Enhanced Extreme Learning Machines for Image Classification

CUI DONGSHUN

Interdisciplinary Graduate School
Energy Research Institute @ NTU

2019

Enhanced Extreme Learning Machines for Image Classification

CUI DONGSHUN

INTERDISCIPLINARY GRADUATE SCHOOL

A thesis submitted to the Nanyang Technological University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

2019

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

29/3/2019
....
Date

Cui Dongshun
....
Cui Dongshun

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is of sufficient quality and grammatical clarity to be examined. To the best of my knowledge, it is free of plagiarism and the research and writing are those of the candidate except as acknowledged in the author Attribution Statement. To the best of my knowledge, the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

Apr 2, 2019

Date

Huang Guangbin

Huang Guangbin

Authorship Attribution Statement

This thesis does not contain any publication documents. Therefore, this form does not apply for my thesis submission processes.

29/3/2019	Cui Dongshun
....	....
Date	Cui Dongshun

Acknowledgements

First, I would like to thank my supervisor Prof. Huang Guang-Bin for his support and excellent supervision. Thank you for helping me overcome the life and academic obstacles I met these years, and providing help in research thinking and art sense. I will always remember the good advice I have received from you these years. I am also grateful to my co-supervisor Assoc. Prof. Zhang Jie and Assoc. Prof. Lin Zhiping who have given me their kind supervision. Thank you for your time spent on helping me find my way to be a researcher.

Great thanks to my project leader, Dr. Yang Yan, who has always been leading me to a young professional researcher when I was in BMW-NTU Future Mobility Lab. She is enthusiastic and always prepared whenever I have questions, and I cannot achieve such works without her help in my first year here. Special thanks to Dr. Kai Hu, Dr. Liyanaarachchi Lekamalage Chamara Kasun, Ms. Liu Tianchi, Mr. Song Kang, Mr. Han Wei, Mr. Zhang Guanghao and Mr. Li Yue when I was in Delta-EEE Joint Research Lab, with whom I have worked closely and published several papers in the past few years. Thank you for the interesting discussions and celebration gatherings.

Thanks to the support from Nanyang Technological University (NTU), Energy Research Institute @NTU (ERIAN), Interdisciplinary Graduate School (IGS), BWM-NTU Future Mobility Lab and Delta-EEE Joint Research Lab at School of Electrical and Electronic Engineering at NTU.

Last but not least, I would like to thank my parents, my sister, my wife and all other family members for their love, understanding, encouragements, and support.

Table of Contents

Abstract

Image Classification is one of the critical computer vision tasks, and it is also the foundation of the related tasks like object detection/recognition/segmentation, which all need to identify the positive and negative samples. Methods proposed for image classification can be divided into two groups: traditional image processing based methods and modern machine learning based methods. The main difference between these two categories is how to extract the features, including the common features of the intraclass samples and the distinguishing features of the interclass samples. For traditional image processing based methods, researchers focus on extracting features manually after the mere observation of the samples. While for modern machine learning based methods, and researchers tend to focus on designing networks based on the training data, building a model, and evaluating the model on the test data.

Among numerous machine learning methods, we choose the Extreme Learning Machines (ELMs) for our image classification applications. ELM has been proposed in more than a decade ago, as a single layer feedforward neural network, its primary objective is designing classification and regression models. ELM and its variants have been applied widely in the past years in many fields, including computer vision tasks, time sequence signals analysis, and so on. A basic ELM network consists of an input layer L_1, a hidden layer L_2, a output layer L_3, and the connections between L_1 and L_2, and L_2 and L_3. From the view of data flow, it can be summarized as raw data or features of the samples are fed into the input layer, then flow into the hidden layer and the output layer by being operated with similar connections. The most prominent highlight of ELM is the weights and biases in the connections between L_1 and L_2 are randomly assigned. The aim of ELM is to computing the weights between L_2 and L_3 based on minimizing the sum of the norm of the weights and the errors of predicted outputs and target outputs. In this thesis, we focus on enhancing four aspects of ELM networks and apply them to the image classification.

The first work in this thesis focuses on the input layer of the ELM network. As introduced above, raw data or extracted features of the samples are fed into the network. Raw data is the simple straight-line input data for general image classification, but for some semantic images (e.g., smile images, gesture images), the purposefully designed feature extraction methods are also essential. In our first work, we take smile image classification task (or called smile detection) as an example to show the effectiveness of our handcrafted feature extraction method named "pairwise distance vector". Compared with raw data and other existing state-of-the-art smile image classification methods, the extracted features have advantages of low dimension, easy to understand and visualize, and high efficiency.

The second work in this thesis focuses on the connections between the input layer and the hidden layer. Researchers have been assuming the random weights and biases between L_1 and L_2 obey uniform distribution or Gaussian distribution since the ELM was put forward. However, this may lead to the requirement of a significant amount of memory and time to process them, especially when there are many ELM networks in one system. Inspired by the discovery on fruit flies' olfactory systems, we proposed a method named sparse binary ELM (or Fly-ELM), which adopts sparse binary random projections between the input layer and hidden layer. The Fly-ELM achieves comparable classification accuracy, but costs much less memory and requires less training and testing time when compared with the related methods.

The third work in this thesis is extending the single-layer network to a multi-layer network by using ELM Auto-Encoder (ELM-AE). Due to the simple structure, the performance of the ELM networks on large datasets has a certain gap with the existing state-of-the-art methods. Motivated by the PCA Network (PCANet), we have extended the single layer ELM network to multiple layers by using ELM-AE and named our new method as CFR-ELM. The results show that CFR-ELM obtains higher accuracies than PCANet on some large public datasets for image classification.

The last work in this thesis focuses on the output layer of the ELM network. In the past years, the reflection of the similarity of the input samples on target codes (the coding of the labels) hasn't been taken into account. However, for image classification tasks, the intuitive impression is that similar objects should be assigned relevant target codes. In this thesis, we have analyzed different target coding methods and evaluated the performance of them on image classification.

Overall, the thesis contributes to these four aspects. From the view of efficient input data, we have designed a handcrafted feature extraction method for smile images classification. From the perspective of the distribution of random weights between the input layer and the hidden layer, we have proposed and proved the effectiveness of the sparse binary ELM. Inspired by the deep architecture of deep learning, we have extended the single layer to multiple layers of ELM networks to achieve good performance on large image classification datasets. Finally, from the point of target coding, we have introduced and evaluated different target coding methods for image classification.

List of Figures

List of Tables

List of Acronyms

AE	Auto-Encoder
ANNs	Artificial Neural Networks
CFR	Compact Feature Representation algorithm
DNN	Deep Neural Networks
ELM	Extreme Learning Machine
ELM-AE	Extreme Learning Machine Auto-Encoder
GPU	Graphics Processing Unit
HOG	Histogram of Oriented Gradients
HSI	hyperspectral imagery
KNN	K-Nearest Neighbor
LBP	Local Binary Patterns
LRF	Local Receptive Fields
OCR	Optical Character Recognition
PCA	Principal Component Analysis
SIFT	Scale-Invariant Feature Transform
SVM	Support Vector Machines
WTA	Winner Take All

Chapter 1

Introduction

Chapter 1 begins with the background and motivation for studying image classification based on extreme learning machines. Then this chapter continues to set the scope of the study on how to adopt the enhanced extreme learning machines into image classification by combining traditional feature extraction, sparse binary projection, deep architecture, and target coding, respectively. Furthermore, the objectives and contributions of this thesis are explained. At last, the structure of this thesis is presented.

1.1 Background and Motivation

Image classification (also known as image recognition) refers to the task of extracting/learning features from images and classifying them into different categories [3, 4]. That is, a label is assigned to the given image based on the decision from the extracted/learned features and the classifier [5]. From the perspective of the types of images, image classification can be further divided into digits image classification [6], texture image classification [7], scene image classification [8], face/smile/gesture image classification [9, 10, 11], general image (including different kinds of objects) classification [12, 13], and so on. Image classification is an important foundational for many computer vision fields, like object detection/tracking [14, 15], semantic/instance segmentation [16, 17], etc. Since all these tasks are related to the classification of positive samples (e.g., true object and instance) and negative samples (e.g., background) in region level or pixel level.

From the definition of image classification, we know that there are two main parts of a primary image classification system: feature extraction module and classification module. Feature extraction/learning is about how to extract/learn effective features from the raw images, while the classification module is about how to use classifiers to classify raw images by using the extracted features. Researchers have proposed a large number of methods for feature extraction and designed kinds of classifiers in recent decades.

Feature extraction methods (feature descriptors) usually refer to the traditional image processing methods, which can be roughly devided into two categories: local region-based feature descriptors (e.g., Scale-Invariant Feature Transform (SIFT) [18], Local Binary Patterns (LBP) [19], and Histogram of Oriented Gradients (HOG) [20]) and global region-based feature descriptors (e.g., Gabor features [21] and Principal Component Analysis (PCA) [22]). While feature learning usually refers to the modern computer vision methods that can automatically find the features required for image classification tasks from raw images. Features learning also can be roughly divided into categories: supervised learning (with labeling information) and unsupervised learning (without labeling information).

Classifiers are utilized to analyze the extracted/learned features, associate these features with the labels of raw images in the training stage, and predict the labels of test images in the testing stage. Many famous classifiers have been proposed, including Fisher's linear discriminant [23], k-nearest neighbor (KNN) [24], Support Vector Machine (SVM) [25, 26], Artificial Neural Networks (ANNs) [27], random decision forests [28] and so on. Specifically, current popular neural networks like Extreme Learning Machine [29] and Deep Neural Network (DNN) [13] all belong to ANNs.

Noted that feature learning and classification can also be associated together, and that is what researchers have been doing nowadays. ELM, DNN, and other neural network-based methods can learn features from the raw images and complete the classification task at the same time in one network [30, 31].

1.2 Research Scope and Questions

We have introduced the two main steps for image classification, and in this section, the research scope and questions will be put forward. Many traditional methods have been proposed for image classification, and most of them focus on designing handcrafted feature descriptors (find more details in Chapter 2). These descriptors usually work well for some specific tasks like corner point detection (classification of pixels), edge detection, and shape detection. Then, as the improvement of the requirements for practical applications and the development of the classifiers, traditional methods meet the modern neural network-based methods like ELM and DNN.

ELM is a very promising algorithm for both feature learning and classification. As a single-hidden-layer feedforward neural network, ELM has attracted more and more attention from both academia and industry. ELM theories show that hidden neurons can be assigned randomly and ELM has been applied in many applications, including clustering, classification, and regression [32, 33]. In recent years, researchers have proposed local receptive fields based extreme learning machine (ELM-LRF) [34] and ELM Auto-Encoder (ELM-AE) [35], which further enhance ELM's representation learning ability locally and globally.

Compared with ELM, DNN has also begun to shine in the field of computer vision over the past few years. With the development of graphics processing units (GPUs), graphics computing power has become stronger, leading deeper and deeper neural networks have been designed and implemented for image classification. Here, we have listed some popular and famous deep neural network-based methods for image classification in chronological order, they are Alex network (AlexNet) [13], ”VGG” network (VGGNet) [36], and deep residual network (ResNet) [37]. These networks are proposed and have been adopted to win many image classification competitions.

However, in this thesis, we focus our research on enhancing the ELM network for image classification rather than choosing DNN-based methods due to the following reasons:

(1) Strong theoretical derivation. The ELM has rigorous theoretical proof, while DNN is more like an engineering technique, which has no solid theoretical proof yet.

(2) Fast in both training and testing stages. ELM networks usually have much fewer parameters than DNNs, and all parameters in ELMs either are randomly generated or computed immediately without any iterations.

(3) Easy for implementation and visualization. ELM is much easier than DNN in implementation and visualization since DNN is like a “black box” and it is hard to visualize.

ELM belongs to machine learning, and like many other machine learning methods, it has been applied to image classification and many other tasks. However, there are still some research gaps in the ELM-based image classification methods, and four of them are listed as follows.

Firstly, as we have introduced above, many researchers just fed the raw data into the input layer of ELM networks. Compared with traditional feature extractions methods, it is more convenient to use raw data as the input data, and it is more suitable for general cases. However, for the specific tasks, extracted features are usually more efficient, and flexible than raw data and the extracted features are generally meaningful and easy to understand. That is, for the specific tasks (like smile image classification), it is not a good choice to use the raw data as the input data immediately.

Secondly, one of the core ideas of ELM is that weight matrix **W**, and bias matrix **B** between the input layer and the hidden layer are randomly generated based on a continuous sampling distribution probability and independent of training data [32]. Huang *et al.* first introduced and implemented uniform random weights and biases [38] for ELM on regression and classification problems. From that on, most ELM researchers follow the similar way of generating **W** and **B**, while the majority of the remaining researchers initialize **W** and **B** from the normal/Gaussian distribution [39, 40]. Even ELM with random uniform or normal distributed weights and biases has been applied in many fields, and remarkable results have been obtained, it still has some inconveniences that have not been solved. For examples, random uniform weights are not easy for understanding or visualization, and hard for hardware implementation due to the precision problem. Besides, random weight matrices with a float/double value for each entry takes up a large amount of memory.

Thirdly, ELM is a single-layer neural network, and most of its variants share the single or shallow structure. We know that with the development of neural networks, more and more work has been done on feature learning by adopting deep neural networks, which have achieved impressive results in feature learning with its ability to unfold the potential of the data with many hidden layers [36, 41]. There is no doubt that network-based methods with deep architecture outperform most of the ones with single or shallow architecture for computer vision tasks, even there are millions of parameters need to be tuned for such networks usually, and it may require months to train. So, deep architecture should be considered into the ELM networks.

Fourthly, target coding (coding for the labels of the samples) is an indispensable part of image classification. With years of research, there are lots of work that have been done on how to extract useful features using ELMs [42, 43, 44, 45], but little work has been done on target coding for ELM. By far, most people adopt one-of-k target coding methods instantly when they use ELM classifiers, while others choose the numeric coding methods to simplify the computation. One-of-k (one-hot) coding (specifically refer to “encoding” in this thesis) comes from electronics, and there is only one ”hot” value in the coding list. It was introduced into the modern data science by [46], and the one “hot” value is replaced with “1” for simplicity reasons. One-of-k coding is the most frequently used target coding method for machine-learning based classification applications in recent years [33, 47, 48]. More target coding methods should be introduced and explored for ELMs.

1.3 Objectives and Contributions

The objective of this thesis is enhancing four aspects of ELM networks for image classification problems. We have listed and analyzed four existing research gaps in Section 1.2, and here we will explain our contributions which correspond to these four gaps.

In our first work, we focus on the data which is fed to the input layer of the ELM network. We propose a novel feature extraction method for smile image classification by using a short and snappy feature descriptor named Pair-wise Distance Vector, which is extracted only from the facial landmarks around the mouth. We have tested our algorithms on the public smile image classification database, and the results indicate that our method is better than the state-of-the-art methods with higher accuracy and lower dimension of features. Particularly, we list the results of the comparison between the raw data and the extracted features in Chapter 3.

In our second work, we concentrate on the distribution of the random weights between the input layer and the hidden layer. Inspired by the findings of the fruit fly olfactory system, combining this "fly algorithm" and the earlier work on random binary ELM as well as sparse solutions for ELM, we propose a novel ELM-based image classification method named sparse binary extreme learning machine. In short, we name it as Fly-ELM to highlight its biological property found in the fruit fly olfactory system. Fly-ELM introduces the sparse binary random matrix into ELM networks, which is different from the consecutive binary/ternary ELMs and sparse binary random winner-take-all (SBR-WTA). The sparse binary projection restricts the number of non-zero coefficients, which guarantees non-overfitting and low computational cost. Meanwhile, the random projection with nonlinearity satisfies the universal approximating capability theory of ELM networks.

In our third work, we embed the deep architecture into the ELM networks and propose a Compact Feature Representation algorithm (CFR-ELM) by using Extreme Learning Machine (ELM) under a shallow (could be extended to deep) network framework. CFR-ELM consists of a compact feature learning module and a post-processing module. Each feature learning module in CRF-ELM performs the following operations: 1) patch-based mean removal; 2) ELM-AE based feature learning; 3) Max pooling to make the features more compact. Post-processing module is inserted after the feature learning module and simplifies the features learn by the feature learning modules by hashing and block-wise histogram. We have tested CFR-ELM on four typical public image classification databases, and the results demonstrate that our method outperforms the state-of-the-art techniques.

In our fourth work, we analyze and explore the effects of one-of-k coding methods on ELM classifiers for image classification. Besides, motivated by the properties of orthogonality and equal weight (each coding has the same number of non-zero elements) of Hadamard coding [49, 50, 51], we have compared its effects with one-of-k coding. Two simple coding methods ordinal coding and binary coding have also been compared to show the effectiveness of one-of-k coding and Hadamard coding. To the best of our knowledge, this is the first discussion on target coding for ELM on image classification.

Noted that even these four contributions are presented individually, they can be arbitrarily combined to accomplish various tasks. We introduce them one by one by order of the data flows through the ELM networks in this thesis.

1.4 Thesis Overview

The remaining chapters of this thesis are organized as follows. (This thesis interpolates material from the papers written by the author, and all these papers' first author is the author of this thesis.) Chapter 2 goes through the related work on ELM networks and existing image classification methods, including the traditional methods and learning-based algorithms. Particularly, we review and analyze the ELM-related methods on image classification in detail.

Chapter 3 uses the material from References [52, 53]. We take smile images classification as a carrier to demonstrate our novel feature extraction method named pair-wise distance Vector, which can fully reflect object's shape and has properties of invariance of translation, rotation, and scaling.

Chapter 4 adopts the material from Reference [54]. We propose a novel neurology-based method named Fly-ELM by embedding sparse binary random projections into the existing ELM network for image classification. We also present the architecture of the Fly-ELM, and each module of the network will be demonstrated and derived clearly.

Chapter 5 uses the material from References [45, 55]. We propose a novel feature representation method named CFR-ELM by using ELM-AE for image classification. CFR-ELM is a multi-stage feature learning network, which contains a patch mean-removal process and ELM-AE in each stage. The former removes the influence of illumination, while the latter learns the representation of the input patches, filters the raw samples, and outputs the practical features. Two post-processing steps (binary hashing and block-wise histogram) are needed to simplify the final output features.

Chapter 6 adopts the material from Reference [56]. We explore the effectiveness of target coding on ELM classifiers for image classification. In particular, we have analyzed two linearly dependent coding (ordinal coding and binary coding) and two linearly independent coding (one-of-k coding and Hadamard coding).

Chapter 7 summarizes this thesis and points out several future research directions.

Chapter 2

Literature Review

Chapter 2 begins with the introduction of basic ELM network, some ELM variants are presented and explained, and the ELM-AE is highlighted. Then this chapter continues to review some typical methods for image classification, including traditional image processing based methods and machine learning based methods. At last, the ELM-related methods for image classification are reviewed and introduced in detail.

2.1 Extreme Learning Machines (ELMs)

The feedforward neural networks were developed in the 1990s [57] and spread quickly due to their abilities on approximating any measurable function to any desired degree of accuracy. That is, regarding image classification problems, we can always find complex unknown mappings for the training samples and their corresponding labels by using feedforward neural networks. The challenging problem is how to find or calculate the weights and biases in the networks at the expense of reasonable consumption. The existing methods can be divided into two categories roughly according to the need of iteration:

(1) Inerative methods: gradient-descent learning based methods (e.g., back-propagation (BP [58]) and convolutional neural network (CNN [59])), kernel learning based methods (e.g., support vector machine (SVM [60])), and so on.

(2) Non-iterative methods: least square (LS) learning based methods (e.g., extreme learning machines (ELMs [29])).

ELM is the core of our research, but SVM is also a relatively popular learning algorithm, and it is also adopted in our work. So, here we give a brief review of SVM. SVM constructs an optimal hyperplane for the independent samples in feature or kernel space to maximize the functional margin $\frac{2}{||w||}$ (w is the normal vector to the hyperplane) and minimizing the misclassification cost ξ (approximately equivalent to the number of misclassification samples) simultaneously. Smaller misclassification cost means better fitting for training data while a larger functional margin implies better generalization ability for test data. Fig. 2.1 illustrates the principle of SVM where the functional margin $\frac{2}{||w||}$ and clipped ℓ_1-norm misclassification cost function ξ are shown.

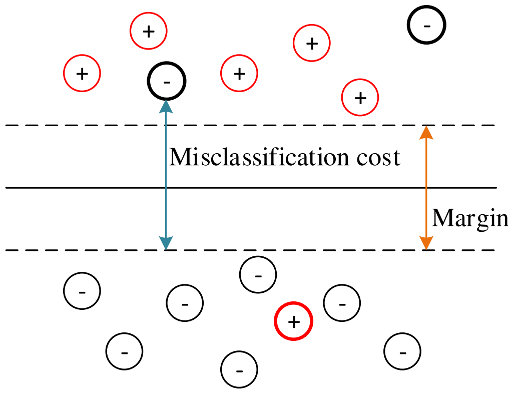


Figure 2.1: The principle of support vector machine.

The mathematical optimization formulation of SVM is:

$$\begin{aligned} &\underset{\mathbf{w},\xi,b}{\textit{minimize}} \quad \frac{1}{2}\|\mathbf{w}\|_2^2 + C\|\boldsymbol{\xi}\|_1, \\ &\text{subject to} \quad y_i(\mathbf{w}\phi(x_i)+b) \geq 1-\xi_i, \quad \xi \geq 0. \end{aligned} \tag{2.1}$$

Where ξ is a non-negative slack variable, b is the bias term, y is the label, and C is the hyperparameter of SVM which indicates the trade-off between training accuracy and generalization ability. The advantage of SVM is the sparseness of the solution and good generalization ability even in a very high dimensional space. The disadvantage of SVM is that its large memory requirement and slow computation in large-scale tasks. SVM also leads to imbalanced distortion of data for multi-class classification.

ELMs (shown in Fig. 2.2), originally proposed as "generalized" single-hidden-layer feedforward networks (including but not limited to sigmoid neural networks, RBF networks, threshold networks, Fourier series, Wavelets, etc.), has been extensively studied in the past decade and attracted increasing attention from academia and industry [38, 30, 61, 62, 63]. ELM theories demonstrate that hidden neurons can be set randomly in many applications, including clustering, feature learning, regression, and classification [64, 45, 65, 66]. Many variants of the ELM have been proposed in the last decade, and most of our work concentrate on the early ELM (called "basic ELM" here), explore and enhance its abilities of image classification and feature learning. In the following paragraphs, we will summarize the basic ELM in detail and some ELM variants in brief.

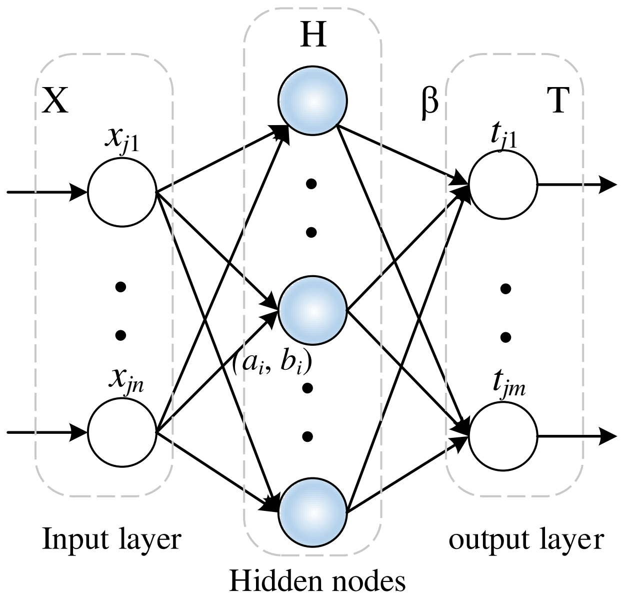


Figure 2.2: The diagram of extreme learning machine (ELM).

Basic ELM. ELM is a very promising algorithm for classification and feature learning. The weights (**W**) and bias (**B**) between the input layer and the hidden layer are randomly assigned, while only the weights (β) between the hidden layer and the output layer need to be computed by solving a linear equation. This network has been proved to have good generalization performance with high training and test speed with remarkable classification and regression abilities [29, 67, 30, 64]. ELM assigns the parameters (a_i, b_i) of the hidden nodes randomly without any iterative tuning [38], which makes it much faster than traditional single hidden layer feedforward neural networks. In classification, assume there are N samples and the training set is $\{(x_j,t_j)|x_j \in R^n, t_j \in R^m, j = 1, \cdots, N\}$, where n is the dimension of each feature vector $\mathbf{x}$, and m is the length of output vector $\mathbf{t}$ (or $\mathbf{T}$) (m is 1 here because smile detection is a binary classification problem, and t_i is -1 or 1). The hidden node output function $G(\mathbf{a}_i, b_i, \mathbf{x})$, and $i = 1, \cdots, L$ (L is the number of hidden nodes). So the hidden layer output matrix is $\mathbf{H}$, and it can be easily calculated since we already know that the $\mathbf{x}$ and $(\mathbf{a}_i, b_i)$ are randomly assigned.

$$\mathbf{H}(\mathbf{x}) = \left[G(\mathbf{a}_1, b_1, \mathbf{x}), \cdots, G(\mathbf{a}_L, b_L, \mathbf{x})\right]. \tag{2.2}$$

Set β as the output weights, and β_i as the weight vector connecting the ith hidden node and output node. Then, the output of ELM is:

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}). \tag{2.3}$$

From the above two equations, we can get $\mathbf{H}\beta = \mathbf{T}$, that is

$$\beta = \mathbf{H}^{\dagger}\mathbf{T}, \tag{2.4}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose pseudoinverse of $\mathbf{H}$. Once we obtain β, together with the same weight matrix W between the input layer and the hidden layer, we can predict the label of new input features.

ELM networks have the “universal approximation capability” [68, 62, 61] and “classification capability” [30]. That is, with the randomly assigned hidden neurons and appropriate output weights, ELMs can approximate any target function (“universal approximation capability”) and separate arbitrary disjoint regions of any shapes (“classification capability”) by only tuning the amount of the hidden neurons. So, it is very intuitive to apply ELM networks to regression problems and classification problems. Besides, ELM networks can be adopted in the fields of compression [69], clustering [70] and feature learning [71, 72].

In past decades, many researchers have developed many ELM variants, and some famous variants are listed below in the order of the time when they are first proposed, and brief comparisons with basic ELM are presented.

(1) Online sequential extreme learning machine (OS-ELM) [73, 74]. Liang *et al.* proposed a sequential learning algorithm based on the basic ELM called OS-ELM in [74]. In basic ELM, we feed the training samples into the network at once, and it works quite well when the amount of the training samples are small. Limited by storage space and computing power, it becomes more and more challenging to train the network when the amount of training samples increases dramatically. OS-ELM is proposed to solve this kind of problems by feeding the samples in the form of batching and update the weights between the hidden layer and the output layer iteratively. The idea of iterative update affects the work of many people afterward, including the third and fourth contributions in this thesis.

(2) Regularized extreme learning machine [75, 30, 76]. From the above explanation of basic ELM, we know that we calculate the weights β directly by using the matrix operation, and there is no constraint on it. While the objective of the regularized ELM is to minimize both the training errors and the norm of β, and it can be written as:

$$\underset{\beta}{\operatorname{argmin}} \frac{1}{2}||\beta||^2 + \frac{1}{2}C||\mathbf{H}\beta - \mathbf{Y}||^2, \tag{2.5}$$

where C is called the regularization factor and it is a hyperparameter. Researchers have proved that the performance of the regularized ELM has improved a lot in most cases without increasing training time [75]. Our second work in this thesis is motivated by the regularized ELM, and we review it again with more details in Section 4.2.

(3) Extreme Learning Machine Auto-Encoder (ELM-AE) [35, 55]. ELM-AE is proposed in [35] to learn features of the input data using singular values for big data processing. The most noticeable difference between ELM-AE and basic ELM is that the target output is the same as the input in ELM-AE, while the target output is the label information in basic ELM. Constrained by orthogonality, the weights between the hidden layer and the output layer can learn discriminative features of

the input samples. That is, β is calculated by Equ. 2.4 and

$$\beta^T \beta = \mathbf{I}, \tag{2.6}$$

where $\mathbf{I}$ is the identity matrix. Our third work is deeply motivated by ELM-AE and another existing method called PCA network, and we will introduce the theory and implementation procedure of ELM-AE in detail in Section 5.3.

(4) Local receptive field based extreme learning machine (ELM-LRF) [34]. Basic ELM and ELM-AE adopt full connections and response to the input samples globally, while Bai *et al.* introduced local connections to ELM and recept the input data locally. Besides, ELM-LRF shows that each hidden node in ELM can be a combination of several hidden nodes, which can be taken as a subnetwork. The combination of the hidden nodes can be random, and the weights can be randomly generated just like basic ELM. The idea of local connections and subnetworks of ELM-LRF inspire our first work and the third work.

2.2 Image Classification

In this section, we will review some typical image classification techniques. One of the critical foundations is the images which are used for classification, so first of all, we list some famous public datasets for image classification tasks. Similarly, we introduce them in order of time, while the complexity of these datasets is in the same order.

(1) Coil-20/100 datasets [77, 78] were estimated in 1996, and contains 20 and 100 different objects respectively. Each image contains only one object, and the object is fine segmented, that is, the background of each image is black.

(2) MNIST dataset [59] was created in 1998, and all the images in this dataset are handwritten digits from zero to nine. The background of each digit is also filled with black pixels.

(3) ETH-80 dataset [79] was collected in 2003, and the background is colored. ETH-80 contains 80 objects from 8 chosen categories, that is, each category has ten subcategories.

(4) OCR letter dataset [80] was built in 2004. It is similar to the MNIST dataset but contains handwritten letters rather than digits.

(5) GENKI-4K [81] dataset was collected in 2009. As one of the most famous smiling and non-smiling images database, it has been used in many smiling image classification (or smile detection) related work. In this thesis, this dataset is adopted in Chapter 3 to evaluate the performance of our novel method by combining hand-craft features and ELM classifier.

(6) CIFAR10/100 datasets [12] are created in 2009. Each image contains one object with a complex background.

Different datasets are collected under different scenarios, and the more complicated the scene, the harder it is for image classification. All the datasets mentioned above will be introduced in detail in the following corresponding following chapters. In this thesis, we have chosen different public datasets to test our proposed methods, and the criteria for selecting a dataset is that it could best describe the characters of our approaches.

In the following paragraphs, we will introduce several existing typical image classification methods. Computer vision began in the late 1960s and image classification is one of the first tasks. In 1966, Seymour Papert [82] started a summer vision project, which attempted to construct a significant part of a visual system, and the final goal of this project is to name/classify objects with the known objects by matching the descriptions/features.

Studies from the 1970s focused on extracting features of the basic elements like corner [83], line [84], edge [85], and shape [86]. Image classification related tasks mainly based on these basic descriptions and matching techniques in the 20th century, and most of them are only suitable for the ideal cases [87, 88]. It is notable that Haralick *et* *al.* proposed a novel image classification method by computing texture features in [3], which inspires the following texture-based approaches. The texture is an innate property of surfaces of objects, and it is a pattern that could be used to identify different objects. Three easily computable textural features f_1, f_2, f_3 are expressed as:

(1) Augular Second Moment:

$$f_1 = \sum_i \sum_j p^2(i,j). \tag{2.7}$$

(2) Contrast:

$$f_2 = \sum_{n=0}^{N_{g-1}} n^2 \left(\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j) \right), \quad (|i-j| = n). \tag{2.8}$$

(3) Correlation:

$$f_3 = \frac{\sum_i \sum_j (ij) p(i,j) - \mu_x \mu_y}{\sigma_x, \sigma_y}. \tag{2.9}$$

Here, $p(i,j)$ is the (i,j)th entry in the image, N_g is the quantized levels, μ_x, μ_y, σ_x, and σ_y are the means and standard devisions of p_x and p_y, while $p_x(i)$ and $p_y(j)$ are the ith and jth sum of the rows of $p(i,j)$ respectively.

The rapid development of image classification originated in the early 2000s, and the landmark event is that Paul Viola and Michael Jones proposed a machine learning approach for simultaneous object detection (recall that object detection is essentially an image classification problem of distinguishing between positive and negative sample images). This method consists of three main keys: Haar-like features, AdaBoost classifier, and the Cascade structure. Haar-like features are represented as the difference of the sum of two/three/four rectangle areas in a digital image, and the integral image is introduced to calculate the sum of a rectangle area efficiently. Assume the pixel value of an image at (i,j) is $p(i,j)$ and the pixel value of its corresponding integral image at

(i, j) is $p_{int}(i, j)$, then we have

$$p_{int}(i, j) = \sum_{i' \leq i} \sum_{j' \leq j} p(i', j'). \tag{2.10}$$

The illustration of an integral image is shown in Fig. 2.3, pixel value at location 1 (indicated as $p_{int}(1)$) in the integral image is the sum of pixels in the area A in the raw image (indicated as S_A). Then we have $p_{int}(1) = S_A$, $p_{int}(2) = S_A + S_B$, $p_{int}(3) = S_A + S_C$ and $p_{int}(4) = S_A + S_B + S_C + S_D$. So, we can calculate the sum of all pixel values of any region in a linear time, for example, $S_D = p_{int}(4) + p_{int}(1) - p_{int}(2) - p_{int}(3)$.

