, Dec. 2015, pp. 1–15.

[137] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 142–158, 2016.

[138] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. of Advances in Neural Information Processing Systems*, Montral, Quebec, Canada, Dec. 2014, pp. 2672–2680.

[139] M. Brookes, "The matrix reference manual," *Imperial College London, [online] http://www.ee.imperial.ac.uk/hp/staff/dmb/matrix/intro.html*, 2005.

[140] D. J. MacKay, "Bayesian interpolation," *Neural computation*, vol. 4, no. 3, pp. 415–447, 1992.

[141] T. Fletcher, "Relevance vector machines explained," *University College London: London, UK, http://home.mit.bme.hu/ horvath/IDA/RVM.pdf*, 2010.