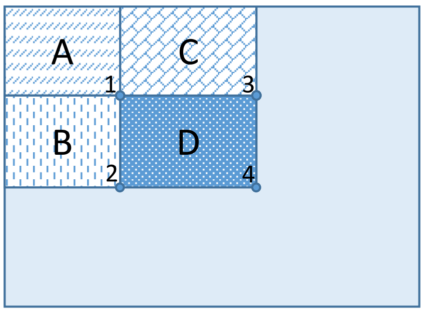


Figure 2.3: Illustration of an integral image.

The second step is feeding the extracted haar-like features and learn classifiers based on AdaBoost. AdaBoost is an iterative algorithm, which trains different weak classifiers based on the same training samples. In a real application, many weak classifiers are chained together to form a strong classifier to achieve increased detection accuracy.

The third step is detection with cascade structure, whose schematic is shown in Fig. 2.4, and “1”, “2” and “3” represent the AdaBoost classifiers. From the schematic, we can see that the cascade structure detection completes coarse-to-fine processing by striking out false samples step by step.

With these three steps, an efficient face detection method is proposed, and this method has a profound impact on the following object detection (and image classification) approaches. By following the framework of efficient feature extraction methods and appropriate classifiers, researchers have presented many practical image classification systems with high performance.

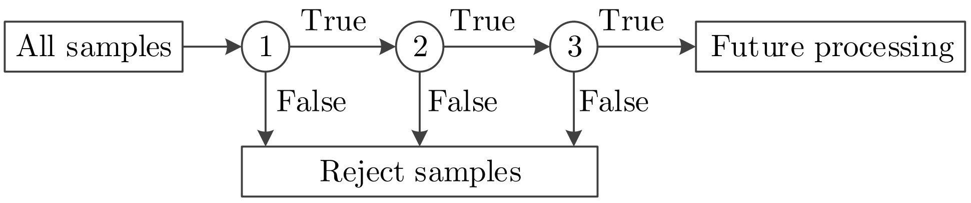


Figure 2.4: Cascade structure detection.

Later, Pietikainen *et al.* proposed an excellent feature extraction method named local binary patterns (LBPs) and applied it to the multiresolution gray-scale and rotation invariant texture images classification system in 2002 [19], and to the practical face recognition system in 2006 [89].

After that, methods of combining LBPs and suitable classifiers (e.g., SVM and ELM) have been implemented in many other fields such as gender/age/ethnicity classification and facial expression classification with excellent performance [90, 91, 92, 47]. In this thesis, we have compared some of our work with LBP-based methods, so here we give a brief review of this famous feature descriptor. The principle of LBP is very straightforward. First, each pixel value is compared with its eight neighboring pixels. If the adjacent pixel value is smaller than the center value, it is assigned with the value 0; otherwise, it is assigned with 1. Then, the eight adjacent values are treated as an 8-bit binary number and this binary number is converted to a decimal value. The center pixel is then assigned with this decimal number. The LBP basic operator is shown in Fig. 2.5.

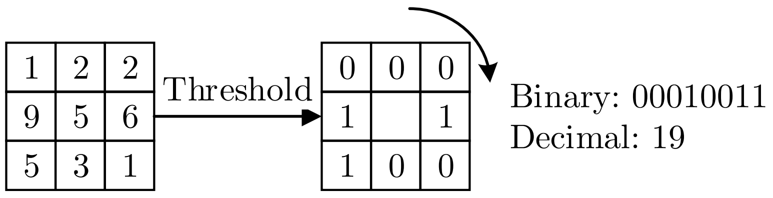


Figure 2.5: Basic local binary pattern operator.

This process of extracting features first and then putting features into one or many classifiers is still prevalent until now.

Deep neural networks was developed for image classification in the 1990s, and the first popular deep neural network is proposed in [6] for handwritten digit image classification. In 2012, benefit from the growth of computing power of GPUs, DNN-based (or deep learning based) image classification methods became very hot. As one of the most representative deep neural networks, AlexNet was implemented on ImageNet (the large image classification dataset at that time) and achieved the state-of-the-art result [13]. After that, VGGNet [36] was proposed in 2014, and it achieved better results with more layers than Alexnet. Later, ResNet [37] and DenseNet [93] were developed to explore the ability of cross-layer connections. The deep architecture of these methods has inspired our third work in this thesis, and we have analyzed the difference between DNN-based methods and ELM-based methods in Chapter 1. We haven't compared our methods with DNN-based methods since our main work is on how to enhance the ELM networks for image classification.

Even deep learning based methods have achieved very high accuracies on some specific datasets, they still suffer from some inherent disadvantages like the lack of solid theoretical proof, consuming enormous computing resources, costing long training time and so on [55].

2.3 Existing ELM-Based Methods for Image Classification

Lots of ELM-based algorithms for image classification have been proposed, and they can be grouped into the primitive application and advanced application.

The primitive ELM-based application means algorithms only use the ELMs as classifiers. In [47], a framework for hyperspectral imagery (HSI) classification is presented by extracting LBP features and using ELM as a classifier. As we have introduced above, LBP descriptor is proposed for texture classification, and then it is applied to many other fields like describing the local spatial pattern. Here, Li *et al.* extracted LBP features of the images from real hyperspectral datasets and then fed them into the ELM networks.

They have compared their proposed method with other related approaches and found that LBP representations are quite useful in HSI spatial features, and the ELM classifier outperforms SVM-based algorithms.

Many other primitive uses of ELMs mostly within the framework of extracting features and classifying these features with ELMs [94, 95, 96, 97]. Most of them work quite well for some specific applications, but not suitable for other uses. In this thesis, our first proposed system is for smiling images and non-smiling images classification, and the novel feature descriptor we have designed is based on the facial landmarks since facial landmarks detection methods are more and more accurate and efficient nowadays [98]. From the view of the ELM network, our first work focuses on its input layer.

The advanced ELM-based application means the methods modify the structures of existing ELMs, including diverse optimization of the weights [32, 35], various connection methods between different layers [34, 43, 99], changeable output layers [69] and so on.

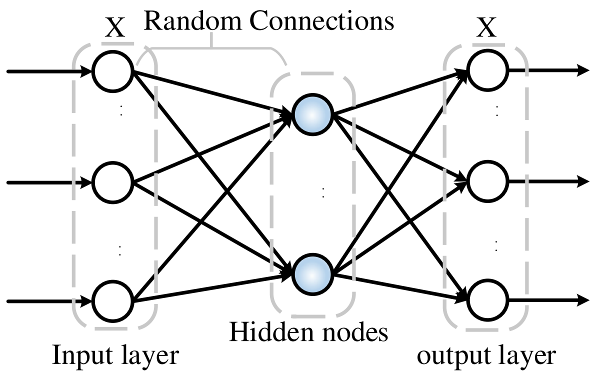


Figure 2.6: Structure of the ELM-AE.

Kasun *et al.* [35] proposed a famous variant of basic ELM called ELM auto-encoder (ELM-AE, shown in Fig. 2.6) for representation learning and dimension reduction, and applied it to image classification. In the ELM-AE, the output nodes are the same as the input nodes, and random weights and biases between the input layer and the hidden layer are orthogonal here. Theory of ELM-AE has shown that output weights could represent the raw input data efficiently for image classification problems. An ELM-based method for texture was also proposed later for image classification by taking the output weights of the ELM as feature vectors in [100].

In 2015, Huang *et al.* [34] introduced a local receptive field into ELM networks and proposed local receptive field based ELM (ELM-LRF) for classification problems. Unlike full connections in basic ELM, ELM-LRF adopts local connections between the input layer and the hidden layer, and learn the output weights by using ridge regression. Later, multi-layer ELM [99] and hierarchical ELM [43] were proposed which extend single layer ELM to multiple (hidden) layer networks.

Our last three work in this thesis all belong to the advanced ELM-based algorithms, and our second work and third work are directly inspired by ELM-LRF and Hierarchical ELM. More details will be presented in the following corresponding chapters.

Chapter 3

ELM-Based Pairwise Distance Vector for Smiling and Non-Smiling Images Classification

Chapter 3 begins with the introduction of applications of facial expression recognition and smile detection (smiling and non-smiling images classification), then we analyze the gaps of current smile detection algorithms, and propose our novel algorithm based on ELM and the snappy set of features extracted from a few of facial landmarks around the mouth. Furthermore, we have theoretically proved that the proposed feature extraction method has properties of invariance of translation, rotation, and scaling. At last, We have tested our ELM-based algorithms on the smile images classification database, and the results indicate that our method is better than the state-of-the-art methods with higher accuracy and lower dimension of features.

3.1 Introduction

Facial expressions recognition has attracted increasing attention in past years [101, 102] and smile (as one of the most common facial expressions) detection has become one of the hotspots recently. Accurate identification of smiles can verify a person's true feelings [103, 104]. Smile detection is to distinguish between smiling face images and non-smiling face images.

Many researchers have proposed various methods for smile images classification in the last decade. In general, most of them have a facial feature extraction module followed by classification. There are two groups of feature extraction methods. The first group is based on extracting features from the raw images, while the second group relies on extracting features from the facial landmarks (facial key points which can be adopted for face recognition, head pose estimation, face morphing, etc.). A brief introduction of these two groups of feature extraction methods is given, and the dimensions of the feature vectors are presented in the following two paragraphs.

Shinohara et al. [105] extracted Higher-order Local Auto-Correlation (HLAC) features at each pixel in the raw image. The HLAC features are integrated with a weight map to form a feature vector, and the dimension of this vector is proportional to the total number of pixels in the image. Gabor Energy Filters (GEF) [106] was applied, and Gabor vector was adopted as features in [10]. The dimension of Gabor feature vector is $O(10^5)$ in their work. Shan put forward an efficient approach for smile detection by comparing the pixel intensities and used the binary results of each comparison (which is represented numerically as 1 or 0) as features in [107]. The dimension of the features is $O(10^6)$ in his work. Later on, he improved this method by replacing the comparison operators with the intensity difference between arbitrary two pixels in [108] because the intensity difference provides more information than comparison operators. A mixed feature vector was used in [109] by combining three popular feature descriptors: Local Binary Pattern (LBP) [110, 111], Local Phase Quantization (LPQ) [112] and Histogram of Oriented Gradients (HOG) [113]. The dimension of the mixed feature is reduced to

500 by Principal Component Analysis (PCA) [114]. The features mentioned above are all extracted from raw aligned face images, since [10] has shown that image alignment is helpful to increase the accuracy of object detection.

Huang et al. [115] adopted the $x-$direction distance between the left mouth corner and right mouth corner as a measure to distinguish whether there exists a smiling face. The criterion is the $x-$direction distance of the smiling face is larger than a standard length plus a threshold T_{smile}. However, the threshold is hard to determine, and this method is not robust since people can have various head pose in real scenarios and the distance between the left and right mouth corner can vary considerably. The x and y coordinates of eight particular mouth corners were used as input to a three-layer feedforward network in [2], but we know that absolute locations of mouth are quite unstable in different facial images [116, 117, 118].

In this chapter, we propose a novel smile images classification system by using a short and snappy feature descriptor for smile detection named Pair-wise Distance Vector, which is extracted only from the facial landmarks around the mouth. Commonly, we annotate 68 facial landmarks within seven facial regions, including eyebrows (10 landmarks), eyes (12 landmarks), nose (9 landmarks), mouth (20 landmarks) and face contour (17 landmarks). The motivation of only using facial landmarks around the mouth in this chapter is that a smile is performed by moving cheeks and lips [119]. To verify the effectiveness of facial landmarks around the mouth in detecting a smile, we calculate the importance of all the facial components (facial subregions, including eyebrows, eyes, nose, mouth, face contour.) by using MUG Facial Expression Database (some examples of this database are shown in Fig. 3.1), which contains the manual-annotated facial landmarks for 401 images of 26 subjects [120]. The details are as follows:

- Each smile-face image and its corresponding neutral images are treated as a pair of images, and their facial landmarks are aligned according to three fixed position (centers of the left eye, right eye and the apex of the nose)
- For the facial landmarks within each facial component of each pair images, the

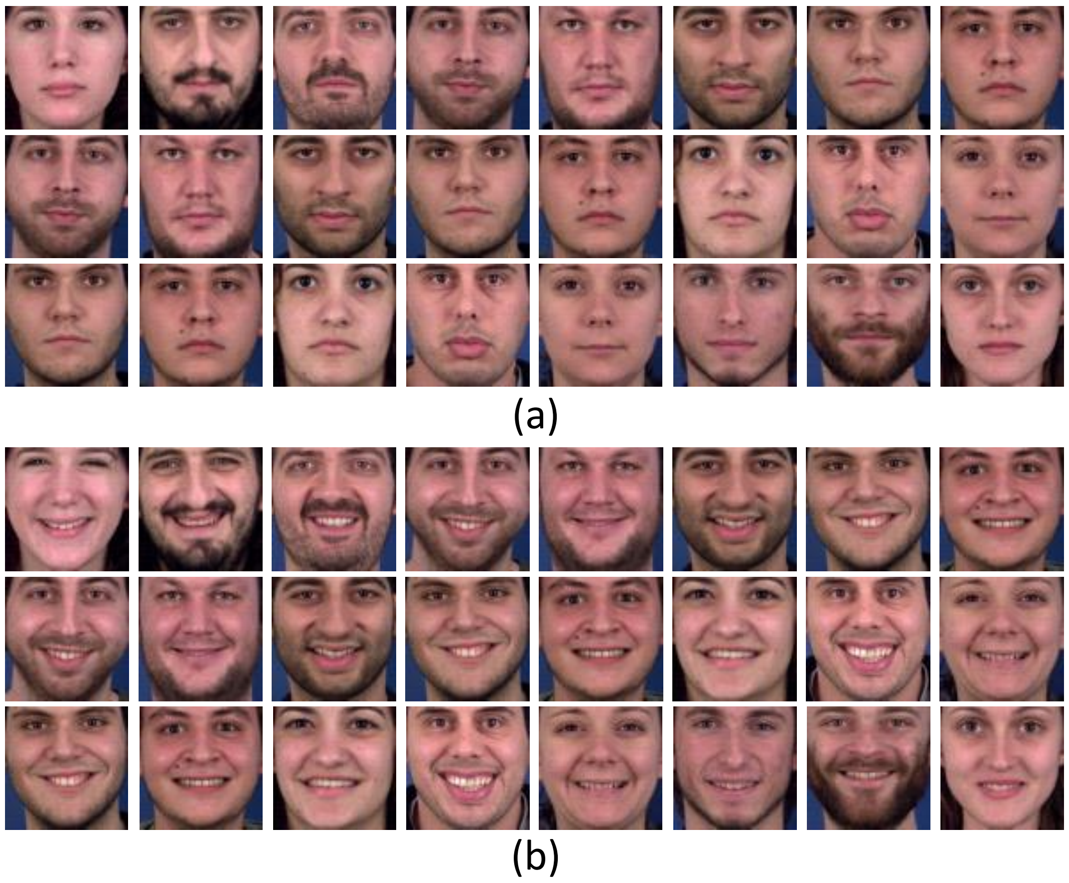


Figure 3.1: An example of MUG Facial Expression Database. (a) shows neutral images and (b) shows the corresponding smile-face images.

movements are calculated, and the variance of these movements is calculated.

- All these variances are added together for each corresponding facial component, and their proportions are shown in Fig. 3.2.

Apparently, the movements of facial landmarks around the mouth achieve the maximum variance, which implies that mouth deformation can indicate a smile.

After extracting Pair-wise Distance feature vectors from the training samples, we adopt Extreme Learning Machine (ELM) [121] as a classifier to train a model due to ELM's outstanding performance regarding accuracy and speed of training and testing.

Our work has four main contributions:

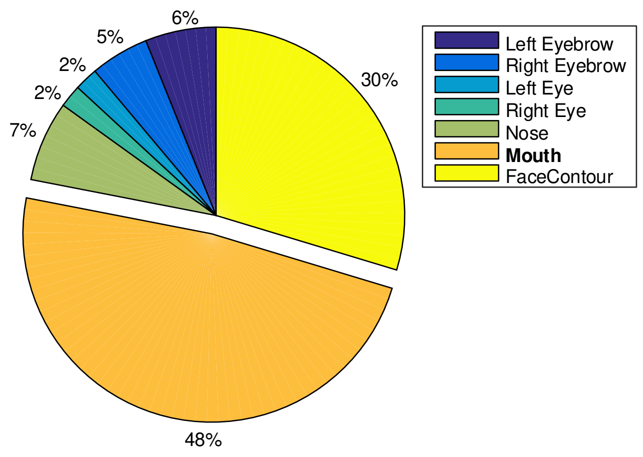


Figure 3.2: Impact proportions of seven facial components for smiling face compared with neutral.

(a) We verify the effectiveness of mouth deformation as an indicator of a smile.

(b) We introduce a new framework which takes detection of face and facial landmarks as a preprocessing step, extracts a feature vector from facial landmarks, and classify these feature vectors with a classifier (like ELM).

(c) We propose a novel and snappy feature extraction method that extracts a feature vector only from the facial landmarks around the mouth. This feature vector can describe the shape of the mouth with little distortion no matter how the shape is changed. We have proved that the extracted feature vector has the property of invariance of translation, rotation, and scaling.

(d) We provide an updated GENKI-4K benchmark by relabeling all the images and removing the duplicate images. We also evaluate our proposed algorithm again according to the new annotations.

The remainder of this chapter is organized as follows. Section 3.2 introduces the related work of our method, including CFAN for facial landmarks detection and ELM for classification. In Section 3.3, we describe the proposed smile detection method using the novel features (Pair-wise Distance Vector). The experiments and results are discussed in detail in Section 3.4, and the conclusion is given to summarize this chapter finally.

3.2 Related Works

As shown in Fig. 3.3, we need a preprocessing step to locate the essential facial landmarks and a classifier to determine between smiling and non-smiling faces based on the extracted features. In our work, we adopt Coarse-to-Fine Auto-encoder Networks (CFAN) and ELM for these two modules respectively. These two related work and the novel feature vector (introduced in Section 3.3) are integrated and form the proposed novel smile detection algorithm. More details of these two techniques are presented below.

3.2.1 Facial Landmarks Detection

The foundation of our study is the accurate detection of facial landmarks, and research on facial landmarks detection (or estimation) can be divided into two categories [98]: holistic feature extraction based and local feature extraction based. The representative holistic feature extraction method is Active Appearance Models (AAM) [122], and the typical local feature extraction technique is Constrained Local Model (CLM) [123]. In this chapter, we obtain facial landmarks automatically by adopting CFAN [98], has achieved the highest accuracy at present to the best of our knowledge. Besides, CFAN (with Matlab codes) has been evaluated on a common desktop machine with Intel Core i7-3770 Quad-Core Processor 3.4GHz CPU and 32GB DDR3 SDRAM. The program runs rather efficiently by processing more than 40 fps per second.

CFAN consists of a global Stacked Auto-encoder Network (SAN) and several local SANs. The mapping F of global SAN from input image I to shape S is taken as a stacked auto-encoder.Local SANs share the similar objective function to the global SAN by replacing the input image with the output of global SAN. More details of CFAN are introduced in Section 3.3.

Global SAN can quickly reach the approximate locations of the facial landmarks and

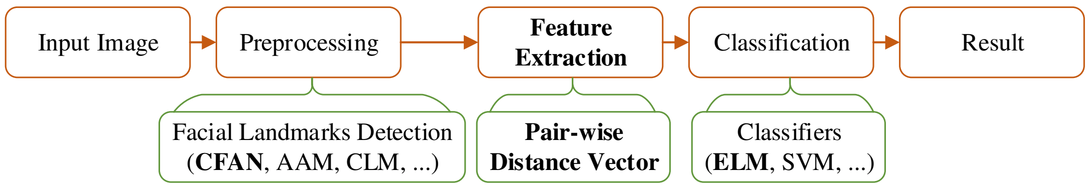

Input Image
Preprocessing
Feature Extraction
Classification
Result
Facial Landmarks Detection (CFAN, AAM, CLM, ...)
Pair-wise Distance Vector
Classifiers (ELM, SVM, ...)

Figure 3.3: Flowchart for smile detection.

local SANs refine these positions step by step.

3.3 ELM-Based Smile Image Classification Using Distance Vector

Our framework for smile detection is shown in Fig. 3.3. Firstly, a step of preprocessing is performed on the input image to locate all the facial landmarks (68 facial landmarks in total in our work) by CFAN. Then, the proposed novel feature extraction method is performed on these facial landmarks, and Pair-wise Distance Vector is extracted Finally, ELM is carried out to classify smile based on these feature vectors. Particularly, the necessity of using ELM for the specific task is that the effectiveness of ELM on classification problems has been proved from theory and practice. Therefore, our technique approach consists of three main parts: picking up the facial landmarks around the mouth, extracting pair-wise distance vector, and classifying the features with ELM.

3.3.1 Landmarks Extraction for Smile Detection

The facial landmarks around the mouth are extracted using a modified CFAN. First of all, a face detector is performed to find the boundary box of the face roughly in the input image, and then facial landmarks are located within the facial region. Kinds of face detectors have been proposed in the past decades [124, 125, 118, 126, 127], and many detectors can be employed here. Exploring the best detector is out of the scope of this chapter. Instead, we focus on tuning the exist CFAN (which could locate 68 facial landmarks) to extract the 20 landmarks around the mouth which are required for smile detection.

CFAN can be divided into two parts: the global SAN and local SANs. The global SAN estimates the rough locations of facial landmarks by using global raw features at a low-resolution image. The objective of facial landmarks detection is to find a mapping function $F(x)$ which project the input face image x to the coordinates of the facial landmarks (donated as $L(x)$). By stacking k auto-encoders (the mapping function of i^{th} layer is f_i), we have the objective function as:

$$F^* = arg \min_F ||L_g(x) - f_k(f_{k-1}(...f_1(x)))||_2^2, \tag{3.1}$$

where $F = \{f_i | i \in [1,k]\}$ and L_g is the actual locations of the facial landmarks. Global SAN is optimized by adopting a regularization term and an unsupervised pre-train process to overcome over-fitting and local minimum respectively (details can be found in [98]). The outputs of global SAN are the approximate locations of facial landmarks.

The local SANs are designed similarly to tune these locations further. Different from the raw global features adopted in the stage of global SAN, local SANs use the local shape-indexed features (i.e., Scale-invariant feature transform (SIFT)[18]) as input.

CFAN in [98] outputs 68 facial landmarks in total, shown in Fig. 3.4 (c). However, only facial landmarks around the mouth are needed for smile detection, and 48 facial landmarks are superfluous. So in our work, we only adopt the 20 facial landmarks around the mouth and three anchors (centers of two eyes and the tip of the nose). These three key facial landmarks are chosen because they are the most commonly used reference facial landmarks for face alignment [128, 129, 130, 131]. They will be ignored in the final layer of the auto-encoder network on the stage of local SANs, which reduces about two-thirds of the computational complexity.

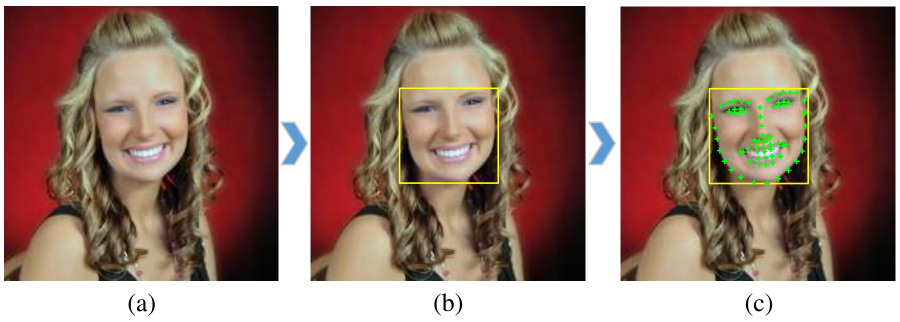


Figure 3.4: The result of preprocessing. (a) shows the original image comes from Helen database [1]. (b) is the result of face detection (marked with yellow rectangle). (c) is the result of facial landmarks based on coarse-to-fine auto-encoder network (marked with a green cross).

3.3.2 Pair-Wise Distance Vector

We obtain 68 facial landmarks with the preprocessing step. Twenty facial landmarks around the mouth are used according to the reason we have analyzed in Section 3.1. These 20 facial landmarks and their sequence numbers are shown in Fig. 3.5.

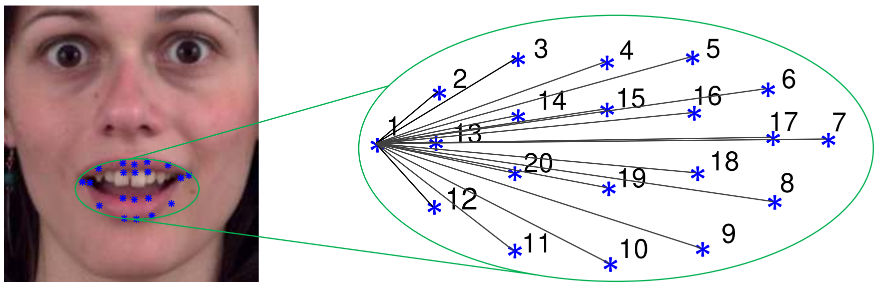


Figure 3.5: Twenty facial landmarks around the mouth.

Assume there are N facial landmarks in a geometric figure, and the coordinates of an arbitrary point P_i $(i \in [1,n])$ are (x_i, y_i). The Euclidean distance matrix of these N facial landmarks is:

$$
\begin{aligned}
M_{ED} = & \begin{bmatrix} |\mathbf{v}_{11}| & |\mathbf{v}_{12}| & \cdots & |\mathbf{v}_{1n}| \\ |\mathbf{v}_{21}| & |\mathbf{v}_{22}| & \cdots & |\mathbf{v}_{2n}| \\ \vdots & \vdots & \ddots & \vdots \\ |\mathbf{v}_{n1}| & |\mathbf{v}_{n2}| & \cdots & |\mathbf{v}_{nn}| \end{bmatrix} \\
= & \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{bmatrix} \\
= & \begin{bmatrix} 0 & d_{12} & \cdots & d_{1n} \\ d_{12} & 0 & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{1n} & d_{2n} & \cdots & 0 \end{bmatrix},
\end{aligned}
\tag{3.2}
$$

where $\mathbf{v}_{ij}$ $(i, j \in [1, n])$ is the vector from point P_i to point P_j. The norm of $\mathbf{v}_{ij}$ is d_{ij} which stands for the Euclidean distance between P_i and P_j. We know the coordinates of P_i and P_j are (x_i, y_i) and (x_j, y_j) respectively, so $M_{ij} = |\mathbf{v}_{ij}| = d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} = d_{ji} = M_{ji}$, and M_{ED} is a symmetric matrix. Extract the upper triangular part of this matrix (diagonal elements are ignored since they are all zeros) and convert these elements into a vector D:

$$
D = [d_{12}, d_{23}, \cdots, d_{1n}, d_{23}, \cdots, d_{2n}, \cdots, d_{(n-1)n}]. \tag{3.3}
$$

After normalizing D with its ℓ_1 norm, we obtain a feature vector F:

$$
F = Normalization(D) = \frac{D}{\|D\|_1}, \tag{3.4}
$$

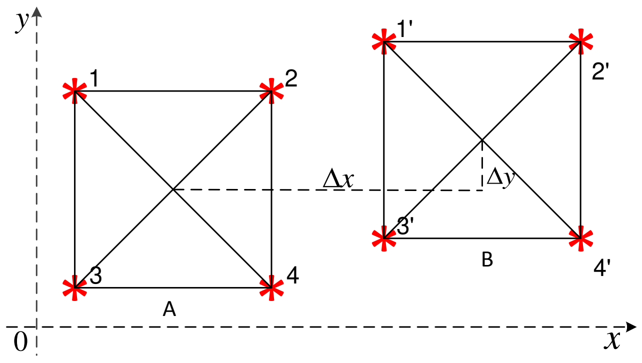


Figure 3.6: Sketch for translation invariance.

and

$$\begin{aligned}\|D\|_1 = &|d_{12}| + |d_{13}| + \cdots + |d_{1n}| + |d_{23}| + \cdots + \\ &|d_{2n}| + \cdots + \left|d_{(n-1)n}\right|.\end{aligned} \tag{3.5}$$

This feature vector F is named as Pair-wise Distance Vector, and the length of F is $n(n-1)/2$. F is a shape descriptor which can adequately reflect an object's shape since we can reconstruct the same normalized figure as the original figure by F. Besides, this feature vector has properties of invariance of translation, rotation, and scaling, and the proofs are as below.

Translation Invariance

There are two figures in Fig. 3.6, A and B. A is a geometric figure, and B is the translated geometric figure of A, which has an offset Δx in the horizontal direction and Δy in the vertical direction. The coordinates of point P_i in A are (x_i, y_i), then the coordinates of point $P_{i'}$ in B are (x'_i, y'_i), which equal to $(x_i + \Delta x, y_i + \Delta y)$.

Proof: Assume the distance between P_i and P_j in A is d_{ij}, and the distance between $P_{i'}$

and $P_{j'}$ in B is d'_{ij}. Then we have:

$$\begin{aligned}(x'_i - x'_j)^2 &= ((x_i + \Delta x) - (x_j + \Delta x))^2 \\ &= ((x_i - x_j) + (\Delta x - \Delta x))^2 \\ &= (x_i - x_j)^2.\end{aligned} \tag{3.6}$$

Similar to equation (3.6), we have:

$$\begin{aligned}(y'_i - y'_j)^2 &= (y_i + \Delta y) - (y_j + \Delta y))^2 \\ &= (y_i - y_j)^2.\end{aligned} \tag{3.7}$$

Then,

$$\begin{aligned}d'_{ij} &= \sqrt{(x'_i - x'_j)^2 + (y'_i - y'_j)^2} \\ &= \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} = d_{ij}.\end{aligned} \tag{3.8}$$

So $D' = D$, and finally, we know the feature vector $F' = F$, which means F is a translation invariant shape descriptor.

Rotation Invariance

In Fig. 3.7, A is an original geometric figure, and B is the rotated geometric figure with rotation angle θ. So coordinates of point $P_{i'}$ in B are (x'_i, y'_i), which equals to $(x_i cos\theta - y_i sin\theta, x_i sin\theta + y_i cos\theta)$.

Proof: Similar to the proof of translation invariance, we have:

$$\begin{aligned}(x'_i - x'_j)^2 =&((x_i cos\theta - y_i sin\theta) - (x_j cos\theta - y_j sin\theta))^2 \\ =&(x_i - x_j)^2 cos^2\theta + (y_i - y_j)^2 sin^2\theta \qquad , \\ &- 2(x_i - x_j)(y_i - y_j) sin\theta cos\theta\end{aligned} \tag{3.9}$$

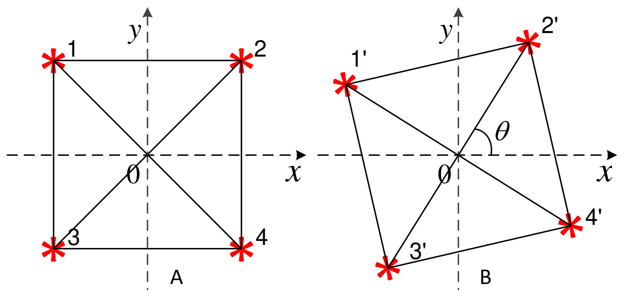


Figure 3.7: Sketch for rotation invariance.

and

$$\begin{aligned}(y'_i - y'_j)^2 =&((x_i sin\theta + y_i cos\theta) - (x_j sin\theta + y_j cos\theta))^2 \\ =&(x_i - x_j)^2 sin^2\theta + (y_i - y_j)^2 cos^2\theta \\ &+ 2(x_i - x_j)(y_i - y_j) sin\theta cos\theta.\end{aligned} \tag{3.10}$$

Based on equation (3.9) and equation (3.10), we have:

$$\begin{aligned}(x'_i - x'_j)^2 + (y'_i - y'_j)^2 =&(x_i - x_j)^2(sin^2\theta + cos^2\theta) \\ &+ (y_i - y_j)^2(sin^2\theta + cos^2\theta) \\ =&(x_i - x_j)^2 + (y_i - y_j)^2.\end{aligned} \tag{3.11}$$

Then the distance between $P_{i'}$ and $P_{j'}$ in B is:

$$\begin{aligned}d'_{ij} &= \sqrt{(x'_i - x'_j)^2 + (y'_i - y'_j)^2} \\ &= \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} = d_{ij}.\end{aligned} \tag{3.12}$$

So the feature vector $F' = F$, and F is a rotation invariant shape descriptor.

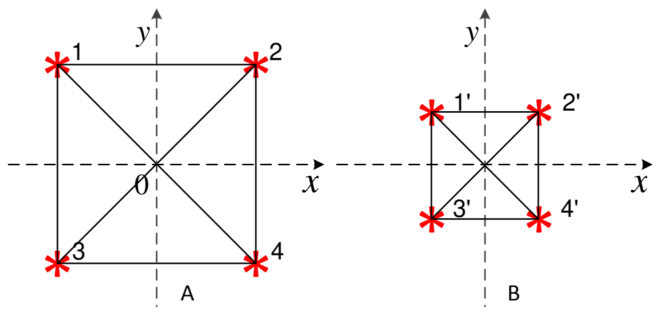


Figure 3.8: Sketch for scaling invariance.

Scaling Invariance

In Fig. 3.8, A is an original geometric figure and B is the scaled geometric figure of A with a scaling factor α $(\alpha > 0)$. The coordinates of point $P_{i'}$ in B are (x_i', y_i'), which equal to $(\alpha x_i, \alpha y_i)$.

Proof: Similar to the above two proofs, we have:

$$\begin{aligned} d'_{ij} &= \sqrt{(x'_i - x'_j)^2 + (y'_i - y'_j)^2} \\ &= \sqrt{(\alpha x_i - \alpha x_j)^2 + (\alpha y_i - \alpha y_j)^2} \\ &= \alpha\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} = \alpha d_{ij}. \end{aligned} \tag{3.13}$$

Then, $D' = \alpha D$. So the feature vector $F' = \frac{D'}{\|D'\|_1} = \frac{\alpha D}{\|\alpha D\|_1} = \frac{\alpha D}{\alpha \|D\|_1} = \frac{D}{\|D\|_1} = F$, and F is a scaling invariant shape descriptor.

3.3.3 ELM for Smile Detection

The last step of our technical approach for smile detection is using ELM to classify the feature vectors we have got into two categories: smile(or positive, denoted as “1”) and non-smile(or negative, denoted as “-1”). In the section of related work, we have briefly introduced the theories of the early ELM which were developed in 2006 [38]. In 2012, Huang *et al.* [30] introduced a user-specified parameter C (regularization factor) into the ELM theory under the Karush-Kuhn-Tucker (KKT) conditions [132].

The dimension of the features we have achieved is 190, which is much smaller than the training samples (there are 4000 samples in the dataset we use, and 3000 of them are used to train the model. See Section 3.4 for more details). Refer to the analysis of constrained-optimization-based ELM in [30], the relationship between hidden layer output matrix $\mathbf{H}$, output weight matrix β between the hidden layer and output layer, and output vector $\mathbf{Y}$ of the output layer is established as:

$$\mathbf{H}\beta - \mathbf{Y} + \frac{\mathbf{I}}{C}(\mathbf{H}^T)\beta = 0. \tag{3.14}$$

Here $\mathbf{H} = G(\mathbf{WX}+\mathbf{B})$, where $\mathbf{W}$ and $\mathbf{B}$ are weight and bias matrices between input layer and the hidden layer, $\mathbf{X}$ represents the input samples, and $G(\cdot)$ is an activation function.

Solve the equation, we have

$$\beta = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{Y}. \tag{3.15}$$

Smile detection is a binary classification case, and one single-output node is enough. So the decision function of the classifier is

$$f(\mathbf{x}) = \operatorname{sign}\left(h(x)\beta\right). \tag{3.16}$$

So, $f(\mathbf{x}) = 1$ indicates that the predicted label of sample $\mathbf{x}$ is positive. Otherwise, the predicted label is negative.

The training procedure of the proposed smile detection system is shown in Algorithm 1, and the test procedure is shown in Algorithm 2.

```
Algorithm 1: ELM Training
  Input: Training image I and their corresponding labels Y. The number of
         hidden nodes $L$ and the normalization factor $C$.
  Output: An ELM model which contains the weight matrices W and $\beta$, and
          bias matrix B.
1 Use CFAN to locate 20 facial landmarks around mouth: $\mathbf{P} \leftarrow CFAN(\mathbf{I})$ ;
2 Compute Euclidean distance matrix according to Equ. 3.2: $\mathbf{M} \leftarrow \mathbf{P}$ ;
3 Compute pair-wise distance feature vectors according to Equ. 3.3: $\mathbf{D} \leftarrow \mathbf{M}$ ;
4 Normalize D with its $\ell_1$-norm according to Equ. 3.4: $\mathbf{X} \leftarrow \mathbf{D}$ ;
5 Generate W and B randomly, and their dimensions are both $L \times N_1$, where $N_1$ is
    the length of each input sample. ;
6 Calculate the hidden layer output matrix according to Equ. 3.14:
    $\mathbf{H} \leftarrow (\mathbf{W}, \mathbf{X}, \mathbf{B})$;
7 Calculate the output weight according to Equ. 3.15: $\beta \leftarrow (\mathbf{H}, \mathbf{Y}, C)$;
8 return W, $\beta$, and B;
```

```
Algorithm 2: ELM Testing
  Input: Test image $I$, the weight matrices W and $\beta$, and bias matrix B.
  Output: A smile indicator $y$, which is -1 (non-smile) or 1(smile).
1 Use CFAN to locate 20 facial landmarks around mouth: $P \leftarrow CFAN(I)$ ;
2 Compute Euclidean distance matrix according to Equ. 3.2: $M \leftarrow P$ ;
3 Compute pair-wise distance feature vectors according to Equ. 3.3: $D \leftarrow M$ ;
4 Normalize $D$ with its $\ell_1$-norm according to Equ. 3.4: $\mathbf{x} \leftarrow D$ ;
5 Calculate the hidden layer output matrix according to Equ. 3.14:
    $H \leftarrow (\mathbf{W}, \mathbf{x}, \mathbf{B})$;
6 Calculate the predictor according to Equ. 3.16: $y \leftarrow (H, \mathbf{x})$ ;
7 return the indicator $y$;
```

3.4 Experimental Results

In our experiment, we test the proposed method on GENKI-4K [81, 133] database. First, a brief introduction of GENKI-4K is given in subsection 3.4.1, and the primary challenges are analyzed. Subsection 3.4.2 shows the visualization of our proposed feature vector. We evaluate related classifiers by using the same database in subsection 3.4.3. Then we compare our feature extraction method with two related methods based on facial landmarks and some state-of-the-art smile detection algorithms in Subsection 3.4.4 and Subsection 3.4.5 respectively. In Subsection 3.4.6, we evaluate our proposed approach in terms of the accuracy on various head poses. Finally, we update the GENKI-

4K database by correcting some false annotations and removing the duplicates in Subsection 3.4.7.

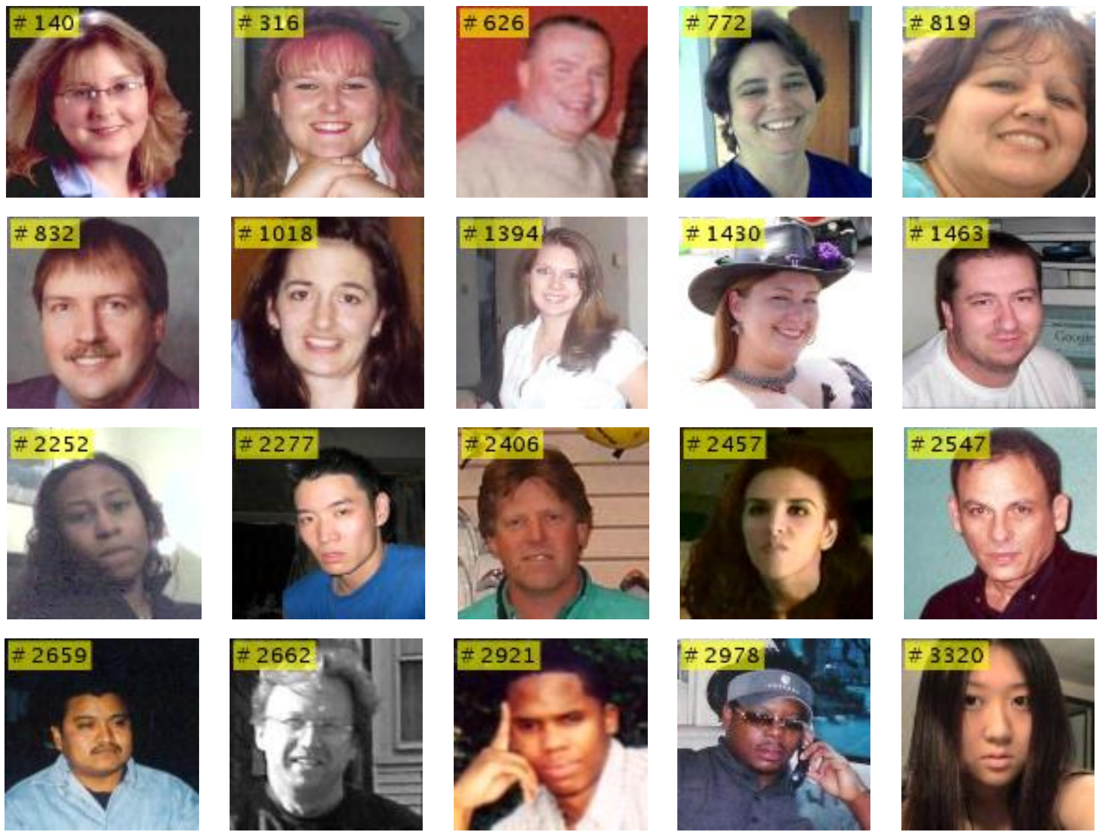


Figure 3.9: Images randomly selected from GENKI-4K database.

3.4.1 GENKI-4K Database

In our experiment, GENKI-4K database is adopted. This database is the most famous database for smile detection and has been widely used in many studies [134, 10, 135, 108, 136, 137, 109, 138]. It contains 4000 real-world scene images, and each image has one valid facial region (2162 smiling faces and 1838 non-smiling faces). These facial images span a broad range of illumination conditions, geographical locations, personal identity, and ethnicity. Fig. 3.9 shows some images which are randomly selected from GENKI-4K database. Images in the first two rows show smiling faces while the rest images show non-smiling faces. This database has fully reflected the two most challenging issues for detecting smiling faces in real applications. One is the head pose, and the

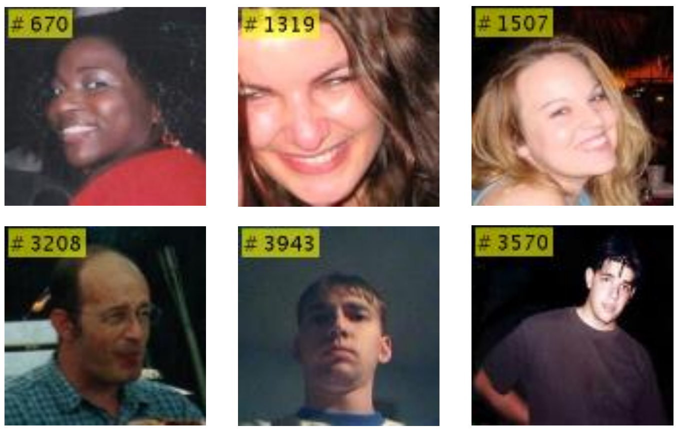


Figure 3.10: Examples for various head poses selected from GENKI-4K database. Images from top left to bottom right are smiling and non-smiling faces with an angle of yaw, pitch, and roll respectively.

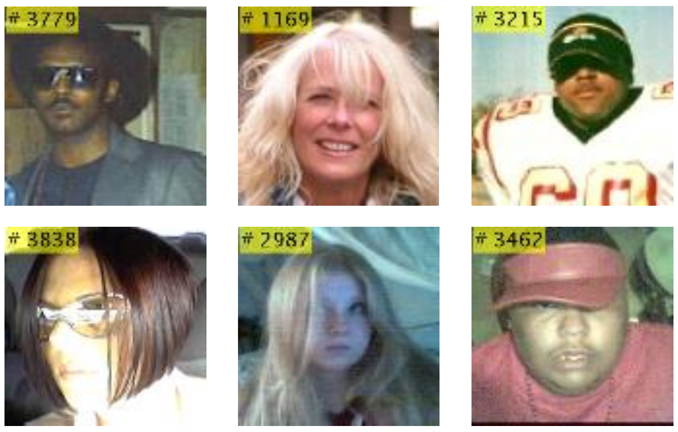


Figure 3.11: Examples of three kinds of facial occlusions from GENKI-4K database. Occlusions covering facial images from the left column to right column are glasses, hairs, and hats respectively.

other is the occlusion. Fig. 3.10 shows some examples of the three degrees of freedom of a human head by the egocentric rotation angles, which are yaw, pitch, and roll [139].

Some facial images with occlusions (glasses, hairs, and hats) are shown in Fig. 3.11.

In the experiment, our method is based on facial landmarks around the mouth, and it does not have these problems. Our algorithm works quite well as long as no occlusion exists in the mouth region, which is a more common condition than no occlusion in the face region (none of the 4000 images in GENKI-4K database has an occlusion covering the mouth). Moreover, we adopt Coarse-to-Fine Auto-Encoder Networks to detect facial

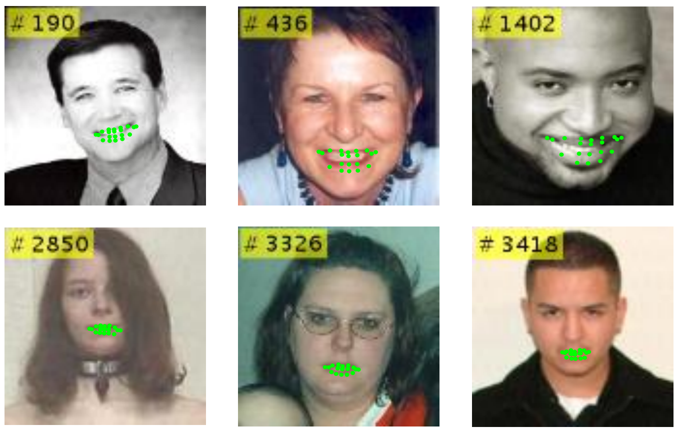


Figure 3.12: Some implement results of coarse-to-fine auto-encoder network on GENKI-4K database.

landmarks around the mouth, which is quite robust to variation of head pose. Some detection results are shown in Fig. 3.12.

3.4.2 Visualization of Pair-Wise Distance Vector

To visualize the high-dimension Pair-wise Distance Vector directly, we perform the PCA on it in the experiment. Here, we extracted 20 facial landmarks around the mouth for all M $(= 4000)$ images in GENKI-4K. The dimension L of Pair-wise Distance Vector is 190 $(= (20-1) \times 20 \div 2)$. We follow the following five steps to visualize the proposed vector:

(1) Combine all features together as a matrix $X = [F_1; F_2; \cdots; F_M]$.

(2) Compute all principal components scores S of X and eigenvalues $E = [e_1, e_2, \cdots, e_L]$ of the covariance matrix of X according to the definition of PCA.

(3) Calculate the sum of eigenvalues $\sum_{i=1}^{i=L} e_i$ and find the first K eigenvalues which satisfy the condition that $\sum_{i=1}^{i=K} e_i \geq T \times \sum_{i=1}^{i=L} e_i$ (here we set $T = 0.85$, so $K = 4$).

(4) Extract first K columns of S and each column is then treated as a variable.

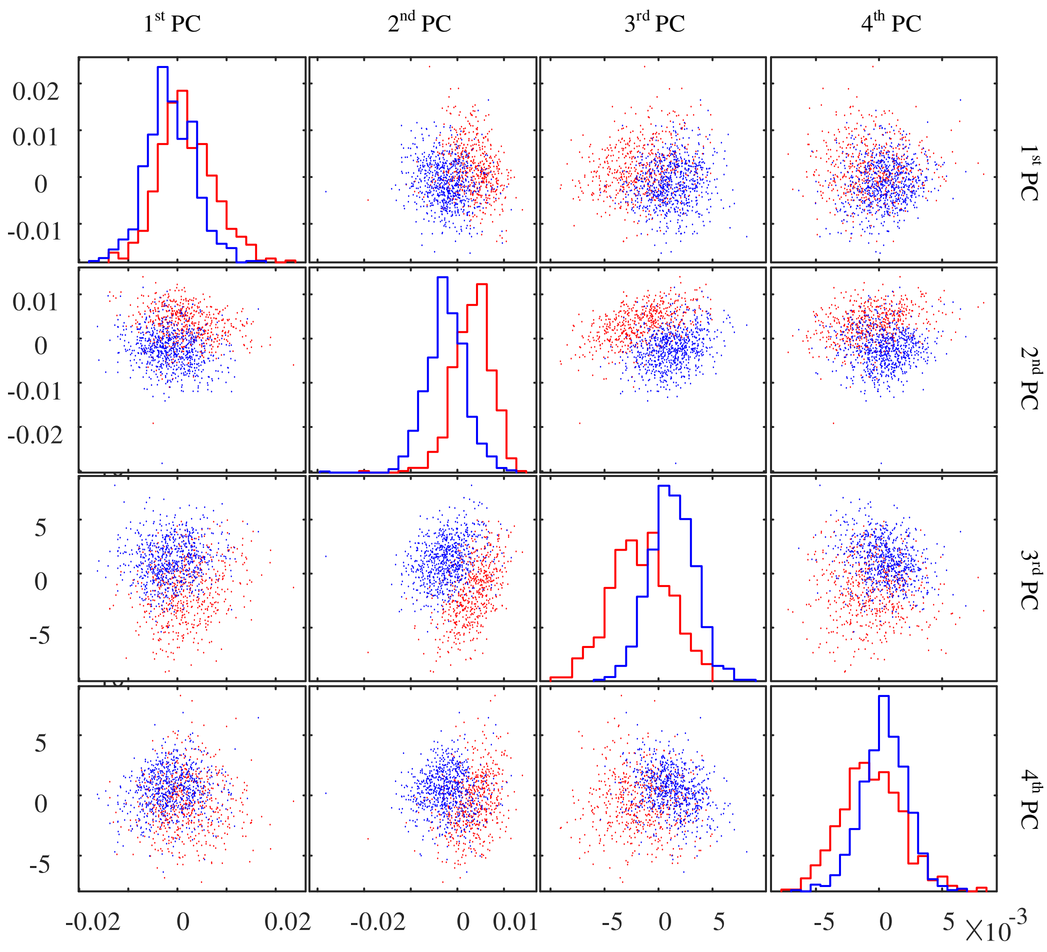


Figure 3.13: Visualization of pair-wise distance vectors. Red dots indicate positive samples (smiling faces) while blue dots denote negative samples (non-smiling faces). 1^{st}, 2^{nd}, 3^{rd} and 4^{th} PC represent first, second, third and fourth principal component respectively.

(5) Display each pair of variables as a scatterplot, along with a univariate histogram for each variable, shown in Fig. 3.13.

Two conclusions can be drawn from the scatterplots above. First, the proposed feature extraction method is effective because the scatterplots between the second and third principal components are somewhat separable. Second, features cannot be precisely separated by linear classifiers.

3.4.3 Comparison of Different Classifiers

To evaluate the performance of commonly used classifiers on smile detection, we compare two different baseline classifiers with ELM. Raw pixel values and Pair-wise Distance Vector are extracted from facial images as the training and testing inputs of the classifiers.

Raw pixel values are automatically cropped from the gray level of original images by a face detector followed by scaling to reach a normalized size 48×48, so the dimension of the raw-pixel-value feature vector is 2304.

Brief introductions and parameter settings of the three different classifiers are listed below:

AdaBoost. Adaptive boosting algorithm (AdaBoost) is an iterative algorithm, which trains different weak classifiers with the various parameters on the same training samples and builds a strong classifier by computing a weighted sum of all the weak classifiers [140]. Gentle AdaBoost (with decision trees as the weak classifiers) is adopted in our experiment (same with AdaBoost used in [10] and [138]) because this boosting method is more efficient in practical applications [141].

SVM. Support Vector Machine (SVM) is essentially a binary classification model, which obtains several support vectors by maximizing the geometric margin between negative and positive samples [142, 143]. In our experiment, the type of kernel function is radial basis function, and the gamma in kernel function ranges from 10^{-5} to 10^{5} in the process of cross-validation.

ELM. ELM (Extreme Learning Machine) has been introduced in Section 3.2. The number of hidden neurons is set to two times the dimensions of the input features.

Cross-validation is implemented for each classifier to make a valid estimation of performance, and we adopt a K-fold (K is set to 4 in our experiment) cross-validation [144]. All the 4000 samples from GENKI-4K database are divided into K groups, and only

Table 3.1: Comparison of different classifiers.

Accuracy (%)	Raw pixel values	Proposed features
AdaBoost	79.22 ± 1.12	91.54 ± 0.67
SVM	77.47 ± 1.18	92.70 ± 0.85
ELM	79.30 ± 1.64	93.42 ± 1.46

one group is to be tested while others are used to train the model. So there are K results of the classification accuracy for each classifier. The mean of all these K results is calculated as the accuracy of each classifier, along with the standard deviation, shown in Table 3.1. The results suggest that ELM performs better than AdaBoost and SVM both for raw-pixel-value features and the features proposed in this chapter.

3.4.4 Comparison with Related Methods Based on Facial Landmarks

To show the effectiveness of our proposed features extracted from the facial landmarks, we compare two related methods which are also based on facial landmarks. The proposed feature extraction method relies on the accurate detection of facial landmarks, which is also the key preprocessing step of methods developed by Huang et al. [115] and Kowalik et al. [2]. The smile detection approaches proposed in [115] and [2] have been explained in the fourth paragraph of Section 3.1 and detail steps of implementation are introduced as follows:

In [2], x and y coordinates of eight facial landmarks around the mouth (shown in Fig. 3.14, including the left and right mouth corners (P'_1 and P'_5), three facial landmarks on the upper lip ($P'_{2\sim4}$) and three facial landmarks on the lower lip ($P'_{6\sim8}$) are used as features. However, there are 20 facial landmarks ($P_{1\sim20}$, shown in Fig. 3.2) around a mouth in the implementation of CFAN landmark detector. So, a corresponding mapping between P'_i ($i \in [1,8]$) and P_j ($j \in [1,20]$) is built as:

- $P'_1 \leftarrow P_1, P'_5 \leftarrow P_7$;
- $P'_2 \leftarrow$ the center point of P_2, P_3, and P_{14};

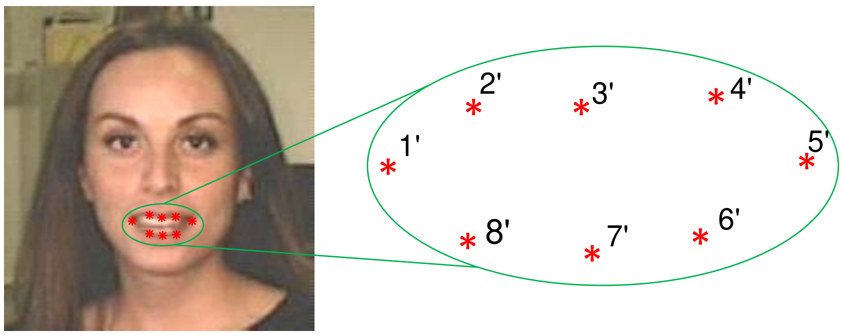


Figure 3.14: Sketch for eight facial landmarks around mouth used in [2].

- $P'_3 \leftarrow$ the center point of P_4 and P_{15};
- $P'_4 \leftarrow$ the center point of P_5, P_6, and P_{16};
- $P'_6 \leftarrow$ the center point of P_8, P_9, and P_{18};
- $P'_7 \leftarrow$ the center point of P_{10} and P_{19};
- $P'_8 \leftarrow$ the center point of P_{11}, P_{12}, and P_{20}.

We can easily obtain the eight facial landmarks from the existing 20 facial landmarks for each image. Finally, ELM is applied as the classifier to this coordinate-based features.

The method proposed in [115] is only based on left and right mouth corners, and a smile is detected if $x-$direction distance between these two corners is larger than a fixed value D_T. We define the accuracy of this threshold-based method on GENKI-4K as:

$$Acc_{tb} = \max_{D_T}\{\frac{1}{M}\sum_{i=1}^{n}[(D_i > D_T) == L_i]\}, \tag{3.17}$$

where $i \in [1, M]$ indicates the i^{th} images in the database. D_i denotes the distance between left and right mouth corners, and $L_i \in [-1, 1]$ denotes the logical value of negative ($L = -1$) or positive ($L = 1$) label.

The comparison between the above two methods and the method proposed in this chapter is shown in Table 3.2:

Table 3.2: Comparison of different methods based on facial landmarks.

Methods based on facial landmarks	Accuracy (%)
Coordinate-based method + ELM	84.57 ± 0.29
Threshold-based method	57.79[1]
Pair-wise Distance Vector + ELM	93.42 ± 1.46

[1] This accuracy is a constant according to Equ. 3.17.

The reason why threshold-based method achieves the lowest accuracy among these three methods based on facial landmarks is that the left and right mouth corners are not stable, and the distance between them alone is not sufficient enough to indicate the movement of pixels around the mouth when people show a smile. Our proposed feature extraction method achieves higher accuracy than the Coordinate-based method when the same classifier (ELM) is adopted.

3.4.5 Comparison with the State-of-the-Art Smile Detection Methods

To evaluate the performance of our proposed method, we compare it with some state-of-the-art smile detection methods in this section. Many algorithms for smile detection have been proposed in recent years since GENKI-4K database was created by MPLab and introduced in [10]. Among the state-of-the-art methods, some use individual feature while others combine these features to increase the detection accuracy further. Our proposed feature vector is one kind of individual features, so only methods based on individual features are implemented here to estimate the performance of Pair-wise Distance Vector in our experiment. There are four approaches that have achieved the state-of-the-art methods in recent years, which adopt Gabor, LBP, Pixel Comparisons, and Self-Similarity of Gradients (GSS) as features respectively. The details of parameter settings are as follows.

Gabor and LBP features. Gabor and LBP are adopted as features and tested on GENKI-4K in [10], and detailed parameter settings of them are firstly introduced by

Table 3.3: Comparison of different algorithms of smile detection.

Algorithms			Accuracy (%)
Feature	Dimension	Classifier	
Gabor[1]	23040	SVM	89.68 ± 0.62
LBP[1]	944	SVM	87.10 ± 0.76
Pixel Comparisons[2]	500	AdaBoost	89.70 ± 0.45
GSS[3]	576	Ada+SVM	91.11 ± 0.47
Pair-wise Distance Vector	190	ELM	93.42 ± 1.46

[1] This method is proposed in [10], and the accuracy is from [108].
[2] This method is proposed in [108], and the accuracy is also from [108].
[3] This method is proposed in [138], and the accuracy is also from [138].

Shan [108]. Each face image is normalized to a 48×48 patch, which contains 2304 pixels. Bank of Gabor filters are applied to each face patch at eight orientations and five spatial frequencies, and the length of outputs are reduced to 10 by down-sampling. LBP features are computed by adopting a 59-label LBP $(8, 2, u2)$ to all 16 subblocks divided from each face path. The dimensions of Gabor feature and LBP feature are 23040 and 944 respectively.

Pixel-Comparison feature. A method using Pixel-Comparison feature is proposed in [107] and enhanced in [108] by replacing comparison operators with intensity difference. The dimension of comparison features is $(W \times H) \times (W \times H - 1) \times 10$ (W and H are the width and height of face patch, and there are ten types of pixel comparison operators), and feature selection is utilized to reduce its length to 500.

GSS feature. GSS feature is developed by Gao et al. [138], and the feature is based on HOG31 feature map. The dimension for cell-based and block-based HOG feature maps are $m \times m \times 31$ and $n \times n \times 31$ respectively, where 31 denotes the dimension of HOG feature map, m denotes the cell number in a cell-based HOG31, and n indicates the block number in a block-based HOG31. The length of GSS feature vector is $[n^2 \times (n^2 - 1)/2] \times [\lfloor (m-n)/k \rfloor + 1]^2$ (here, k is the cell-based stride between two nearest neighboring blocks).

Table 3.3 shows the performance of methods based on the features introduced above.

We compare these methods in terms of the accuracy of them and the dimensions of the features they adopt. From the view of accuracy, our proposed smile detection system exceeds other state-of-the-art methods by about two to five percent. From the view of efficiency, we know that Gabor, LBP, Pixel Comparisons, and GSS are global feature descriptors and they are very time consuming when the resolution of the input image is high because they perform pixel-level calculations and each pixel in the image is involved in the operation. Our method is based on the shallow network CFAN and several simple calculations of Euclidean distances. CFAN can be very efficient by using some tricks like dropout and max pooling. Moreover, the classifier ELM is usually much faster than SVM and AdaBoost since ELM doesn't have iterations and the training is straightforward [38]. In summary, our method achieves the highest accuracy, the dimension of the proposed feature vector is the smallest, and our proposed system is more efficient than others.

3.4.6 Impact of Pose Variation

To evaluate the proposed approach in terms of the accuracy on various head poses, especially on different roll angles, we test our algorithm on specified samples in this section. We show some examples of failure and success in Fig. 3.15 and Fig. 3.16. We can see that our proposed methods work not well on images which have large head poses of yaw and pitch, and work quite well on images which have little head poses of yaw and pitch (even for those who may have large roll angles). We have proved the proposed feature vector has properties of invariance of translation, rotation, and scaling in Section 3.3. Among them, the property of invariance of rotation benefits smile detection results on the facial images which have roll head poses. We say a facial image has a head pose of only roll rotation is that the angle of yaw rotation and pitch rotation are both not more than ten degrees ($|\alpha| \leq 10^\circ, |\beta| \leq 10^\circ$).

Fig. 3.17 shows the relationship between the number of images and the pose upper limit of yaw and pitch. There are 2038 images (1169 positive images and 869 negative

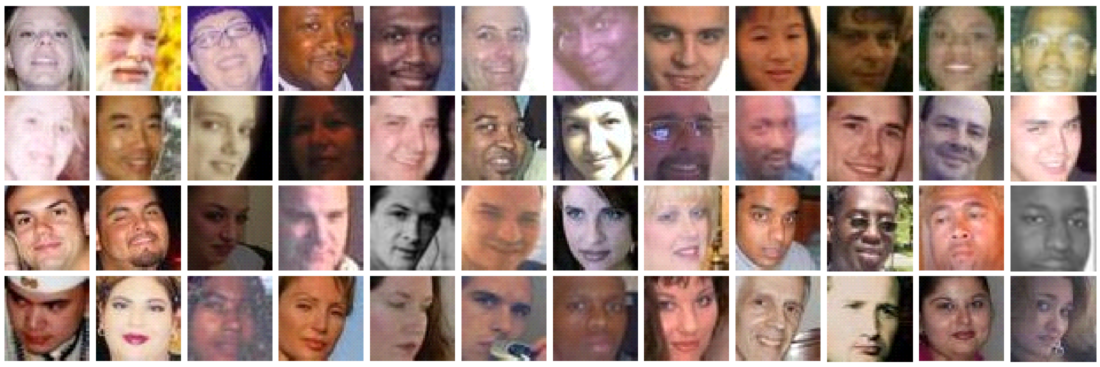


Figure 3.15: Examples of failure on smile detection.

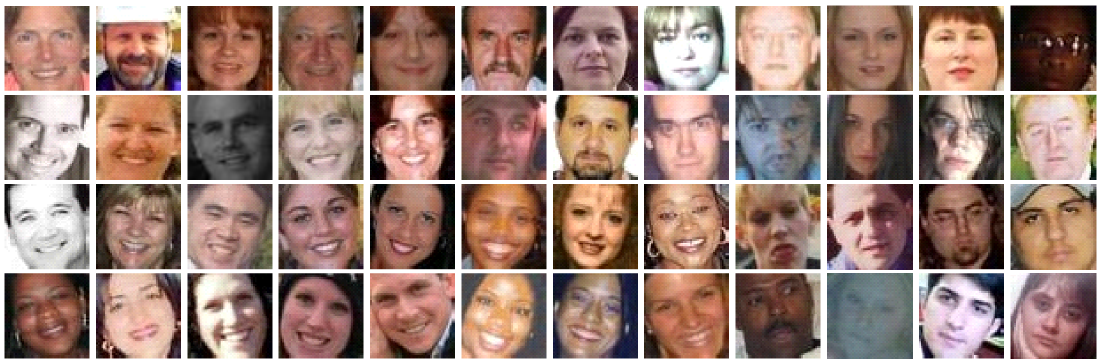


Figure 3.16: Examples of success on smile detection.

samples) with the head pose of only roll rotation in total. Among these 2038 images, we investigate the cumulative distribution based on the pose roll range, shown in Table. 3.4. For these 2038 images, we test our proposed method again, and the accuracy is 94.01%, which is higher than the accuracy of our proposed method on the whole database.

Table 3.4: Distribution of the 2038 images based on the roll poses range.

Number of Images	Positive	Negative	Total
$[0°, 5°]$	659	582	1241
$(5°, 10°]$	312	190	502
$(10°, 15°]$	139	71	210
$(15°, 20°]$	59	26	85

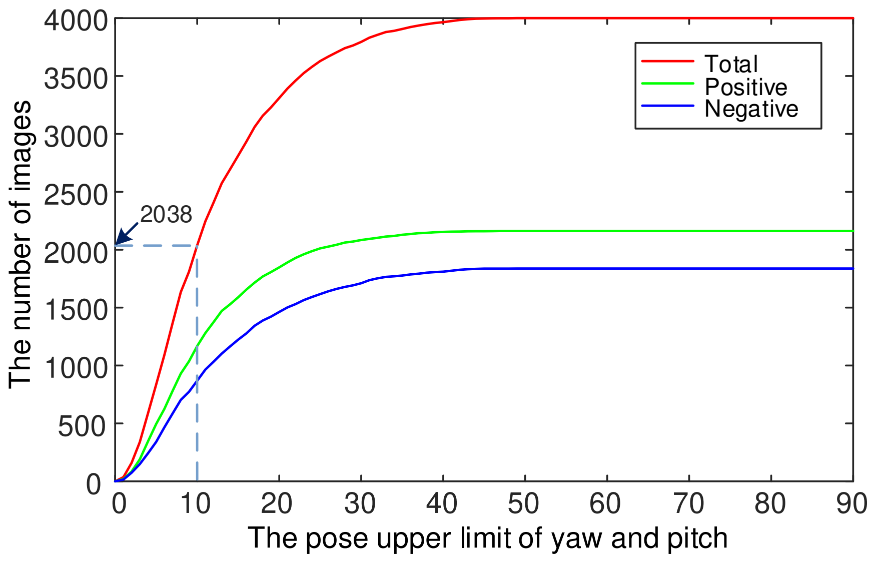


Figure 3.17: The relationship between the number of images and the pose upper limit of yaw and pitch.

3.4.7 Updated GENKI-4K Database

In this section, we update the GENKI-4K database by correcting the false annotations and removing the duplicates. Besides, we test our method on the updated database.

Correct Annotations. After we have checked all the images in this database, we found that some of the annotations are not proper. Finally, incorrect annotations of 12 positive samples and 24 negative samples are found, and they are shown in Fig. 3.18. Incorrect annotated positive samples are images which should have been labeled as negative (non-smile) but are labeled as positive (smile), and incorrect annotated negative samples are images which should be labeled as positive. These wrong annotations affect the results of smile detection algorithms.

Remove Duplicates. Among these 4000 images from GENKI-4K, there are 14 groups of duplications (shown in Fig. 3.19). Duplicate samples should be eliminated from the database because they do not add any new information [145].

After we correct these annotations and remove the duplicate samples, we test our proposed algorithm on the updated database, and find that the accuracy increases 0.81%.

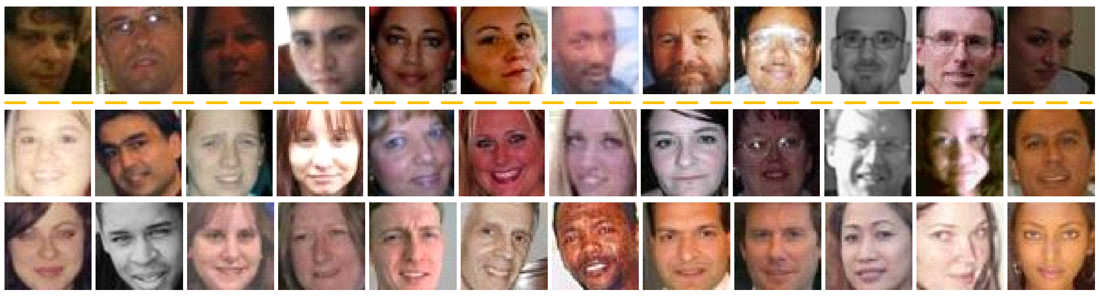


Figure 3.18: Wrong positive and negative samples. Images above the orange dotted line show the incorrect annotations of positive samples in the database and images below the orange dot lines show the incorrect annotations of negative samples.

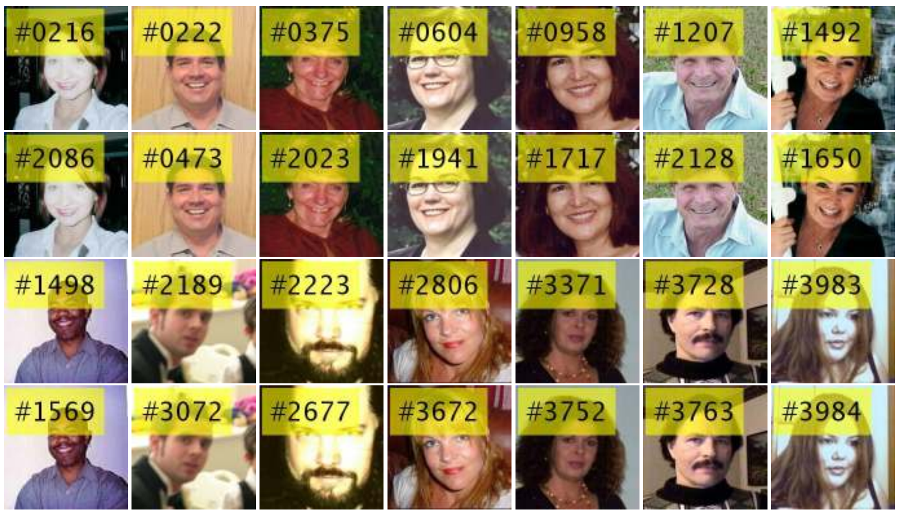
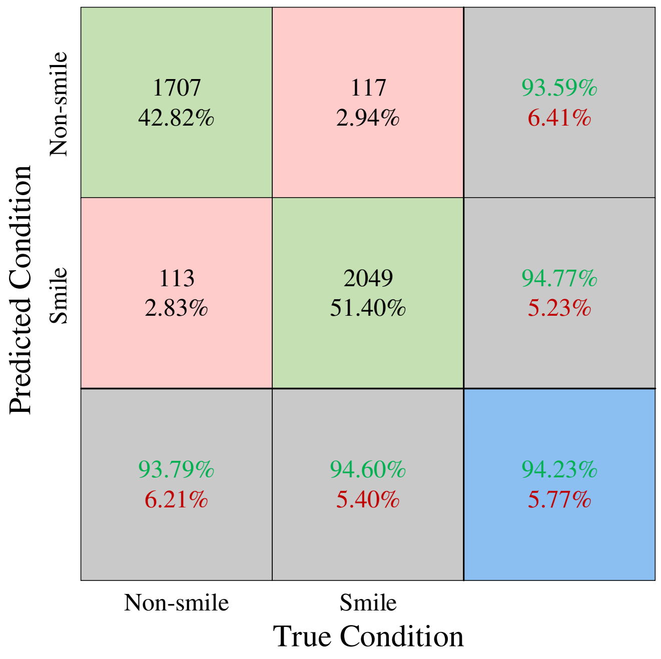


Figure 3.19: All 14 groups of duplications in GENKI-4K database. Images in Row 2 and Row 4 are the corresponding duplications of the images in Row 1 and Row 3 respectively.

The confusion matrix of our proposed method is shown in Fig. 3.20. There are 3986 images in total (14 images are removed from the database since there are 14 groups of duplicates. Among these 14 images, there are eight positive samples and six negative samples, so there are 2154(=2162-8) positive samples and 1832(=1838-6) negative samples remained with the original annotations). The false annotations of 12 positive samples and 24 negative samples are corrected, and there are 2166(=2154-12+24) positive samples and 1820(=1832+12-24) negative samples finally.



Predicted Condition \ True Condition	Non-smile	Smile	
Non-smile	1707 42.82%	117 2.94%	93.59% 6.41%
Smile	113 2.83%	2049 51.40%	94.77% 5.23%
	93.79% 6.21%	94.60% 5.40%	94.23% 5.77%

Figure 3.20: Confusion matrix for the true condition and the predicted condition.

3.5 Summary

In this chapter, we focus on smile detection system-level design and put forward a novel feature extraction method named Pair-wise Distance Vector, which can adequately reflect an object's shape and has properties of invariance of translation, rotation, and scaling. Facial landmarks are obtained by CFAN in the step of preprocessing, and classification is done by adopting ELM. Experimental results indicate that the accuracy of our approach is higher and the dimension of our proposed feature vector is lower than the state-of-the-art methods. Different classifiers are evaluated according to both raw-pixel-value features and Pair-wise Distance Vector features, and ELM outperforms AdaBoost and SVM. Further, we explore the impact of pose variation on our proposed method, and the experiment result verifies the effectiveness of our algorithm on the images which have the head pose of only roll rotation. Finally, we correct the incorrect annotations of the GENKI-4K database and remove the duplicate images. We have tested our algorithm on the updated database and found the accuracy increases further.

Chapter 4

Fly-ELM: Sparse Binary Extreme Learning Machine

Chapter 4 begins with the analysis of artificial and biological learning mechanisms of ELM networks, and the research gap on how to reduce the memory cost when keeping the accuracy. Then in this chapter, we introduce the sparse binary projection into ELM networks and propose the sparse binary ELM (also called “Fly-ELM” since this method is mainly inspired by the results from experiments based on the fly olfactory system). Furthermore, we have proved that the distances between arbitrary different input samples can be preserved in the proposed sparse binary ELM. At last, we have shown that comparing with the regularized ELM, the proposed Fly-ELM achieves comparable accuracy, but costs much less memory and requires less training and testing time in the benchmark of the optical character image classification database.

4.1 Introduction

We have introduced our work on the input layer of the ELM network in Chapter 3, and in this chapter, we concentrate on the distribution of the random weights between the input layer and the hidden layer. From our previous analysis we know that hidden nodes in ELM can be some type of super-nodes which are sub-networks consisting of some hidden nodes [61]. Recent research on ELM has been focusing on its hierarchical architectures and implementations [34, 146, 35, 99, 147].

The key theory of ELM is that learning can be made without tuning hidden nodes as long as these hidden nodes are non-piecewise continuous no matter whether they are artificial neurons, basis functions, or biological neurons. That means:

(1) the properties of hidden nodes (such as biological neurons) can be inherited from their ancestors;

(2) hidden nodes can be transferred from one system to another;

(3) hidden nodes can be randomly generated independently of training data. A random hidden node means all its parameters are randomly generated independently of applications.

ELM theory has been proved rigorously in theory [29, 38, 61, 62] and more and more its biological evidences have been observed recently [148, 149, 150, 151, 152]. Although ELM theories show that hidden node parameters can be randomly generated accordingly to any continuous distribution, in practice, in most cases, hidden node parameters are generated based on uniform, normal or Gaussian distribution [39, 40, 66]. It is noted that it is not easy to understand and visualize hidden node parameters with random uniform distribution, and it is hard to realize hardware implementation for such kind hidden nodes due to the precision issue.

The consecutive binary/ternary ELMs are proposed by initializing the weights of the connections between the input layer and the hidden layer with {0, 1} and {-1, 0, 1} respectively in Heeswijk and Miche [153]. Take the binary ELM for example. Assume that there are N_i input variables, and the number of hidden neurons is N_h. Then the total possible amount of the generated binary weight vectors is 2^{N_i}. First, the weights include all $\binom{N_i}{1}$ subsets of 1 variable, then all $\binom{N_i}{2}$ subsets of 2 variables, until there are N_h vectors. When N_i increases (e.g., N_i is bigger than 100 with high probability for computer vision tasks), the total number of possible weight vectors becomes huge, then the consecutive binary ELM will discard much information which might be very useful.

Huang, et al. [63] proposes ELM with random binary hidden nodes to achieve easier hardware implementation, which is of full connection. Although Huang [32, 154] analyze that the output connections of ELM networks can be adjusted based on sparse coding, sparse solution of binary ELM has neither been studied nor found in biological learning mechanism. Recently, some neurologists experimented with the fruit flies for odor classification (or similarity retrieval) to explore the theory and biological foundation of the similarity search system [155]. They have shown that the fly olfactory circuit assigns a "tag" for each odor with three steps, including divisive normalization, sparse binary random connection (key module), and a winner-take-all (WTA) circuit. The last two steps play the key role in the fly olfactory circuit, "fly algorithm" is a sparse binary random winner-take-all (SBR-WTA). This neural-based algorithm works well for similarity retrieval problems, but still suffers a gap for accurate classification problems (more details are explained in Section 4.2.2).

Inspired by the findings of the fruit fly olfactory system, combining this "fly algorithm" and the earlier work on random binary ELM [63] as well as sparse solutions for ELM [32, 154], this chapter proposes a novel ELM-based image classification method named sparse binary extreme learning machine (in short, Fly-ELM, in order to highlight its biological property found in the fruit fly olfactory system). Fly-ELM introduces sparse binary random matrix into ELM networks, which is different from the consecutive binary/ternary ELMs and SBR-WTA. The sparse binary projection restricts the number of non-zero coefficients, which guarantees non-overfitting and low computational cost

[156]. Meanwhile, the random projection with nonlinearity satisfies the universal approximating capability theory of ELM [29, 38, 157, 32].

The structure of this chapter is as follows. Section 4.2 introduces the previous related work, including basic regularized ELM and SBR-WTA. Section 4.3 presents and analyzes the proposed novel method Fly-ELM in detail and Section 4.4 shows the experiments and the discussion.

4.2 Related Works

4.2.1 Basic Regularized Extreme Learning Machine

ELM is a single-hidden-layer forward neural network, which is proposed to break the "barriers between conventional artificial learning techniques and biological learning mechanism" by Huang et al. [154]. There are many ELM variants like basic regularized extreme learning (regularized ELM [30]), online sequential extreme learning machine (OS-ELM [73, 74]), extreme learning machine autoencoder(ELM-AE [35, 55]), local receptive fields based extreme learning machine (ELM-LRF [34]), and so on. In this chapter, the regularized ELM is adopted and evaluated which has three key techniques:

(1) Hidden layer matrix (e.g., weight and bias matrix between the input layer and the hidden layer for additive nodes, frequencies and phases matrix for Fourier series, etc.) are assigned randomly.

(2) Output matrix between the hidden layer and the output layer is calculated directly.

(3) Regularization factor is introduced to balance the separating margin distance and training error, and it can reduce the risk of overfitting issue.

Assuming there are N samples and each sample $\mathbf{x}_i$ ($i \in [1,N]$) has N_I variables x_{ij} ($j \in [1,N_I]$), then the input layer of the ELM network has N_I neurons. The corresponding label vector of the i-th input sample is $\mathbf{y}_i = [y_{i1}, y_{i2}, \cdots, y_{iN_O}]$, and N_O is the length of the coding [56] of each label, which decides the number of output neurons. Given N_H hidden neurons, then the weight vector between the input layer and the hidden layer $\mathbf{w}_j = [w_{j1}, w_{j2}, \cdots, w_{jN_H}]$, and the weight vector between the hidden layer and the output layer $\beta_k = [\beta_{k1}, \beta_{k2}, \cdots, \beta_{km}]$. In matrix format, $\mathbf{X} = [\mathbf{x}_1; \mathbf{x}_2; \cdots; \mathbf{x}_N]$, $\mathbf{W} = [\mathbf{w}_1; \mathbf{w}_2; \cdots; \mathbf{w}_{N_I}]$, $\beta = [\beta_1; \beta_2; \cdots; \beta_{N_H}]$, and $\mathbf{Y} = [\mathbf{y}_1; \mathbf{y}_2; \cdots; \mathbf{y}_N]$. Also, the bias matrix between the input layer and the hidden layer is represented as $\mathbf{B}$, and the length of it is N_H. More importantly, $\mathbf{W}$ and $\mathbf{B}$ are randomly generated based on some statistic distributions like uniform distribution, normal distribution, and so on.

The objective of the regularized ELM is to minimize both the training errors and the norm of β. We analyze the ELM network layer by layer, then the output of hidden layer is $g(\mathbf{XW} + \mathbf{B})$, and $g(\cdot)$ is an activation function (sigmoid function by default). So the training errors can be expressed as $||g(\mathbf{XW} + \mathbf{B})\beta - \mathbf{Y}||^2$. Finally, the objective function of the regularized ELM is written as:

$$\underset{\beta}{\operatorname{argmin}} \frac{1}{2}||\beta||^2 + \frac{1}{2}C||\mathbf{H}\beta - \mathbf{Y}||^2, \tag{4.1}$$

where $\mathbf{H} = g(\mathbf{XW} + \mathbf{B})$ and C is the regularization factor which can be assigned by users.

According to ELM theory [30], when $N > N_I$, the solution of Eq. (4.1) is:

$$\beta = \mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{Y}, \tag{4.2}$$

and when $N <= N_I$, the resolution is:

$$\beta = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H} \right)^{-1} \mathbf{H}^T\mathbf{Y}. \tag{4.3}$$

Noted that $\mathbf{I}$ is an identity matrix.

By denoting $\mathbf{H}\mathbf{H}^T$ (or $\mathbf{H}^T\mathbf{H}$) as $\boldsymbol{\Omega}$ (a kernel function), we obtain the solution of the regularized ELM. The difference between regularized ELM and basic ELM is that the regularization factor C is introduced into the former, which makes it more powerful to reduce the risk of overfitting.

4.2.2 Sparse Binary Random Winner-Take-All (SBR-WTA)

Sparse matrices are the matrices which have only a very few non-zero elements, and sparse binary matrices are one particular case of sparse matrices which only allows binary values 0 and 1 [158, 159]. Because of the high storage efficiency and low computational cost [160], sparse binary matrices have been successfully applied to many computer science tasks like structures representation [161], compressed sensing [162].

Recently, Dasgupta et al. [155] proposed a neural algorithm of which neurological foundation is the fruit fly olfactory circuit and this circuit can be taken as a three-layer feedforward neural network. These three layers are named PNs (projection neurons) layer, KCs (Kenyon cells) layer, and APL (anterior paired lateral) layer respectively. The first layer has N_I PNs corresponding to the N_I receptor neuron types (for the fruit fly olfactory circuit, there are 50 odorant receptor neuron types, so $N_I = 50$). Each receptor neuron performs a mean-centering normalization, which is similar to the common preprocessing step “divisive normalization” [163]. There are N input samples and each sample is represented as $\mathbf{x}_i = [x_{i1}, x_{i2}, \cdots, x_{iN_I}]$ ($i \in [1,N]$). Assume that the second layer of SBR-WTA neural network has N_H KCs, then the size of the projective matrix $\mathbf{W}$ is $N_I \times N_H$. $\mathbf{W}$ is a sparse binary random matrix, and each entry W_{jk} is set independently with probability p.The corresponding output of the KCs is $\mathbf{h}_i = \mathbf{x}_i\mathbf{W}$. It has been deduced in Dasgupta et al. [155] that the squared ℓ_2-norm of $\mathbf{h}_i$ is

$$\mathbb{E}\|\mathbf{h}_i\|^2 = pN_H\left((1-p)\|\mathbf{x}_i\|^2 + p(\textstyle\sum \mathbf{x}_i)^2\right), \tag{4.4}$$

where $\sum \mathbf{x_i}$ is the sum of all the vector $\mathbf{x_i}$'s entries. Then we have

$$\mathbb{E}\|\mathbf{h}_i - \mathbf{h}_{i'}\|^2 = pN_H \left((1-p)\|\mathbf{x}_i - \mathbf{x}_{i'}\|^2 + p\left(\sum(\mathbf{x}_i - \mathbf{x}_{i'})\right)^2\right). \tag{4.5}$$

In Eq. (4.5), $\mathbf{x}_i$ and $\mathbf{x}_{i'}$ are arbitrary two different input samples, and their corresponding hidden layer outputs are $\mathbf{h}_i$ and $\mathbf{h}_{i'}$ respectively. We know that after mean-centring normalization, $\mathbf{x}_i$ and $\mathbf{x}_{i'}$ have roughly the same distribution, then $\sum(\mathbf{x}_i - \mathbf{x}_{i'}) \approx 0$. The distance between $\mathbf{x}_i$ and $\mathbf{x}_{i'}$ are preserved by the expectation of the distance between $\mathbf{h}_i$ and $\mathbf{h}_{i'}$. According to [155], the distance between $\mathbf{h}_i$ and $\mathbf{h}_{i'}$ is concentrated tightly by its expectation when $\mathbf{x}_i$ and $\boldsymbol{x}_{i'}$ are sparse enough. The sparse binary random projection preserves the distances of input samples.

There are four strategies to process the obtained $\mathbf{h_i}$ and generate a sparse vector $\mathbf{h'_i}$ for the corresponding odor in the fruit fly olfactory circuit from KCs layer to APL layer. The dimensions of $\mathbf{h}_i$ and $\mathbf{h}'_i$ equal to the number of the Kenyon cells (hidden neurons) N_H. Assume there are κ non-zero elements in $\mathbf{h}'_i$, then these four strategies are expressed below:

- Random selection:
$$\mathbf{h}'_{ij} = \begin{cases} \mathbf{h}_{ij}, \text{ if } \mathbf{h}_{ij} \text{ is one of the } \kappa \text{ randomly selected entries of } \mathbf{h}_i. \\ 0, \quad \text{otherwise.} \end{cases}$$
- Top selection (WTA):
$$\mathbf{h}'_{ij} = \begin{cases} \mathbf{h}_{ij}, \text{ if } \mathbf{h}_{ij} \text{ is one of the } \kappa \text{ biggest entries of } \mathbf{h}_i. \\ 0, \quad \text{otherwise.} \end{cases}$$
- Bottom selection:
$$\mathbf{h}'_{ij} = \begin{cases} \mathbf{h}_{ij}, \text{ if } \mathbf{h}_{ij} \text{ is one of the } \kappa \text{ smallest entries of } \mathbf{h}_i. \\ 0, \quad \text{otherwise.} \end{cases}$$
- All selection: $\mathbf{h}'_i = \mathbf{h}_i$.

With the generated sparse vectors of the odors, an approximate nearest neighbors search [164] is performed to classify or identify the odors. Euclidean distance is adopted to measure the similarity of the samples, and compared with the abilities of generalization of the ELM, the approximate nearest neighbors search only performs a very rough classification.

4.3 Sparse Binary Extreme Learning Machine (Fly-ELM)

Motivated by the sparse binary random projection mechanism from the fruit fly olfactory system and ELM theories on binary hidden neurons, this chapter proposes a method called sparse binary ELM (Fly-ELM), whose architecture is shown in Fig. 4.1.

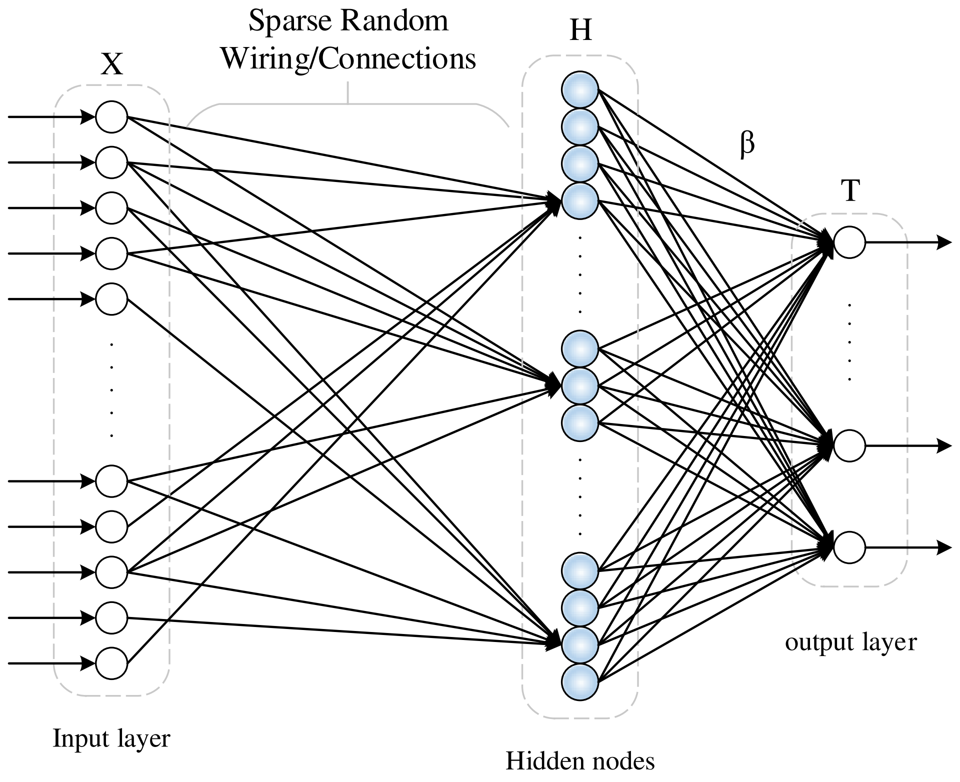


Figure 4.1: The architecture of sparse binary extreme learning machine.

There are five key components in the network of the Fly-ELM. The first component is the input layer, which is fed with the samples (including the raw data or features of the samples). Following the naming rules of the ELMs, we use $\mathbf{x}_i$ to represent the i-th sample data, and $\mathbf{X} = [\mathbf{x}_1; \mathbf{x}_2; ...; \mathbf{x}_N]$ donates all the input samples. Each vector $\mathbf{x}_i$' length N_I is decided by the preprocessing step, and the number of the neurons in the input layer is also N_I. Take image samples as an example.

- If the raw image is used as the input, the dimension is the amount of all the pixels in the image.
- If some feature extraction or learning methods (like LBP [19], SIFT [165], ELM-AE [35], CFR-ELM [55], etc.) are adopted to analyze the image, the dimension is the length of the output features of these methods.

Noted that methods like data normalization might be embedded between the input layer and the raw data (or extracted features), but these methods usually do not change the dimension of the input data.

In the theory of the ELMs, the weight matrix $\mathbf{W}$ and the bias matrix $\mathbf{B}$ between the input layer and the hidden layer are randomly generated by following the continuous sampling distributions. The most widely used is the continuous uniform distribution [70]. The cumulative density function of the uniform distribution on (-1,1) is given by

$$F(x) = \begin{cases} 0, \ \ x < -1 \\ (x+1)/2, \ \ -1 \leq x < 1 \\ 1, \ \ x \geq 1. \end{cases} \tag{4.6}$$

Assume that there are N_H hidden neurons in the hidden layer, then the sizes of $\mathbf{W}$ and $\mathbf{B}$ are $N_I \times N_H$ and $N \times N_H$ respectively. All elements in $\mathbf{W}$ are generated randomly based on the uniform distribution, while for $\mathbf{B}$, only the first row is randomly generated and the rest rows repeat the first row. The output of the hidden layer for input samples is

$\mathbf{H} = g(\mathbf{XW} + \mathbf{B})$, whose size is $N \times N_H$, where $g(\cdot)$ represents any nonlinear piecewise continuous activation functions, and some commonly used of them are listed below:

- Sigmoid function: $g(x) = \frac{1}{1+e^{-x}}$.
- Tanh function: $g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.
- Radial basis function: $g(x) = g(\|x\|)$.
- Sinc: $g(x) = \frac{\sin(x)}{x}$ for $x \neq 0$, and $g(x) = 1$ for $x = 0$.

Noted that activation functions are implemented at the single element level, so they do not change the dimensions of their arguments. For the sake of storage, assume all the elements of $\mathbf{W}$ and $\mathbf{B}$ are in double precision. Based on the IEEE double-point standards [166], each double data occupies 64 bits in computer memory. For the current ELM-based methods, $64N_H(N_I + 1)$ bits are required to store the $\mathbf{W}$ and $\mathbf{B}$. In addition, when the input samples are randomly projected from the input layer and the hidden layer, there are three kinds of representations: compressed representation ($N_H < N_I$), equal dimension representation ($N_H = N_I$), and expanded representation ($N_H > N_I$). They all are the results of dense projections, which need many more mathematical operations (plus and multiply) in computing.

Motivated by the findings from the neurologists on the research of the fly olfactory circuit, this chapter proposes to adopt the sparse binary random projection rather than the dense random projection in the ELM network. Since it is ELM theory that shows binary random hidden neurons can be used in neural networks and the universal approximation capability can be guaranteed [61], the proposed method is named as sparse binary extreme learning machine or Fly-ELM. The proposed method keeps the advantages of random projections because it has been proved that random projections provide better theoretical guarantees and better bounds [167, 168, 169, 61]. The weight matrix $\mathbf{W}$ between the input layer and the hidden layer is sparse binary randomly generated. Unlike traditional ELMs, we have abandoned the bias matrix $\mathbf{B}$, then we have $\mathbf{H} = g(\mathbf{XW})$.

In the network of the Fly-ELM, each neuron in the hidden layer is connected with r input neurons (see Fig. 4.1). Mathematically, each column of $\mathbf{W}$ is a sparse binary vector with r non-zero elements. The hyperparameter r is called hash length, and it can be empirically set between 1 and N_I, and r is assigned to six in the experiments of evaluating the fly algorithm [155]. In Section 4.4, different values are assigned to r, and the corresponding training and test accuracies are evaluated.

Seen from Section 4.2.2, sparse binary random projection preserves the distances between any different input samples. Eq. (4.5) shows the expected distances between the outputs of the arbitrary two different input samples. However, as observed from Eq. (4.4) the realistic distance could be very different when the variance of $||\mathbf{h}_i||$ is significant.

Theorem. Given the sparse binary weight matrix $\mathbf{W}$, if $N_H \geq \frac{5}{\varepsilon^2\delta}\left(2r + \frac{N_I||\mathbf{x}_i||_4^4}{||\mathbf{x}_i||^4}\right)$, then with probability at least $1-\delta$, $\left|\frac{||\mathbf{h}_i||^2}{\mathbb{E}||\mathbf{h}_i||^2} - 1\right| \leq \varepsilon$ holds, which is equivalent to that $||\mathbf{h}_i||$ is tightly concentrated around its expected value. Here, δ and ε are arbitrary values range from 0 to 1.

Proof. Based on Chebyshev's inequality [170], we have

$$P\left(\left|||\mathbf{h}_i||^2 - \mathbb{E}||\mathbf{h}_i||^2\right| \geq \varepsilon\mathbb{E}||\mathbf{h}_i||^2\right) \leq \frac{\sigma(||\mathbf{h}_i||^2)}{(\varepsilon\mathbb{E}||\mathbf{h}_i||^2)^2}. \tag{4.7}$$

Here, $\sigma(\cdot)$ is the variance symbol, and we know that $\sigma(||\mathbf{h}_i||^2) = N_H||h_{i1}||^2$ since $||\mathbf{h}_i||^2 = \sum_j^{N_H} h_{ij}$ and all entries follow the uniform distribution. Similarly, we have $\mathbb{E}||\mathbf{h}_i||^2 = N_H\mathbb{E}h_{ij}^2$. Then, Eq. (4.7) is simplified as

$$P\left(\left|||\mathbf{h}_i||^2 - \mathbb{E}||\mathbf{h}_i||^2\right| \geq \varepsilon\mathbb{E}||\mathbf{h}_i||^2\right) \leq \frac{\sigma(h_{i1}^2)}{N_H(\varepsilon\mathbb{E}h_{i1}^2)^2}. \tag{4.8}$$

Since $\sigma(h_{i1}^2) = \mathbb{E}h_{i1}^4 - (\mathbb{E}h_{i1}^2)^2$, calculate $\mathbb{E}h_{i1}^4$ as below:

$$\begin{aligned} \mathbb{E}h_{i1}^4 &= \mathbb{E}(x_{i1}w_{11} + x_{i2}w_{21} + \cdots + x_{iN_I}w_{N_I1})^4 \\ &\leq (p + 4p^2N_I)\|\mathbf{x}_i\|^4 + (3p^2 + 6p^3N_I)\|\mathbf{x}_i\|^4 + p^4(\sum \mathbf{x}_i)^4. \end{aligned} \tag{4.9}$$

Refer to the supplementary materials of [155] for the detailed derivation process of the Eq. (4.9). After plugging Eq. (4.4) and Eq. (4.9) into Eq. (4.8), one can obtain the lower limit of the N_H. That is, for the sparse binary random projections, the distances between arbitrary different input samples can always be preserved by exploiting enough hidden neurons.

For the sake of storage, the weight matrix **W** between the input layer and the hidden layer needs only $N_H * r$ bits for sparse binary projection, which is much less than that requested in the dense random projection. In addition, the binary random projection also performs feature selection of the input data, which can simplify the models and enhance the generalization ability of the proposed network.

After the hidden layer output matrix is obtained, the next step is to find the output weight matrix β of the connections between the hidden layer and the output layer. This part is entirely the same with standard ELM networks. Compute β by optimising the objection function Eq. (4.1) as given in Eq. (4.2) and Eq. (4.3) in Section 4.2.1.

Compared with the regularized ELM, our proposed Fly-ELM has two main differences.

- The distances under sparse binary random projection have been proved to be preserved in the Fly-ELM while the general random projections only preserve the distances under some specific conditions (refer to [168] for more details).
- Sparse binary entries can be generated and stored more efficiently than dense random entries.

Even though the proposed Fly-ELM and the binary/ternary ELM both focus on how to initialize the weights between the input layer and the hidden layer in the ELM network, the former is different from the later. The former's amount of non-zero entries in each column of **W** is fixed, and each column is a sparse vector, which has many advantages as we have explained above. But for the binary/ternary ELM, the amount of non-zero entries may be dense.

The proposed method is inspired by the SBR-WTA algorithm which has a biological basis. Here we list two main differences between the Fly-ELM and the SBR-WTA.

- Values of all outputs of the hidden layer are used in the Fly-ELM while the indices of partial outputs of the hidden layer are adopted in the SBR-WTA.
- Weights are optimized by minimizing the loss of the true labels and predicted labels in the Fly-ELM [61] while SBR-WTA does not have this constraint.

In the experiments, the performance of the proposed Fly-ELM with the regularized ELM and the SBR-WTA on the public dataset are compared. The analysis will also be given based on the results of the experiments. The proposed method has not been compared with binary/ternary ELM because this work focuses on computer vision tasks, and the dimensions of the data/feature are usually very high. High dimension input variables will lead a massive amount of possible weight vectors in the binary/ternary ELM, and most of which will be abandoned, which means most of the useful information will be discarded.

4.4 Experimental Results

Three experiments have been conducted in order to evaluate the proposed Fly-ELM. Since the proposed method is inspired by the regularized ELM, and SBR-WTA, the Fly-ELM will be compared with these two algorithms in the first experiment. In the third experiment, the performance of the Fly-ELM under different hash length r will be investigated.

All these methods are tested on the widespread public optical character recognition (OCR) dataset [80]. The OCR dataset was collected by Rob Kassel at MIT Spoken Language Systems Group for testing the abilities of handwritten letters recognition systems. It contains 52,152 sample images, and the width and height of each image are 8 and 16 respectively. Some samples from the OCR dataset are shown in Fig. 4.2, from which it can be seen that the samples of each letter vary a lot on the aspects of slant degrees, the thickness, and the pressure to the chapter.

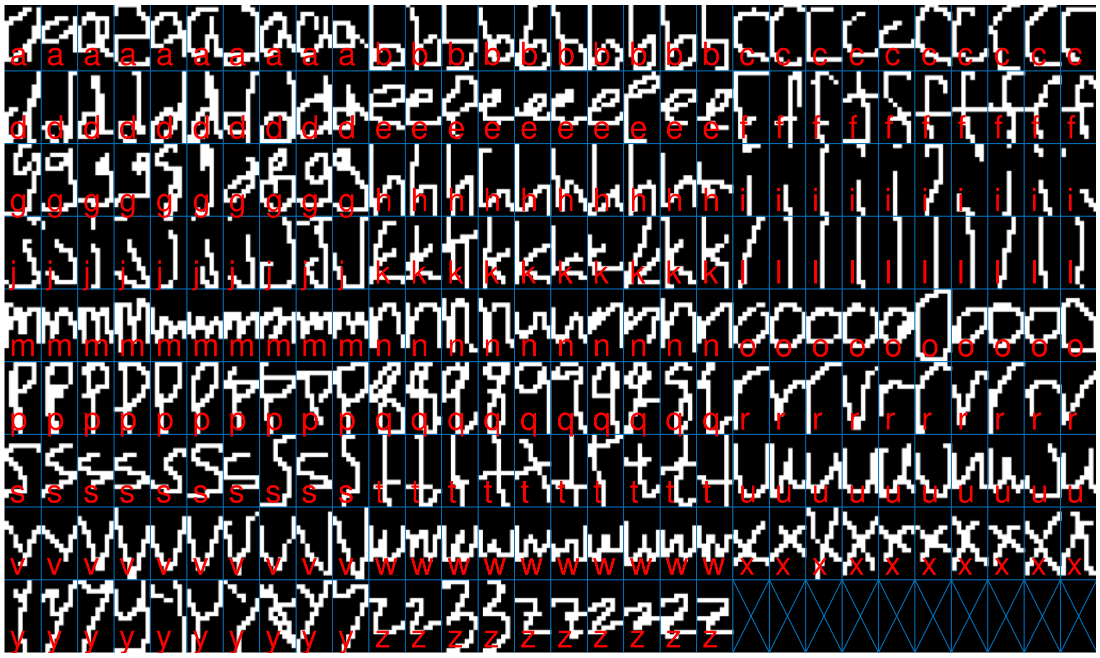


Figure 4.2: Ten randomly selected images of each letter from the optical character recognition (OCR) dataset. The ground truth of each image is marked in red color at the bottom of each image.

All these images are grouped into ten folders, nine of which are used for training and the remaining one is used for testing. All these ten folders are taken by turns for testing, so ten test accuracies are obtained, and the average of all the accuracies is the final test accuracy. The experiments are carried out in the regular personal desktop with a 3.60 GHz Intel(R) Core(TM) i7-4790 processor and 32 GB memory running Matlab 2017b.

4.4.1 Comparison of the Fly-ELM with the regularized ELM

For the Fly-ELM, there are three hyperparameters, the number of non-zero entries in each sparse binary vector r, the regularization factor C, and the number of hidden neurons N_H. While for the regularized ELM, it only has the last two hyperparameters. So to compare the proposed method with the regularized ELM, the control variable method is adopted.

When N_H is fixed at 2,000, we set C as 2^τ, and τ ranges from -10 to 10. Different values are assigned to r, and the corresponding training accuracy and test accuracy are achieved. The results are shown in Table 4.1, from which it can be concluded that the regularized ELM performs a little better than Fly-ELM when the amount of hidden neurons is fixed. By considering that the proposed weight matrix **W** takes up only about one thousandth $((r \times N_H)/(64 \times N_H \times (N_I + 1)))$ of the former, this little bit of performance degradation is acceptable.

Table 4.1: Comparison of the regularized ELM and Fly-ELM when N_H is fixed to 2,000.

Methods		Training Accuracy(%)	Test Accuracy(%)
Regularized ELM		83.81	80.33
Fly-ELM	$r = 2$	83.33	79.72
	$r = 3$	83.23	79.58
	$r = 4$	83.07	79.39
	$r = 6$	82.78	79.50
	$r = 8$	82.09	78.98

It is known that ELMs are based on the empirical risk minimization, and they tend to overfit when the models become complex [75, 171]. For the regularized ELM, the regularization factor adds the probability of reducing overfitting, but it cannot reduce this risk to zero when more hidden neurons are added. To have a fair comparison and avoid the overfitting, the Fly-ELM is compared with basic ELM by constraining the training accuracy as 95%.

Table 4.2 shows the test accuracies of the regularized ELM and the Fly-ELM (similar to

Table 4.2: Comparison of the regularized and the Fly-ELM when the training accuracy is approximately equal to 95%.

Methods		N_H	Test Accuracy(%)
Regularized ELM		7700	86.31
Fly-ELM	$r = 2$	10400	84.33
	$r = 3$	11000	85.92
	$r = 4$	11700	86.20
	$r = 6$	14700	86.68
	$r = 8$	18600	87.11

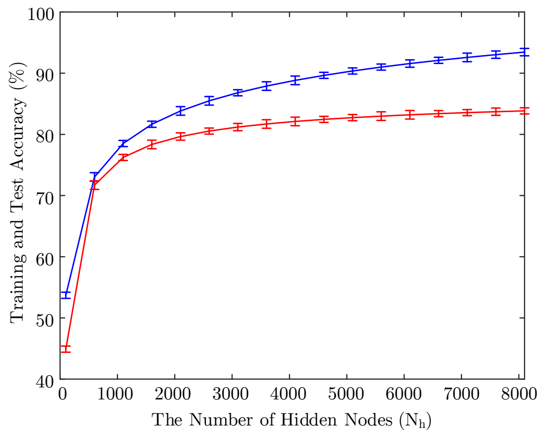
the previous one, different values are assigned to r). As observed from the results, when the training accuracy is about 95%, the test accuracy of Fly-ELM achieves higher accuracy than the regularized ELM by setting $r = 8$. Noted that compared to the regularized ELM, the Fly-ELM needs more hidden neurons to achieve the same training accuracy.

4.4.2 Comparison of the Fly-ELM with the SBR-WTA

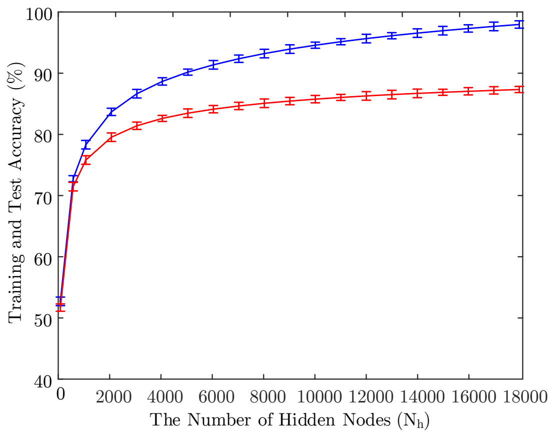
The proposed Fly-ELM is partially motivated by the SBR-WTA method, which has a neurological foundation. The SBR-WTA algorithm has been proved to have the classification ability in [155], and it is compared with the Fly-ELM in the OCR dataset.

According to Section 4.2.2, there are three hyperparameters for the SBR-WTA, the probability p, the number of hidden neurons (KCs) N_H, and the amount of non-zero elements in the output of hidden neurons κ. Noted the relationship between the probability p and the hash length r: $p = r/N_I$ ($N_I = 128$ for the OCR dataset). Following the settings in [155], assign N_H and κ as 2,000 and eight, respectively. For the Fly-ELM, N_H is also elected as 2,000, and other implementation details are the same as the previous experiments. Results of the test accuracy and running time (RT, consists of training time (Ttr) and testing time (Tte)) are obtained for different values of r, shown in Table 4.3.

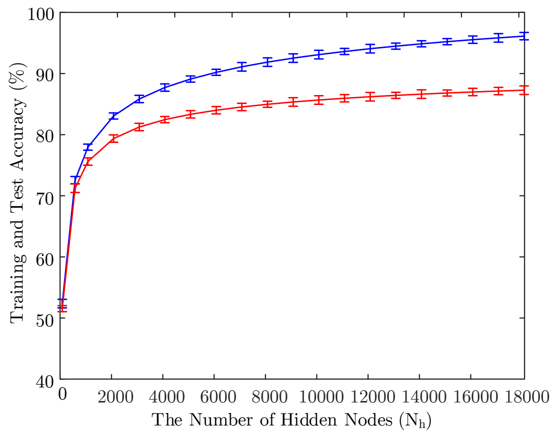
We can conclude that the accuracies of our method are much higher than that of the SBR-WTA from Table 4.3. Moreover, in terms of the running time, the proposed method costs about a sixth of the latter.



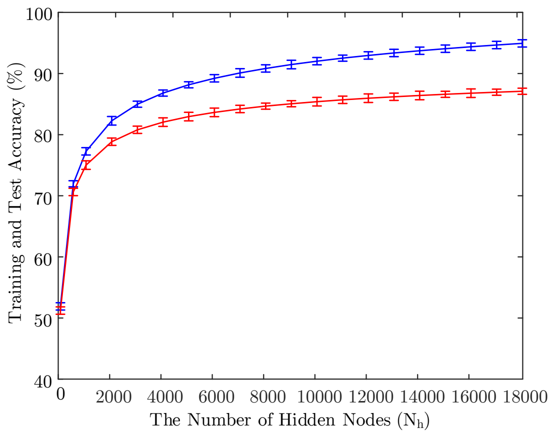
(a) $r = 2$



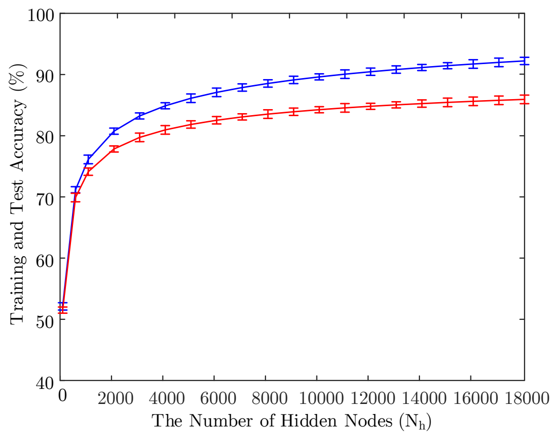
(b) $r = 3$



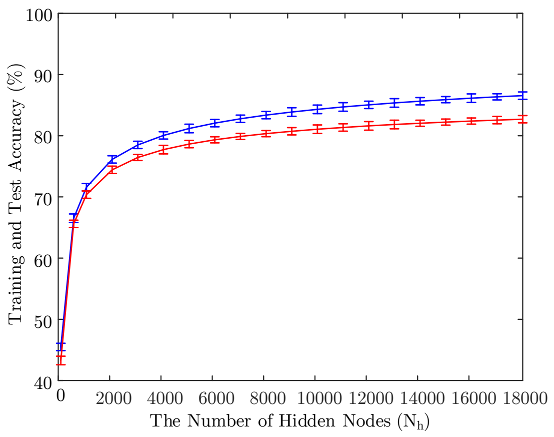
(c) $r = 6$



(d) $r = 8$



(e) $r = 12$



(f) $r = 24$

Figure 4.3: Comparison of different hash length r. The training accuracy curves are drawn in blue color, while the test accuracy curves are in the red.

Table 4.3: Comparison of the Fly-ELM with the SBR-WTA.

r	SBR-WTA				Fly-ELM			
	Accuracy (%)	Ttr (s)	Tte(s)	RT (s)	Accuracy(%)	Ttr (s)	Tte(s)	RT (s)
2	54.10	31.69	4.25	35.94	79.72	5.69	0.81	6.50
3	52.82	31.73	4.33	36.06	79.58	5.14	0.87	6.01
4	53.94	31.81	4.38	36.19	79.39	5.34	0.83	6.17
6	53.84	31.89	4.53	36.42	79.50	5.33	0.84	6.17
8	55.60	31.89	4.52	36.41	78.98	5.21	0.86	6.07

4.4.3 Analysis of the Impact of Different Hash Length

As analyzed beforehand, the hash length r does impact the performance of the Fly-ELM. The training accuracy and test accuracy under different values of r are further investigated. The number of hidden neurons in each Fly-ELM network varies from 100 to 18000 with a step of 50. Noted that for r=2, the maximum of N_H is 8100 since $\binom{128}{2} < 8100+50$.

Set different values to r, and the curves of the corresponding training accuracy and test accuracy are shown in Fig. 4.3, from which two conclusions can be drawn:

- The training and test accuracies increase sharply in the beginning and then increase slowly when the number of the hidden neurons increases and when the hash length r is fixed.
- The training and test accuracies increase first and then decrease with the increase of the hash length r.

In summary, three experiments have been conducted to evaluate the performance of our Fly-ELM in this section. Firstly, the proposed method is compared with the regularized ELM on test accuracy by using the control variable method. Compared with the regularized ELM, similar or even better results are obtained with far less storage required. Secondly, compare the Fly-ELM with the SBR-WTA algorithm, it is found that the proposed method performs much better on the aspects of test accuracy and running time. Thirdly, the impact of different values of r on the performance of the Fly-ELM is analyzed in detail.

4.5 Summary

This chapter proposes a novel neurology-based method named Fly-ELM by embedding sparse binary random projections into the existing ELM network. The results of experiments on the public OCR dataset show that the Fly-ELM outperforms the regularized ELM and the SBR-WTA algorithm.

The work in this chapter is a first attempt at combining sparse binary random projections with the ELM neural network. More effort could be made on applying these projections to other ELM variants in the future. Also, the Fly-ELM will be implemented in hardware platforms due to their particularity of limited memories, which will benefit more real-world applications.

Chapter 5

Compact Feature Learning by Using Extreme Learning Machine

Chapter 5 begins with the brief review of deep neural networks on feature learning, and analysis of a shallow feature learning method PCANet. Then we propose our compact feature learning algorithm based on ELM network referred to CFR-ELM, and the details of it are explained step by step, including its overall framework, the key unit, and post-processing modules. At last, four experiments have been conducted, and the results have demonstrated that our ELM-based method outperforms the existing state-of-the-art methods.

5.1 Introduction

Feature representation/learning is an critical part for many computer vision tasks, such as object detection [20, 172, 173], object recognition [174, 175], object segmentation [16, 176] and image classification [177, 178, 179]. With years of research, a lot of work has been done on how to extract efficient and discriminative features manually or automatically. But this is still a hot research field because of facing challenges from illumination, occlusion, deformations and so on. Nowadays, neural network-based feature representation methods have made remarkable achievements, and they can be broadly categorized as deep and shallow feature learning.

Many deep feature learning algorithms have been proposed for image classification problems. Ciregan *et al.* [180] proposed a multi-column deep neural network which is a wide and deep artificial neural network architecture, which was claimed could match human performance on tasks like traffic signs image classification. A pyramid convolutional neural network was proposed for face representation by adopting a greedy-filter-and-down-sample operation in [181]. Ouyang *et al.* [173] proposed deformable deep convolutional neural networks to learn features for object detection. Shojaeilangari *et al.* [42] used extreme sparse learning to represent facial images for robust facial emotions recognition and achieved a high accuracy. However deep feature learning algorithms require a lot of computational resources, a large amount of training time and huge storage space [36, 41].

In contrast, shallow feature learning algorithms usually don't have these disadvantages and for some tasks achieve performance comparable to deep feature learning algorithms. For example, Coates *et al.* [182] analyzed the feature learning ability of single-layer networks and achieved a high performance when the number of hidden nodes and other parameters are pushed to their limits. A method of unsupervised representation learning based on ELM is proposed on in [43], in which, ELM-AE was adopted as the learning unit, and a transferred layer was introduced. Besides, local contrast normalization (LCD) and whitening were employed as pre-processing steps. Chan *et al.* [183] proposed a simple learning architecture named PCA network (PCANet for short, and PCA is the abbreviated form of 'Principle Component Analysis'), which is quite intuitive and can be efficiently estimated. PCANet has attained remarkable results, but computing PCA costs a lot of time and resources [44].

Our work is partially motivated by the recent results in dimension reduction using ELM [44] which proved the ELM's generalization capability of learning local feature with fast speed. Besides, our work is inspired by PCANet architecture for its astonishing performance on image classification and pooling method adopted in many deep learning methods for its compact ability on feature extraction. All of these lead us to come up with a novel ELM-based shallow framework to learn compact features for image classification.

In the following sections, we introduce the related work on PCANet and ELM briefly. Then, we explain the whole compact feature representation method (CFR-ELM) we proposed, including the details of each feature learning module, max pooling module and the overall framework for image classification which has a support vector machine (SVM) as a classifier. The proposed algorithm is tested on four typical image classification databases, and conclusions are given finally.

5.2 Related Works

PCANet was introduced as a shallow feature learning algorithm in [183] and can be used as a simple baseline for deep feature learning algorithms. It contains three different types of modules: pre-processing, PCA filter convolution, post-processing. In general, the first two types of modules are combined together to form one feature learning layer. The overall architecture of PCANet is created by connecting the feature learning layers and the post-processing, and a two-stage PCANet architecture is shown in Fig. 5.1.

In the pre-processing state, the sliding window with no stride is used to extract patches

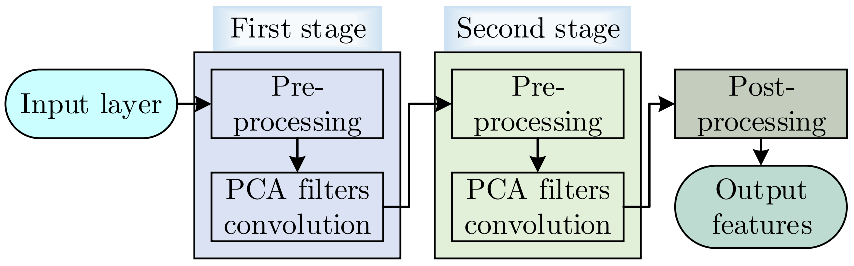


Figure 5.1: A schematic of two-stage PCANet.

and remove the mean as:

$$\bar{\mathbf{x}}_k = \mathbf{x}_k - \frac{\sum_{i=1}^{n} \mathbf{x}_{k_i}}{n}\mathbf{1}, \tag{5.1}$$

where $\mathbf{x}_k$ denotes the k-th patch which contains n pixel values, and $\mathbf{1}$ is an all one vector whose size is the same with $\mathbf{x}_k$. All patches are combined together to form a normalized matrix $\mathbf{X}$. After that, a PCA filter convolution operation is done. The objective function of PCANet is to minimize the reconstruction error with a set of filters, which is expressed as

$$\underset{\mathbf{V}}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{V}\mathbf{V}^T\mathbf{X}\|_{\mathrm{F}}^2, \tag{5.2}$$

where $\mathbf{V}$ is an orthogonal matrix, and $\|\cdot\|_{\mathrm{F}}^2$ is the Frobenius norm. The solution of the objective function is known as the principal eigenvectors of $\mathbf{X}$'s covariance matrix. First, l principal vectors (called PCA filters) are chosen as a convolution core to extract the features of the original images.

The second stage is similar to the first stage, and the main difference is that the total number of patches to be processed is L (the number of filters in the first stage) times of the first stage. With the increase of stages, the amount of patches to be processed becomes larger and larger.

From the above analysis, we know that PCA learns a linear transformation, and it is not enough for representing the complex real-world features. ELM can make up this by its inherent abilities of linear and nonlinear representation. Extreme learning machines

(ELM) was proposed along with the interdisciplinary development of effective machine learning theories and techniques in 2006 [38]. ELM has been attracting the increasing attention from more and more researchers due to the amazing abilities of classification, regression, feature learning, sparse coding, and compression [30]. It has successfully been applied in more and more real industrial applications and usually achieves comparable or better results than many conventional learning algorithms at much fast learning speed.The basic ELM is expressed as

$$\beta = \mathbf{Y}(g(\mathbf{WX}+\mathbf{B}))^{\dagger}. \tag{5.3}$$

In Equ.5.3, $\mathbf{X}$ is a $D \times N$ matrix that denotes the input data, where D is the number of features, and N is the number of samples. $\mathbf{W}$ is a $N_h \times D$ weight matrix between the input layer and the hidden layer, and the elements in W is randomly initialized. $\mathbf{B}$ is a $N_h \times N$ bias matrix of the hidden neurons, where the first column is randomly initialized, and other columns are the duplicates of the first column. $\mathbf{Y}$ is the target data, and it is replaced by X for the auto-encoder, that is, $\mathbf{Y} = \mathbf{X}$. β is a $D \times N_h$ weight matrix between the output layer and hidden layer. $g(\cdot)$ is an activation function, which can be a Sigmoid function (default), a Sine function, a Radbas function, etc. The $(\cdot)^{\dagger}$ is a Moore-Penrose pseudoinverse operator.

Feature learning with ELM auto-encoder (ELM-AE) for big data is introduced in [35]. Recently, dimension reduction with extreme learning machine is analyzed in [44]. In this chapter, ELM-AE is adopted as the core function of learning the features of input patches.

Most closely to our work is the hierarchical extreme learning machine (H-ELM) proposed in [43], which also uses ELM-AE to extract local receptive features and the training patches are randomly selected. In our proposed method, we drop out no patches and adopt a full connection network, which simplifies the network and need fewer parameters. Besides, we have a max pooling layer for each feature learning state for learning a compact feature representation, and the network architecture of our method is far simpler since we have no whitening pre-processing and the connections between the first layer and last layer.

5.3 Compact Feature Representation Using ELM

5.3.1 Overall Framework

To learning compact features efficiently, we proposed a shallow feature-learning architecture using ELM and the framework is shown in Fig. 5.2. Each central processing unit of this framework consists of three parts: patch-based ELM Auto-Encoder (AE), patch-based ELM decoder, and a max-pooling operation. After the main process, two post-processing modules are added, and they are binary hashing and block-wise histogram respectively.

For image classification, assume the database has N images I_i ($i \in [1,N]$, and the resolution of each image is $W \times H$) and the corresponding labels are Y_i. The target of learning a more efficient representation $R(\cdot)$of I_i is to minimize the loss function $\sum_{i=1}^{N} \|C(R(I_i) - Y_i)\|$, where $C(\cdot)$ denotes a classifier. In our work, the support vector machine (SVM, [184]) is adopted to find the patterns of the learned features.

5.3.2 Unit of CFR-ELM

To remove the influence of illumination, we implement a patch operation of subtracting the mean. Assume the width and the height of the patch are w_p and h_p respectively, then we convert each sliding $w_p \times h_p$ window of the image I_i into a column. All the columns construct a matrix X_i by subtracting their means one by one. The width of X_i is $(W - w_p + 1) \times (H - h_p + 1)$ and the height is $w_p \times h_p$.

An auto-encoder based on ELM is implemented to learn the features from all the X_i,

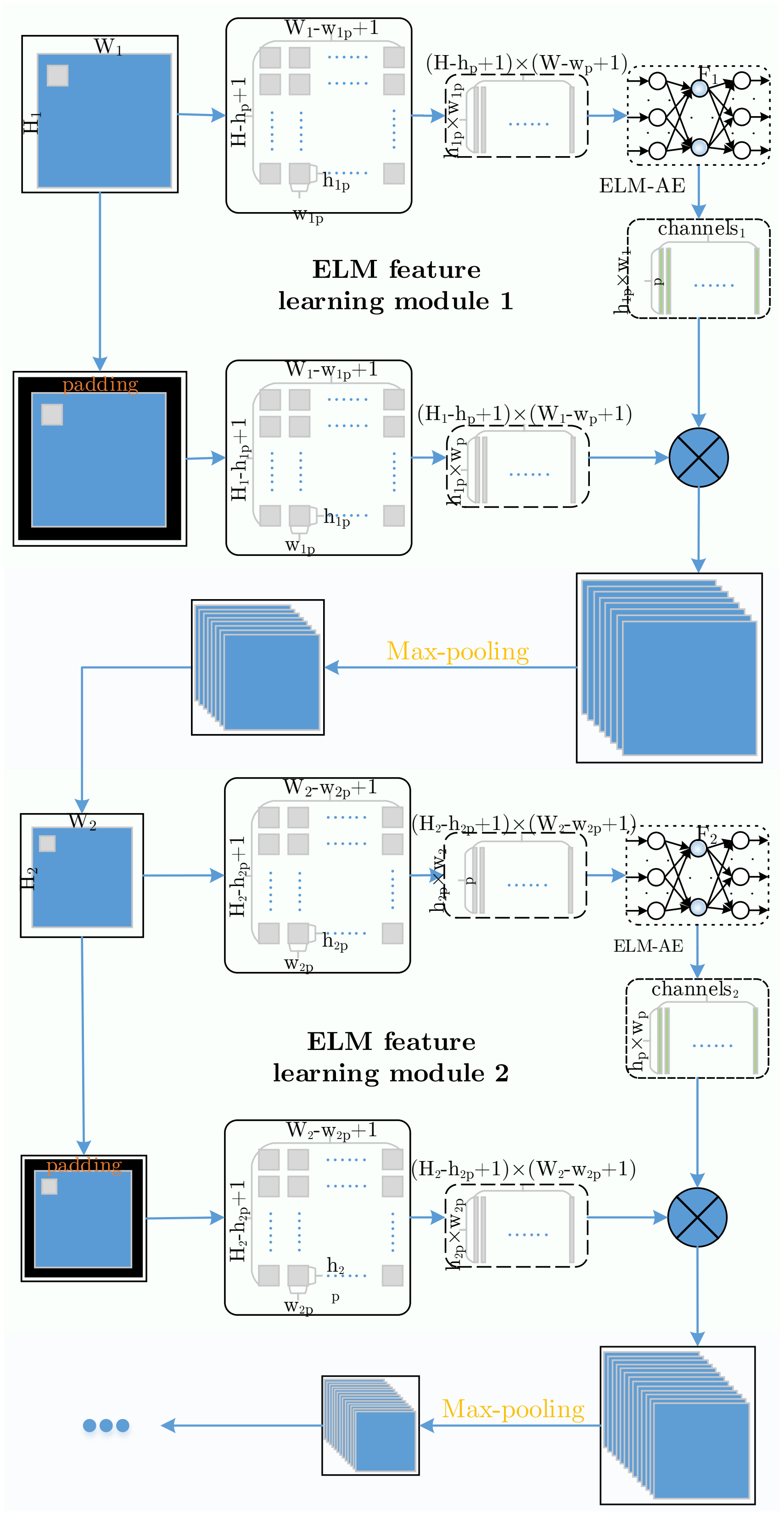


Figure 5.2: The framework of CFR-ELM.

which are denoted as X. We have introduced the details of ELM in Section 5.2. Three differences between ELM-AE and the basic ELM are listed below:

- The input weight $\mathbf{W}$ and the bias $\mathbf{B}$ are both orthogonal matrixes. That is, we have $\mathbf{W}^T\mathbf{W} = \mathbf{I}$ and $\mathbf{B}^T\mathbf{B} = \mathbf{I}$ with regard to ELM-AE.
- The output weight β is an orthogonal matrix. That is, $\beta^T\beta = \mathbf{I}$.
- The output of the ELM-AE network equals to the input.

Here, the number of hidden nodes N_{F_k} corresponds to the number of convolution filters F_k that we use in the k-th stage of the network. The filters are named as "channels" that are presented in Fig. 5.2. N_{F_k} can be learned by optimization, and empirically it can be assigned a number less than 10 due to the limitation of the resources.

With the filters we get from ELM-AE, we can compute the outputs of the patches (overall denoted as X) from the raw samples. The outputs are used to represent the patterns of the raw samples. To keep the information on the edges, we pad each sample with zeros in its surroundings. In our experiments, we pad $w-1$ columns and $h-1$ rows of zeros before the first element and after the last element along the horizontal and vertical directions respectively. After that, the same process of mean-removal patch operation is performed. Finally, we can achieve all the patches X of one sample, whose output O in the k-th stage of the network is expressed as

$$O = F_k * X, \tag{5.4}$$

where F_k is the filter banks that we have achieved from the ELM-AE, and X indicates the input samples that have been processed with zero-padding and mean-removal operations. The output O have N_{F_k} layers for each X in the k-th stage.

Noted that the size of each channel after the first ELM feature learning module is still $W_1 \times H_1$ with the cropping operation has been performed to countervail the impact of padding. So, the dimension of the output O of X increases to $W_1 \times H_1 \times N_{F_k}$, which requests lots of storage space and a large amount of computation. To learn a compact feature representation, O is fed to a max pooling layer (shown in Fig. 5.2 which is highlighted with golden poppy font).

Max pooling (MP) can significantly reduce the dimension of the input representation with losing limited valid information by the assumption that features are contained in the subregions. Each non-overlap subregion of O is processed with a max filter, and the maximum values are retained to represent the corresponding subregions. Assume the size of the subregion of max pooling is $W_{MP} \times H_{MP}$, then after passing the max pooling layer, the dimension of O will be $\lfloor W_1/W_{MP} \rfloor \times \lfloor H_1/H_{MP} \rfloor \times N_{F_K}$. Performances on image classification are provided and compared in Section 5.4.

Units of CFR-ELM are connected in sequence, and each unit is similar to the first unit but can have different parameters. Two units are adopted in our proposed shallow and compact feature representation architecture (denoted as S).

5.3.3 Post-processing Modules and Classification

CFR-ELM contains two post-processing modules: binary hashing and block wise histogram. They play an important role to transfer the output of the central processing units into a more efficient form. From the explanation of ELM, we can achieve N_F output images for each sample, and all the output images are reshaped into the same size with the input images. Here, we have $N_F = \prod_{k=1}^{S} N_{F_k}$.

There are $N_{F_{S-1}}$ outputs Ψ for one raw sample in the stage $S-1$, and each $\psi_\sharp \in \Psi$ ($\sharp \in [1, N_{F_{S-1}}]$) is processed by F_S filters in the stage S and generates F_S corresponding outputs Θ. Each $\theta_\hbar \in \Theta$ ($\hbar \in [1, F_S]$) is converted to a binary data matrix $\theta'_\hbar$ with a zero threshold. A hashing processing is performed on all the θ', and the operation is expressed as

$$T_{\sharp} = \sum_{\hbar=1}^{F_S} 2^{F_S-\hbar} \theta'_{\hbar}. \tag{5.5}$$

The block-wise histograms are computed for all the blocks we achieved with the sliding window technique. Assume the size of the block is represented as $w_b \times h_b$ and the overlap rate of the blocks is denoted as ε, then the stride size of the sliding windows is $\langle (1-\varepsilon)w_b \times (1-\varepsilon)h_b \rangle$ ($\langle \cdot \rangle$ is a rounding operator). We go through the entire $T_{\sharp}$ with this stride and compute the block-wise histogram for each block and combine them together to form the final features.

SVM is adopted to make a classification decision based on the obtained compact features. It constructs an optimal hyperplane for the independent samples in feature space to maximize the functional margin and minimizing the misclassification cost (approximately equivalent to the number of misclassification samples) simultaneously. The advantage of SVM is the sparseness of the solution and good generalization ability even in a high dimension space.

5.4 Experimental Results

With the development of techniques for feature extraction and image classification, the accuracy has been very high for many datasets. To better exhibit the effectiveness of our proposed method, we make the test more difficult by decreasing the number of training samples significantly and take more samples for testing. In our experiments, we test our algorithms on four classical image databases: a variant of MNIST (from Mixed National Institute of Standards and Technology [185]), Coil-20/100 (from Columbia Object Image Library [77, 78]), ETH-80 (from Eidgenössische Technische Hochschule Zürich [79]), and CIFAR-10 (from Canadian Institute for Advanced Research [12]). The first dataset is for digits classification, the second and third datasets are for controlled-environment objects classification, and the last one is for real-world objects classification.

To have a fair comparison with PCANet, the parameters assigned in our method are the same with the optimized PCANet (refer to [183] for more details on how to find the optimized parameters). We adopt two-stage($S = 2$) network, and the number of filters N_{F_1} and N_{F_2} are assigned to eight. The size of patches is 7×7, and max pooling size is 2×2. The block size w_b and h_b are both assigned as seven. Error Rate is adopted to evaluate the performance of the experiments, and ER equals to $\frac{\sum_{i=1}^{n}([P_i \neq T_i])}{N}$, where the value of $[\star]$ 1 when $\star$ is true and 0 if $\star$ is false. So, we know that the smaller ER is, the better the performance is. We have evaluated all the possible overlap rate ε from 0 to 0.9 by a step of 0.1, so for each dataset, we can obtain ten error rates.

The size of all the samples used in our experiments is consistent with the original resolution of these databases. We consider that obtaining higher accuracy by resizing the original images is unfair and not reasonable due to the curse of dimension. Additionally, we use less training samples and more test samples for the above three databases for a better illustration of the effectiveness of our algorithms. Even though our algorithm has the ability of generalization, some few parameters still should be noted. In the following subsections, we evaluate the performance of our method and PCANet on these databases separately. Few of the related state-of-the-art results are also be compared briefly.

5.4.1 Digits Image Classification

We first evaluate our proposed method on a variant of the widely used handwritten digits (0 - 9) image database MNIST, whose original version includes 60k training samples and 10k samples, and the state-of-the-art error rate is 0.21% [186]. The variant of MNIST has much less training samples and more test samples, which means that it is much more challenging. Specifically, 10k, 2k, and 50k samples are used for training, validation, and test respectively. Classification of Handwritten Digits is not an easy task since *thickness*

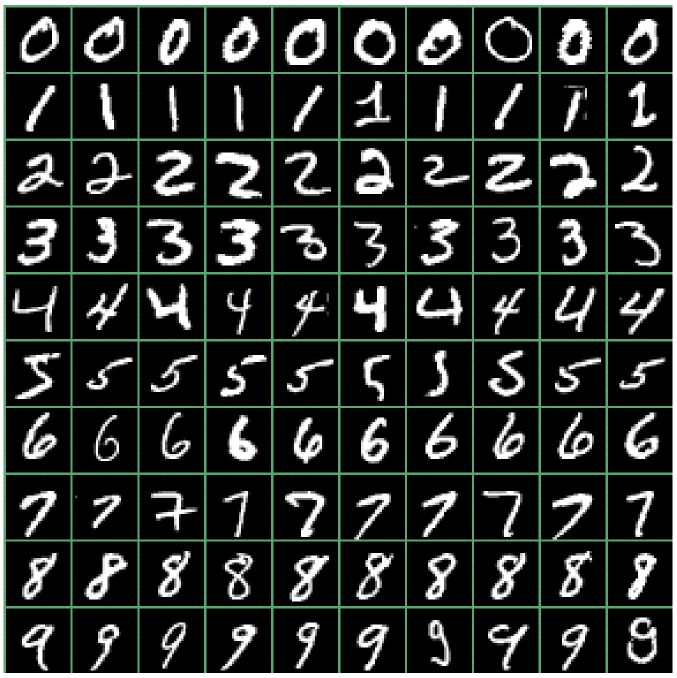


Figure 5.3: Several sample images from MNIST dataset.

and *rotation angles* may differ a lot even for the same digit. The resolution of each image is 28×28, and the amounts of these ten digits have an approximate uniform distribution. We have shown some samples from MNIST in Fig. 5.3 and the total number of each digit for training and test in Fig. 5.4.

For a better visualization of the mean-removal process, we have randomly selected ten samples for each digit from the database and show their mean-removal results in Fig. 5.5.

With the parameters we have introduced above, we have achieved ten error rates corresponding to the ten overlap rates ε. The results of PCANet and our method are indicated in Table.5.1 (the lower error rate has been set to bold). After the comparison, we know that our method has got a 0.980% error rate and achieved lower error rates for the nine out of ten overlap rates and it performs 4.18% (the average of the improvement rates) better than PCANet.

We have also performed our algorithm on the original version of MNIST dataset and obtains 0.49% error rate which outperforms PCANet's 0.62%.

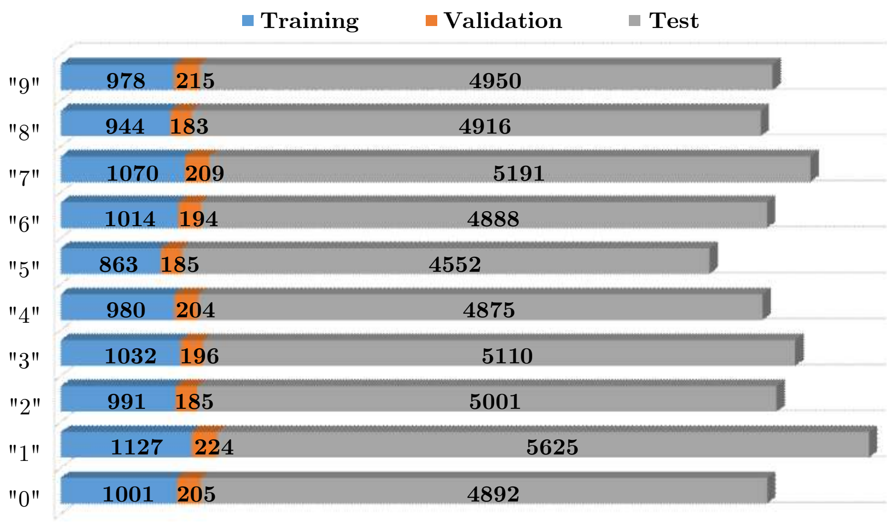


Figure 5.4: Amounts of the 10 digits in the MNIST variant.

5.4.2 Coil-20 and Coil-100 Databases

Coil-20 is a gray-level black-background image database of 20 objects (shown in Fig. 5.6 (a)). Objects are placed in a 360° turntable, and the table rotates 5° each time. So 72 images are captured for each object. Each image is cropped with a rectangle along the boundary of the object, and all images are resized to 128×128 by maintaining the shape of the objects (shown in Fig. 5.6 (b)). Coil-100 database is collected in the same controlled environment, but with more objects and all the images are colored. In the previous, they selected 1/3 or 1/2 of the samples for training. To make the classification problems more challenging, we split the samples into six groups and randomly select one of them for training, and the other groups are used for the test.

The parameters setting for our method and PCANet are same with to the parameters assigned on MNIST except that the max pooling size is 2×2. Error rates are list in Table.5.2 and Table.5.3

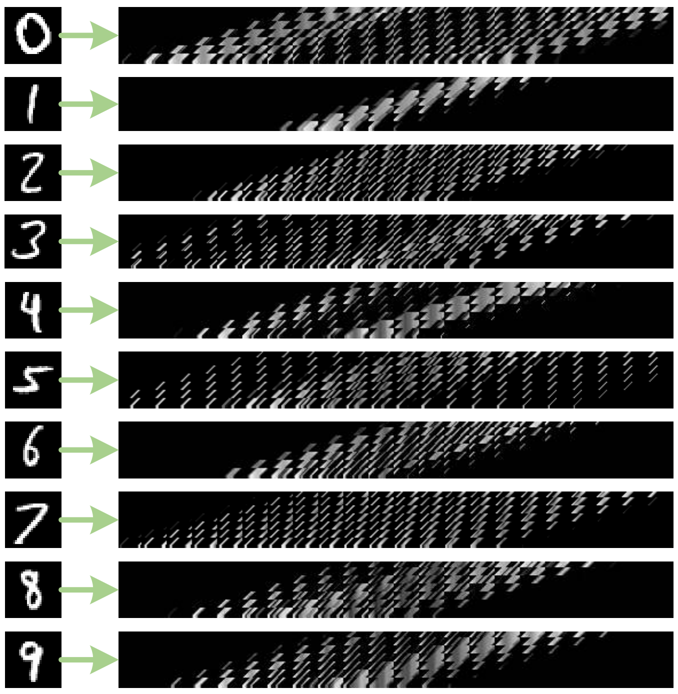
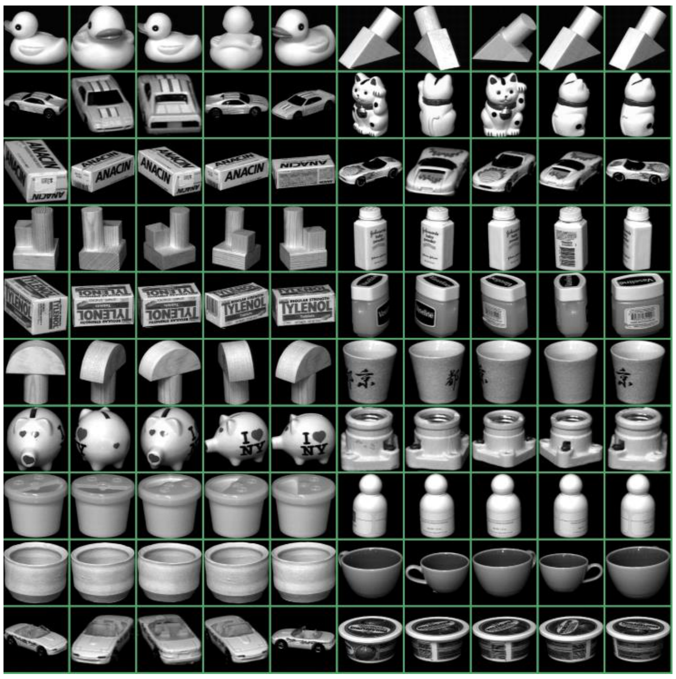


Figure 5.5: Results of the mean-removal patch operation. Each 28×28 image after the 7×7 patch mean-removal process.

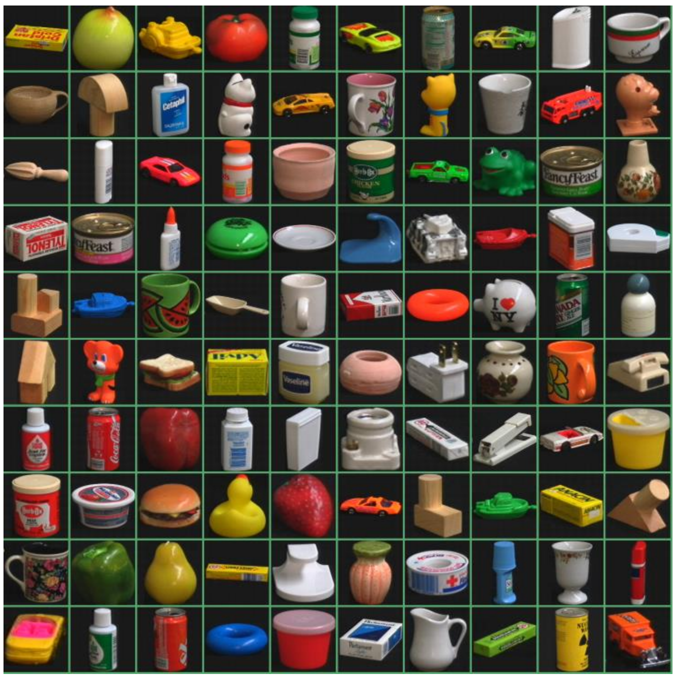
5.4.3 ETH-80 Database

Even though the images of both ETH-80 database and Coil-100 database are colored and of the same size, the former is more challenging than the latter due to the following reasons:

- Color Background. Images of ETH-80 have a non-uniform blue chromakeying background while images of Coil-100 have a black background. Non-uniform background increases the useless information and affects the classifier's performance further.
- Subcategories. ETH-80 contains 80 objects (shown in Fig. 5.7) from 8 chosen categories (or super categories), that is, each category has ten subcategories. Different subcategories are adopted for training and test, which requires that the feature learning algorithms be sufficiently capable to learn the common characteristics of various objects of same super categories.



(a)



(b)

Figure 5.6: Several sample images from Coil-20 (a) and Coil-100 datasets (b).

Table 5.1: Results of PCANet and CFR-ELM with and without max pooling on the MNIST variant dataset. The best results are obtained when the max pooling size is assigned as 1×1, which means the max pooling layer here can be ignored and the results with and without max pooling are the same, so the results only have two columns.

Overlap Rate (ε)	ER & ER+MP ($[1,1]$) (%)	
	PCANet	CFR-ELM
0.0	1.150	**1.082**
0.1	1.118	**1.052**
0.2	1.118	**1.022**
0.3	**1.066**	1.094
0.4	1.072	**1.010**
0.5	1.072	**1.034**
0.6	1.020	**0.990**
0.7	1.050	**1.040**
0.8	1.058	**1.022**
0.9	1.058	<u>**0.980**</u>

In the previous work [79, 187], they use not less than half of the dataset for training. While in our experiments, eight out of ten subcategories are used as test samples for each category.

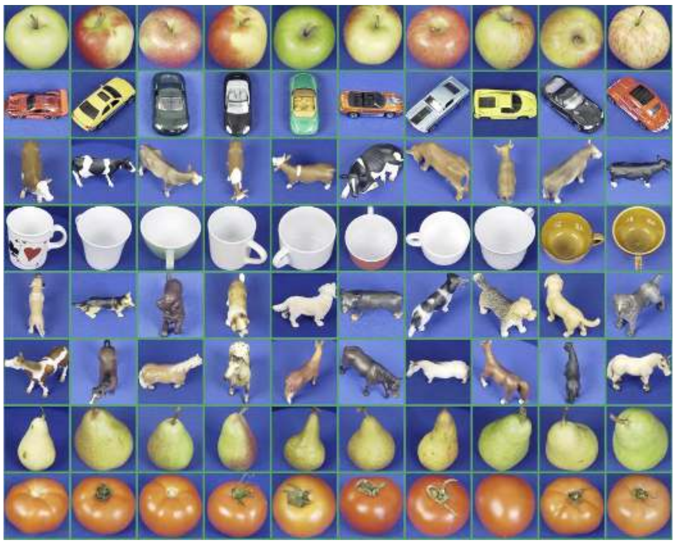


Figure 5.7: Several sample images from ETH-80 dataset.

We have compared our method with PCANet and the error rates are shown in Table.5.4 and the max pooling size is assigned as 2×2. It is clear that our method outperforms PCANet under all overlap rate.

Table 5.2: Results of PCANet and CFR-ELM with and without max pooling on the Coil-20 dataset.

Overlap Rate (ε)	ER (%)		ER + MP (%)	
	PCANet	CFR-ELM	PCANet	CFR-ELM
0.0	9.500	**8.000**	6.667	**5.500**
0.1	9.500	**8.000**	6.417	**4.750**
0.2	9.500	**8.083**	6.417	**5.167**
0.3	9.667	**8.167**	6.417	**5.333**
0.4	9.500	**<u>7.417</u>**	6.500	**4.750**
0.5	9.500	**8.250**	6.500	**4.500**
0.6	9.667	**<u>7.417</u>**	6.083	**4.417**
0.7	9.833	**7.583**	5.833	**<u>3.917</u>**
0.8	9.750	**7.750**	6.167	**5.000**
0.9	9.750	**7.833**	6.167	**4.667**

Table 5.3: Results of PCANet and CFR-ELM with and without max pooling on the Coil-100 dataset.

Overlap Rate (ε)	ER (%)		ER + MP (%)	
	PCANet	CFR-ELM	PCANet	CFR-ELM
0.0	11.830	**9.050**	6.183	**5.600**
0.1	11.680	**10.300**	6.083	**5.683**
0.2	11.680	**<u>8.550</u>**	6.083	**5.267**
0.3	11.780	**8.920**	6.033	**5.850**
0.4	11.770	**10.400**	5.967	**<u>5.133</u>**
0.5	11.770	**10.270**	5.967	**5.533**
0.6	11.850	**9.150**	5.617	**5.400**
0.7	11.870	**10.470**	5.700	**5.367**

5.4.4 CIFAR-10 Database

The CIFAR-10 dataset is more challenging than all the above databases. It contains 60k 32×32 color images in 10 classes, and each class has the same amount. All the images are collected from the real-world scene, and the images are cropped without removing the various background. Besides, objects are difficult to identify due to the different conditions of illumination, occlusion, and non-alignment. Some samples are randomly selected from CIFAR-10 database and shown in Fig. 5.8.

In our experiment, the experimental setup here is similar to [12, 188, 183]. However,

Table 5.4: Results of PCANet and CFR-ELM with and without max pooling on the ETH-80 dataset.

Overlap Rate (ε)	ER (%)		ER + MP (%)	
	PCANet	CFR-ELM	PCANet	CFR-ELM
0.0	20.351	**19.512**	18.052	**16.374**
0.1	**20.236**	21.113	16.054	**15.815**
0.2	20.236	**19.817**	17.744	**15.815**
0.3	20.122	**19.970**	17.687	<u>**15.100**</u>
0.4	20.160	**18.674**	16.924	**15.772**
0.5	20.160	**18.483**	17.565	**15.772**
0.6	19.817	**19.627**	**16.128**	16.301
0.7	**19.588**	19.855	**16.859**	16.872
0.8	19.360	<u>**18.293**</u>	17.219	**16.638**
0.9	**19.360**	20.922	17.500	**16.638**

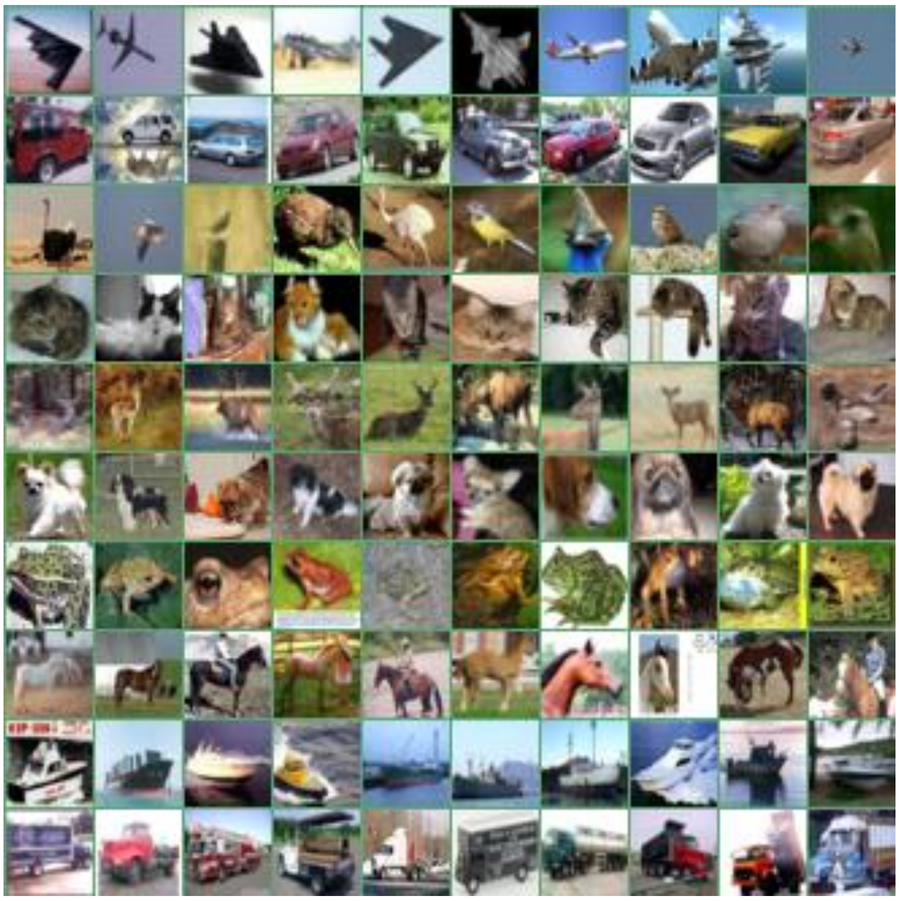


Figure 5.8: Several sample images from CIFAR-10 dataset.

due to the limited memories, we are 30k samples in the training set, and the remains are used for the test. And we only go through the overlap rate from 0 to 0.7 by the step of 0.1. For the experiments with max pooling layers, W_{MP} and H_{MP} are both set to 4. The results of our proposed method on CIFAR-10 are given in Table.5.5.

From the results we can see, our method outperforms PCANet in the majority of cases with or without map-pooling operation.

Table 5.5: Results of PCANet and CFR-ELM with and without max pooling on the CIFAR-10 dataset.

Overlap Rate (ε)	ER (%)		ER + MP (%)	
	PCANet	CFR-ELM	PCANet	CFR-ELM
0.0	27.606	**26.697**	26.130	**24.447**
0.1	27.720	<u>**25.865**</u>	**24.698**	24.954
0.2	27.710	**26.290**	25.909	**24.324**
0.3	27.447	**26.121**	26.094	**25.267**
0.4	26.440	**25.951**	25.260	<u>**23.760**</u>
0.5	**26.194**	26.689	**24.520**	25.157
0.6	**26.110**	26.376	24.930	**24.117**
0.7	26.550	**26.136**	26.058	**24.854**

5.5 Summary

In this chapter, we propose a novel compact feature representation method named CFR-ELM, which has a shallow network architecture. A detailed introduction is given to its framework, which contains critical units based on ELM, post-processing modules, and the classification using SVM. We have explained the reason why CFR-ELM can learn efficient and compact features by adopting ELM Auto-Encoder and the max pooling operation. The experiments on four typical diverse databases prove the effectiveness of our proposed method since CFR-ELM achieves impressive results by learning a better compact representation of the input data. Different overlap rates have been analyzed, and experiments with and without max pooling layers have been implemented and compared. The error rate of CFR-ELM is generally lower than PCANet.

Chapter 6

Target Coding for Extreme Learning Machine

Chapter 6 begins with the introduction of the definition of target coding and the most popular target coding method one-of-k coding. Then in this chapter, we have introduced Hadamard coding to ELM networks and explained how it works in details. Furthermore, we have analyzed and compared linearly independent coding methods and linearly dependent coding methods. At last, we have implemented all these target coding methods on the OCR letter dataset, and the results show that different target coding methods will indeed affect the performance of the same classifier.

6.1 Introduction

Feature extraction/learning and Target coding are two essential parts for supervised learning, especially for classification problems. Target coding (or labels encoding) is an indispensable part of the supervised learning algorithms. With years of research, there are lots of work have been done on how to extract effective features using ELMs [42, 43, 44, 45]. However, little work has been done on target coding for ELM. By far, most people adopt one-of-k target coding methods instantly when they use ELM classifiers, while others adopt the numeric coding methods to simplify the computation.

One-of-k (one-hot) coding comes from electronics, and there is only one ”hot” value in the coding list. It is introduced into the modern data science by [46], and the one ”hot” value is replaced by ‘1’ for simplicity reasons. One-of-k coding is the most frequently used target coding method for machine-learning based classification applications in recent years [33, 47, 48]. The assumptions of one-hot coding are that each target is independent and the similarities of any two targets is a constant value of zero.

However, this is not reasonable since similar targets should have a bigger similarity than dissimilar targets. This assumption is limited due to the complex relationship between targets. For example, C is a combination of A and B, then A’s target code can be ‘$10x$’, B’s code can be ‘$01y$’ and C’s code can be ‘$11z$’ (here, ‘x’, ‘y’ and ‘z’ indicate other properties of A, B, and C respectively). Sanjoy et al. have proved that the fly olfactory circuit assigns more similar ‘tags’ to the more similar pairs of odors [155]. Jiang et al. proposed an asymmetric hashing method to encode the training data and test data separately, and deep supervised learning is adopted to learning the features and update the hashing codes [189].

In this chapter, we have explored the effects of one-of-k coding methods on ELM classifiers. Besides, motivated by the properties of orthogonality and equal weight (each coding has the same number of non-zero elements) of Hadamard coding [49, 50, 51], we have compared its effects with one-of-k coding. Two simple coding methods ordinal coding and binary coding have also been compared to show the effective of one-of-k coding and Hadamard coding. To the best of our knowledge, this is the first discussion on target coding for ELM.

The structure of this chapter is as follows. We give a brief introduction on ELM-C and target coding in Section 6.2. Detailed analysis of target coding methods for ELM classifiers are explained in Section 6.3 and results are presented in Section 6.4. Finally, Section 6.5 summarizes our contribution and the future work.

6.2 Related Works

We have reviewed the basic ELM in Section 2.1, here we reintroduce ELM networks for classification with the training part and test part. The definition and an example of target coding are presented after ELM.

Extreme Learning Machine for Classification. ELM is a single (hidden) layer forward neural network which has been proposed in [38]. The weights ($\mathbf{W}$) and bias ($\mathbf{B}$) between the input layer and the hidden layer are randomly assigned, while only the weights (β) between the hidden layer and the output layer need to be computed by solving a linear equation. This network has been proved to have good generalization performance with high training and test speed with remarkable classification and regression ability [29, 67, 30, 64].

Assume there are N_1 training samples $\mathbf{X}_1$, and the feature dimension of each sample is N_I. We use $\mathbf{Y}_1$ to denote the label matrix which consists of coding vectors of the training target, and the length of each coding vector is N_O. The number of hidden nodes is N_H, then the dimension of $\mathbf{W}$ and $\mathbf{B}$ is $N_I \times N_H$ and $N_1 \times N_H$ respectively. Noted that each row in $\mathbf{B}$ is a duplicate of the first row. Now, with the full-connected single-layer forward network, the output of hidden nodes is

$$\mathbf{H}_1 = g(\mathbf{X}_1\mathbf{W} + \mathbf{B}), \tag{6.1}$$

where $g(*)$ is an activation function.

Training this network is equivalent to solving a linear system

$$\mathbf{Y}_1 = \mathbf{H}_1\beta, \tag{6.2}$$

and the minimum norm least squares solution is

$$\beta = \mathbf{H}_1^{\dagger}\mathbf{Y}_1. \tag{6.3}$$

Here $\dagger$ is the Moore-Penrose inverse operator, and the dimension of β is $N_H \times N_O$.

For the N_2 testing samples $\mathbf{X}_2$, similar steps are performed, and the output of hidden nodes are

$$\mathbf{H}_2 = g(\mathbf{X}_2\mathbf{W} + B), \tag{6.4}$$

and the predicted labels are

$$\widehat{\mathbf{Y}}_2 = \mathbf{H}_2\beta. \tag{6.5}$$

Here, $\widehat{\mathbf{Y}}_2$ is the predicted output of the test samples. To evaluate the classification accuracy, $\widehat{\mathbf{Y}}_2$ should be compared with $\mathbf{Y}_2$, and more details will be discussed on performance evaluation in the following sections.

To avoid the singularity of the matrix H, a unified learning network of ELM is proposed in [30] which minimizes the training error, and the norm of the output weight matrix by introducing a regularization factor C. The target is to minimize $\frac{1}{2}\|\beta\|^2 + \frac{1}{2}C\sum_{i=1}^{N}|\xi_i|^2$.

Here, ξ_i is the i-th training error. The solution of this function is

$$\beta = \begin{cases} \mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}_1\mathbf{H}_1^T\right)^{-1} \mathbf{Y}_1, N_1 < N_H. \\ \left(\frac{\mathbf{I}}{C} + \mathbf{H}_1\mathbf{H}_1^T\right)^{-1} \mathbf{H}^T\mathbf{Y}_1, N_1 \geq N_H. \end{cases} \tag{6.6}$$

Target Coding. Coding techniques have been developing for many years in the field of information science [190, 191], and various methods have been proposed for different propose like data compression, forward error correction, and encryption. Recently, coding for targets has gained a growing attraction from the machine learning community, especially supervised learning like classification [192, 193]. The detailed definition of target coding is given in [51], and here we restate it briefly as follows.

A target coding (code) $\mathscr{T}$ is defined as a matrix whose dimension is $n \times l$, where n is the number of unique categories and l is the length of a vector whose elements come from an integer set $\mathscr{S}$ (*alphabet set*). Each item in $\mathscr{S}$ is called a *symbol*, so every element in $\mathscr{T}$ is a symbol.

For example, we have a dataset which contains three instances. These three instances are divided into two groups, and their categories are ‘dog’, ‘cat’, ‘dog’ in turn. Assume $\mathscr{S} = \{0,1\}$, and $l = 4$. Then one possible assignment of target coding is

$$\mathscr{T} = \begin{bmatrix} 0,1,0,1 \\ 1,0,1,0 \\ 0,1,0,1 \end{bmatrix}, \tag{6.7}$$

so the coding of ‘dog’ is $[0,1,0,1]$, and ‘cat’ is $[1,0,1,0]$.

6.3 Target Codings for Extreme Learning Machine

From Equ.6.3 and Equ.6.5, we have

$$\widehat{\mathbf{Y}}_2 = \mathbf{H}_2\mathbf{H}_1^{\dagger}\mathbf{Y}_1. \tag{6.8}$$

Substitute Equ.6.1 and Equ.6.4 into Equ.6.8, we can obtain

$$\widehat{\mathbf{Y}}_2 = \left[g(\mathbf{X}_2\mathbf{W}+\mathbf{B})\right]\left[g(\mathbf{X}_1\mathbf{W}+\mathbf{B})\right]^{\dagger}\mathbf{Y}_1. \tag{6.9}$$

Assume $\mathbf{M} = \left[g(\mathbf{X}_2\mathbf{W}+\mathbf{B})\right]\left[g(\mathbf{X}_1\mathbf{W}+\mathbf{B})\right]^{\dagger}$, then $\mathbf{M}$ is determined by the training samples $\mathbf{X}_1$, test samples $\mathbf{X}_2$, weight matrix $\mathbf{W}$ and bias matrix $\mathbf{B}$. For ELM, $\mathbf{W}$ and $\mathbf{B}$ are randomly generated. So $\mathbf{M}$ will be a determined matrix once we know the training and test samples.

$$\widehat{\mathbf{Y}}_2 = \mathbf{M}\mathbf{Y}_1. \tag{6.10}$$

For the classification problem of a test set, one sample is a *true sample* when its predicted label is equal to its true label. Otherwise, it is a *false sample*. The accuracy of the classifier is

$$Accuracy = \frac{\#TS}{\#TS+\#FS}, \tag{6.11}$$

where $\#TS$ and $\#FS$ represent the number of true samples and false samples respectively.

For ELM classification, the true label matrix and predicted label matrix of a test set are denoted as $\mathbf{Y}_2$ and $\widehat{\mathbf{Y}}_2$. According to Equ.6.11, we can derive that the test accuracy of an ELM classifier is

$$Accuracy = \frac{\sum_{i=1}^{N}\mathfrak{B}\left(y_{2_i}, \mathfrak{S}\left(\mathbf{Y}_2, \widehat{y}_{2_i}\right)\right)}{N_2}, \tag{6.12}$$

where $\mathfrak{B}$ is a logical function and $\mathfrak{S}$ is a nearest-neighbor search function. Their analytical formulas are

$$\mathfrak{B}(\mathbf{v_1},\mathbf{v_2}) = \begin{cases} 1, \mathbf{v_1} = \mathbf{v_2} \\ 0, \mathbf{v_1} \neq \mathbf{v_2} \end{cases} \tag{6.13}$$

and

$$\underset{y_2'}{\operatorname{argmin}} \, \mathfrak{S}(\mathbf{Y}_2, \widehat{y_2}_i) = \{y_2' \in \mathbf{Y}_2 \mid D(y_2', \widehat{y_2}_i) = \min_{y_2''} D(y_2'', \widehat{y_2}_i)\}, \tag{6.14}$$

where $\mathbf{v_1}$ and $\mathbf{v_2}$ are arbitrary equal-length vectors, and $D(\cdot)$ is a distance matrix.

Currently, researchers usually set the final output of an ELM classifier as the index of the maximum element in the output vector of the last layer. This operation is compatible with the evaluation protocol we introduced above. The proof is as follows:

Assume the i-th element t_i is the biggest of an n-length output vector and other elements are represented as t_j ($j \in [1,n]$, and $j \neq i$). The i-th element of the target vector is 1, and other elements are all 0. According to nearest neighbors definition. $D_i = (t_i - 1)^2 + \sum_j {t_j}^2 = \sum_{k \in [1,n]} {t_k}^2 + 1 - 2t_i$. Since t_i is the biggest element, so D_i achieves the smallest distance between the target coding vector and the output coding vector.

We know that ELM has universal approximation capability, that is, ELM can approximate any continuous target functions. Here, we only consider the linear situation. A group of fix-length coding vectors have two situations: linearly dependent and linearly independent. The definition of linearly dependent of a set of vectors $V = \{\overrightarrow{v_1}, \overrightarrow{v_2}, ..., \overrightarrow{v_n}\}$ is: there exists a group of not all-zero scalars $\{a_1, a_2, ..., a_n\}$, such that $\sum_{i \in [1,n]} a_i \overrightarrow{v_i} = \overrightarrow{0}$. While for linearly independent, there doesn't exist such not all-zero scalars. So for $\mathbf{Y}_1$ and $\mathbf{Y}_2$, they can be derived (or selected) from linearly dependent coding vectors or linearly independent coding vectors.

There are many kinds of linearly dependent coding. Actually, a group of vectors must be linearly dependent if the dimension of the vector L_C is smaller than the number of vectors N_C. The most popular linearly dependent coding are ordinal coding and binary coding.

Ordinal coding. Ordinal coding means each class of target is represented as an ordinal integer, shown in Table 6.1. This is the simplest way to encode targets, and ordinal coding can not only be used for classification but also for regression.

Table 6.1: Ordinal coding for different classes.

Classes	Ordinal coding
#1	1
#2	2
#3	3
#$\cdots$	$\cdots$
#N	n

Binary coding. Binary coding indicates that each class of target is represented as a binary number. An example is shown in Table.6.2, where the dimension of each binary coding is two, and the amount is four. Binary coding can be used for single-label classification and multi-label classification.

Table 6.2: Two-bit binary coding for different classes.

Classes	Two-bit binary coding
#1	0 0
#2	0 1
#3	1 0
#4	1 1

There are also many methods for linearly independent coding, and the two most widely adopted coding are one-of-k coding and Hadamard coding.

One-of-k coding. Assume the dimension of one target code vector is K, then one-of-k coding means the *k*-th element of the vector of the *k*-th class is one and other elements are all zero. An simple example is shown in Table 6.3.

Hadamard coding. The Hadamard code (shown in Table.6.4) is one of the popular

Table 6.3: One-of-k coding for different classes.

Classes	one-of-k coding
#1	$1\,0\,0\,0\cdots 0$
#2	$0\,1\,0\,0\cdots 0$
$\#\cdots$	$\cdots$
#N	$0\,0\,0\,0\cdots 1$

Table 6.4: Hadamard coding for different classes.

Classes	Hadamard coding
#1	1 1 1 1
#2	1 0 1 0
#3	1 1 0 0
#4	1 0 0 1

error-correcting codes which are usually adopted in communication systems. Hadamard code is generated by using the Hadamard matrix $\mathscr{H}$, and the dimension (denoted as n) of $\mathscr{H}$ should be not less than the amount (denoted as m) of code we need. Each element of a Hadamard matrix is either -1 or $+1$, and $\mathscr{H}\mathscr{H}_T = nI$, where T is a transpose operator and I is a unit matrix. One way to produce a new Hadamard matrix is to embed an old Hadamard matrix into a specified template which is,

$$\mathscr{H}_{new} = \begin{bmatrix} \mathscr{H}_{old} & \mathscr{H}_{old} \\ \mathscr{H}_{old} & -\mathscr{H}_{old} \end{bmatrix}.$$

After we have the relationship of $\mathscr{H}_{new}$ and $\mathscr{H}_{old}$, we also need some initialization kernels. Here here we only list three cases: $n \in \{p * 2^q, n \geq 2 | p \in [1, 12, 20], q \in \mathbb{N}\}$, and the three kernels are

$$\mathscr{H}_2 = \begin{bmatrix} + & + \\ + & - \end{bmatrix},$$

$$\mathscr{H}_{12} = \begin{bmatrix}
++++++++++++ \\
+-+-+++---+- \\
+--+-+++---+ \\
++--+-+++--- \\
+-+--+-+++-- \\
+--+--+-+++- \\
+---+--+-+++ \\
++---+--+-++ \\
+++---+--+-+ \\
++++---+--+- \\
+-+++---+--+ \\
++-+++---+--
\end{bmatrix},$$

and

$$\mathscr{H}_{20} = \begin{bmatrix} ++++++++++++++++++++ \\ +--++----+-+-++++--+ \\ +-++----+-+-++++--+- \\ +++----+-+-++++--+-- \\ ++----+-+-++++--+--+ \\ +----+-+-++++--+--++ \\ +---+-+-++++--+--++- \\ +--+-+-++++--+--++-- \\ +-+-+-++++--+--++--- \\ ++-+-++++--+--++---- \\ +-+-++++--+--++----+ \\ ++-++++--+--++----+- \\ +-++++--+--++----+-+ \\ +++++--+--++----+-+- \\ ++++--+--++----+-+-+ \\ +++--+--++----+-+-++ \\ ++--+--++----+-+-+++ \\ +--+--++----+-+-++++ \\ +-+--++----+-+-++++- \\ ++--++----+-+-++++-- \end{bmatrix}.$$

So, for $p = 1$, $\mathscr{H}_4 = [\mathscr{H}_2, \mathscr{H}_2; \mathscr{H}_2, -\mathscr{H}_2]$, $\mathscr{H}_8 = [\mathscr{H}_4, \mathscr{H}_4; \mathscr{H}_4, -\mathscr{H}_4]$, ...

For $p = 12$, $\mathscr{H}_{24} = [\mathscr{H}_{12}, \mathscr{H}_{12}; \mathscr{H}_{12}, -\mathscr{H}_{12}]$, $\mathscr{H}_{48} = [\mathscr{H}_{24}, \mathscr{H}_{24}; \mathscr{H}_{24}, -\mathscr{H}_{24}]$, ...

For $p = 20$, $\mathscr{H}_{40} = [\mathscr{H}_{20}, \mathscr{H}_{20}; \mathscr{H}_{20}, -\mathscr{H}_{20}]$, $\mathscr{H}_{80} = [\mathscr{H}_{40}, \mathscr{H}_{40}; \mathscr{H}_{40}, -\mathscr{H}_{40}]$, ...

In order to be consistent with the above target coding method, we map '+1' and '-1' in the Hadamard matrix we produced to '0' and '1' respectively. Besides, we delete the elements in the first column and the first row because they are all the same. So the dimension of each Hadamard code is $n-1$. A greedy search method introduced in [51] is implemented in our experiments.

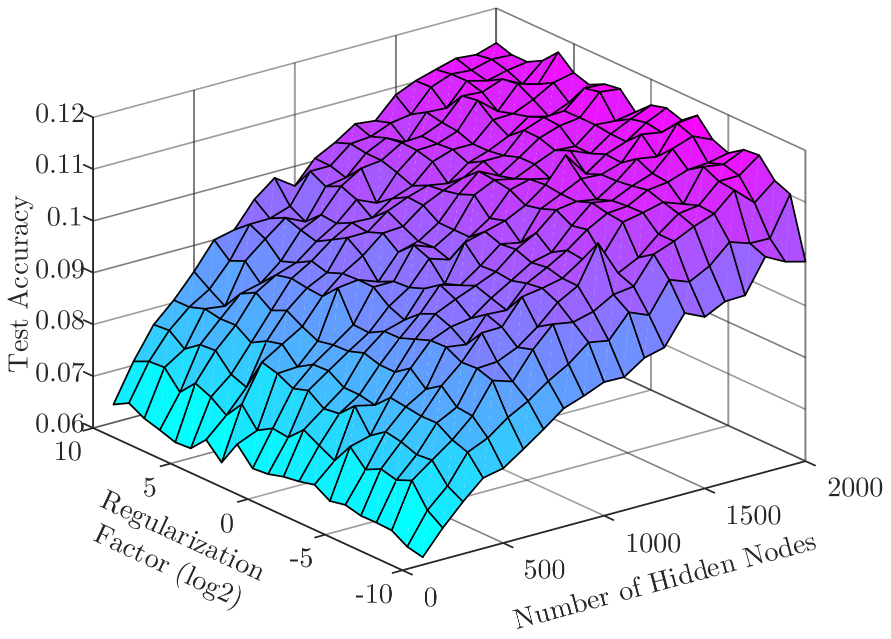
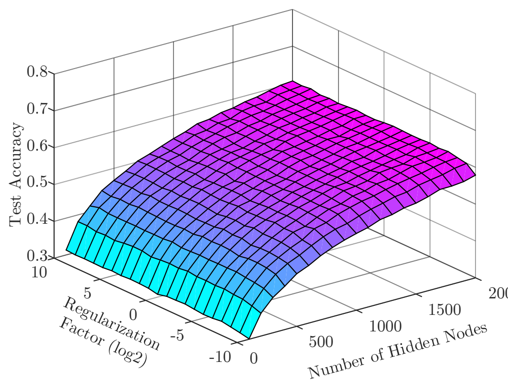


Figure 6.1: Training accuracy and test accuracy with ordinal target coding on OCR dataset using ELM.

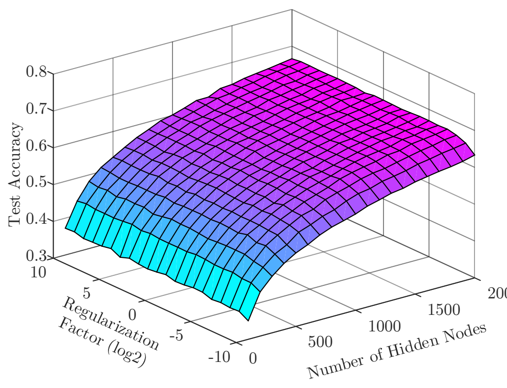
6.4 Experimental Results

We adopt an OCR letter dataset to show the effects of the four above-mentioned target coding methods on ELM classification. We have implemented the proposed method in Chapter 4 on the OCR letter database, and here we will review it briefly. OCR letter database is collected by Rob Kassel [194], and it contains 52152 samples of 26 letters (a-z). Each letter in this database is normalized and stored as a 16×8 binary image. Some examples from the OCR database are shown in Fig. 4.2), from which we can see that letters are difficult to recognize due to the large and varied deformation (e.g., partial occlusion and rotation).

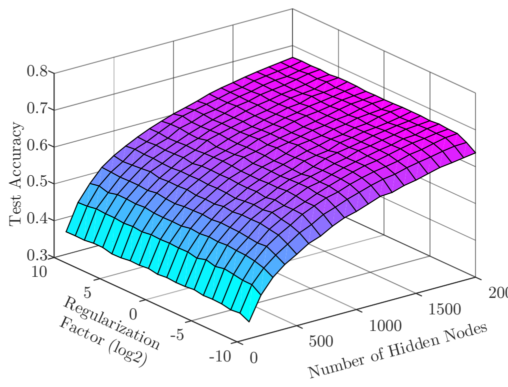
In the experiments, we feed the same feature data (raw pixel values) into the classifi-



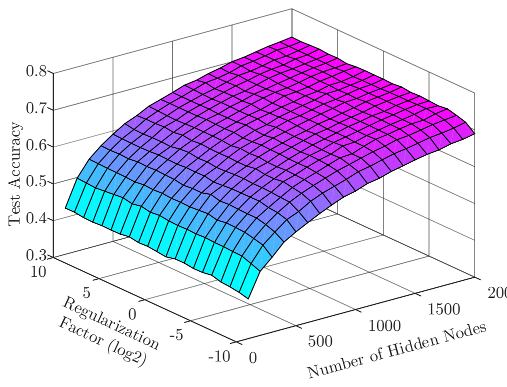
(a) $n = 5$



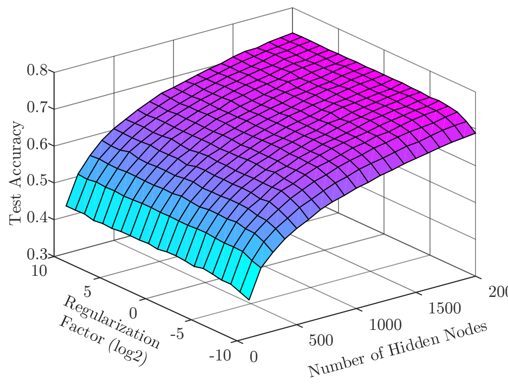
(b) $n = 7$



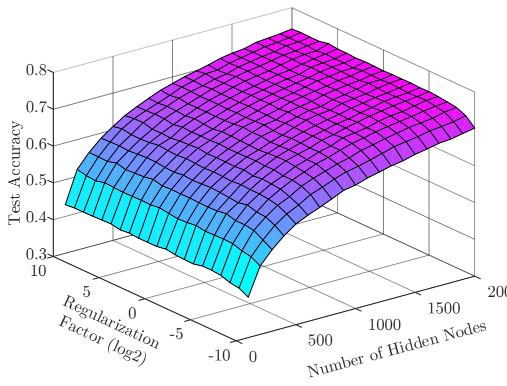
(c) $n = 9$



(d) $n = 11$



(e) $n = 13$



(f) $n = 15$

Figure 6.2: Training accuracy and test accuracy with binary target coding on OCR letter dataset using ELM.

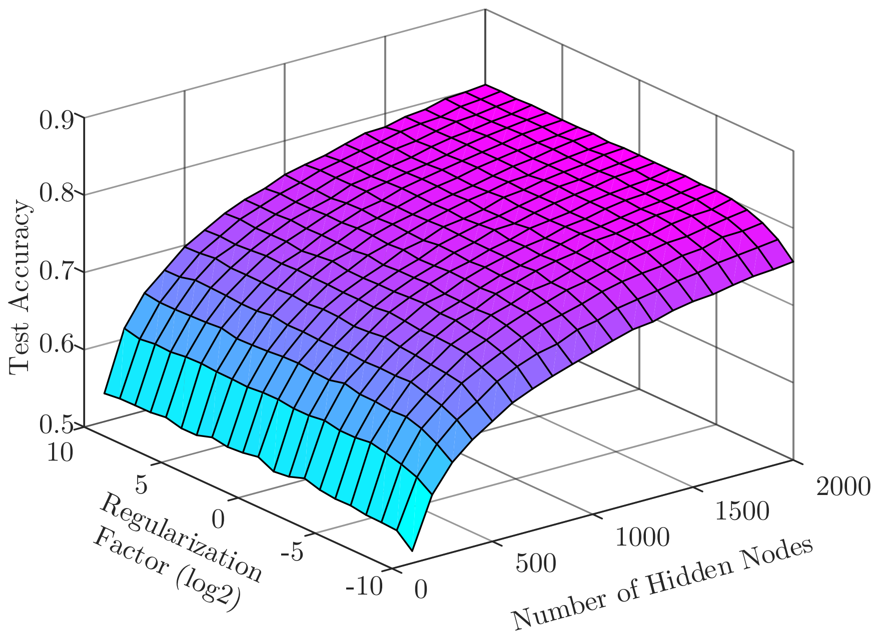


Figure 6.3: Training accuracy and test accuracy with one-of-k target coding on OCR letter dataset using ELM.

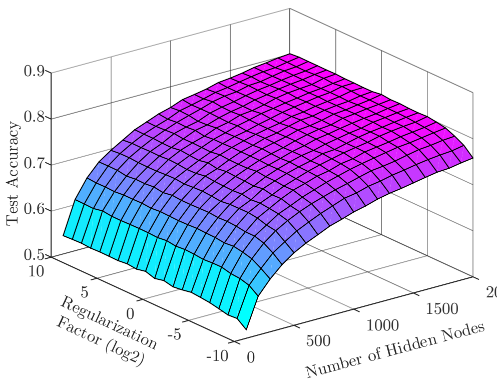
cation network, and all the parameters in the ELM classifier are kept consistently, including activation function, the number of hidden nodes and regularization factor. The activation function is uniformly set to the Sigmoid function, which is defined as

$$Sig(x) = \frac{1}{1+e^{-x}}. \tag{6.15}$$

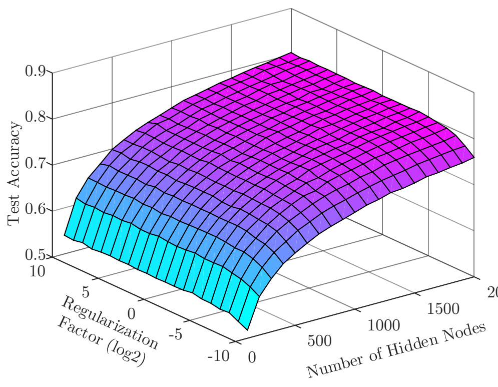
The number of hidden nodes ranges from 100 to 2000 with a step of 100, and the regularization factor is set to 2^i where i goes from -10 to 10 by a unit step.

We follow the default 10-fold cross-validation protocol to train and test the four target coding methods we have introduced in Section.6.3.

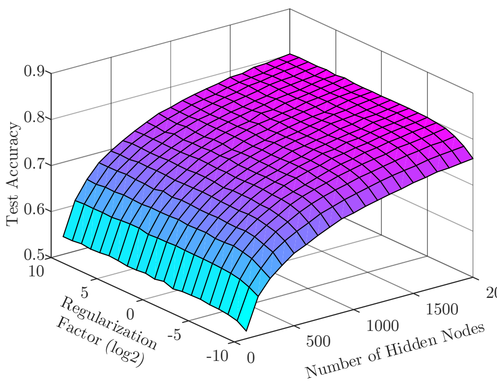
Ordinal target coding is an intuitive one-to-one mapping. 26 positive integers are required by the OCR letter database since it has 26 different letters (classes). According to the above parameters setting, the results of ordinal target coding shown in Fig. 6.1. We can see that the classification accuracy is getting higher with the number of hidden



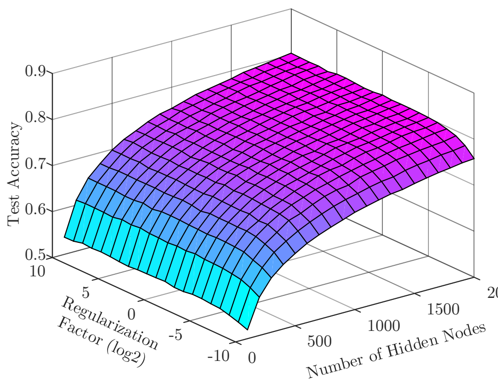
(a) $n = 31$



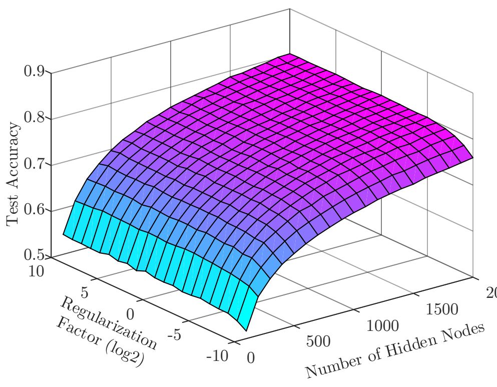
(b) $n = 39$



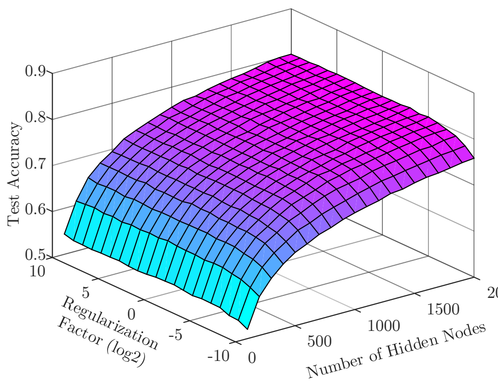
(c) $n = 47$



(d) $n = 63$



(e) $n = 79$



(f) $n = 127$

Figure 6.4: Training accuracy and test accuracy with Hadamard target coding on OCR letter dataset using ELM.

nodes increase.

We use n to indicate the length of a binary coding vector. Then, we have tested six different values of n, and they take the value from 5 to 15 by a step of 2 (n cannot be smaller than 5. Otherwise, this coding method cannot provide 26 different coding vectors). So here we have six corresponding results for binary target coding, and they are shown in Fig. 6.2. We can see that the performance becomes better as n increases.

Previously, we have introduced that each vector from one-of-k target coding only has one non-zero element. So for the OCR letter database, the one-of-k target coding corresponds to a 26-dimension identity matrix, and each row vector stands for a different letter respectively. The results of one-of-k target coding are shown in Fig. 6.3, and the accuracy increases as the number of hidden nodes rises.

Similar to the operation on binary target coding, we also selected six values for Hadamard target coding. After we obtained the corresponding Hadamard code, we randomly selected 26 coding vectors and assigned them to the 26 letters. The classification results with Hadamard target coding by using ELM on the OCR letter database are shown in Fig. 6.4.

We summarized the best results of the above experiments in Table.6.5, from which we can see that

- Also as linearly dependent coding, the binary method performs better than the ordinal method.
- For linearly independent coding, one-of-k target coding and Hadamard target coding are evenly matched.
- Linearly independent coding methods are better than linearly dependent coding methods in these experiments.

Table 6.5: Training and test accuracies of different target coding methods.

Target Coding Method		Training Accuracy	Test Accuracy
ordinal		0.1187	0.1172
one-of-k		0.8381	0.8033
Hadamard	n = 31	0.8378	0.8045
	n = 39	0.8380	0.8043
	n = 47	0.8382	0.8047
	n = 63	0.8381	0.8043
	n = 79	0.8375	0.8038
	n = 127	0.8376	0.8040
Binary	n = 5	0.6421	0.6088
	n = 7	0.6967	0.6697
	n = 9	0.7072	0.6705
	n = 11	0.7594	0.7250
	n = 13	0.7666	0.7331
	n = 15	0.7821	0.7484

6.5 Summary

In this chapter, we have explored the effectiveness of target coding on ELM classifiers. In particular, we have analyzed two linearly dependent coding (ordinal coding and binary coding) and two linearly independent coding (one-of-k coding and Hadamard coding). Experiments have been implemented on the OCR letter database to show the different effects of different target coding, and discussion is given based on the results. We have shown that different target coding methods indeed affect the performance of ELM classifiers.

Chapter 7

Conclusions and Future Works

Chapter 7 summarizes the four contributions in this thesis, and some recommendations for further research are given.

7.1 Conclusions

In this thesis, we have presented four enhanced aspects of ELMs. These contributions are individual but are all closely associated with the ELM networks. From the input end to the output end, we have proposed and explained the novel handcraft features pairwise distance vector and fed them into the input layer, introduced sparse binary random projection to concretize the weights between the input layer and the hidden layer, extended the architecture of the ELM network, and explored the performance of different target coding methods.

In our first work, we have proposed an ELM-based smiling and non-smiling images classification system by using a novel feature extraction method and the ELM. We focus on smile images classification system-level design and put forward a novel feature extraction method named Pair-wise Distance Vector, which can adequately reflect an object's shape and has properties of invariance of translation, rotation, and scaling. We have compared our algorithm with the existing state-of-the-art methods on the public smile detection dataset, and the results have shown that the accuracy of our ELM-based approach is higher and the dimension of our proposed feature vector is lower than the state-of-the-art methods.

In our second work, we focus on the optimization of the weights between the input layer and the hidden layer. We have introduced the sparse binary projection into ELM networks and proposed a novel ELM method named Fly-ELM, which is inspired by fruit flies' olfactory systems. The relationship between Gaussian random projections and sparse random binary projections is analyzed, and the results of the proposed Fly-ELM implemented on public OCR image classification dataset have shown that our algorithm achieves comparable accuracy but costs much less memory.

In our third work, we have concentrated on extending the depth of ELM networks and designed a novel compact feature learning method named CFR-ELM. CFR-ELM has a multi-layer architecture, and each module in it performs three operations: 1) patch-based mean removal; 2) ELM auto-encoder (ELM-AE) to learn features; 3) Max pooling to make the features more compact. We have tested our proposed method on four representative image classification datasets, and the results of these experiments have shown the effectiveness of our proposed method.

In our last work, we have explored the effectiveness of different target coding methods on ELM classifiers. The Hadamard coding is firstly introduced into ELM networks, and ordinal coding, binary coding, one-of-k coding, and Hadamard coding are all analyzed. We have also used the OCR dataset to verify the different effects of different target coding methods and obtained the conclusion that different target coding methods indeed affect the performance of ELM classifiers.

Noted that in this thesis, we have chosen different public datasets to test our proposed methods, since we focus on different aspects to enhance the ELM networks. As we have introduced, these four aspects include the extracted features fed into the input layer, the random binary projections between the input layer and the hidden layer, the deep architecture of the hidden layers, and the target coding methods for the output layer. In the first work, we choose smile images dataset, and smile detection is a typical specific application of image classification techniques. In the second and fourth work, we choose the simple OCR dataset because we focus on the comparison of our enhanced ELM with the basic/kernel ELM. Our third work focuses on feature learning for image classification, and we have tested our algorithm on four famous general image classification datasets to show that our proposed method outperforms other relevant state-of-the-art methods.

7.2 Future Works

Even though we have explored and enhanced four different aspects of ELM networks, we have realized that there are still many areas that can be explored, and many gaps can be filled in the future.

As we have introduced, we have proposed a novel and snappy feature descriptor in our first work, and this descriptor should be suitable for many other fields which have already obtained the landmarks. For example, after we get the locations of body key points, we could apply our algorithm to head pose estimation and action recognition effortlessly. The performance and generalization of our approach should be further examined when applied to other tasks.

There are still many directions to improve our last three work. For example, our second work shows the efficient of the sparse binary mapping between the input layer and the hidden layer, and it may be integrated with the sparse ELM, which constrains the sparsity of the output layer. Our third work is mainly motivated by PCANet, and we have achieved excellent results by embedding ELM-AE into the inspired architecture. Similarity, there are plenty of ELM variants and other shallow (or deep) neural networks have been proposed these years, the combinations and integration of them should develop more interesting outcomes. For our fourth work, since we have shown that different target coding methods indeed affect the performance ELM classifiers, this could be dug into further to find the optimal target coding methods in the future.

We have presented our four contributions on enhancing four aspects of ELM networks for image classification individually, and each component has been analyzed and explained clearly. However, we have not associated some or all of them into one system since there are so many combinations. The potential interaction between them and how to adjust them to optimize the whole system deserve further study.

Author's Publications

Journal Papers:

1. **Dongshun Cui**, Liyanaarachchi Lekamalage Chamara Kasun, Guanghao Zhang, Wei Han, Guang-Bin Huang and Yiqiang Chen. "Fly-ELM: Sparse Binary Extreme Learning Machine", *Cognitive Computation*, [Major Revision], 2018.
2. **Dongshun Cui**, Guang-Bin Huang and Tianchi Liu. "ELM based Smile Detection Using Distance Vector," *Pattern Recognition*, vol. 79, pp. 356-369, 2018.
3. **Dongshun Cui**, Guanghao Zhang, Kai Hu, Wei Han and Guang-Bin Huang. "Face recognition using total loss function on face database with ID photos," *Optics & Laser Technology*, pp. 1-7, 2017.

Conference Papers:

1. **Dongshun Cui**, Guanghao Zhang, Wei Han, Liyanaarachchi Lekamalage Chamara Kasun, Kai Hu and Guang-Bin Huang. "Compact Feature Representation for Image Classification Using ELMs," *Proceedings of IEEE International Conference on Computer Vision (ICCV) Workshops*, pp. 1015-1022, 2017.

2. **Dongshun Cui**, Kai Hu, Guanghao Zhang, Wei Han, Guang-Bin Huang. "Target Coding for Extreme Learning Machine," *Proceedings of ELM-2017*, 2017. [Accept].

3. **Dongshun Cui**, Guanghao Zhang, Kai Hu, Wei Han and Guang-Bin Huang. "Face Recognition Benchmark with ID Photos," *International Symposium on Artificial Intelligence and Robotics (ISAIR)*, pp. 27-35, 2017.

4. **Dongshun Cui**, Guang-Bin Huang, Liyanaarachchi Lekamalage Chamara Kasun, Guanghao Zhang and Wei Han. "ELMNet: Feature Learning using Extreme Learning Machines," *Proceedings of IEEE International Conference on Image Processing (ICIP)*, pp. 1857-1861, 2017.

5. **Dongshun Cui**, Guang-Bin Huang, Tianchi Liu. "Smile detection using Pair-wise Distance Vector and Extreme Learning Machine," *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN)*, pp. 2298-2305, 2016.

Bibliography

[1] V. Le, J. Brandt, Z. Lin, L. Bourdev, and T. S. Huang, "Interactive Facial Feature Localization," in *European Conference on Computer Vision*. Springer, 2012, pp. 679–692.

[2] U. Kowalik, T. Aoki, and H. Yasuda, "BROAFERENCE–A Next Generation Multimedia Terminal Providing Direct Feedback on Audience's Satisfaction Level," in *IFIP Conference on Human-Computer Interaction*. Springer, 2005, pp. 974–977.

[3] R. M. Haralick, K. Shanmugam *et al.*, "Textural Features for Image Classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. pp, no. 6, pp. 610–621, 1973.

[4] B. Javidi, *Image Recognition and Classification*. Marcel Dekker New York, 2002.

[5] A. Bosch, A. Zisserman, and X. Munoz, "Image Classification Using Random Forests and Ferns," in *Proceedings of the IEEE International Conference on Computer vision*. IEEE, 2007, pp. 1–8.

[6] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten Digit Recognition with a Back-Propagation Network," in *Advances in Neural Information Processing Systems*, 1990, pp. 396–404.

[7] G. Zhao and M. Pietikainen, “Dynamic Texture Recognition Using Local Binary Patterns with an Application to Facial Expressions,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 915–928, 2007.

[8] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, “Learning Multi-Label Scene Classification,” *Pattern Recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.

[9] M. A. Turk and A. P. Pentland, “Face Recognition using Eigenfaces,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition.* IEEE, 1991, pp. 586–591.

[10] J. Whitehill, G. Littlewort, I. Fasel, M. Bartlett, and J. Movellan, “Toward Practical Smile Detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 11, pp. 2106–2111, 2009.

[11] S. Mitra and T. Acharya, “Gesture Recognition: A Survey,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 3, pp. 311–324, 2007.

[12] A. Krizhevsky and G. Hinton, “Learning Multiple Layers of Features from Tiny Images,” *Technical Report*, vol. 1, no. 4, 2009.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[14] C. P. Papageorgiou, M. Oren, and T. Poggio, “A General Framework for Object Detection,” in *Proceedings of the IEEE International Conference on Computer vision.* IEEE, 1998, pp. 555–562.

[15] A. Yilmaz, O. Javed, and M. Shah, “Object Tracking: A Survey,” *ACM Computing Surveys*, vol. 38, no. 4, pp. 1–45, 2006.

[16] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[17] K. Li, B. Hariharan, and J. Malik, "Iterative Instance Segmentation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3659–3667.

[18] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[19] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution Gray-Scale and Rotation Invariant Texture Classification With Local Binary Patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[20] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1. IEEE, 2005, pp. 886–893.

[21] B. S. Manjunath and W.-Y. Ma, "Texture Features for Browsing and Retrieval of Image Data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 837–842, 1996.

[22] S. Wold, K. Esbensen, and P. Geladi, "Principal Component Analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[23] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K.-R. Mullers, "Fisher Discriminant Analysis with Kernels," in *Proceedings of the IEEE signal processing society workshop on Neural networks for signal processing IX*. IEEE, 1999, pp. 41–48.

[24] J. M. Keller, M. R. Gray, and J. A. Givens, "A Fuzzy K-Nearest Neighbor Algorithm," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. pp, no. 4, pp. 580–585, 1985.

[25] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," in *European Conference on Machine Learning*. Springer, 1998, pp. 137–142.

[26] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for Support Vector Machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1–27, 2011.

[27] C. Bishop, C. M. Bishop *et al.*, *Neural networks for pattern recognition*. Oxford University Press, 1995.

[28] T. K. Ho, "The Random Subspace Method for Constructing Decision Forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[29] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks," in *Proceedings of IEEE International Joint Conference on Neural Networks*, vol. 2. IEEE, 2004, pp. 985–990.

[30] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme Learning Machine for Regression and Multiclass Classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 513–529, 2012.

[31] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," in *Proceedings of the 22nd ACM International Conference on Multimedia*. ACM, 2014, pp. 675–678.

[32] G.-B. Huang, "An Insight into Extreme Learning Machines: Random Neurons, Random Features and Kernels," *Cognitive Computation*, vol. 6, no. 3, pp. 376–390, 2014.

[33] G. Huang, G.-B. Huang, S. Song, and K. You, "Trends in Extreme Learning Machines: A Review," *Neural Networks*, vol. 61, pp. 32–48, 2015.

[34] G.-B. Huang, Z. Bai, L. L. C. Kasun, and C. M. Vong, "Local Receptive Fields Based Extreme Learning Machine," *IEEE Computational Intelligence Magazine*, vol. 10, no. 2, pp. 18–29, 2015.

[35] L. L. C. Kasun, H. Zhou, G.-B. Huang, and C. M. Vong, “Representational Learning with Elms for Big Data,” *Intelligent System*, 2013.

[36] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[37] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[38] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme Learning Machine: Theory and Applications,” *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[39] G. Feng, G.-B. Huang, Q. Lin, and R. Gay, “Error Minimized Extreme Learning Machine with Growth of Hidden Nodes and Incremental Learning,” *IEEE Transactions on Neural Networks*, vol. 20, no. 8, pp. 1352–1357, 2009.

[40] E. Soria-Olivas, J. Gomez-Sanchis, J. D. Martin, J. Vila-Frances, M. Martinez, J. R. Magdalena, and A. J. Serrano, “BELM: Bayesian Extreme Learning Machine,” *IEEE Transactions on Neural Networks*, vol. 22, no. 3, pp. 505–509, 2011.

[41] M. Blot, M. Cord, and N. Thome, “Max-Min Convolutional Neural Networks for Image Classification,” in *Proceedings of IEEE International Conference on Image Processing*. IEEE, 2016, pp. 3678–3682.

[42] S. Shojaeilangari, W.-Y. Yau, K. Nandakumar, J. Li, and E. K. Teoh, “Robust Representation and Recognition of Facial Emotions Using Extreme Sparse Learning,” *IEEE Transactions on Image Processing*, vol. 24, no. 7, pp. 2140–2152, 2015.

[43] W. Zhu, J. Miao, L. Qing, and G.-B. Huang, “Hierarchical Extreme Learning Machine for Unsupervised Representation Learning,” in *Proceedings of IEEE International Joint Conference on Neural Networks*. IEEE, 2015, pp. 1–8.

[44] L. L. C. Kasun, Y. Yang, G.-B. Huang, and Z. Zhang, "Dimension Reduction with Extreme Learning Machine," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3906–3918, 2016.

[45] D. Cui, L. L. C. Kasun, G.-B. Huang, G. Zhang, and W. Han, "ELMNet: Feature Learning using Extreme Learning Machines," in *International Conference on Image Processing*. IEEE, 2017, pp. 1857–1861.

[46] C. W. Omlin and C. L. Giles, "Stable Encoding of Large Finite-State Automata in Recurrent Neural Networks with Sigmoid Discriminants," *Neural Computation*, vol. 8, no. 4, pp. 675–696, 1996.

[47] W. Li, C. Chen, H. Su, and Q. Du, "Local Binary Patterns and Extreme Learning Machine for Hyperspectral Imagery Classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 7, pp. 3681–3693, 2015.

[48] J. Cao, K. Zhang, M. Luo, C. Yin, and X. Lai, "Extreme Learning Machine and Adaptive Sparse Representation for Image Classification," *Neural Networks*, vol. 81, pp. 91–102, 2016.

[49] S. W. Golomb and L. D. Baumert, "The Search for Hadamard Matrices," *The American Mathematical Monthly*, vol. 70, no. 1, pp. 12–17, 1963.

[50] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Elsevier, 1977.

[51] S. Yang, P. Luo, C. C. Loy, K. W. Shum, X. Tang *et al.*, "Deep Representation Learning with Target Coding," in *Proceedings of AAAI Conference on Artificial Intelligence*, 2015, pp. 3848–3854.

[52] D. Cui, G.-B. Huang, and T. Liu, "Smile Detection Using Pair-Wise Distance Vector and Extreme Learning Machine," in *Proceedings of IEEE International Joint Conference on Neural Networks*. IEEE, 2016, pp. 2298–2305.

[53] ——, “ELM Based Smile Detection Using Distance Vector,” *Pattern Recognition*, vol. 79, pp. 356–369, 2018.

[54] D. Cui, L. L. C. Kasun, G. Zhang, W. Han, G.-B. Huang, and Y. Chen, “Fly-ELM: Sparse Binary Extreme Learning Machine,” *Cognitive Computation*, 2018.

[55] D. Cui, G. Zhang, W. Han, L. L. C. Kasun, K. Hu, and G.-B. Huang, “Compact Feature Representation for Image Classification Using ELMs,” in *Proceedings of IEEE International Conference on Computer Vision Workshops*. IEEE, 2017, pp. 1015–1022.

[56] D. Cui, K. Hu, G. Zhang, W. Han, and G.-B. Huang, “Target Coding for Extreme Learning Machine,” in *Proceedings of ELM-2017*. Springer, 2017, pp. 27–35.

[57] K. Hornik, M. Stinchcombe, and H. White, “Multilayer Feedforward Networks are Universal Approximators,” *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[58] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning Internal Representations by Error Propagation,” California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[59] 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, 1998.

[60] C. Cortes and V. Vapnik, “Support Vector Machine,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[61] G.-B. Huang and L. Chen, “Convex Incremental Extreme Learning Machine,” *Neurocomputing*, vol. 70, no. 16, pp. 3056–3062, 2007.

[62] G.-B. Huang, L. Chen, C. K. Siew *et al.*, “Universal Approximation Using Incremental Constructive Feedforward Networks with Random Hidden Nodes,” *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.

[63] G.-B. Huang, Q.-Y. Zhu, K. Mao, C.-K. Siew, P. Saratchandran, and N. Sundararajan, "Can Threshold Networks Be Trained Directly?" *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 53, no. 3, pp. 187–191, 2006.

[64] B. Wang, W. Zhao, Y. Du, G. Zhang, and Y. Yang, "Prediction of Fatigue Stress Concentration Factor Using Extreme Learning Machine," *Computational Materials Science*, vol. 125, pp. 136–145, 2016.

[65] K. Hu, W. Yang, and X. Gao, "Microcalcification Diagnosis in Digital Mammography using Extreme Learning Machine Based on Hidden Markov Tree Model of Dual-Tree Complex Wavelet Transform," *Expert Systems with Applications*, vol. 86, pp. 135–144, 2017.

[66] S. Wang, C. Deng, W. Lin, G.-B. Huang, and B. Zhao, "NMF-Based Image Quality Assessment Using Extreme Learning Machine," *IEEE Transactions on Cybernetics*, vol. 47, no. 1, pp. 232–243, 2017.

[67] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme Learning Machines: A Survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.

[68] G.-B. Huang and L. Chen, "Enhanced Random Search Based Incremental Extreme Learning Machine," *Neurocomputing*, vol. 71, no. 16, pp. 3460–3468, 2008.

[69] Z. Bai, G.-B. Huang, D. Wang, H. Wang, and M. B. Westover, "Sparse Extreme Learning Machine for Classification," *IEEE Transactions on Cybernetics*, vol. 44, no. 10, pp. 1858–1870, 2014.

[70] G. Huang, S. Song, J. N. Gupta, and C. Wu, "Semi-Supervised and Unsupervised Extreme Learning Machines," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2405–2417, 2014.

[71] F. BenoíT, M. Van Heeswijk, Y. Miche, M. Verleysen, and A. Lendasse, "Feature Selection for Nonlinear Models with Extreme Learning Machines," *Neurocomputing*, vol. 102, pp. 111–124, 2013.

[72] S. Ding, N. Zhang, J. Zhang, X. Xu, and Z. Shi, "Unsupervised Extreme Learning Machine with Representational Features," *International Journal of Machine Learning and Cybernetics*, vol. 8, no. 2, pp. 587–595, 2017.

[73] G.-B. Huang, N.-Y. Liang, H.-J. Rong, P. Saratchandran, and N. Sundararajan, "On-Line Sequential Extreme Learning Machine," *Computational Intelligence*, vol. 2005, pp. 232–237, 2005.

[74] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.

[75] W. Deng, Q. Zheng, and L. Chen, "Regularized Extreme Learning Machine," in *IEEE Symposium on Computational Intelligence and Data Mining*. IEEE, 2009, pp. 389–395.

[76] J. M. MartíNez-MartíNez, P. Escandell-Montero, E. Soria-Olivas, J. D. MartíN-Guerrero, R. Magdalena-Benedito, and J. GóMez-Sanchis, "Regularized Extreme Learning Machine for Regression Problems," *Neurocomputing*, vol. 74, no. 17, pp. 3716–3721, 2011.

[77] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia Object Image Library (COIL-20)," *Technical Report*, 1996.

[78] S. Nayar, S. Nene, and H. Murase, "Columbia Object Image Library (COIL-100)," *Technical Report*, 1996.

[79] B. Leibe and B. Schiele, "Analyzing Appearance and Contour Based Methods for Object Categorization," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2. IEEE, 2003, pp. II–409.

[80] B. Taskar, C. Guestrin, and D. Koller, "Max-Margin Markov Networks," in *Advances in Neural Information Processing Systems*, 2004, pp. 25–32.

[81] http://mplab.ucsd.edu, "The MPLab GENKI Database, GENKI-4K Subset."

[82] S. A. Papert, "The Summer Vision Project," 1966.

[83] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," in *Alvey Vision Conference*. Citeseer, 1988, pp. 147–151.

[84] R. O. Duda and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[85] J. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. pp, no. 6, pp. 679–698, 1986.

[86] H. Kauppinen, T. Seppanen, and M. Pietikainen, "An Experimental Comparison of Autoregressive and Fourier-Based Descriptors in 2D Shape Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 2, pp. 201–207, 1995.

[87] J. C. Harsanyi and C.-I. Chang, "Hyperspectral Image Classification and Dimensionality Reduction: An Orthogonal Subspace Projection Approach," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 32, no. 4, pp. 779–785, 1994.

[88] A. Vailaya, A. Jain, and H. J. Zhang, "On Image Classification: City images vs. landscapes," *Pattern Recognition*, vol. 31, no. 12, pp. 1921–1935, 1998.

[89] T. Ahonen, A. Hadid, and M. Pietikainen, "Face Description with Local Binary Patterns: Application to Face Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. pp, no. 12, pp. 2037–2041, 2006.

[90] C. Shan, “Learning Local Binary Patterns for Gender Classification on Real-World Face Images,” *Pattern Recognition Letters*, vol. 33, no. 4, pp. 431–437, 2012.

[91] C. Shan, S. Gong, and P. W. McOwan, “Facial Expression Recognition Based on Local Binary Patterns: A Comprehensive Study,” *Image and Vision Computing*, vol. 27, no. 6, pp. 803–816, 2009.

[92] Z. Yang and H. Ai, “Demographic Classification with Local Binary Patterns,” in *International Conference on Biometrics*. Springer, 2007, pp. 464–473.

[93] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2017, pp. 4700–4708.

[94] E. Malar, A. Kandaswamy, D. Chakravarthy, and A. G. Dharan, “a Novel Approach for Detection and Classification of Mammographic Microcalcifications Using Wavelet Analysis and Extreme Learning Machine,” *Computers in Biology and Medicine*, vol. 42, no. 9, pp. 898–905, 2012.

[95] A. Samat, P. Du, S. Liu, J. Li, and L. Cheng, “E^{2}LMs: Ensemble Extreme Learning Machines for Hyperspectral Image Classification,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 4, pp. 1060–1069, 2014.

[96] C. Chen, L. Zhou, J. Guo, W. Li, H. Su, and F. Guo, “Gabor-Filtering-Based Completed Local Binary Patterns for Land-Use Scene Classification,” in *Proceedings of IEEE International Conference on Multimedia Big Data*. IEEE, 2015, pp. 324–329.

[97] Q. Li, Q. Peng, J. Chen, and C. Yan, “Improving Image Classification Accuracy with ELM and CSIFT,” *Computing in Science and Engineering*, pp. 1–8, 2018.

[98] J. Zhang, S. Shan, M. Kan, and X. Chen, "Coarse-to-Fine Auto-Encoder Networks (CFAN) for Real-Time Face Alignment," in *European Conference on Computer Vision*. Springer, 2014, pp. 1–16.

[99] J. Tang, C. Deng, and G.-B. Huang, "Extreme Learning Machine for Multilayer Perceptron," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 4, pp. 809–821, 2016.

[100] J. J. d. M. S. Junior and A. R. Backes, "ELM Based Signature for Texture Classification," *Pattern Recognition*, vol. 51, pp. 395–401, 2016.

[101] S. Wan and J. Aggarwal, "Spontaneous Facial Expression Recognition: A Robust Metric Learning Approach," *Pattern Recognition*, vol. 47, no. 5, pp. 1859–1868, 2014.

[102] S. Elaiwat, M. Bennamoun, and F. Boussaid, "A Spatio-temporal Rbm-Based Model for Facial Expression Recognition," *Pattern Recognition*, vol. 49, pp. 152–161, 2016.

[103] U. Hess and P. Bourgeois, "You smile–I smile: Emotion Expression in Social Interaction," *Biological Psychology*, vol. 84, no. 3, pp. 514–520, 2010.

[104] S. M. Imran, S. M. Rahman, and D. Hatzinakos, "Differential Components of Discriminative 2d Gaussian-Hermite Moments for Recognition of Facial Expressions," *Pattern Recognition*, vol. 56, pp. 100–115, 2016.

[105] Y. Shinohara and N. Otsu, "Facial Expression Recognition Using Fisher Weight Maps," in *Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition*. IEEE, 2004, pp. 499–504.

[106] J. G. Daugman, "Complete discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 36, no. 7, pp. 1169–1179, 1988.

[107] C. Shan, “An Efficient Approach to Smile Detection,” in *Proceedings of IEEE International Conference and Workshops on Automatic Face and Gesture Recognition*. IEEE, 2011, pp. 759–764.

[108] ——, “Smile Detection by Boosting Pixel Differences,” *IEEE Transactions on Image Processing*, vol. 21, no. 1, pp. 431–436, 2012.

[109] L. An, S. Yang, and B. Bhanu, “Efficient Smile Detection by Extreme Learning Machine,” *Neurocomputing*, vol. 149, pp. 354–363, 2015.

[110] L. Wang and D.-C. He, “Texture Classification using Texture Spectrum,” *Pattern Recognition*, vol. 23, no. 8, pp. 905–910, 1990.

[111] M. Heikkilä, M. Pietikäinen, and C. Schmid, “Description of Interest Regions with Local Binary Patterns,” *Pattern Recognition*, vol. 42, no. 3, pp. 425–436, 2009.

[112] Z. Lei and S. Z. Li, “Fast Multi-Scale Local Phase Quantization Histogram for Face Recognition,” *Pattern Recognition Letters*, vol. 33, no. 13, pp. 1761–1767, 2012.

[113] S. Tian, U. Bhattacharya, S. Lu, B. Su, Q. Wang, X. Wei, Y. Lu, and C. L. Tan, “Multilingual Scene Character Recognition with Co-Occurrence of Histogram of Oriented Gradients,” *Pattern Recognition*, vol. 51, pp. 125–134, 2016.

[114] L. Zhang, W. Dong, D. Zhang, and G. Shi, “Two-Stage Image Denoising by Principal Component Analysis with Local Pixel Grouping,” *Pattern Recognition*, vol. 43, no. 4, pp. 1531–1549, 2010.

[115] Y.-H. Huang and C.-S. Fuh, “Face Detection and Smile Detection,” in *Proceedings of IPPR Conference on Computer Vision, Graphics and Image Processing*, 2009, p. 108.

[116] O. Rudovic, I. Patras, and M. Pantic, “Facial Expression Invariant Head Pose Normalization Using Gaussian Process Regression,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 28–33.

[117] A. Dhall, K. Sikka, G. Littlewort, R. Goecke, and M. Bartlett, "A Discriminative Parts Based Model Approach for Fiducial Points Free and Shape Constrained Head Pose Normalisation in the Wild," in *Proceedings of IEEE Winter Conference on Applications of Computer Vision*. IEEE, 2014, pp. 1028–1035.

[118] X. Zhu and D. Ramanan, "Face Detection, Pose Estimation, and Landmark Localization in the Wild," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 2879–2886.

[119] L. R. Rubin, "The Anatomy of a Smile: its Importance In the Treatment of Facial Paralysis," *Plastic and Reconstructive Surgery*, vol. 53, no. 4, pp. 384–387, 1974.

[120] N. Aifanti, C. Papachristou, and A. Delopoulos, "The MUG Facial Expression Database," in *Proceedings of the International Workshop on Image Analysis for Multimedia Interactive Services*. IEEE, 2010, pp. 1–4.

[121] Q.-Y. Zhu, A. K. Qin, P. N. Suganthan, and G.-B. Huang, "Evolutionary Extreme Learning Machine," *Pattern Recognition*, vol. 38, no. 10, pp. 1759–1763, 2005.

[122] B. Fasel and J. Luettin, "Automatic Facial Expression Analysis: A Survey," *Pattern Recognition*, vol. 36, no. 1, pp. 259–275, 2003.

[123] D. Cristinacce and T. Cootes, "Automatic Feature Localisation with Constrained Local Models," *Pattern Recognition*, vol. 41, no. 10, pp. 3054–3067, 2008.

[124] G. Yang and T. S. Huang, "Human face detection in a complex background," *Pattern Recognition*, vol. 27, no. 1, pp. 53–63, 1994.

[125] A. Colombo, C. Cusano, and R. Schettini, "3d Face Detection Using Curvature Analysis," *Pattern Recognition*, vol. 39, no. 3, pp. 444–455, 2006.

[126] Y. Ban, S.-K. Kim, S. Kim, K.-A. Toh, and S. Lee, "Face Detection Based on Skin Color Likelihood," *Pattern Recognition*, vol. 47, no. 4, pp. 1573–1585, 2014.

[127] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, "A Convolutional Neural Network Cascade for Face Detection," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5325–5334.

[128] B. Moghaddam, T. Jebara, and A. Pentland, "Bayesian Face Recognition," *Pattern Recognition*, vol. 33, no. 11, pp. 1771–1782, 2000.

[129] D. Jiang, Y. Hu, S. Yan, L. Zhang, H. Zhang, and W. Gao, "Efficient 3d Reconstruction for Face Recognition," *Pattern Recognition*, vol. 38, no. 6, pp. 787–798, 2005.

[130] X. Cao, Y. Wei, F. Wen, and J. Sun, "Face Alignment by Explicit Shape Regression," *International Journal of Computer Vision*, vol. 107, no. 2, pp. 177–190, 2014.

[131] Z. Zhang, P. Luo, C. C. Loy, and X. Tang, "Learning Deep Representation for Face Alignment with Auxiliary Attributes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 5, pp. 918–930, 2016.

[132] J. A. Suykens and J. Vandewalle, "Training Multilayer Perceptron Classifiers Based on a Modified Support Vector Method," *IEEE Transactions on Neural Networks*, vol. 10, no. 4, pp. 907–911, 1999.

[133] J. Whitehill and J. R. Movellan, "A Discriminative Approach to Frame-by-Frame Head Pose Tracking," in *Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition*. IEEE, 2008, pp. 1–7.

[134] M. Eckhardt, I. Fasel, and J. Movellan, "Towards Practical Facial Feature Detection," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 03, pp. 379–400, 2009.

[135] G. Littlewort, J. Whitehill, T. Wu, I. Fasel, M. Frank, J. Movellan, and M. Bartlett, "The Computer Expression Recognition Toolbox (CERT)," in *Proceedings of IEEE International Conference and Workshops on Automatic Face and Gesture Recognition*. IEEE, 2011, pp. 298–305.

[136] D. McDuff, R. El Kaliouby, T. Senechal, M. Amr, J. F. Cohn, and R. Picard, “Affectiva-MIT Facial Expression Dataset (AM-FED): Naturalistic and Spontaneous Facial Expressions Collected” In-the-Wild”,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 2013, pp. 881–888.

[137] L. Zhang, D. Tjondronegoro, and V. Chandran, “Facial Expression Recognition Experiments with Data from Television Broadcasts and the World Wide Web,” *Image and Vision Computing*, vol. 32, no. 2, pp. 107–119, 2014.

[138] Y. Gao, H. Liu, P. Wu, and C. Wang, “A New Descriptor of Gradients Self-Similarity for Smile Detection in Unconstrained Scenarios,” *Neurocomputing*, vol. 174, pp. 1077–1086, 2016.

[139] E. Murphy-Chutorian and M. M. Trivedi, “Head Pose Estimation in Computer Vision: A Survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 4, pp. 607–626, 2009.

[140] L. Rokach, “Taxonomy for Characterizing Ensemble Methods in Classification Tasks: A Review and Annotated Bibliography,” *Computational Statistics and Data Analysis*, vol. 53, no. 12, pp. 4046–4072, 2009.

[141] M. Liu, “Fingerprint Classification Based on Adaboost Learning from Singularity Features,” *Pattern Recognition*, vol. 43, no. 3, pp. 1062–1070, 2010.

[142] H.-C. Kim, S. Pang, H.-M. Je, D. Kim, and S. Y. Bang, “Constructing Support Vector Machine Ensemble,” *Pattern Recognition*, vol. 36, no. 12, pp. 2757–2767, 2003.

[143] S. An, W. Liu, and S. Venkatesh, “Fast Cross-Validation Algorithms for Least Squares Support Vector Machine and Kernel Ridge Regression,” *Pattern Recognition*, vol. 40, no. 8, pp. 2154–2162, 2007.

[144] G. C. Cawley and N. L. Talbot, "Efficient Leave-One-Out Cross-Validation of Kernel Fisher Discriminant Classifiers," *Pattern Recognition*, vol. 36, no. 11, pp. 2585–2592, 2003.

[145] S. Barua, M. M. Islam, X. Yao, and K. Murase, "MWMOTE–Majority Weighted Minority Oversampling Technique for Imbalanced Data Set Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 2, pp. 405–425, 2014.

[146] G.-B. Huang, J. Wu, and D. C. Wunsch, "Hierarchical Extreme Learning Machines," *Neurocomputing*, vol. 277, pp. 1–3, 2018.

[147] Y. Wang, Z. Xie, K. Xu, Y. Dou, and Y. Lei, "An Efficient and Effective Convolutional Auto-Encoder Extreme Learning Machine Network for 3d Feature Learning," *Neurocomputing*, vol. 174, pp. 988–998, 2016.

[148] M. Rigotti, O. Barak, M. R. Warden, X.-J. Wang, N. D. Daw, E. K. Miller, and S. Fusi, "The Importance of Mixed Selectivity in Complex Cognitive Tasks," *Nature*, vol. 497, pp. 585–590, 2013.

[149] O. Barak, M. Rigotti, and S. Fusi, "The Sparseness of Mixed Selectivity Neurons Controls the Generalization-Discrimination Trade-Off," *Journal of Neuroscience*, vol. 33, no. 9, pp. 3844–3856, 2013.

[150] R. I. Arriaga, D. Rutter, M. Cakmak, and S. S. Vempala, "Visual Categorization with Random Projection," *Neural Computation*, vol. 27, no. 10, pp. 2132–2147, 2015.

[151] J. Xie and C. Padoa-Schioppa, "Neuronal Remapping and Circuit Persistence in Economic Decisions," *Nature Neuroscience*, vol. 19, no. 6, pp. 855–861, 2016.

[152] E. L. Rich and J. D. Wallis, "What Stays the Same in Orbitofrontal Cortex," *Nature Neuroscience*, vol. 19, no. 6, pp. 768–770, 2016.

[153] M. van Heeswijk and Y. Miche, "Binary/Ternary Extreme Learning Machines," *Neurocomputing*, vol. 149, pp. 187–197, 2015.

[154] G.-B. Huang, "What Are Extreme Learning Machines? Filling the Gap Between Frank Rosenblatt's Dream and John Von Neumann's Puzzle," *Cognitive Computation*, vol. 7, no. 3, pp. 263–278, 2015.

[155] S. Dasgupta, C. F. Stevens, and S. Navlakha, "A Neural Algorithm for a Fundamental Computing Problem," *Science*, vol. 358, no. 6364, pp. 793–796, 2017.

[156] Y. Xia, K. He, P. Kohli, and J. Sun, "Sparse Projections for High-Dimensional Binary Codes," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3332–3339.

[157] H.-J. Rong, G.-B. Huang, N. Sundararajan, and P. Saratchandran, "Online Sequential Fuzzy Extreme Learning Machine for Function Approximation and Classification Problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 4, pp. 1067–1072, 2009.

[158] J. Martinovic, J. Dvorskỳ, and V. Snášel, "Sparse Binary Matrices," *ITAT*, vol. 2005, pp. 103–116, 2005.

[159] J. Snaider and S. Franklin, "Extended Sparse Distributed Memory and Sequence Storage," *Cognitive Computation*, vol. 4, no. 2, pp. 172–180, 2012.

[160] H. Höfling and R. Tibshirani, "Estimation of Sparse Binary Pairwise Markov Networks Using Pseudo-Likelihoods," *Journal of Machine Learning Research*, vol. 10, no. Apr, pp. 883–906, 2009.

[161] D. A. Rachkovskij, "Representation and Processing of Structures With Binary Sparse Distributed Codes," *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 2, pp. 261–276, 2001.

[162] W. Lu, K. Kpalma, and J. Ronsin, "Sparse Binary Matrices of LDPC Codes for Compressed Sensing," in *Data Compression Conference*, 2012, pp. 1–10.

[163] S. R. Olsen, V. Bhandawat, and R. I. Wilson, "Divisive Normalization in Olfactory Population Codes," *Neuron*, vol. 66, no. 2, pp. 287–299, 2010.

[164] A. Andoni and P. Indyk, "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions," in *Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 2006, pp. 459–468.

[165] D. G. Lowe, "Object Recognition from Local Scale-Invariant Features," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2. IEEE, 1999, pp. 1150–1157.

[166] W. Kahan, "IEEE Standard 754 for Binary Floating-Point Arithmetic," *Lecture Notes on the Status of IEEE*, vol. 754, pp. 1–30, 1996.

[167] E. Bingham and H. Mannila, "Random Projection in Dimensionality Reduction: Applications to Image and Text Data," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2001, pp. 245–250.

[168] Q. Shi, C. Shen, R. Hill, and A. van den Hengel, "Is Margin Preserved after Random Projection?" in *Proceedings of the 29th International Conference on Machine Learning*. Omnipress, July 2012, pp. 591–598.

[169] T. I. Cannings and R. J. Samworth, "Random-Projection Ensemble Classification," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 79, no. 4, pp. 959–1035, 2017.

[170] P. J. Huber, "The Behavior of Maximum Likelihood Estimates under Nonstandard Conditions," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1. Berkeley, CA, 1967, pp. 221–233.

[171] R. Savitha, S. Suresh, and H. Kim, "A Meta-Cognitive Learning Algorithm for an Extreme Learning Machine Classifier," *Cognitive Computation*, vol. 6, no. 2, pp. 253–263, 2014.

[172] T. Liu, Z. Yuan, J. Sun, J. Wang, N. Zheng, X. Tang, and H.-Y. Shum, "Learning to Detect a Salient Object," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 2, pp. 353–367, 2011.

[173] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy *et al.*, "DeepID-Net: Deformable Deep Convolutional Neural Networks for Object Detection," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2403–2412.

[174] M. Guillaumin, J. Verbeek, and C. Schmid, "Is That You? Metric Learning Approaches for Face Identification," in *Proceedings of the IEEE International Conference on Computer Vision*. IEEE, 2009, pp. 498–505.

[175] M. Blum, J. T. Springenberg, J. Wülfing, and M. Riedmiller, "A Learned Feature Descriptor for Object Recognition in RGB-D Data," in *Proceedings of IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 1298–1303.

[176] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning Rich Features from RGB-D Images for Object Detection and Segmentation," in *European Conference on Computer Vision*. Springer, 2014, pp. 345–360.

[177] Z. Guo, L. Zhang, and D. Zhang, "A Completed Modeling of Local Binary Pattern Operator for Texture Classification," *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1657–1663, 2010.

[178] Y. Huang, Z. Wu, L. Wang, and T. Tan, "Feature Coding in Image Classification: A Comprehensive Study," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 3, pp. 493–506, 2014.

[179] F. Liu, G. Lin, and C. Shen, "CRF Learning with CNN Features for Image Segmentation," *Pattern Recognition*, vol. 48, no. 10, pp. 2983–2992, 2015.

[180] D. Ciregan, U. Meier, and J. Schmidhuber, “Multi-Column Deep Neural Networks for Image Classification,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3642–3649.

[181] H. Fan, Z. Cao, Y. Jiang, Q. Yin, and C. Doudou, “Learning Deep Face Representation,” *arXiv preprint arXiv:1403.2802*, 2014.

[182] A. Coates, A. Ng, and H. Lee, “An Analysis of Single-Layer Networks in Unsupervised Feature Learning,” in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2011, pp. 215–223.

[183] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, “PCANet: A Simple Deep Learning Baseline for Image Classification?” *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5017–5032, 2015.

[184] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “LIBLINEAR: A Library for Large Linear Classification,” *Journal of Machine Learning Research*, vol. 9, no. Aug, pp. 1871–1874, 2008.

[185] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, “An Empirical Evaluation of Deep Architectures on Problems with Many Factors of Variation,” in *Proceedings of the 24th International Conference on Machine Learning*. ACM, 2007, pp. 473–480.

[186] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, “Regularization of Neural Networks Using Dropconnect,” in *Proceedings of the International Conference on Machine Learning*, 2013, pp. 1058–1066.

[187] M. Hayat, M. Bennamoun, and S. An, “Deep Reconstruction Models for Image Set Classification,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 4, pp. 713–727, 2015.

[188] R. Memisevic, “Gradient-Based Learning of Higher-Order Image Features,” in *Proceedings of the IEEE International Conference on Computer Vision*. IEEE, 2011, pp. 1591–1598.

[189] Q.-Q. Jiang and W.-J. Li, “Asymmetric Deep Supervised Hashing,” in *Proceedings of AAAI Conference on Artificial Intelligence*. AAAI Press, 2018.

[190] R. W. Hamming, *Coding and Theory*. Prentice-Hall, 1980.

[191] T. S. Rappaport, *Wireless Communications: Principles and Practice*. IEEE, 1996, vol. 2.

[192] D. J. Hsu, S. M. Kakade, J. Langford, and T. Zhang, “Multi-Label Prediction via Compressed Sensing,” in *Advances in Neural Information Processing Systems*, 2009, pp. 772–780.

[193] M. M. Cisse, N. Usunier, T. Artieres, and P. Gallinari, “Robust Bloom Filters for Large Multilabel Classification Tasks,” in *Advances in Neural Information Processing Systems*, 2013, pp. 1851–1859.

[194] R. H. Kassel, “A Comparison of Approaches to On-Line Handwritten Character Recognition,” Ph.D. dissertation, Massachusetts Institute of Technology, 1995.